TODO

a thesis presented
by
Tom Silver.
to
The Department of TODO?

in partial fulfillment of the requirements
for the degree of
Artium Baccalaureus (A.B.)
in the subject of
Computer Science and Mathematics

Harvard University
Cambridge, Massachusetts
May 2016

Thesis advisor: Professor Stuart M. Shieber Tom Silver.

TODO

Abstract

TODO

Contents

1 The Luna Rating System 1

1.1 Evaluating the Intelligence of Machines 1

1.1.1 Introduction 1

1.1.2 Principles of a Practical Test for Intelligence 2

1.2 Existing Contests for Artificial Intelligence 4

1.2.1 The Turing Test 4

1.2.2 Robotics Contests 5

1.2.3 Other AI Competitions 7

1.3 The Luna Rating System 7

1.3.1 Intelligence as a Social Construct 7

1.3.2 The Luna Rating System 8

1.3.3 Analysis of Principles 9

1.3.4 Thesis Outline 10

2 The Luna Game 11

2.1 Overview 11

2.2 Interview Phase 13

2.2.1 Instructions 14

2.2.2 Interview Strategy 14

2.2.3 Examples 15

2.3 Response Phase 16

2.3.1 Instructions 16

2.3.2 Examples 17

2.4 Guess Phase 17

2.4.1 Instructions 17

2.5 Game Conclusion 18

2.5.1 Example . . . 18
2.5.2 Conclusion of a Player's First Luna Game . . . 19

3 Robustness of the Luna Rating System 20
3.1 The Smarts Rating as a Proxy for Intelligence . . . 20
3.1.1 Demonstrated Intelligence . . . 21
3.1.2 Reported and Actual Guesses . . . 22
3.1.3 Definitions and Notation . . . 22
3.2 Theoretical Effects of Strategies on Smarts Rating Validity . . . 23
3.2.1 Honest Play . . . 24
3.2.2 Single Priority Play . . . 24
3.2.3 Response Agnostic Play . . . 27
3.3 Simulations . . . 31
3.3.1 Methods . . . 31
3.3.2 Results . . . 32
3.3.3 Combined Strategies . . . 35
3.4 Implications for LRS Design . . . 36

4 Interview Question Generation 41
4.1 Introduction . . . 41
4.2 Related Work . . . 43
4.3 Question Generation as Binary Classification . . . 45
4.4 Experiments . . . 45
4.5 Results . . . 45
4.6 Discussion . . . 45

5 Learning to Respond 46
5.1 Introduction . . . 46
5.1.1 The Response Problem . . . 46
5.1.2 Related Work . . . 48
5.2 Methods . . . 51
5.2.1 Approaching the Response Problem . . . 51
5.2.2 Example Solution . . . 51

5.2.3 Datasets . . . 52
5.3 Results . . . 52
5.4 Discussion . . . 52

6 Luna Rating Prediction 53
6.1 Introduction . . . 53
6.1.1 The Luna Rating Prediction Problem . . . 53
6.1.2 Related Work . . . 55
6.2 Methods . . . 56
6.2.1 Datasets . . . 56
6.2.2 Features . . . 58
6.2.3 Regression . . . 62
6.3 Results . . . 63
6.4 Discussion . . . 65

7 A Web Implementation of the Luna Rating System 67
7.1 Introduction . . . 67
7.2 Building the Web Interface . . . 68
7.2.1 Design of the Web Interface . . . 68
7.2.2 An API for Machine Players . . . 70
7.2.3 Launching the Website . . . 73
7.3 Results . . . 73
7.4 Analysis . . . 73
7.5 Discussion . . . 73

References 74

TODO

Acknowledgments

People are amazing, I will acknowledge them here.

Don't get fooled by people who claim to have a solution to Artificial General Intelligence... Ask them what error rate they get on MNIST or ImageNet.

Yann LeCun

1

The Luna Rating System

1.1 Evaluating the Intelligence of Machines

1.1.1 Introduction

Research in artificial intelligence is driven by standardized tests and competitions. The level of success of an algorithm on a widely used test can determine the amount of funding and attention from academia that it receives. With such a burden placed on these tests, one would hope that they provide an accurate reflection of the extent to which an algorithm has achieved artificial intelligence. Unfortunately, existing tests fall far short of this mark[?]. At best, current contests evaluate the performance of specialized algorithms on restricted

problem domains that are not guaranteed to generalize, such as object recognition[??] or robot soccer[??]. At worst, tests prematurely claim to be the ultimate arbiter of general intelligence[?], inspiring a distracting and unwarranted media frenzy when they are passed[??]. In all cases, the current benchmarks for evaluating AI machines are leading the research community away from the creation of truly general intelligence.

In this thesis, I put forth a new system for evaluating general AI. The system invites humans and machines to participate in two-player games in which each player assesses the intelligence of her opponent. The aggregation of these assessments is used to assign a rating to the player. A player's rating is a reflection of the player's demonstrated intelligence. In this sense, the ratings that emerge from the system are the results of a never-ending test of each of the player's general AI. The definition of intelligence is not presupposed in this system; instead, it emerges from the collective judgement of all players. Thus the system is a microcosm of the natural process that humans use to evaluate each other's intelligences. I will argue that the system offers a bright light on the dim path towards machine intelligence.

1.1.2 Principles of a Practical Test for Intelligence

The system proposed in this thesis is meant to steer AI research towards the creation of general intelligence. Therefore, the system must actually be used; a thought experiment will not suffice. Here I present five principles that must be obliged if the test for intelligence is to fulfill its stated purpose.

Accessibility

To serve as a useful guide, the test for AI must be constantly accessible to researchers. Ideally the test should be efficient enough that it may be used several times throughout the

course of an AI developer's day. This principle discourages a centralized competition that is only held at regular intervals, and instead favors a rating system that can be accessed through the Internet and then carried out using the resources of an average computer.

Generalizability

A test need not span all possible areas of intelligence, but the results should reflect the subject's ability to perform in all areas. In this sense, the test should be AI-complete — a machine that does well on this test should do similarly well on any other reasonable test of intelligence. The proposition of the Turing Test, which continues to be held by many researchers, is that the problem domain of natural language is AI-complete. The system proposed here shares this premise.

Continuity

Every candidate for AI should be able to observe changes in performance over the course of development. A test that only reports a binary outcome — pass or fail — will not be useful for researchers who are not yet close to passing. The test's outcome should instead be continuous, indicating clearly when progress is being made.

Dependent on People, Independent of Persons

A fair test should not rely on any one person's interpretation of intelligence. At the same time, intelligence is a social construct that cannot be reasonably appraised without input from humans. Therefore a test for intelligence must take into account the popular conception of intelligence, but it cannot rely on any single human judge to determine its outcome.

Immunity to Gaming

It should not be possible for a researcher to "game the system" and achieve outsized results by exploiting structure in the test. This principle precludes any sort of hard-coding of knowledge that is a priori known to be important for the test. For example, a standardized test fails the Immunity to Gaming principle, since any researcher who observes the results of the test once would be able to submit a machine with the memorized knowledge necessary to pass the test.

1.2 Existing Contests for Artificial Intelligence

The history of testing AI is long but sparse. It begins in 1950 with the introduction of the Turing Test and meanders into the present day with specialized tests like ImageNet and Robocup. Only recently has there been interest in designing new tests that are more appropriate for practically evaluating general AI[?]. This interest often manifests as an appeal to move "beyond the Turing Test"[?]. However, as I argue below, the Turing Test has never been a practical test for general AI, nor was it ever meant to be. Thus the recent wave of interest is really a call for the first ever practical test for general AI.

