Rushy Panchal 12/8/14

Senior Honors Focus: Humanities

Topic: What is the most suitable Machine Learning algorithm to create a Game AI that is indistinguishable from a human player?

Guiding Questions:

1. What is artificial intelligence in relation to games?
2. How and why is learning AI useful in games and in other applications?
3. What applicable machine learning algorithms currently exist, and how do they work?
4. How can we test the humanness of an AI?
    a. What is a Turing Test?
    b. How can the Turing Test be applied to Game AI?
5. What factors might render an AI distinguishable from a human player?
6. What form of strategy should the AI use to mimic a human player?
7. Which algorithm is the most appropriate choice for Game AI?
    a. Are the other algorithms more suitable for different applications of AI?

Key References**:**

Spronck, Pieter, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. "Adaptive Game AI with Dynamic Scripting." *Machine Learning* 63.3 (2006): 217-48. *Springer*. Kluwer Academic Publishers. Web. 7 Oct. 2014. <http://dx.doi.org/10.1007/s10994-006-6205-6>.

This article focuses on a special algorithmic approach to Game AI: dynamic scripting. Dynamic scripting is the process of creating a list of actions based on a set of rules (218). The actual learning algorithm comes from modifying the rule-base (set of rules). Each rule has a certain weight, which is the probability that it will be used. Machine learning occurs by modifying the weights based on observations (221 – 222). The rule-base can also be dynamically generated by the program through the use of evolutionary algorithms (236). For the game to actually be entertaining for users, the AI must seem human (it cannot play too well). To do this, the weights of “good” rules can be intentionally kept low (239). From this article, I learned about a new Machine Learning algorithm, which is especially tailored for Game AI. Although this algorithm is similar to Neural Networks, it is much simpler and so I believe it will be easier to use.

West, Robert L., Christian Lebiere, and Dan J. Bothell. "Cognitive Architectures, Game Playing, and Human Evolution." *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*(2006): 103-23. *Rensselaer Cognitive Science Department*. Cambridge University Press, May 2008. Web. 12 Oct. 2014. <http://www.cogsci.rpi.edu/~rsun/book/West-Lebiere-Bothell.pdf>.

The authors write about game theory and how humans make decisions. One major conclusion is that humans do not play games optimally (win most of the time in the long term). Instead, we tend to play maximal strategies. That is, humans look for patterns, even in random games where the opposing agent's actions cannot be determined, and try to win in the short term. To model human strategy, therefore, game AI must use maximal tactics. However, this would require the AI to predict outcomes when playing against a human. Because the human is also using maximal strategy, the AI faces the difficult task of determining whether one maximal plan is better than another. In addition, game AI are very good at implementing optimal tactics, so this presents another obstacle. From this article, I gained a greater understanding of how we, as humans, play games. By learning this, I can attempt to model an AI's strategy based on this maximal model.

Nilsson, Nils J. *Introduction to Machine Learning*. Stanford: Stanford U, 2005. Stanford University, 19 June 2010. Web. 17 Aug. 2014. <http://robotics.stanford.edu/~nilsson/MLBOOK.pdf>.

This book draft contains very detailed descriptions on Machine Learning and its associated algorithms. He goes into detail in multiple popular algorithms (including Neural Networks, Decision Trees, and both Supervised and Unsupervised Learning). Nilsson also provided examples for each algorithm. One aspect that I found extremely interesting was his analysis of statistical noise. Because most of these algorithms rely on pattern-recognition in colossal sets of data, they must be able to account for imperfections in the data. He proposes using numerical bias to help find the best solution. I had not considered this prior to reading this source, so it was very insightful.

The book draft greatly added to my understanding of Artificial Intelligence and Machine Learning. Most of the algorithms that he described are very complex, but I was able to grasp them better after reading them. In addition, by reading about these learning methods, I was able to create and further the idea for my own algorithm.

“Machines will be capable, within twenty years, of doing any work a man can do.”

– Herbert Simon

Indistinguishable AI: Searching for Ourselves

You wake up one day to find out that your best friend, whom you have known for your entire life, is not human. He was able to convince you that he was human, making you unaware that he is simply a program running on a computer – just a bunch of data on a hard drive. Although the idea of computer programs imitating humans seems like a distant, science-fiction dystopia, it is more realistic than many believe. As technology progresses, the difference between reality and the digital world fades away. We can simulate real-life experiences to the extent that we cannot differentiate between what is real and what we truly perceive. Modern video games emulate different scenarios that blend the differences between reality and fantasy. A large aspect of these video games is Artificial Intelligence (AI), which drives non-player characters and opponents. Many games now employ AI that seem to be human players because of their ability to adapt to changing situations. However, there are numerous different methods to implement AI, each with its own benefits and drawbacks.

Commonly, Artificial Intelligence is described as any computer program that is able to perform different actions based on different conditions. For example, most non-player characters in games are referred to as AI, simply because their actions are dependent on the situation. Some programs that act as chat bots may seem to exhibit intelligence, even though they simply choose a response based on matched keywords. However, AI is not simply the ability to make decisions. Instead, a program is a qualified AI when it is able to modify its own behavior based on its experience. Stephen Hawking believes that “intelligence is the ability to adapt to change.” Indeed, intelligent behavior is derived from this ability. Specifically, self-modifying AI should utilize Machine Learning (ML), a special branch of Artificial Intelligence research that focuses on self-improving programs.

Self-improving, or learning, AI are already used in numerous different applications. Technology and software industries frequently utilize learning AI. Google and Facebook make use of Deep Learning[1] in their image recognition software. Every time their software correctly identifies the subject of an image, it becomes better at doing so again due to its learning algorithms.[2] Apple's Siri, a digital assistant, uses voice and text analysis to understand what it is told. Likewise, Google's search engine uses numerous algorithms[3] to identify what the user is searching for and find the best matching results through a complex ranking process.[4] Although the task of distinguishing the subject of an image and understanding conversation is fairly trivial for humans, it is enormously complex for computers. As a result, most software companies use Machine Learning algorithms to create self-improving programs that can accomplish these tasks. Without learning AI, our technology would be very different, for the worse.

As technology continues to seem more realistic, AI start to seem more human. Human-like AI can bridge the gap between the digital and the "real" world. Current researchers are still unable to definitively state how the human mind functions. Philosophers continue to argue about what it means to be human and what our purpose is. By creating AI that are similar to us,

[1] Deep Learning is an algorithm in which the software is trained on successively more complex data.
[2] Quoc V. Le., Marc'aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng, "Building High-level Features Using Large Scale Unsupervised Learning," *Google Research* (2012). <http://research.google.com/archive/unsupervised_icml2012.pdf>.
[3] An algorithm is a method or approach of solving a particular problem. In this case, Google's search algorithm is the method by which the server finds what the user is looking for.
[4] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Stanford Infolab*. <http://infolab.stanford.edu/~backrub/google.html>.

we are able to learn more about ourselves. Philip K. Dick's novel *Do Androids Dream of Electric Sheep?* depicts a coming-of-age for the human race as a whole. Various characters learn about what being human truly means through their interactions with intelligent robots.[5] Similar to the characters in the novel, the human race will be able to understand who we are by creating and working with humanlike programs.

In video games, learning AI provide increased entertainment value because they constantly evolve. While some players may manipulate a flaw in static AI,[6] they are unable to take advantage of a game AI that makes use of Machine Learning.[7] In general, there are two forms of game AI. Some AI attempt to play the best they can. Most of these AI use searching functions to make the best-possible decision. IBM's Deep Blue beat Garry Kasparov in 1997 by utilizing a complex search function, evaluating nearly 200 million positions per second.[8] Similarly, IBM's supercomputer, Watson, applied an enormous search and evaluation process to determine answers in its *Jeopardy!* match against Ken Jennings and Brad Rutter. However, it also utilized self-modification to improve itself over time.[9] Arthur Samuel created a simple Checkers AI, but was disappointed when he repeatedly defeated it. So, he modified it such that

[5] Philip K Dick, *Do Androids Dream of Electric Sheep?*, (New York: Ballantine, 1996).

[6] A static AI is one that does not change, and simply applies the same strategy repeatedly.

[7] Aaron Khoo and Robert Zubek, "Applying Inexpensive AI Techniques to Computer Games," *IEEE Intelligent Systems* 17.4 (2002): 48. <http://www.cs.northwestern.edu/~khoo/downloads/papers/IEEEIntelSystems2002InexpensiveAI.pdf>.

[8] IBM, "Deep Blue," *IBM 100*. <http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>.

[9] IBM, "IBM Watson: The Science Behind an Answer," *YouTube: IBM* (2011). <http://www.youtube.com/watch?v=DywO4zksfXw>.

it would learn how to play better based on previous games.[10] All of these AI work to find the best possible action given their situation.

On the other hand, some AI attempt to provide the most entertainment value. This often leads to the AI not playing the best possible. By playing the best it can, the AI can quickly become undefeatable, which is not as enjoyable for the user. Instead, it attempts to play at an average level so that it sometimes wins and sometimes loses. Each of these AI uses a different method in the game. There is a variety of machine learning algorithms that are suitable for different tasks. For human-like learning game AI, there are four potential algorithms: neural networks, dynamic scripting, the Learnable Evolution Model, and Markov Decision Processes.

Neural networks are an attempt to model how the brain works.[11] Just as the brain uses a highly-connected, complex network of physical neurons to perform its functions, neural networks make use of small nodes which have very specific tasks. Each neuron (also called a perceptron) in a neural network has five major components and one minor component. The inputs of the neuron are values (usually numeric) provided either externally or from other neurons. In addition, there is a weight associated with each input. This weight is multiplied with the input to provide the actual value used by the neuron. To account for statistical noise, there is usually a bias attached with each neuron. This bias allows the neural network to consider deviations in the data because the data available to the network are rarely perfect. Because it is not essential for the actual learning process, the bias is considered a minor component. The

[10] Arthur Samuel, "Some Studies in Machine Learning Using the Game of Checkers. II-Recent Progress," (1967). <http://researcher.watson.ibm.com/researcher/files/us-beygel/samuel-checkers.pdf>.

[11] Daniel Shiffman, and eds. Shannon Fry, "Neural Networks," *The Nature of Cod*e: 445. <http://natureofcode.com/book/chapter-10-neural-networks/>.

activation function directly models those of actual neurons in the brain; it decides whether or not the neuron should “fire” (release an output). If the function decides not to release an output, then the output defaults to zero. A value of zero does not affect any other neurons.

Finally, the neuron has an output, which is the value returned by combining the inputs in some fashion, which varies depending on the neural network.[12] For most neurons, this is just the sum of the weighted inputs. By itself, a singular neuron cannot do much. However, it can achieve amazing feats if many neurons are connected into a network. The learning process itself occurs by modifying the weights associated with each input. By modifying weights on the neural connections, specific neurons can be given increased priority, because of how the output is calculated. A larger weight results in a larger output for the given neuron, which increases the neuron’s effect on the network. To cause beneficial learning, neurons that lead to greater output can have their weights increased. To increase the speed of learning, at the expense of accuracy, a learning constant can be used. The learning constant is multiplied by the output of each neuron, which amplifies the change. A greater learning constant causes faster learning, as well as a loss of accuracy in the training. Similarly, a smaller constant slows down the learning process, leading to greater accuracy overall.

The next algorithm is dynamic scripting. This is a modification upon existing game AI. Originally, game AI followed specific scripts of actions based on the situation. For example, in a first-person-shooting game, if the AI recognizes an enemy, it executes the associated script with enemy detection (most likely, this script would consist of shooting at the enemy).[13] The

[12] Shiffman, 447.
[13] Pieter Spronck, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma, "Adaptive Game AI

dynamic scripting algorithm generates the scripts of actions based on the AI’s previous actions. Each observation/action pair is stored in a database of rules. There can be multiple possible actions for each observation, so each pair has a certain weight, which is the probability of it being chosen. Similar to neural networks, the weight can be increased if the pair leads to favorable outcomes, or decreased if it leads to unfavorable outcomes. This system, of course, relies on some way to judge performance of the action.[14] In the case of random situations, a “weight-history” mechanism can be implemented. For example, if a specific situation causes an overall bad rule (one that is not as beneficial as another) to be favored, then it will lead to unfavorable outcomes in the future. However, by maintaining a history of the weights for each rule, there is room for adjustment and rolling changes back.[15] This strategy leads to a more effective and versatile AI.

The third major machine learning algorithm is the set of evolutionary algorithms (EA), which is also often referred to as the Learnable Evolution Model (LEM). Similar to neural networks, LEM models a biological process. Instead of the brain, however, LEM is an example of Darwinian evolution.[16] Initially, the algorithm creates a population of possible “solutions” (individuals) to the problem. In this case, the problem (or goal) is playing the game well. After running the AI with the current generation, a fitness function determines which individuals performed well.[17] The weakest individuals are removed, whereas the fittest are evolved. The

with Dynamic Scripting," Machine Learning 63.3 (2006): 218. <http://dx.doi.org/10.1007/s10994-006-6205-6>.
[14] Spronck et al, 221 – 223.
[15] Spronck et al, 230.
[16] Darwinian evolution is the process through which species evolve to increase their chances of survival.
[17] Reed Simpson, "Evolutionary Artificial Intelligence in Video Games," (2012): 2. <https://wiki.umn.edu/pub/UmmCSciSeniorSeminar/Fall2012PapersAndTalks/ReedSimpson.pdf>.

evolution takes place in one of two ways: mutation or crossovers. Mutation occurs by randomly changing a part of the individual. The two fittest individuals are usually evolved through crossover: the two distinct parts of the individuals are combined. This is a usually a more promising evolution than random mutation, but it is also slower because the crossover is computationally expensive.[18] The algorithm "learns" the best solution through the evolution process. If using random mutation, it may be beneficial to keep track of previous generations in case of a harmful mutation.[19] This is similar to how rules in dynamic scripting can become progressively worse due to random error.[20]

The final algorithm is the Markov Decision Process (MDP). This is essentially a method to choose the best out of all possible courses of action for the AI. For very complex problems, the AI chooses out of a subset of all possible decisions because it is infeasible to consider every single decision. Compared to LEM, the MDP algorithm solely requires a fitness evaluation function. The algorithm works by creating a tree of all (or most) possible courses of action. Each node of the tree is followed by all possible situations after taking that action, and so on.[21]

This model is a decentralized algorithm because it assumes that the agent (game AI) is not omniscient. That is, the AI does not know everything about its current situation and must make decisions based on its limited knowledge.[22] Because the AI only has knowledge from its own observations, it is similar to how a human player has limited knowledge, and so it will

[18] Simpson, 3.
[19] Simpson. 3.
[20] Shiffman, 230.
[21] Shlomo Zilberstein and Christopher Amato, "Achieving Goals in Decentralized POMDPs," (Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems, 2009): 593, MIT Computer Science and Artificial Intelligence Laboratory, 27 Sept. 2014 <http://rbr.cs.umass.edu/papers/AZaamas09.pdf>.
[22] Zilberstein et al, 594.

seem more human. A specific course of action is a traversal or path in the tree. To make the best possible decision, the tree has to be complete; the tree must consider all possible scenarios. Even for small games, this becomes a colossal tree. For example, the simple game of Tic-Tac-Toe has a total possible 9! (9 factorial) different moves, which equates to 362,880 different nodes in the tree. For much more complex games, such as Chess, the tree would be of astronomical proportions due to the games' branching factors. Thus, the tree can be dynamically generated (created as needed).[23] This loses accuracy, but is a much more feasible task.

To test the "humanness" of an AI, Alan Turing proposed his famous Turing Test. Essentially, the Test is performed in a question-answer based format. An interrogator asks questions to two subjects. One subject is a human, and the other is an AI. The interrogator does not know that one of the subjects is an AI, and his goal is to determine if both are human. On the other hand, the AI's goal is to appear human. Based on the responses, if the interrogator cannot determine that one or the other is an AI, then he must conclude that both are human.[24] The largest implications of such a conclusion is that the AI has successfully imitated a human. An important aspect of the Test is that the AI being tested cannot be penalized for a physical trait. For example, all questions/answers are given electronically, so that the AI is not at a physical disadvantage. This is because Turing's original goal was to evaluate whether or not the AI has humanlike intelligence, not whether it appears to be human physically.