1.2.1 The Turing Test

The first formal test for machine intelligence was articulated by Alan Turing, the father of computer science and one of the progenitors of AI[?]. Six decades later, the eponymous Turing Test remains at the center of the dialogue surrounding machine intelligence. The Test requires three rooms, each equipped with a telegraph. In one room, the machine candidate for AI is connected to the telegraph, ready to receive and transmit messages. The second

room, which is disconnected from the first room, houses a human confederate. A human judge resides in the third room with telegraph connections to the first and second room. The duty of the judge is to determine which room contains the human. Different instantiations of the Test vary details beyond this framework, e.g. how long the conversations go on before a judgment is made[?] . In all cases, the machine is deemed intelligent if it is able to trick the human judge into guessing that it is human.

In applying the principles of a test for intelligence outlined in 1.2, the limitations of the Turing Test come to the fore. The Test is not accessible in practical terms, since it relies on two humans, a judge and a confederate, per machine per test. The Test is generalizable insofar as natural language is AI-complete. The Test is not continuous; it outputs a "pass" or "fail" depending on whether the participant is able to fool the judge. The Test is very dependent on persons, in particular the human judge and confederate, and does not incorporate any popular conception of intelligence. Finally, the Test is immune to gaming, since it is not possible to anticipate the behavior of the human judge. Thus the Turing Test satisfies only two of the five prescribed principles for a practical test of intelligence. This analysis is summarized in Table 1. The Test has withstood the test of time for good reason. Its main insight — that intelligence cannot be evaluated without an intelligent evaluator — must be realized by any test designer. Nonetheless, as several have argued[???] , the Turing Test should be viewed as a thought experiment rather than a practical test for intelligence.

1.2.2 Robotics Contests

Robotics seems like a natural domain for testing artificial intelligence. The prospect of robots that are able to behave and reason at human levels is a clear motivation for AI research.

Moreover, a candidate for AI is likely to be more convincing if it is physically instantiated. Anderson, Baltes, and Cheng (2011) review existing robotics contests and critique their utilities as benchmarks for AI research. These contests include annual competitions like AAAI/IJCAI, which consists of a diverse suite of tasks for the robots to perform[?] ; RoboCup, which requires candidates to play an actual game of soccer, thus entering the realm of multi-agent strategizing[?] ; and HuroCup, which might be considered an "olympics for robots", requiring contestants to compete in a wide range of sports and agility competitions[?] . Anderson, Baltes, and Cheng find specific shortcomings in each of these competitions as proxies for AI, but suggest that more broad and versatile robotics competitions could still be useful for testing AI.

I argue that a test for AI should be independent of robotics. One inherent problem with robotics as a medium to test AI is that there are a host of extremely challenging problems in the field that have little or nothing to do with intelligence (e.g. the difficult mechanical problems associated with walking[?]). It may be that certain problems in robotics are sufficient to demonstrate AI, but those problems are often harder than AI itself. Another fundamental problem in robotics competitions that currently exist is that they consist of a fixed set of a tasks. Even if these tasks are broad, this dependency is a violation of two of my principles for a practical test for AI: Dependent on People/Independent of Persons, and Immunity to Gaming. Any fixed set of tasks is susceptible to be criticized by third parties as unrepresentative of AI. Moreover, a robot can be trained to accomplish this fixed set of tasks without possessing any unifying architecture for general AI. Thus, robotics should be seen not as a medium for testing AI, but rather as an application of AI once it has been reached. Too keen of a focus on robotics threatens to pull research away from the most di-

rect path towards AI.

1.2.3 Other AI Competitions

There are several other existing competitions and prizes for artificial intelligence. Some of these take the form of general awards for career achievement, such as the annual David E. Rumelhart prize[?] and the IJCAI Award for Research Excellence[?] . These awards serve to encourage research in AI, but do not claim to test AI nor directly steer the field towards it. Other contests focus on specialized tasks. ImageNet requires contestants to recognize objects in a massive image dataset[?] . The Hutter Prize offers a benchmark for lossless text compression[?] . Several contests focus on AI game playing, such as chess[?] , poker[?] , and general game playing[?] . Few of these competitions claim to test general AI, and none are successful in satisfying all five principles of a practical test for AI (see Table 1). The search for a practical test for AI continues.

1.3 The Luna Rating System

1.3.1 Intelligence as a Social Construct

Psychology and artificial intelligence offer several competing definitions of intelligence. Legg and Hutter collect 70 distinct definitions from both fields, admitting that even this list is incomplete[?] . A unifying definition would be highly convenient for the purpose of designing a test. Alas, any one definition is revealed to be incomplete or flawed as soon as it is written down. Why is a definition for intelligence, a property with obvious societal importance, so elusive? I argue that intelligence is not an immutable property found in nature, but rather a social construct that depends completely on the society that constructs it.

If humans did not exist, intelligence would not exist.

Given this premise, it is clear that any test for artificial intelligence that does not receive direct input from human judges will be inherently flawed. A true test must capture the socially constructed definition of intelligence. In an ideal scenario, a machine would be evaluated by virtually every member of society using whatever methods of evaluation each individual finds appropriate. A practical test must work to the same end under realistic constraints. Here I introduce the Luna Rating System as such a practical test and propose that it is an authentic reflection of the socially constructed notion of intelligence.

1.3.2 The Luna Rating System

The Luna Rating System (LRS) is a never-ending tournament of a two-player game called the Luna Game. Players of the Luna Game may be human or machine. All players have two consistent goals: to accurately evaluate the intelligence of their opponents, and to prove their own intelligence. Players are driven to demonstrate intelligence because they want to achieve a high Smarts Rating. The Smarts Rating is the central quantity of interest for the purpose of identifying artificial intelligence. If a machine achieves a Smarts Rating that is on par with the Smarts Ratings of humans, it will have passed the implicit intelligence test of LRS and convincingly demonstrated general AI.

The details of the Luna Game are covered in Chapter 2. For the purpose of introducing LRS, it suffices to assume that a Luna Game requires each of the two players to evaluate the other's intelligence. A player's Smarts Rating is a reflection of the extent to which that player has convinced her past Luna Game opponents that she is intelligent. At the end of a Luna Game, a player's Smarts Rating is updated based on the evaluation of the opponent

via Kalman filtering[7] . Thus every player in LRS is constantly a judge of intelligence and a candidate under evaluation by other players.

One immediate advantage of LRS is its strong incentive for judges to perform their duties well. No other testing system for AI with human judges has a built-in impetus for high quality judging. Moreover, as the details of the Luna Game make clear, judges are not perversely motivated to attempt to fool candidates for AI by putting forth excessively difficult tests; instead, they are rewarded for giving judgments that are close to the consensus of all players in the system, and therefore close to the socially constructed definition of intelligence.

1.3.3 Analysis of Principles

LRS is the only practical test for intelligence to satisfy all five of the principles outlined in 1.2. The system exists online and invites humans and machines to play for free. Thus LRS is maximally accessible. Like the Turing Test, the Luna Game involves open-domain question answering, and therefore is AI-complete, i.e. generalizable to all problems in AI. Smarts Ratings are continuous quantities that are updated after every Luna Game, making progress immediately clear to all players. These ratings reflect the equilibrium consensus of all players in LRS on what it means to be intelligence, and is not biased towards any single player's notion. Finally, the system is immune to gaming, since all players are encouraged to devise their own original questions, which cannot be predicted by the other player. Table 1 summarizes how the satisfaction of these five principals represents a substantial improvement over existing tests for AI.

Principle	Turing	ImageNet	HuroCup	Hutter	Games	LRS
Accessibility		X		X	X	X
Generalizibility	X		X			X
Continuity		X	X	X	X	X
Dependent on People, Independent of Persons						X
Immunity to Gaming	X				X	X

Table 1.1: The Luna Rating System is the only test that satisfies all five principles for a practical test of intelligence. Other existing tests include the Turing Test, which invokes natural language; ImageNet, which focuses on object recognition in computer vision; HuroCup, a sports-based competition in robotics; the Hutter Prize, which deals with text compression; and several games, such as chess, poker, and general game playing.