[23] Zilberstein et al, 595.
[24] Alan M. Turing, "Computing Machinery and Intelligence," *Mind* 59.236 (1950): 434. <http://mind.oxfordjournals.org/content/LIX/236/433.full.pdf>.

For Game AI, the Turing Test is often modified to be more applicable. One major variation is simply watching the AI in action. The *Soar Quakebot* AI was tested by observing its gameplay in a First-Player Shooter (FPS) game. Eight judges watched it play against human characters, and then rated how human it seemed.[25] Similarly, the other modified Test has the AI play against normal human players. Instead of judges rating the AI, however, the players themselves do so. Both tests use a blind procedure, so that the human players/judges do not know they are playing against an AI prior to rating it.

With the *Quakebot* AI, the researchers found several factors to be clear indicators that a player is not human. The goal of this game-playing AI was to seem human, while playing against numerous human players with varied experience in the game. The game-play was judged by eight experts of the game who did not know that they were spectating an AI. The researchers varied four main characters of the bot in their trials: decision time, strategy, number of available tactics, and aiming accuracy.[26] Skill level was measured as a ratio of subject/agent kills/points compared to the number of kills/points that the expert scored. The expert is the player with the most experience in the game. As expected, when decision time was decreased, the skill level increased, in a fairly linear fashion.[27] Similarly, increasing aiming accuracy led to greater skill levels. However, tactics and strategy used did not necessarily affect skill,[28] due to the narrow realm of the study. The judges were asked to give a humanness rating (from a level

[25] John E. Laird and John C. Duchi, "Creating Human-like Synthetic Characters with Multiple Skill Levels: A Case Study Using the Soar Quakebot," *AAAI Spring Symposia* (2001): 55. <http://www.aaai.org/Papers/Symposia/Spring/2001/SS-01-02/SS01-02-012.pdf>.
[26] Laird et al, 54.
[27] Laird et al, 56.
[28] Laird et al, 57.

of 1 to 10) of the AI, and then a single Boolean (true or false) prediction on whether or not the subject was an AI. The most experienced judge was the sole person able to distinguish AI and human players perfectly.[29] As a result, it may not be a very complex task to trick another player into believing a game AI is human. However, the AI should also not be distinguishable from expert players, so the *Quakebot* AI is not perfected. As indicated in Figure 1, the relationship between decision time and humanness level is even more interesting.[30]

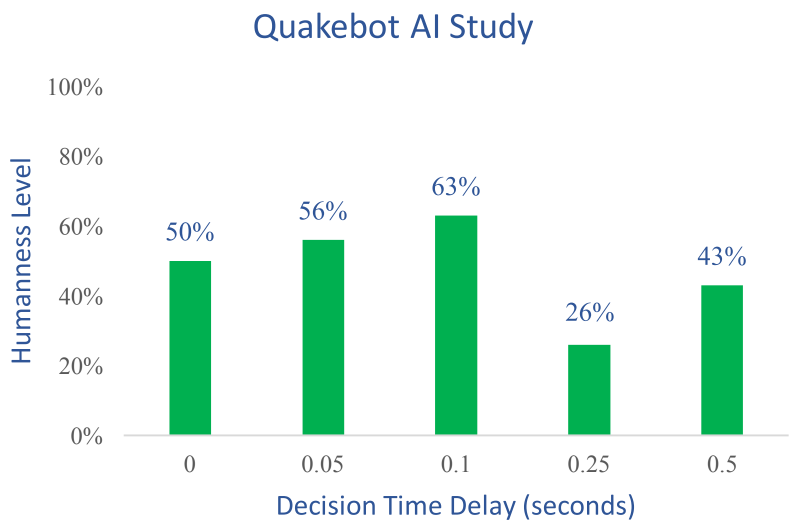

Figure 1: Quakebot AI's decision time versus its judged humanness level.

Even though the AI seemed relatively human in the two fastest times, 0 and 0.05 seconds, it seemed the most human at a delay time of 0.1 seconds. The AI does not seem human at all during the 0.25 and 0.5 delay times because it would make careless errors, such as

[29] Laird et al, 57.
[30] Laird et al, 56.

bumping into walls.[31] Although the strategy did not affect the humanness level significantly, any AI that does not vary its tactics will also seem monotonic and not human.

To overcome the potential failures of seeming human, there are numerous factors a game-playing AI must consider. As seen from the *Quakebot* study, the AI must be able to have a moderate decision time. This will vary from game to game, so it should be able to modify its decision times based on player feedback. In addition, the AI should utilize human-like strategy. One major issue with implementing human strategy is that modern game theory does not predict human behavior well.[32] Whereas most AI algorithms utilize optimal strategy, which maximizes the potential reward/benefits in the long-term, game-playing analysis shows that humans often play with maximal strategy. Instead of focusing on the long-term goal, humans often repeatedly modify their tactics to adjust to minor changes in the situation, in attempt to maximize the short-term rewards. In other terms, we often attempt to maximize short-term gains at the risk of losing, whereas optimal players, which includes most AI, attempt to minimize losing at the risk of not winning.[33] Because most AI algorithms tend toward optimal strategy, they would require several modifications as well as advanced machine learning techniques to adapt maximal strategies.

Another complication arises with utilizing maximal strategy: it is very difficult to predict the outcome of two competing maximal agents,[34] such as a human playing against a human-like

---

[31] Laird et al, 57.
[32] Robert L. West, Christian Lebiere, and Dan J. Bothell, "Cognitive Architectures, Game Playing, and Human Evolution," *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (2006): 103. <http://www.cogsci.rpi.edu/~rsun/book/West-Lebiere-Bothell.pdf>.
[33] West et al, 104.
[34] West et al, 105.

AI. This is due to the fact that maximal strategy often has rapidly changing tactics during the gameplay. As a result, a learning strategy based on analyzing maximal strategies is often very complex and inconsistent in results. In an analysis of humans playing games based on randomness, such as rolling dice or rock-paper-scissors-shoe, West and his colleagues show that the human players attempt to find and exploit short-term patterns, even though these do not exist in random games.[35] As a solution, they suggest that AI could learn to mimic human strategy by analyzing gameplay history; however, this could lead to an equilibrium of maximal strategy. Such an equilibrium leads to both competing agents (game-players) searching for patterns against each other, and this increases further difficulty associated with predicting maximal strategy.[36] So, the best strategy is using a mix of optimal and maximal strategy, especially when playing against a maximal player, such as a human. Similar to humans, the AI cannot be perfect. Instead, it must have a small degree of randomized "human" error in its strategy.[37]

Turing believes that humans are similar to machines.[38] Because of this similarity, he states that it is definitely possible that machines will be able to modify themselves and add to their own complexities.[39] Hence, an AI can seem human by following human strategy, especially through maximal game-playing. In addition to maximal strategy, an AI should utilize entropy[40] to achieve intelligent behavior. Alex Wissner-Gross states that all intelligent actions attempt to

[35] West et al, 109.
[36] West et al, 105 - 108.
[37] Turing, 456.
[38] Turing, 459.
[39] Turing, 457.
[40] Entropy is the potential for random/chaotic behavior.

"maximize future freedom of action." The AI engine *Entropica* is able to learn how to perform common tasks by simply maximizing the entropy in the situation.[41] Therefore, an AI that appears to be intelligent, and thus human, must keep future possibilities available by using this strategy.

To actually compare these algorithms, there are five major features to look into: feasibility, predictability of outcome, ability to adapt to new situations, the potential to seem human, and the time taken/computational resources[42] required. These features are required for an AI to seem human through self-improvement. Feasibility checks how practical the algorithm is to implement for game AI, in terms of complexity and applicability. If the algorithm is too complex to be used or not applicable to game AI, it will most likely be not used. In addition, the outcome should be predictable. Some algorithms do not always provide the same results to the same stimuli; this would render an AI useless because it will not perform the predicted actions. The Machine Learning algorithm should be able to adapt to new scenarios. This goes hand-in-hand with predictability, because the developer[43] should be confident that the AI will be able to handle situations that it has not encountered before. This is especially crucial for game AI, because the developer cannot test every possible situation that the AI will encounter. The entire objective of this form of game AI is to seem human, so the algorithm should be able to accomplish this task. For example, it should have modifiable parameters that can affect its humanness. Finally, the algorithm should be able to run in a relatively quick

[41] Alex Wissner-Gross, "A New Equation for Intelligence," *TED* (2013). <http://www.ted.com/talks/alex_wissner_gross_a_new_equation_for_intelligence>.
[42] In this context, computational resources refers to the processor cycles and random-access memory required.
[43] The developer is the programmer/creator of the AI.

amount of time and not use excessive resources. If the AI is not able to make decisions quickly, it will neither be practical nor seem human. In addition, the algorithm should not require astronomical amounts of memory or processing power. All five aspects are highly interrelated, and so the chosen algorithm should meet all (if not most) constraints to be considered applicable.

Historically, neural networks have been used for very specific purposes. Their design requires very intricate networks for even the simplest of tasks. Most video games are extremely complex, and so the network required for an AI would be colossal. As a result, most game developers do not use neural networks because complexity is a great hindrance to development of AI. The game has to be described in terms of inputs and outputs, both of which are numeric. Then, AI has to make a decision based on this output. Generally, more elaborate networks are harder to manage and create, and so they are not necessarily feasible for game AI, especially for even minimally complex games. While finding a solution with a neural network is generally fast, training the network is very time-consuming,[44] especially for online training.[45] In addition to being time-consuming, the algorithm requires a lot of processing power which is simply not available for most games. For a game AI to seem human, it must be able to adapt to changing scenarios as they occur. If it does not adapt quickly enough, it can be easily distinguished, as seen in the *Quakebot* study.[46]

[44] Darryl Charles and Stephen McGlinchey, "The Past, Present, and Future of Artificial Neural Networks in Digital Games," *International Conference on Computer Games: Artificial Intelligence, Design and Education* (2004): 3. <http://eprints.ulster.ac.uk/8186/>.

[45] Online Training is when the AI adapts as it is being used in the game, as compared to offline-training, which is when the AI is trained prior to being released.

[46] Laird et al, 56.

Another major drawback of Neural Networks in game AI is their unpredictability. For game developers to release an AI, they must be confident that it will perform predictably. However, this is not the case, because Neural Networks are not very predictable when the training is performed online.[47] Despite this, they are very adaptive to new situations. The design of neural networks is to recognize patterns through weight-alteration, and so it has an amazing capability of learning to handle new situations. Unfortunately, due to its inherent lack of predictability, the actions taken by an AI employing neural networks may not be desirable at all.

As with any algorithm, it should have some sort of difficulty-scaling mechanism to match a player's skill level, so that it plays neither extremely well nor extremely poorly. Although this can be implemented by manually modifying the weights of the neural connections, this can lead to unexpected results. The major flaws in neural networks, when applied to game AI, stem from the intrinsic complexity of the algorithm. Because a network for a game has to be extremely complex, it is hard to predict the outcome of given inputs. This causes inhuman behavior and inadequate adaptability.

Although Darryl Charles and Stephen McGlinchey claim that the Neural Network algorithm has potential to be used in game AI, they are only able to provide one example of an online-learning AI.[48] Coupled with the lack of qualifications of the algorithm, neural networks are not suitable to be used in humanlike game AI. Despite their inapplicability to game AI, neural networks have been applied to and are useful for numerous purposes, including

[47] Charles et al, 3.
[48] Charles et al, 2.

image/handwriting recognition, the travelling-salesman problem[49], and stock-market/financial predictions.

Evolutionary Algorithms can find the best course of action for an AI by mutating previous selections. Similar to Neural Networks, LEM is naturally adaptive. LEM was designed with adaptability in mind, because it utilizes randomized starting populations. In addition to this adaptability, LEM provides a thorough analysis of each solution, and so it can create the best possible decision for the AI. However, it shares a similar flaw with neural networks: unpredictability. When the algorithm encounters a new situation, the developer cannot accurately predict how it will respond. The results are more predictable than neural networks, however, because LEM is much simpler. Also due to its simplicity, it is quick to implement and test, so it has some merit compared to neural networks. On the other hand, it is also slow to run. Because it has to analyze each possible solution, it can take immense amounts of memory and processing time. As shown before, the simple game of Tic-Tac-Toe has a total possible 362,880 moves. Analyzing the outcome for every possible move is an extremely long process. For an online real-time strategy game, there are countless possible interactions. It is relatively impossible to check each of these offline, so the learning must occur online as the algorithm encounters new situations, and this is a time-consuming process.

Despite this thorough analysis, error in the algorithm can propagate.[50] For example, if a single bad mutation occurs during the analysis, this malfunction has a significant chance of

[49] This problem is an optimization problem in which a travelling salesman wants to visit numerous cities in the shortest time possible.
[50] Simpson, 3.

affecting future mutations. As a result, the future solutions are not the best possible, even though this is the goal of the algorithm. The algorithm is not often used to mimic human behavior, because it focuses more on optimization; it is not very effective at modelling human game-playing. Although it shows potential for game AI due to its adaptability, it is rarely used for large game AI, especially in complex games. Instead, it is used frequently in optimizations with strict parameters, such as class scheduling, and models of actual biological evolution.

In theory, the Markov Decision Process algorithm produces the best game AI. Because it can consider all possible moves, the algorithm is able to choose the best course of action. In addition, this form of reinforcement learning is amazing against an optimal player.[51] However, creating a tree of the size required by MDPs is infeasible. It would involve massive amounts of computational resources, as well as take a very long time to process. Furthermore, it has been shown that humans are, in fact, not optimal players. Because of this, human course of action cannot be pre-determined, and so this algorithm would not be always suitable for a game AI. Of course, the AI could search through the tree after every move, but this makes the algorithm even slower. In addition, an AI that is able to play the best possible move every single time would not seem very human.

Both Amato and Zilberstein looked into the Decentralized MDP algorithm. Specifically, they worked on Game AI agents that would work toward a common goal. They argue that this algorithm is the best because it is “guaranteed to produce an optimal solution.”[52] In addition, they state that the algorithm can be modified to incorporate infinite bounds. That is, the AI can

[51] Zilberstein et al, 600.
[52] Zilberstein et al, 599.

search for a solution even when the solution cannot be found in a finite number of steps, and still produce an optimal approximation. Along with this, the algorithm operates on local data of the AI.[53] So, each agent can perform the best course of action without knowing the entire state of the game. They further state that this algorithm is the best game AI algorithm because the AI receives an immediate reward after each action. It is able to constantly improve itself, instead of waiting for each agent to progress to the new state.[54] Whereas some indefinite-horizon MDPs[55] require explicitly checking for the goal, Zilberstein and Amato's algorithm always knows "when the goal is reached."[56] One major aspect that they believe is essential to the success of Machine Learning is scalability. Because their algorithm is not restricted to a single problem size, it is very scalable and thus is able to "significantly outperform current state-of-the-art algorithms."[57]

To help show scalability, they provide a data table of their benchmarked algorithms, showing the average time taken by the algorithm in various problems as compared to other algorithms. Although they claim that this algorithm can find the best solution, the nonlinear programming (NLP) model finds a better solution for two of the problems. However, their MDP algorithm finds a very close solution in a *much* shorter amount of time (4 seconds as compared to 117 seconds for the NLP model).[58] This time does not scale linearly with problem size, and so this speed difference is very significant (see Figure 2). Instead, the algorithm scales in a nearly

[53] Zilberstein et al, 593.
[54] Zilberstein et al, 594.
[55] An indefinite-horizon MDP is an algorithmic case in which the ending point is not necessarily known from the beginning.
[56] Zilberstein et al, 597.
[57] Zilberstein et al, 599 - 600.
[58] Zilberstein et al, 600.

quartic fashion, which grows extremely quickly for large data sets. For the Martian Rover problem,[59] which is arguably the closest to a game AI, their goal-directed MDP finds a much better solution than NLP in almost the same amount of time. It found a net reward[60] of 26.9 in 491 seconds when using 5 different rovers. However, in another trial with 6 rovers, it finds a solution with a final reward of 21.4 in 956 seconds.[61] This colossal time difference for a worse solution shows that although the algorithm may sometimes find the best solution, it does not do so in a consistent amount of time. For a real-time Game AI, this is a very significant factor. An AI that varies in decision times can greatly hinder its effectiveness.