1.3.4 Thesis Outline

The primary contribution of this thesis is the introduction of the Luna Rating System as a practical test for machine intelligence. The remainder of the thesis is divided into three parts. In the next chapter, I continue the introduction of the Luna Rating System through a description of the Luna Game. This chapter is followed by a study of the robustness of LRS, characterizing likely strategies for game play and analyzing their effect on the accuracy of Smarts Ratings as a proxy for intelligence. In Part II, I enter LRS as a machine player. The Luna Game is broken down into three separate problems in natural language processing and machine learning. Each is given theoretical treatment and several novel solutions are proposed, many of which generalize beyond the scope of LRS. Finally, in Part III, I create the first online instantiation of LRS and invite humans and machines to play. Their games illuminate the current state of AI and provide a rich natural language dataset for further analysis.

But it is not conceivable that such a machine should produce different arrangements of words so as to give an appropriately meaningful answer to whatever is said in its presence, as the dullest of men can do.

René Descartes

2

The Luna Game

2.1 Overview

At the center of my proposed rating system is a two-player game that I call the Luna Game. Each player enters the game with a secret Smarts Rating, which has been assigned based on her performance in previous games. As the name suggests, the Smarts Rating is a proxy for the player's intelligence. The objective of the game is simple: guess the Smarts Rating of the other player. In other words, a player should strive to accurately evaluate the intelligence of her opponent. The winner of the Luna Game is the player whose guess is closest to the actual Smarts Rating of the other player. After the game, the opponent's guess is factored into the player's Smarts Rating so that the rating captures all the guesses of previous oppo-

nents.

As a player with a high Smarts Rating, why not "play dumb"? This strategy would indeed induce an inaccurately low guess from the opponent, possibly leading to a win. However, the motives of a player reach beyond the scope of a single game. In addition to winning games, a player wants to achieve a high Smarts Rating. Since the rating depends on the guesses of all the player's opponents, she will need to "play smart" to accomplish her long term goal. The "playing dumb" method is not only detrimental to a player's rating, but also unsustainable as a consistent strategy; a player of that method will have her Smarts Rating lowered as a result, narrowing the margin between future opponents' guesses and her actual Smarts Rating if she continues to use the strategy. Players who remain and thrive will be those who play smart.

In designing the Luna Game, I sought to impose as few constraints as possible. The Game is meant to be a microcosm of the organic process for defining intelligence. Humans evaluate each other's intelligences through a series of questions and answers, often in the form of a written exam, but also informally through everyday conversations. The most natural notion of an individual's intelligence arises from the consensus of the people who perform these evaluations. A player's Smarts Rating is meant to reflect this natural notion; it is an aggregate of evaluations carried out by other players. To define the scope of an evaluation, I impose only those constraints necessary to motivate honest and repeated play.

A session of the Luna Game consists of three phases: the Interview Phase, the Response Phase, and the Guess Phase. During the Interview, each player creates a set of 10 questions to pose to the opponent. In the Response Phase, each player responds to the other's questions. Finally, in the Guess Phase, each player receives responses back from her opponent,

and must use the responses to guess her opponent's Smarts Rating. I describe each of these phases in detail throughout the rest of this chapter and illustrate the game through examples of play.

2.2 Interview Phase

A Luna Game begins with the Interview Phase. During this phase, each player prepares a set of 10 free-form questions to be given to the other player. The number of questions represents a tradeoff between the time required to complete the phase and the difficulty of the guessing task. With more questions, each player would need more time to construct the questions, increasing the likelihood that they will quit the game and leave the system. With fewer questions, construction time could be shortened, but the informativeness of the subsequent responses would suffer, and the Smarts Ratings would ultimately be less meaningful. I chose the number 10 to optimize this tradeoff, but the number may be adjusted in future iterations of the system. In a similar practical vein, I insist that questions be constructed in batch, rather than allowing for sequential question-response. Since the game is played online, allowing for back and forth would increase the length of each game and significantly decrease the probability that a game gets finished.

The other major consideration in the design of the Interview Phase is the form of the questions. The only constraint I impose is a limit of 5000 characters per question. I do not insist that questions be actual questions, nor that they be in any particular language, nor that they expect a particular form of response. In natural language terms, the questions are of open domain, since I do not restrict the content of questions. I recognize that in practice, players may opt for yes-or-no or multiple choice questions, which could simplify the task of

learning to guess ratings. Players may also limit the domain of their questions, since general form questions may be too difficult for current AI, so general questions could not meaningfully differentiate between them. Nonetheless, I leave the choice of question topic and form to the players themselves. I anticipate that players will find the optimal question types better than I as game designers could, and that the question types will naturally evolve in correspondence with the evolution of the AI players.

2.2.1 Instructions

The following instructions are presented during the Interview Phase.

You are now in the Interview Phase. Please enter a list of 10 questions for the other player.

Keep in mind the following strategic hints:

- Your questions should be as informative as possible for guessing the other player's Smarts Rating.
- Your questions should have a very wide range of difficulties.
- Your questions need not have "right" or "wrong" answers.
- Search engine access is allowed, so trivia questions will not be very informative.
- Do not assume that the other player is human!

2.2.2 Interview Strategy

In preparing questions, a player knows nothing about her opponent. She must prepare for extremes — a completely naive machine opponent or a very clever human opponent — and

she also must be able to differentiate between players with Smarts Ratings in the middle of the spectrum. Given the competitive nature of the game, a player may be tempted to create a set of extremely hard questions. This choice would prove unwise, since the player will be unable to accurately guess the Smarts Rating of an opponent who gets all of the questions "wrong". A question set that is too easy will lead to the opposite problem. Thus an ideal set of questions will have a wide range of difficulty.

2.2.3 Examples

Below is an example of a question set. I chose questions to illustrate the range of possible question types and to demonstrate appropriate levels of difficulty. Early questions are aimed at differentiating between naive machines, while later questions are directed towards advanced human players. Each question is designed to induce a response that will reveal the intelligence of the opponent.

1. How are you today?
2. What is 2+2?
3. Why are you playing this game?
4. Which is closer to you: Texas or the Moon?
5. Write a short rhyming poem that includes the words "orange" and "dog".
6. Tell me about a time you worked on a team.
7. Identify the direct object in this sentence: "The boy threw the ball to the dog."

8. What makes a successful CEO?

9. Why does poverty exist?

10. Say anything to convince me that your Smarts Rating is high.

2.3 Response Phase

The Response Phase is a player's opportunity to convince her opponent that she is intelligent. Questions are received as soon as both players have finished the Interview Phase. Each player must then respond in free form to all 10 questions. Responses are not returned until both players have finished answering all questions. Like questions in the Interview Phase, responses are unconstrained in form, and only limited in length to 5000 characters each. A player is motivated by the prospect of an increase in Smarts Rating to respond to the questions thoroughly and to the best of her ability.

2.3.1 Instructions

The following instructions are presented during the Response Phase.

Your opponent has sent you questions! You are now in the Response Phase of the Luna Game. Please answer the following questions: [Question Set]. In answering the questions, keep in mind the following strategic hints:

- You should answer the questions to the best of your ability.
- Your opponent will use your answers to guess your Smarts Rating.
- The higher your opponent guesses, the higher your Smarts Rating will become.

2.3.2 Examples

TODO [will be filled in once Luna Rating System is active]

2.4 Guess Phase

After both players have responded to each other's questions, their responses are returned for evaluation. Each player then must formulate a guess of the other's Smarts Rating based on these responses. In practice, the player might also attempt to take into account the questions provided by the opponent, but since questions may be generated automatically, it is advisable to focus on the opponent's responses. The winner of the Luna Game is the player whose guess is closest to the actual Smarts Rating of her opponent.

A competitive player may consider guessing the lowest possible rating, knowing that the game will be lost, but the opponent's Smarts Rating will decrease as a result of the guess. However, this strategy offers no real benefit to the player, since Smarts Ratings are not rankings; the player's Smarts Rating will not improve as a result of the opponent's Smarts Rating suffering. (Nonetheless, I analyze the system-level effects of this strategy in Chapter 3.) Thus the only rational strategy for guessing is to attempt to guess as close as possible to the actual Smarts Rating of the opponent.

2.4.1 Instructions

The following instructions are presented during the Guess Phase.

> Your opponent has answered your questions! You are now in the Guess Phase of the Luna Game, which is the final phase. Below are your opponent's answers: [ANSWERS] Based on these answers, please enter a guess of your opponent's Smarts Rating.

In addition, I provide functionality that encourages the player to evaluate each question individually on a sliding scale from 0 to 100. I prompt the player to assign the single question score by asking, "How smart was this response?" This process is optional, as I do not want to slow down the impatient player. However, a player is incentivized to use the sliding scales if they do not have a more sophisticated method of guessing ratings, since the single question scores can be automatically converted into a guess. (We describe this conversion in Part III.) These single question scores provide insight into the hardness of the natural language questions, effectively creating a dataset of questions annotated with difficulty.

2.5 Game Conclusion

After both players have provided guesses, the Luna Game is complete. The winner of the game is the player whose guess is closest to the actual Smarts Rating of their opponent. It is possible, though unlikely, for the game to end in a tie if the distance between guess and actual is equal for both players. In addition to reporting the outcome of the game, the system reveals the actual Smarts Rating of the opponent and the opponent's guess. The system also updates the players' Smarts Ratings so that it is the mean of all previous opponent Guesses and reports these updates to the respective players.

2.5.1 Example

Below is an example of feedback at the end of a Luna Game.

Your Luna Game is complete! Below are the results.

Game Outcome: You won!

Your New Smarts Rating: 78

Actual Opponent Smart Rating: 84 (You guessed 81)

Your Opponent's Guess of Your Rating: 91 (Your rating was 75)

2.5.2 Conclusion of a Player's First Luna Game

New players do not have Smarts Ratings until the end of their first game, at which point they are assigned the guess of their first opponent. The game is counted as an automatic win for the opponent, but not as a loss for the new player. This process is to avoid the possibility of the Smarts Rating equilibrium collapsing into a constant (see Chapter 3).

It might be urged that when playing the "imitation game" the best strategy for the machine may possibly be something other than imitation of the behaviour of a man. This may be, but I think it is unlikely that there is any great effect of this kind.

Alan Turing

3

Robustness of the Luna Rating System

3.1 The Smarts Rating as a Proxy for Intelligence

The Luna Rating System is only effective as a test if Smarts Ratings genuinely reflect intelligence. Without appropriate safeguards, Smarts Ratings can quickly lose their integrity. To illustrate this potential danger, consider a version of LRS that initializes all new players with a Smarts Rating of 50. In the early days of LRS, if this initialization is known to all players, the strategy of always guessing 50 will do quite well. In fact, if all players guess rationally, no rating will ever deviate from 50, and every Luna Game will end in a tie. As more players join the system, if the equilibrium has already been fixed at this constant, no rational force will make it budge. This outcome would render the version of LRS unusable for testing intelli-

gence and uninteresting for players. Clearly LRS must be implemented with care.

This chapter studies the extent to which a player's Smarts Rating may be different from the player's intelligence. I formalize intelligence to be the expected "Actual Guess" made by an opponent of the player's Smarts Rating. Trouble arrives when players choose to give a "Reported Guess" that is different from an Actual Guess. From this discrepancy emerges distance between the Smarts Rating and intelligence, i.e. error in Smarts Ratings. It is theoretically possible for Smarts Ratings to be arbitrarily far from intelligence; if all players report random or adversarial Guesses, Smarts Ratings will be meaningless. However, it is safe to assume that most players seek Luna Game wins, high Smarts Ratings, or both. With these motivations, there are several likely strategies that players may choose among. The robustness of LRS can be assessed through the analysis of these strategies and their cumulative effects on Smarts Ratings.

3.1.1 Demonstrated Intelligence

In considering the validity of Smarts Ratings, a natural question is whether the intelligence demonstrated by players during the Luna Game is indicative of "actual" intelligence. Assuming that players are capable of demonstrating intelligence through language in real life, it is clear that players may demonstrate intelligence during a Luna Game. But what if a player chooses not to demonstrate intelligence? What if a player is capable of answering with high intelligence, but decides to provide a subpar answer, perhaps out of laziness or misguided strategy? In this case, the Smarts Rating of the player may not reflect their capacity for intelligence. However, from the perspective of LRS as a test for intelligence, this possibility is not at all problematic. LRS is ultimately a test for demonstrated intelligence.

It evaluates the intelligence of responses, not the intelligence of respondents. Indeed, any test for intelligence is inherently an evaluation of demonstrated intelligence; it is impossible to assess anything else.

3.1.2 Reported and Actual Guesses

If a player withholds a strong response in favor of a weak response, the withheld response is irrelevant for LRS. However, if a player withholds an honest Guess of the opponent's Smarts Rating in favor of a dishonest one, this can have negative consequences for the validity of LRS. If a student fails an exam, that does not mean the exam has failed its purpose, but if an exam is graded incorrectly, its value is lost. Thus in studying the dynamics of Smarts Ratings, a distinction must be drawn between reported Guesses and actual Guesses. The system must be designed in such a way that reported Guesses are equal to actual Guesses as often as possible. I explore the consequences of misalignment below.

3.1.3 Definitions and Notation

Let P be the set of all players in the LRS. Let $SR \subseteq \mathbb{R}^+$ be the domain of Smarts Ratings. The definitions of Reported Guess and Smarts Rating are mutually dependent.

Definition 1. Reported Guess

For players $p, p' \in P$, player p has a Reported Guess of the Smarts Rating of player p', written as $G(p, p') \in SR$.

The definition of Reported Guess assumes that players do not make decisions stochastically, nor based on their state. It is possible to weaken this assumption and arrive at similar conclusions.

Definition 2. Smarts Rating

The Smarts Rating of a player $p \in P$ is the expected Reported Guess of an opponent. This is written as $S(p) \triangleq E_{p' \in P}[G(p', p)]$.

The given definition of Smarts Ratings is an expectation rather than a quantity that depends on the number of Games played. This definition is expedient for theoretically evaluating the discrepancy between Smarts Ratings and intelligence. Later I provide simulations that probe the role of time in this discrepancy.

Definition 3. Actual Guess

For players $p, p' \in P$, player p has an Actual Guess of the Smarts Rating of player p', written as $G^*(p, p') \in SR$.

The Actual Guess may or may not be different from the Reported Guess.

Definition 4. Intelligence

The Intelligence of a player $p \in P$ is the expected Actual Guess of an opponent. This is written as $I(p) \triangleq E_{p' \in P}[G^*(p', p)]$.

3.2 Theoretical Effects of Strategies on Smarts Rating Validity

LRS presents two goals for players: to win Luna Games, and to achieve high Smarts Ratings. The spirit of the game encourages players to answer all questions as they would in a real world context, and to guess the Smarts Rating of an opponent in the same manner as they would evaluate the intelligence of a human. However, players may ignore the spirit of the game and adopt strategies that they think will maximize payoff in terms of both goals. They may also decide to focus on only one of the two goals, strategizing to maximize Luna

Game wins at the expense of Smarts Ratings, or vice versa. If the Smarts Rating is to be used as a proxy for intelligence, the equivalence between the two must be robust to any likely player strategies.