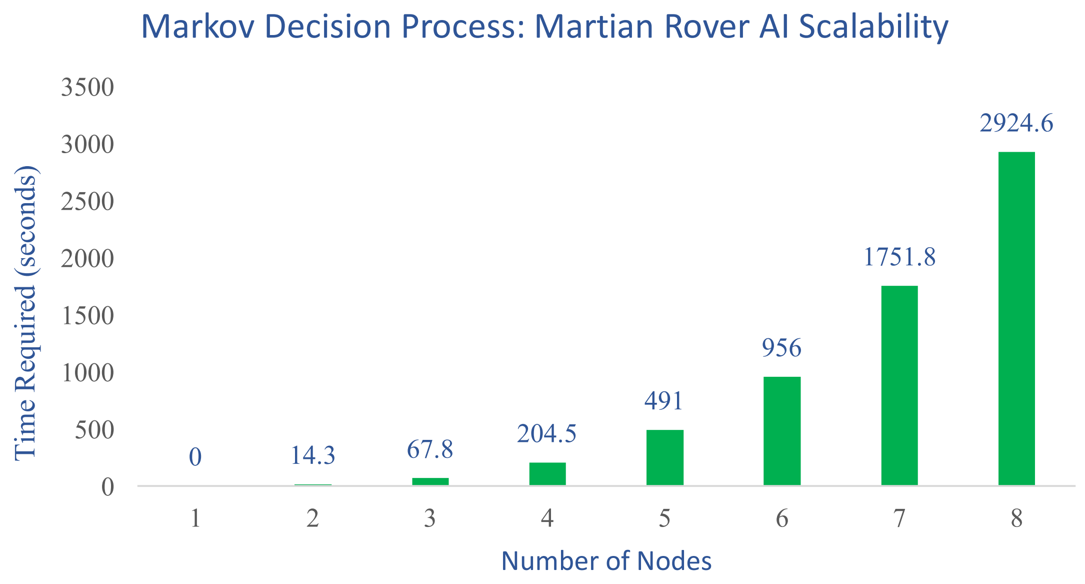

Figure 2: Time required for the Martian Rover Problem using the Markov Decision Process, based on the number of nodes used.[62]

[59] This problem is a theoretical situation in which two (or more) Martian rovers attempt to meet without knowledge of the landscape or their relative locations.
[60] The reward of an MDP algorithm is the total benefit for the AI. A more positive number indicates a greater reward.
[61] Zilberstein et al, 600.
[62] The points for nodes 2, 3, 4, 7, and 8 are projected using power regression. This is possible due to the exponential nature of the problem. These points were not available in the original study.

Despite claiming that the algorithm is consistent and scalable, their data shows that the time taken and results can vary greatly, even with the slightest change in the number of agents. Therefore, their argument is not completely valid. Along with the lack of seeming human, this colossal time requirement results in an algorithm that is not suitable for game AI. However, it is often used in situations where the main agent has incomplete knowledge about the environment, such as the stock-market and financial predictions.

Marc Ponsen, along with Pieter Spronck, Ida Sprinkhuizen-Kuyper, and Eric Postma, focused on the Dynamic Scripting algorithm. Whereas traditional game AI often utilize static scripts,[63] they wanted to generalize the script-generation process. The authors lay out four main features that the AI must consider to be useful in games: speed, effectiveness and consistency, variety, and scalability. Because the algorithm solely requires extracting and modifying rules, it is designed to be very fast. As long as the rules are logical, the algorithm is also consistent and effective. The authors consider, however, that if there are issues in learning, such as randomized error attributed to very specific situations, then the AI may evolve to an inferior state.[64] This error exists in most machine learning algorithms, so the authors contest that this issue is not a problem in the algorithm itself. Because each AI agent would utilize a different "rule base," variety is also inherent in the algorithm design. The AI is scalable "when [it] is enhanced with the top-culling difficulty-scaling mechanism."[65] They conclude that their algorithm is ideal for game AI due to meeting their listed requirements.[66]

[63] A script is a list of commands for the AI to perform, where each command has an associated condition.
[64] Spronck et al, 224.
[65] Spronck et al, 242.
[66] Spronck et al, 246.

Ponsen and his group of researchers provide a more relevant algorithm for Game AI. In fact, they designed it specifically to play various strategy-based games. They also benchmark the algorithm in a strategy game. They compare their dynamic scripted AI to a static AI. In the first trial, they note that various tactics do not have consistent turning points[67]. For seven different tactics, there are fairly high standard deviations (the highest being 64.5 with an average of 51). However, they expected this because the algorithm is not inherently consistent.[68] Compared to the MDP research group, Ponsen acknowledges the drawbacks of their algorithm. However, he also proposes a solution: adding bias to the rule base. When doing this, the standard deviation remains large relative to the average, but the turning points also become a much lower value. The longest time it took for the dynamic AI to start winning, with a biased rule base, was 17 game rounds, with a standard deviation of 8.6 (out of 100 tests).[69] Although Ponsen shows that in general, the algorithm is effective in short amounts of time, he and his research group do not provide concrete data on time benchmarks, unlike Zilberstein. In addition, although they state that their AI met their qualifications and goals, they do not show how they arrived at this conclusion. For the most part, they use inductive reasoning to show the qualities of the AI, without a lot of data.

Whereas Neural Networks, the Learnable Evolution Model, and Markov Decision Processes fall short in one aspect or another, Dynamic Scripting provides the most suitable algorithm for game AI. It has already been used in numerous AI in both dynamic and static

[67] For this game, a turning point is defined as the game round when the dynamic AI starts to win against the static AI.
[68] Zilberstein et al, 233.
[69] Zilberstein et al, 234.

contexts, and so it will be easy to implement in the future. It runs very quickly, due to the algorithm simply modifying weights and looking up rules in a database. Although the solution is not always the best possible, which LEM is able to provide, dynamic scripting is consistently predictable. When modified, it can easily support adaptability. Finally, it is able to create a humanlike AI through precise manipulation of rule weights, as shown with difficulty scaling. Because it meets all of the requirements of a humanlike game-playing AI, the Dynamic Scripting algorithm is the most appropriate choice for such an AI.

Technology is progressing at ever-increasing rates. Game developers follow this pattern by creating games that seem extremely realistic. Whereas in the past, computer programs could easily be distinguished from humans, modern Artificial Intelligence is becoming increasingly human. These AI may lead the dawn of a new era in which we co-exist with AI, unable to tell the two the apart. From our interactions with technology and humanlike AI, we will learn more about ourselves and understand what it truly means to be human.

“Today’s Artificial Intelligence is about new ways of connecting people to computers, people to knowledge, people to the physical world, and people to people.”

– Patrick Winston

Appendix

Algorithm – a method or approach to solving a particular problem

Artificial Intelligence (AI) – self-modifying computer programs that evolves over time

Computational Resources – processing power and computer (random-access) memory required for the algorithm to work properly

Darwinian Evolution – biological process through which species evolve to increase their chances of survival

Deep Learning – machine learning algorithm in which the program evolves by studying successively more-complex data

Developer – programmer/creator of the AI (usually also creator of the game itself)

Machine Learning (ML) – a branch of AI that focuses on programmatic self-improvement

Martian Rover Problem – a theoretical situation in which two (or more) Martian rovers attempt to meet (or find each other) without knowing the other rover's location or any information about the landscape

Maximal Strategy – maximizing short-term rewards, by maximizing wins at the risk of losing

Optimal Strategy – maximizing long-term rewards, by minimizing losses at the risk of not winning

Static AI – "AI" program that does not make use of machine learning, and so it never changes or improves

Turing Test – an experiment to test whether or not an AI seems human

Works Cited

Brin, Sergey, and Lawrence Page. "The Anatomy of a Large-Scale Hypertextual Web Search Engine." *Infolab*. Stanford, n.d. Web. 07 Nov. 2014. <http://infolab.stanford.edu/~backrub/google.html>.