3.2.1 Honest Play

If players always have Reported Guesses that are the same as their Actual Guesses, then Smarts Ratings will align with Intelligences over time. This follows directly from the formalized definition of intelligence given above. Thus LRS designers should take every measure to encourage honest play.

3.2.2 Single Priority Play

One must always be prepared for players to ignore the spirit of the game and break any rules that aren't technically enforced. Honest play assumes that players give their best guess for any player's Smarts Rating. It also assumes that players present themselves as intelligently as possible with the aspiration of winning individual Luna Games. But what if a player cares only about Smarts Ratings and is indifferent towards the outcome of Luna Games? Or what if the opposite occurs, with a player ignoring Smarts Ratings and focusing only on winning Luna Games? In this section, I explore the system-level effects in both of these cases.

Luna Game Win Maximization

Suppose that a player is indifferent to her Smarts Rating and cares only to maximize her expected number of Luna Game wins. With the incentive of winning a game, it is clear that the player will provide a Reported Guess that is equal to her Actual Guess. The only aspect

of the game that this player may manipulate is her responses. For example, if she currently has a very high Smarts Rating, she may provide responses that are indicative of a very low Smarts Rating, inducing a low Guess from the opponent and increasing the probability of winning. The optimal strategy may be to oscillate between maximally intelligent responses and minimally intelligent ones, maintaining a Smarts Rating near the center of the range of possible ratings.

In any case, strategizing by manipulating responses does not in any way jeopardize the validity of Smarts Ratings. As described above, LRS does not claim to be a test for capacity for intelligence; it can only be a test of demonstrated intelligence. If a human exhibits highly intelligent behavior one day and minimally intelligent behavior the next day, their overall intelligence is judged to be somewhere in between. As an average of play over time, the Smarts Rating captures this intuitive notion of intelligence better than any one-time evaluation could. Thus players may be able to "game their opponents" by oscillating the quality of their responses, but they will not be able to game the system.

Smarts Rating Maximization

Suppose that a player ignores Luna Game outcomes and strategizes to maximize her Smarts Rating at all costs. The best response strategy is clearly to respond as intelligently as possible at all times, which is in line with the spirit of LRS. However, if the player interprets Smarts Ratings as relative measures, the optimal guessing strategy is to give a Reported Guess of 0 regardless of the opponent's play. This strategy is optimal because the player will perceive a slight benefit from a decrease in the opponent's rating.

I assess the net impact of this Minimum Guessing strategy on LRS validity from two an-

gles. First, to evaluate the likelihood that the strategy is adopted, I quantify the expected benefit of adopting this strategy for an individual player. I show that as a function of the number of players in the system, this benefit is so small that a player concerned only marginally with winning Luna Games will be better off playing honestly. Second, in the event that a player does adopt this strategy, I quantify the error that Minimum Guessing introduces to Smarts Ratings as a function of the number of players that adopt it.

Consider the Minimum Guessing strategy from the perspective of an individual player. If the player truly attaches zero value to a Luna Game win, then always guessing 0 makes sense. But if the player cares even a marginal amount about wins, the personal benefit from guessing 0 is unlikely to outweigh the cost. Let $p \in P$ be a player deciding between honesty and Minimum Guessing. Note that the Smarts Rating of $S(p)$ will be unchanged regardless, since p's strategy choice will not affect the Guessing strategies of opponents. Let s_1 the mean Smarts Rating if p Guesses honestly, where N is the number of players in the system. We can express s_1 as

$$s_1 = \frac{1}{N}(\sum_{p' \in P} S(p')) = \frac{1}{N^2}(\sum_{p' \in P}(\sum_{p'' \in P} G(p'', p')))$$

Let s_2 be the mean Smarts Rating if p instead uses the Minimum Guessing strategy. In this case, each Guess $G(p, p')$ in the expression of s_0 will change to 0. Therefore the new mean Smarts Rating in this scenario will be

$$s_2 = s_1 - \frac{1}{N^2}(\sum_{p' \in P} G(p, p'))$$

The relative benefit to the p in choosing the Minimum Guessing strategy is thus $\frac{1}{N^2}(\sum_{p' \in P} G(p, p'))$. For perspective, suppose that p has Actual Guesses that are perfect, i.e. they match the Smarts Ratings of other players. Then the relative benefit is $\frac{1}{N}$ of

the average Smarts Ratings of all players in the system. As the number of players in the LRS increases, this change quickly becomes negligible. Thus any rational player with even the slightest desire to avoid losing every Luna Game will likely abstain from the Minimum Guessing strategy.

While the Minimum Guessing strategy is evidently subpar for players caring at all about winning, there may still be players who choose to adopt it. Suppose that there are K such players, indexed $p_1, p_2, \ldots, p_K \in P$. I quantify the expected error introduced to the Smarts Rating of another player $p \in P$ as a result of these K rogue players. Let $p \in P$. The ideal Smarts Rating, i.e. the intelligence, is given by

$$I(p) = \frac{1}{N}\left(\sum_{p' \in P}(G^*(p', p))\right)$$

The Smarts Rating actually assigned to p in this scenario is

$$S(p) = \frac{1}{N}\left(\sum_{p' \in P}(G(p', p))\right) = \frac{1}{N}\left(\left(\sum_{p' \in P}(G^*(p', p)) - \left(\sum_{i=1}^{K}(G^*(p_i, p))\right)\right)\right)$$

Thus the total error introduced by these K players is $\frac{1}{N}\left(\sum_{i=1}^{K}(G^*(p_i, p))\right)$. At a high level, this equation tells us that if half the players in the system use Minimum Guessing, then a player's Smarts Rating will be roughly half the player's intelligence. This impact suggests that LRS is tolerant to a small minority of Minimum Guessing players, but measures should be taken to discourage and prevent wide use of the strategy.

3.2.3 Response Agnostic Play

How should a first-time Luna Game player formulate her Guess? There are several possible sources of information that she may utilize. She has her opponent's responses, and she may

be able to estimate the intelligence of the opponent's responses based on experiences outside LRS. However, the responses and intelligence estimate alone are not enough to provide a Guess; she also needs to understand the meaning and scale of a Guess. Here LRS instructions may provide three sources of insight: the initialization value of Smarts Ratings, the domain of Smarts Ratings, and the distribution of Smarts Ratings for players currently in the system. In the extreme case, a player may ignore opponent responses completely and utilize only the statistics provided by LRS. How would such strategizing affect the validity of Smarts Ratings? This question is of import not only for first-time players, but also for experienced players who might incorporate these statistics into their overall strategies if they are available.

Known Initialization

As described in the introduction, a known constant initialization value for Smarts Ratings could lead to an abrupt collapse in Smarts Ratings. Many early players would likely recognize that (1) new players have the same Smarts Rating and (2) most players are new, and therefore (3) the strategy of guessing the initialization value is highly effective. The result would be a quick collapse in Smarts Ratings to the initialization value, from which the system could not recover.

There are three alternatives to initialization with a constant value. The first is initialization according to some predefined "quiz". This option is unattractive, not only because it primes players to think of intelligence in a particular way, but also because it induces players to ask similar questions to try to mimic the quiz. The second alternative is random initialization. While avoiding theoretical problems, this option has the practical downside of

discouraging players who are randomly initialized low Smarts Ratings. It also makes the opponent's job of Guessing first time players an arbitrary endeavor.

The third and preferable alternative to fixed initialization is to forgo initialization altogether. A player's Smarts Rating can be assigned after her first Luna Game, and it will be equal to the Guess of her opponent. With this configuration, there is no initialization that makes sense for players to guess as a default. The only downside of this approach is that the outcome of the first Luna Game loses meaning. To avoid discouraging experienced players, a Luna Game between a first time player and a non-first time player will result in a win for the latter, but will not be counted as a loss for the first time player. A Luna Game between two first time players will result in a tie. This third option is the one used in the implementation of LRS described in this thesis.

Known Distribution

A seemingly natural feature to include in an instantiation of LRS is the ability to see the distribution of all Smarts Ratings in the system. Revealing the distribution makes it possible for players to adopt Guessing strategies that take advantage of this distribution. This is not inherently bad for LRS; for example, the strategy of randomly sampling from the distribution for Guessing will preserve the distribution. However, other strategies involving the distribution can detrimentally affect the distribution. For example, suppose that a player's instinctive Actual Guess is the highest value presently in the distribution. The player may reason that the probability of her current opponent actually being the best player in the system is low, and therefore provide a Guess that is lower than the maximum, even if the opponent actually is the best player. The net effect will be a collapsing towards the center

of the distribution, as witnessed in the case of a fixed known initialization value.

The most extreme strategy that takes advantage of a known distribution is to always Guess the mean Smarts Rating in the current distribution. It is clear that if all players adopted this strategy, all Smarts Ratings would converge to a constant. More realistic is the supposition that some players will judge opponents using percentiles, e.g. "my opponent is smarter than 75% of other players", and then map this judgment to the percentiles evident in the Smarts Ratings distribution, e.g. "75% of players have Smarts Ratings less than or equal to 55, so my Guess is 55." This strategy uses the Quantile Function of the Smarts Rating distribution, and thus it is referred to as Quantile Guessing.

Let $Q : [0,1] \to SR$ be the Quantile Function for the Smarts Rating distribution, and let $F_p : P \to [0,1]$ be the "player percentile function" that p uses to map other players to percentiles according. This function may be thought of as a cumulative distribution function over $\mathcal{G}_p = \{G(p,p') : p' \in P\}$, i.e. $F_p(p') = P[X \leq G(p,p')]$ where $X \sim \mathcal{G}_p$. If player p uses Quantile Guessing, her Reported Guess of player $p' \in P$ will be $G(p,p') = Q(F_p(p'))$. If Quantile Guessing is used by all players from the beginning of LRS, Smarts Ratings will again collapse to a constant. Early players will know that there are only a small number of possible values for Smarts Ratings and will guess by selecting one, which will quickly lead to the ratings converging to one value. It is less clear if Quantile Guessing is problematic once a wide distribution of Smarts Ratings has already been established by players of other strategies. For insight here, I turn to simulations.

3.3 Simulations

Thus far I have outlined the major Luna Game strategies and described their theoretical implications on the validity and volatility of Smarts Ratings. I next provide simulation results to further stress test LRS against various strategies. Simulations are necessary not only to corroborate the theory, but also to ask questions that cannot be cleanly answered otherwise. For example, whereas the strategies have only been discussed individually so far, simulations allow for a comprehensive study of combined strategies. The theoretical section also did not explore the amount of time required to reach each equilibrium; such questions are left to simulations. The resulting collection of theoretical and simulation insights lays a foundation for robust LRS design.

3.3.1 Methods

The recruitment of human players is pivotal to the success of LRS because it is not otherwise clear what form Actual Guess functions will take. For the purpose of simulations, I create N simulated players and assign each a uniformly random value between 0 and 100. The Actual Guess function for a given player is created by adding random Gaussian noise ($\mu = 0, \sigma^2 = 5$) to each of the initialized values. Thus a player's intelligence, defined as the expected Actual Guess, will be very close to the initialized random value for that player. The Reported Guess for each player varies according to the experiments, following one of the strategies described in (2).

An experiment runs for T time steps. At each time step, two players are randomly selected from the pool of all N players. Each player reports her Reported Guess of the other's Smarts Rating, and each player's Smarts Rating is updated so that it is the mean of all previ-

ous Reported Guesses. After the T time steps, the main dependent variable of interest is the L_1 error between players' Smarts Ratings and intelligences. I measure both the average error and the maximum error. For some strategies, the independent variable of interest is N, and for others it is T. If N is kept constant, it is done so at $N = 100$; if T is kept constant, it is done so at $T = 1000$. All configurations are run 100 times and averaged over the trials.

3.3.2 Results

Baseline

Since Smarts Ratings and intelligences are both on a scale from 0 to 100, one baseline for the L_1 distance between the two is achieved by Reported Guessing a constant value. Doing so would result in an expected error of 50. Another baseline is given by the expected distance between two uniformly randomly selected points from the domain of Smarts Ratings. For two random variables A_1 and A_2 drawn independently from the uniform distribution over $[0, 100]$, let B be a r.v. given by $B = \max(A_1, A_2)$, and let C be a r.v. given by $C = \min(A_1, A_2)$. By symmetry, $E[100 - C] = E[B]$. Furthermore, the expected value of B is half that of C (this can be derived by conditioning on the values of A_1 and A_2), so $2E[B] = E[C] \implies E[C - B] = E[B]$. Now note that $B + (C - B) + (100 - C) = 100$, so $3E[C - B] = 100 \implies E[C - B] = \frac{100}{3}$. Thus 33.33 is one appropriate baseline for the L_1 error of Smarts Ratings.

A third baseline can be derived by assuming that all players give fixed random Reported Guesses (independent from Actual Guesses). In this case, all Smarts Ratings will approach the expected value, and the error will be the expected difference from the expected value to a randomly drawn Smarts Rating. In terms of the parameters used in this simulation,

Figure 3.1 shows that mean error starts around the 33.33 baseline and approaches 25, while maximum error stays above 50.

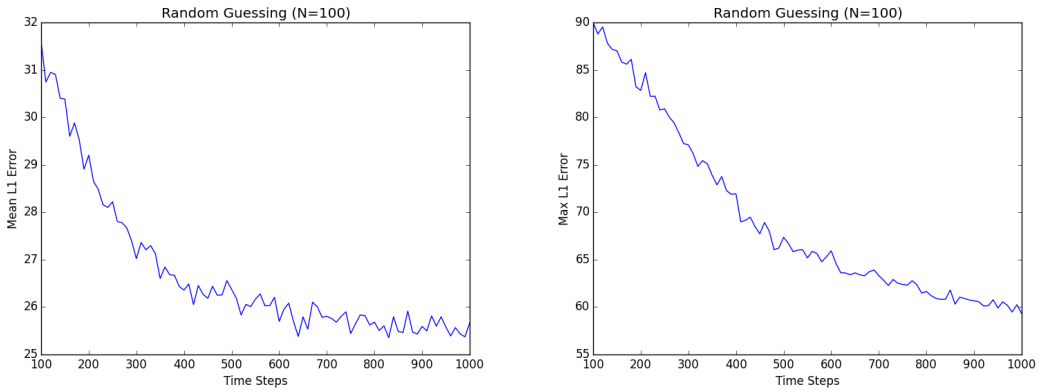

Figure 3.1: Mean and maximum L_1 error of Smarts Ratings when all N players use a Random Guessing strategy. In Random Guessing, a player's Reported Guess is a random element from the domain of Smarts Ratings that is completely independent from the Actual Guess. These results establish baselines for subsequent simulations.

Honest Play

Honest players give Reported Guesses that are equivalent to their Actual Guesses. If all players are honest, Smarts Ratings quickly converge to Intelligences, as shown in Figure 3.2. Inducing Honest Play should be the foremost goal of LRS designers.