Sergey Brin and Lawrence Page are the original creators of PageRank, which is the main algorithmic infrastructure of Google's search engine. In addition, they co-founded Google. As a result, they are very knowledgeable about the how Google worked at the time, which is what the article describes. From this article, I found out more about how Google's search algorithm works.

Charles, Darryl, and Stephen McGlinchey. "The Past, Present, and Future of Artificial Neural Networks in Digital Games." *International Conference on Computer Games: Artificial Intelligence, Design and Education* (2004): n. pag. *Ulster Institutional Repository*. University of Ulster, 1 Feb. 2010. Web. 20 Nov. 2014. <http://eprints.ulster.ac.uk/8186/>.

Dr. Darryl Charles has worked with Artificial Intelligence (and specifically, game AI) for most of his career. He has published numerous articles in journals on his work and research at the University of Ulster in the United Kingdom. Similarly, Dr. Stephen McGlinchey studied Computer Science at the University of Paisley, which is where he received his doctorate in Neural Networks. Because of the length of their careers in AI and neural networks, both Charles and McGlinchey are qualified researchers in this field. Their article focuses on how neural networks are not popular for game developers. In addition, they state the two main reasons on why networks are not popular: unpredictability and the algorithm's slowness. They believe networks may be useful for game AI to scale the difficulty of the game and modify non-player characters to seem more realistic, which provide an enhanced experience for the player. However, they argue that the computational expense associated with networks results in their unpopularity.

From this article, I was able to understand why networks are not used in game AI. Even though they may be useful in terms of effectiveness, the speed and unpredictability render them inapplicable. In addition, I learned about the drawbacks of neural networks, which will greatly help me in my analysis. For the most part, this article presents arguments from different sources in various Machine Learning journals. By mixing these arguments, the authors come up with their own conclusion.

Dick, Philip K. *Do Androids Dream of Electric Sheep?* New York: Ballantine, 1996. Print.

This novel is solely used for a short anecdote on human-like AI and intelligent robots, and how the characters interact with them.

IBM. "Deep Blue." *IBM 100*. IBM, n.d. Web. 07 Nov. 2014. <http://www-03.ibm.com/ibm/history/ibm100/us/en/icons/deepblue/>.

IBM is a leading software development company and is the creator of Deep Blue, the Chess AI which played against Garry Kasparov. This article describes the general algorithm of Deep Blue; I used it for the figure of Deep Blue's search depth.

IBM. "IBM Watson: The Science behind an Answer." *YouTube*. IBM, 18 July 2011. Web. 07 Nov. 2014. <http://www.youtube.com/watch?v=DywO4zksfXw>.

Similar to the Deep Blue article, this is also produced by IBM, which is the company that originally developed the Watson software and infrastructure. This video describes how Watson searches for an answer to a *Jeopardy!* question. In addition, it includes details on how Watson is able to improve over time. I was able to use this information to show how a famous AI uses machine learning and self-improvement.

Khoo, Aaron, and Robert Zubek. "Applying Inexpensive AI Techniques to Computer Games." *IEEE Intelligent Systems* 17.4 (2002): 48-53. Northwestern University. IEEE, July-Aug. 2002. Web. 25 Oct. 2014. <http://www.cs.northwestern.edu/~khoo/downloads/papers/IEEEIntelSystems2002-InexpensiveAI.pdf>.

Both Aaron Khoo and Robert Zubek are PhD candidates at Northwestern University. They have researched Artificial Intelligence, as related to robotics and games, for most of their careers, and so they are well-versed in this field. In this article, they discuss two different AI models they created. The first was a reactive AI, which responded to its observations

(essentially a decision tree). Their goal with this first AI was to respond to an ever-changing environment in which there are other agents (players) who affect the situation. They also wanted to minimize the computational expense of the AI. This was perceived as human by numerous opponents, including experts of the game. In addition, their second model was also seen as human. However, the primary goal of this model was to respond to in-game chat in a human way. This was not completely successful, because being able to understand human chat (which is not always structured) is a very difficult task for computers. Regardless, it did show success in most cases. Although this article is slightly over a decade old, the algorithms that the AI used have not changed in recent times. Their analyses remain valid due to the static nature of these algorithms. From this article, I obtained an in-depth description and analysis of two applied Game AI. Both of the models played in two different games, and so I was able to understand what factors would make an AI seem human (or not). The general methodology of the authors is two-fold: first, they describe how their AI works (briefly). Then, they analyze the AI's performance in various games and how players perceived it.

Laird, John E., and John C. Duchi. "Creating Human-like Synthetic Characters with Multiple Skill Levels: A Case Study Using the Soar Quakebot." *AAAI Spring Symposia* (2001): 54-58. AAAI. Association for Advancement of Artificial Intelligence, 2001. Web. 28 Sept. 2014. <http://www.aaai.org/Papers/Symposia/Spring/2001/SS-01-02/SS01-02-012.pdf>.

Both John Laird and John Duchi have worked with artificial intelligence and machine learning for significant portions of their careers. Duchi teaches at Stanford University as an Assistant Professor of Electrical Engineering and Computer Science, and Laird is a professor of Computer Science at the University of Michigan. Duchi received his doctorate in statistical learning, and so they are both well versed in the respective topics. This case study focuses on a Game AI playing a specific Real-Time-Strategy (RTS) game called Quake. The study uses the Game AI (with limited information about its environment) and varies its parameters (decision time, skill, etc.) while using it to play against an expert in the game. Human players of varying skill levels, as well as a "perfect" bot (in terms of both skill and "humanness") are used as reference points in their matches against the expert. Finally, numerous judges review each player's performance, and rates their humanness level (this is a modified Turing Test). The players include both the AI and the normal players. Overall, the study found that decision time and skill level had the greatest impact on humanness. The study is not very recent, but the data presented still holds true. Whereas computers have become faster with time as according to Moore's Law, the algorithms themselves have not changed and so their data is still valid.

This study aided my understanding of how the Turing Test could be applied to a Game AI. In addition, it provided data on how different factors can affect the "humanness" rating of an AI. The AI they utilized was not learning or self-improving, so I will have to research how these factors could be automatically improved. Because this is a case study, the information is provided in a very straightforward manner: how the study was performed, data collected, and a conclusion. Their analysis of the data was very interesting; they found that some factors (such as tactics used) did not have a major impact on the humanness rating. On the other hand, factors such as decision time (whether too fast or too slow) and accuracy played some of the largest roles.

Le, Quoc V., Marc'aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. "Building High-level Features Using Large Scale Unsupervised Learning." (2012): n. pag. *Google Research*. Google, 2012. Web. 7 Nov. 2014. <http://research.google.com/archive/unsupervised_icml2012.pdf>.

These six authors have worked with Google for numerous years. In addition, they were some of the researchers who helped integrate machine learning into Google's search engine and its features. So, their descriptions of how Google uses unsupervised-learning is definitely valid. Based on the information provided in the article, I was able to describe how Google's image-recognition software is able to improve upon itself overtime.

Michalski, Ryszard S. "Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning." Ed. Floriana Esposito and Lorenza Saitta. *Machine Learning* 38.1-2 (2000): 9-40. *GMU Machine Learning and Inference Library*. Kluwer Academic Publishers, 14 Feb. 2004. Web. 18 Aug. 2014. <http://www.mli.gmu.edu/papers/96-2000/00-2.pdf>.

Ryszard Michalski has years of experience in Machine Learning. He has over three hundred contributions to the Machine Learning and Inference Library (MLI) publications at George Mason University since 1966. In addition, he has pursued the Learnable Evolution Model (LEM) since 1999. Due to his years of work in the field, he is a qualified expert. This article provided the algorithm to use Machine Learning to model biological evolution. In essence, it follows a form of unsupervised learning but with numerous modifications. The article is slightly old, and so the computational expense associated with LEM is not as significant anymore as Michalski states. However, this does not affect how LEM is used in modern applications because the algorithm itself has not evolved, and so the article is not outdated.

By itself, this topic is very interesting. However, it is not completely relevant to my topic. Instead, I was able to learn how LEM works, and I think this can be applied to a broader sense of Machine Learning: using an evolution-based algorithm to cause self-improvement in a program. In addition, LEM can be used to find the best course of action for a game AI. Most of this model is original research, and Michalski cites his own previous works frequently. However, this can be expected because his evolution model is unique.

Nilsson, Nils J. *Introduction to Machine Learning*. Stanford: Stanford U, 2005. Stanford University, 19 June 2010. Web. 17 Aug. 2014. <http://robotics.stanford.edu/~nilsson/MLBOOK.pdf>.

Dr. Nilsson is a reputable figure in the field of Artificial Intelligence. He has worked on it for the better part of two decades and has numerous publications (both books and articles) in the field. This book is a draft of a potential textbook, written during his time at Stanford University. In addition to this book draft, Nilsson has written numerous other articles (focused on AI) while he served as a professor of Engineering and Computer Science at Stanford. The book draft contains very detailed descriptions on Machine Learning and its associated algorithms. He goes into detail in multiple popular algorithms (including Neural Networks, Decision Trees, and both Supervised and Unsupervised Learning). Nilsson also provided examples for each algorithm. One aspect that I found extremely interesting was his analysis of statistical noise. Because most of these algorithms rely on pattern-recognition in colossal sets of data, they must be able to account for imperfections in the data. He proposes using numerical bias to help find the best solution. I had not considered this prior to reading this source, so it was very insightful. This book draft was written in 2005, but his descriptions of how the Machine Learning algorithms work remains valid. The only significant change in the algorithms since 2005 is their unique uses, which are constantly evolving.

The book draft greatly added to my understanding of Artificial Intelligence and Machine Learning. Most of the algorithms that he described are very complex, but I was able to grasp them better after reading them. In addition, by reading about these learning methods, I was able to create and further the idea for my own algorithm. Although Nilsson himself is an expert in this field, he also cites many articles and works to support his descriptions of the algorithms. He also provides mathematical analysis of the algorithms to further show how they work.

Ponsen, Marc, and Pieter Spronck. "Improving Adaptive Game AI with Evolutionary Learning." (2004): 389-96. *ILK Research Group*. Computer Games: Artificial Intelligence, Design and Education, 2004. Web. 26 Oct. 2014. <http://ilk.uvt.nl/~pspronck/pubs/PonsenCGAIDE.pdf>.

Pieter Spronck has taught courses on Data Processing, Game Development, and Artificial Intelligence at Tilburg University (in the Netherlands). Marc Ponsen is a PhD student, focusing on Machine Learning, at Maasthrict University, Netherlands. One of the key sections of this article is an extension on the standard dynamic scripting. Whereas dynamic scripting requires a rule base that does not change, integrating evolutionary algorithms allows the AI to generate new rules. To do so, the state of the game is encoded into a chromosome. This is essentially a list of the current state, used rules, and parameters (for each rule) positioned a certain way. New generations are formed through one of four mutation methods, all of which check for gene similarity. In this case, a gene in the chromosome is a single element in the list. After testing the generic dynamic scripting AI (against a static AI), they incorporated the evolution into it and retested it. This led to increased success (which is measured by victories over the opponent), and so they concluded that the evolutionary algorithm is beneficial. From this article, I learned how the evolutionary process works (which they had proposed in a previous article). In addition, I was able to rank the machine learning algorithms better, because the Dynamic Scripting algorithm can also consider adaptability, due to the evolution. I also found a potential function to evaluate the fitness of a given generation in the evolution process. The article starts with describing their Dynamic Scripting algorithm for the specific game, Wargus, and how it differs from classic static scripting. I already had a general idea about this from their previous article, so this was not too useful. However, they went on to discuss the evolutionary aspects, as well as provide their evaluation of the results of the two AI, in a self-designed experiment. Because how the machine learning algorithm works has not changed, the article's age is not relevant to its validity.

Samuel, Arthur. "Some Studies in Machine Learning Using the Game of Checkers. II-Recent Progress." *IBM Research*. IBM, 5 June 1967. Web. 7 Nov. 2014. <http://researcher.watson.ibm.com/researcher/files/us-beygel/samuel-checkers.pdf>.

Arthur Samuel describes his research, while at IBM, on his checkers AI algorithm. He originally studied at MIT and then went on to work at IBM, and so his work is credible. Although the article is fairly old, I only use it to describe how game AI have been used, especially in older times.

Shiffman, Daniel. "Neural Networks." The Nature of Code. Ed. Shannon Fry. 2012. 444-78. *The Nature of Code*. Web. 14 Sept. 2014. <http://natureofcode.com/book/chapter-10-neural-networks/>.

Mr. Shiffman has worked with open-source projects for quite a while, and has written two books. He is an Associate Arts Professor, focusing on computational media and programming, at New York University. *The Nature of Code* is an introductory programming book, but it includes detailed descriptions and examples of Neural Networks. This chapter on Neural Networks provided a lot of insight on how they work and what they are useful for. They are designed to model neurons and their connections in the human brain, and are able to evolve based on given stimuli. Shiffman focuses mainly on supervised reinforcement learning by adjusting weights, which is a simple (but effective) way of training the Neural Networks. In addition, he describes how the Networks can evolve by modifying the weights of neural connections. After reading this chapter, I believe that a Neural Network may be the most suitable algorithm for a learning AI. Because Shiffman focused mainly on the perceptron (single neurons), I will have to research more into larger-scale networks and back-propagation (the learning algorithm for multi-neuron networks). Shiffman cites a multitude of books and articles that support his algorithm description.

Simpson, Reed. "Evolutionary Artificial Intelligence in Video Games." (2012): n. pag. 2012. Web. 14 Oct. 2014. <https://wiki.umn.edu/pub/UmmCSciSeniorSeminar/Fall2012PapersAndTalks/ReedSimpson.pdf>.

Reed Simpson has studied Artificial Intelligence and Computer Science, and has his degree in Computer Science from the University of Minnesota. This article is his senior research project while studying at the University. It starts off with his explanation of why adaptive Game AI are important: they provide a more immersive and realistic user experience. In addition, Evolutionary Algorithms (EA) provide better adaptive AI than traditional Game AI (which usually are not adaptive and thus fail in unknown situations). EA works by starting with a population of individuals, where each individual is a possible solution to the problem. In the case of a Game AI, the “problem” is playing the game well while also seeming to be another human player. Each individual solution is evolved, through a process of mutation and crossovers. Because the EA model closely follows biological evolution, the AI’s mutation and crossover processes are similar to their respective biological processes. To determine which individuals survive, a fitness function is required.

One interesting aspect is that the fitness function greatly determines the time required for the model to evolve. If the function is not adequate, the evolution can take a very long time. This is one of the major issues with EA: time required and computational expense. For fairly complex games, the EA model takes significant computational resources and time to provide a finished product. In addition, it is hard to predict how the AI will respond to situations, so the EA model may not be suitable for very complex games. From this article, I learned about how this model can be applied to Game AI. Because the author provides examples of its use (in three different games), I know that it is definitely plausible. Although this algorithm can generally find the “best” solution, it can take a very long time to complete. I will definitely be able to use this algorithm in some (probably simpler) form. Simpson structures his article in three sections: why adaptive AI is important, how EA works, and a small study of its use in three different games. The application to the games is mostly original research, whereas the description of the algorithm is a compilation of the works of numerous other experts in the AI field.

Spronck, Pieter, Marc Ponsen, Ida Sprinkhuizen-Kuyper, and Eric Postma. "Adaptive Game AI with Dynamic Scripting." *Machine Learning* 63.3 (2006): 217-48. *Springer*. Kluwer Academic Publishers. Web. 7 Oct. 2014. <http://dx.doi.org/10.1007/s10994-006-6205-6>.