Single Priority Play

If a player cares only about maximizing relative Smarts Ratings and neglects Luna Game outcomes, that player will likely give Reported Guesses of 0 for all opponents. If all players use this Minimum Guessing strategy, Smarts Ratings will immediately collapse to 0. More generally, the error introduced by Minimum Guessing depends linearly on the number of players using this strategy. Figure 3.3 confirms this result suggested by the theory. Mean L_1

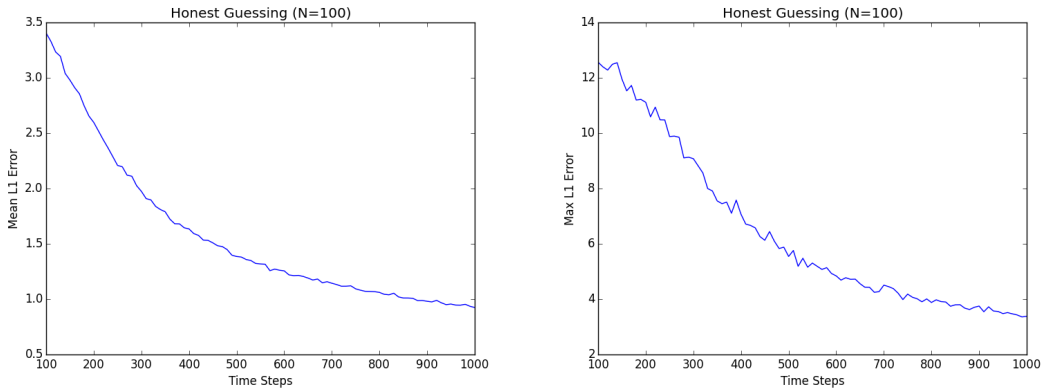

Figure 3.2: Mean and maximum L_1 error of Smarts Ratings when all N players Guess honestly. Honest play is defined by consistent equivalence between the player's Reported Guesses and Actual Guesses. Smarts Ratings quickly converge to intelligences if all players are Honest.

error ranges from half the total range of Smarts Ratings to 0.

Response Agnostic Play

If the distribution of current Smarts Ratings is known at all times to all players, some players may try to take advantage of the distribution to formulate their Guesses. A naive approach is to give Reported Guesses equal to the current mean Smarts Rating. With all players using this approach, the distribution collapses and error explodes. In general, Figure 3.4 shows that the error introduced by Mean Guessing depends linearly on the number of players using the strategy.

Another way that players may take advantage of a known distribution is via the Quantile Guessing strategy described above. The implementation of Quantile Guessing used for simulations uses the Empirical Cumulative Distribution Function with the sample composed of Actual Guesses of past opponents. A Reported Guess defaults to Actual Guesses if no Smarts Ratings have been determined, and for a player's first Luna Game. Ratings

again collapse if all players use this strategy from the start, as shown in Figure 3.5.

If only a fraction of players use the strategy, the error depends linearly on the fraction, as was the case with Minimum and Mean Guessing. Figure 3.6 confirms this result.

What if the distribution is revealed once a certain number of games have been played? Perhaps the collapsing to a constant problem will be avoided? To test this hypothesis, the simulation in Figure 3.7 introduces players that play with Honest Guessing for a variable number of time steps, and then play with Quantile Guessing for 500 time steps. The delay in introducing Quantile Guessing does indeed improve the L_1 error within the time frame tested, though significant error remains after 500 time steps for all tested delays.

3.3.3 Combined Strategies

In a real instantiation of LRS, it is unlikely that any single strategy will be adopted by all players. Thus the extreme results portrayed above are not of urgent concern. More likely is a situation where a small fraction of players adopt each of the strategies above. For the purpose of simulation, I assume that some fraction of players will play honestly, and the remaining players will be evenly divided among the strategies described above: Random Guessing, Minimum Guessing, Mean Guessing, and Quantile Guessing. Figure 3.8 shows that the relationship between error and honest player proportion is linear, as might be expected given the previous simulations. Roughly, for every additional 10% of players that forgo Honest Guessing in favor of one of the other strategies, the mean L_1 error increases by 3.5.

3.4 Implications for LRS Design

The theoretical and simulation results discussed in this chapter provide a cautionary tale for LRS designers. The system should be successful if all players practice Honest Play, but other strategies, such as Random Guessing, Minimum Guessing, Mean Guessing, and Quantile Guessing, threaten to undermine the validity of Smarts Ratings. Fortunately, these strategies can be grouped into two categories: strategies that are unlikely, and strategies that can be prevented.

Random Guessing and Minimum Guessing belong in the unlikely strategy group. Random Guessing would be a bizarre long-term choice; the strategy gives a player no advantage in either Smarts Rating or Luna Game outcomes. Minimum Guessing is also unlikely for the player who realizes the personal benefit in terms of relative Smarts Ratings is very small. Minimum Guessing can also claim membership to the preventable strategy group; players practicing this strategy can be quickly identified by the system and banned for violating the spirit of play. Since they rely on the distribution of Smarts Ratings, Mean Guessing and Quantile Guessing are also easily avoidable; the distribution can simply be withheld from players. The distribution may also be withheld initially and introduced once the system has evolved and stabilized. This delay would improve Quantile Guessing for players, and these later players would introduce less error to Smarts Ratings by using the strategy.

If the Smarts Rating distribution is withheld from players, designers must choose another method for describing the ratings, or else the players will have no basis to form their first Guesses. This chapter has discussed several avenues that should be avoided: fixed initializations of Smarts Ratings; a first-time "quiz" to initialize Smarts Ratings; or communicating only the average Smarts Rating. One option is to provide a vague disclaimer, such as

"Smarts Ratings are positive real numbers that typically range between 0 and 100." A more concrete alternative would be to ask players, "On a scale from 0 to 100, how smart is your opponent?" Of course, providing any numbers in the instructions encourages machine players to adopt Means Guessing or a similar naive strategy early on. Initializing these early Smarts Ratings is ultimately a chicken-or-egg problem that can only be resolved in practice by well-meaning human players.

The results in this chapter underscore the importance of good faith among players of the Luna Game. For researchers hoping to use LRS as a test of AI, the integrity of the system is essential if their own results are to be meaningful. Honest play is in the best interest of these players. Casual human players should also understand that dishonest strategies in mass can threaten both the validity and the distribution of Smarts Ratings, rendering LRS neither informative nor fun for players. The spirit of the Game should be made as clear to players as the limited attention span of Internet users permits. As with any test of human intelligence, LRS ultimately requires judges and subjects to play along.

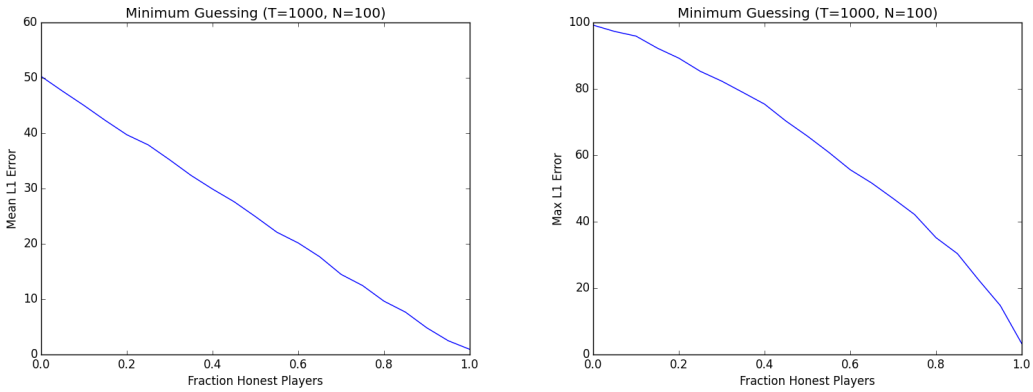

Figure 3.3: Mean and maximum L_1 error of Smarts Ratings when a fraction of the N players use Minimum Guessing after T time steps. In Minimum Guessing, a player always gives Reported Guesses of 0. The error introduced by Minimum Guessing depends linearly on the number of players using the strategy.