Pieter Spronck has taught courses on Data Processing, Game Development, and Artificial Intelligence at Tilburg University (in the Netherlands). Marc Ponsen is a PhD student, focusing on Machine Learning, at Maasthrict University, Netherlands. Ida Sprinkhuizen-Kuyper currently teaches courses in Artificial Intelligence and Evolutionary Algorithms in Radboud University, Nijmegen (Netherlands). Eric Postma is an Artificial Intelligence Professor at Tilburg University. In addition, the article is from the *Machine Learning* Journal, which is very reputable in the field of AI. This article focuses on a special algorithmic approach to Game AI: dynamic scripting. Dynamic scripting is the process of creating a list of actions based on a set of rules. The actual learning algorithm comes from modifying the rule-base (set of rules). Each rule has a certain weight, which is the probability that it will be used. Machine learning occurs by modifying the weights based on observations. The rule-base can also be dynamically generated by the program through the use of evolutionary algorithms. For the game to actually be entertaining for users, the AI must seem human (it cannot play too well). To do this, the weights of “good” rules can be intentionally kept low. From this article, I learned about a new Machine Learning algorithm, which is especially tailored for Game AI. Although this algorithm is similar to Neural Networks, it is much simpler and so I believe it will be easier to use. The authors present the information in two sections. The first focuses on the theoretical aspects of the algorithm, which is what I mainly used. The second is an original case study of a Game AI (which utilizes dynamic scripting). Despite the fact that the article was published almost a decade ago, their research into Dynamic Scripting is valid because the algorithm itself has not changed in that time.

Turing, Alan M. "Computing Machinery and Intelligence." *Mind* 59.236 (1950): 433-60. Oxford Journals. Oxford University Press, 25 Aug. 2008. Web. 25 Sept. 2014. <http://mind.oxfordjournals.org/content/LIX/236/433.full.pdf>.

Alan Turing is often considered the “father of Artificial Intelligence.” He is one of the first (of many after him) who has looked into Artificial Intelligence (AI) and Machine Learning topics. His proposed models are long regarded as the foundations of Artificial Intelligence. He was appointed the position of Reader (senior researcher) at the University of Manchester due to his work in Computer Science (Artificial Intelligence and Cryptography) and Mathematics. That being said, some of his ideas are outdated due to much greater increases in technology. However, his descriptions of how computers should evolve have laid the groundwork for modern researchers. In this article, Turing proposes the famous "Turing Test", which essentially concludes that if a computer cannot be distinguished from a human being, then it must be considered equal to the human being. This conclusion is made by asking both the computer and human a series of questions (without explicitly knowing who is answering in any physical way) and solely basing the judgment off of the answers.

In addition, Turing explicitly defines what a machine is, and what AI consists of. This is very important to my project because the term AI can have many different meanings, depending on the user. Turing defines it as a machine that is capable to learn, and this is the definition that is most applicable to my project. In addition, Turing describes a generic learning algorithm, which is now known as supervised reinforcement learning. He believes a machine can improve by interpreting the consequences of its actions (as either positive or negative results). He also suggests that a learning AI should be created with a child in mind --- it should have the ability to learn, but not much prior knowledge. Turing's way of presenting his ideas is very unique. He first describes the general Turing Test, and then lists counter-arguments against the possibility of a "thinking machine". After each counter-argument, he describes a solution or his view on the matter, and this was very insightful on how common problems could be overcome. He does not cite many works that are strictly about AI. This is most likely due to the lack of AI research at the time.

West, Robert L., Christian Lebiere, and Dan J. Bothell. "Cognitive Architectures, Game Playing, and Human Evolution." *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation* (2006): 103-23. *Rensselaer Cognitive Science Department*. Cambridge University Press, May 2008. Web. 12 Oct. 2014. <http://www.cogsci.rpi.edu/~rsun/book/West-Lebiere-Bothell.pdf>.

Robert West is an Assistant Professor of Psychology (specifically, Computer Applications of Psychology) at Carleton University (Canada). Christian Lebiere and Dan Bothell are both Psychology Professors at Carnegie Mellon University. All three have a significant number of research projects and articles in the field of Cognitive Science. The authors write about game theory and how humans make decisions. One major conclusion is that humans do not play games optimally (win most of the time in the long term). Instead, we tend to play maximal strategies. That is, humans look for patterns, even in random games where the opposing agent's actions cannot be determined, and try to win in the short term. To model human strategy, therefore, game AI must use maximal tactics. However, this would require the AI to predict outcomes when playing against a human. Because the human is also using maximal strategy, the AI faces the difficult task of determining whether one maximal plan is better than another. In addition, game AI are very good at implementing optimal tactics, so this presents another obstacle. From this article, I gained a greater understanding of how we, as humans, play games. By learning this, I can attempt to model an AI's strategy based on this maximal model. The authors present their conclusion in a structured format. They first list their hypotheses, then show results of their "maximal" AI competing in a tournament. Based on this, they show their conclusions. Their conclusions do not differ much from their hypotheses, because the data is a very specific case and so it does not model all of the possible maximal strategies. This study was performed nearly seven years ago. Despite this, their analysis of how humans play games (and human nature) still holds true today.

Wissner-Gross, Alex. "A New Equation for Intelligence." Nov. 2013. *TED*. Web. 27 Sept. 2014.

<http://www.ted.com/talks/alex_wissner_gross_a_new_equation_for_intelligence>.

Dr. Alex Wissner-Gross has worked with AI and machine learning for a significant portion of his life. He is an Institute Fellow at Harvard University as well as a Research Affiliate at the Media Laboratory of the Massachusetts Institute of Technology. In addition, his doctoral thesis at Harvard focused on machine learning, so he is very experienced in the field. In this TEDx Talk, he focuses on intelligence, both for humans and computers. He argues that intelligence can be associated with "maximizing future freedom of action". This essentially means that intelligent actions attempt to create the greatest number of future possibilities. In this fashion, intelligent beings do not trap themselves into one specific path.

I believe this is a very specific definition of intelligence that I will be able to incorporate into Game AIs. In addition, artificial intelligence should model human intelligence, so this provided a different way for a Game AI to mimic human players. Wissner-Gross presents his argument and equation by basing it off of previous research. In addition, he shows an AI Engine called Entropica which displays intelligent behavior through this process of maximizing possibilities. I have not heard much about the engine but I will definitely look into it to see how it can show human-like intelligence.

Zilberstein, Shlomo and Christopher Amato. "Achieving Goals in Decentralized POMDPs." *Proceedings of the Eighth International Conference on Autonomous Agents and Multiagent Systems*. Budapest, Hungary, 2009. 593-600. MIT Computer Science and Artificial Intelligence Laboratory. AAMAS, 2009. Web. 27 Sept. 2014. <http://rbr.cs.umass.edu/papers/AZaamas09.pdf>.

Shlomo Zilberstein and especially Christopher Amato have both dealt with reinforcement based learning algorithms for more than a decade. Zilberstein is a Professor of Computer Science at the University of Massachusetts Amhert, and Amato is a Research Scientist at the Massachusetts Institute of Technology. Amato has numerous papers on how Markov Decision Processes (MDPs) work. This paper focuses on an algorithm of using decentralized MDPs to determine the best course of action for an agent (entity). In a decentralized MDP, there can be numerous agents, with each keeping separate observations. Essentially, the MDP is a way to utilize local observations (which may not be completely accurate), the current state of the problem/game, a list of possible actions, and a list of possible rewards to calculate the best form of action. By using the MDP in a learning process, a policy can be formed. This policy is a course of action when the agent encounters the same state again. Amato also considers indeterminate data, for when the agent does not know everything about the problem or game. MDP is a very human-like way of thinking about a problem, because it considers consequences of actions, even with limited knowledge. Although MDP is generally slow, its execution has become faster since the article was written due to modern computers being much faster. However, the data of execution times that they present are still useful when considered relative to each other.

The paper was fairly complex but with further research, I will be able to utilize the MDP so that a Game AI can play well but also seem human. Amato and Zilberstein both cite some of their previous works, and most of their references are from Artificial Intelligence conferences. Because a lot of the work is theoretical (the paper itself does not deal with a lot of data other than performance times), there is very little room for error. The general format of the paper follows a constructive trend. Each new section builds upon the last, adding a bit more complexity. In this way, I was able to understand the algorithm while not getting lost in the general complex nature.