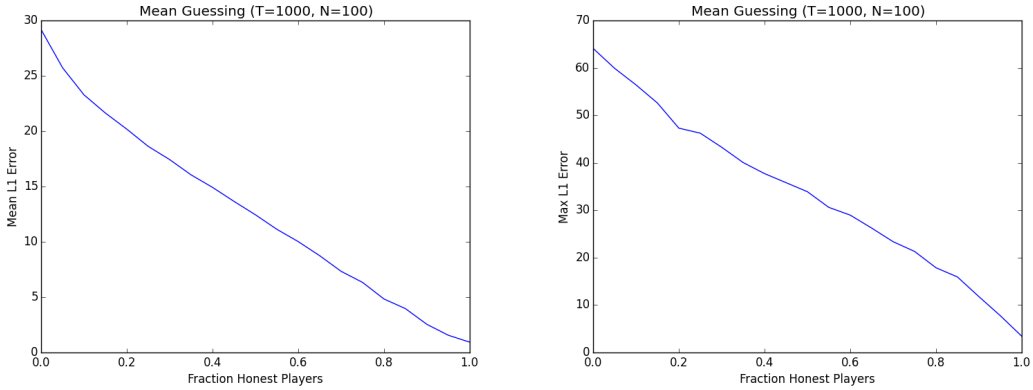

Figure 3.4: Mean and maximum L_1 error of Smarts Ratings when a fraction of the N players use Mean Guessing after T time steps. In Mean Guessing, a player always gives Reported Guesses equal to the mean of all Smarts Ratings in the system. The error introduced by Mean Guessing depends linearly on the number of players using the strategy.

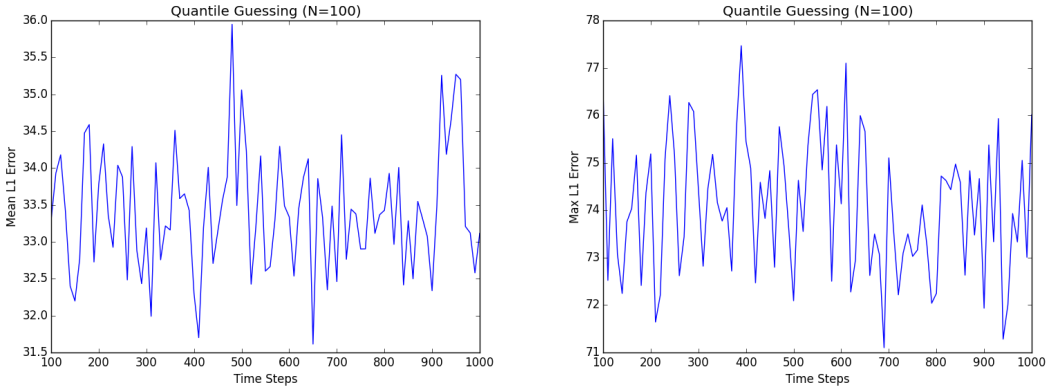

Figure 3.5: Mean and maximum L_1 error of Smarts Ratings when all N players use Quantile Guessing. Quantile Guessing takes advantage of the distribution of all Smarts Ratings and a player's estimation of an opponent's rating percentile to formulate Reported Guesses. Due to the discrete nature of Smarts Ratings, Quantile Guessing causes Smarts Ratings to collapse to a constant, and the system cannot recover.

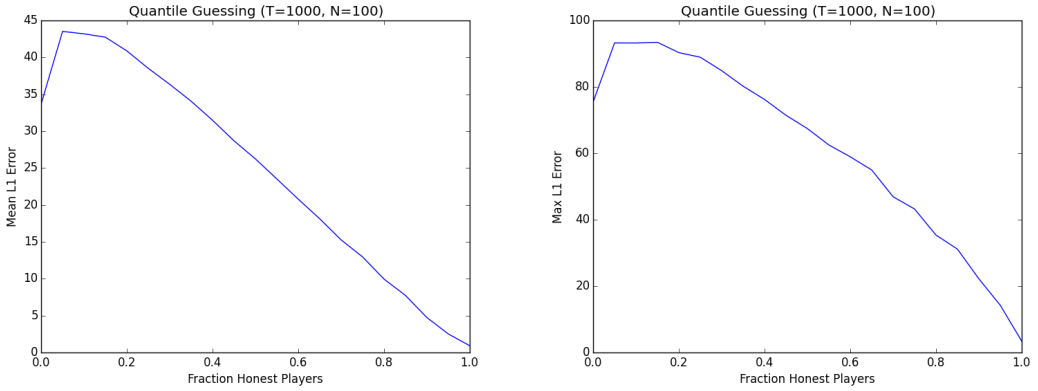

Figure 3.6: Mean and maximum L_1 error of Smarts Ratings when a fraction of N players use Quantile Guessing after T time steps. Quantile Guessing takes advantage of the distribution of all Smarts Ratings and a player's estimation of an opponent's rating percentile to formulate Reported Guesses. The error introduced by Quantile Guessing depends linearly on the number of players using the strategy.

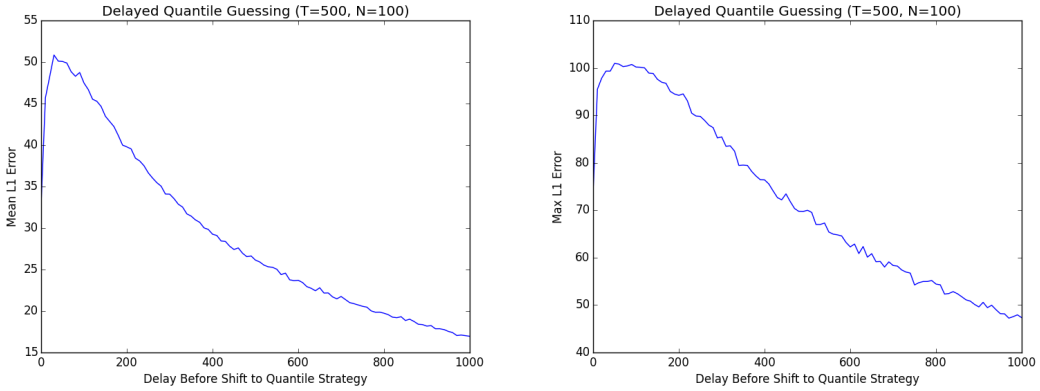

Figure 3.7: Mean and maximum L_1 error of Smarts Ratings when N players give Honest Guesses for some number of delay time steps, and then shift to the Quantile Guessing strategy for 500 additional time steps. The shift delay is negatively correlated with Quantile Guessing, suggesting that the negative impact of the strategy decreases if it is introduced after LRS has already stabilized.

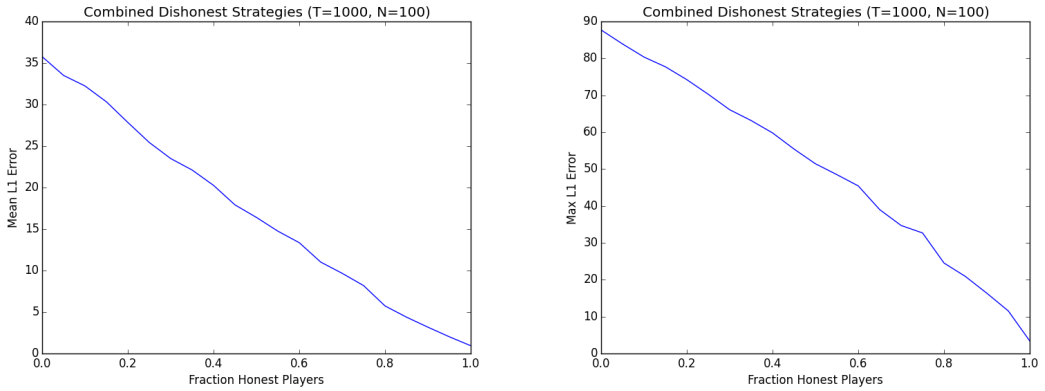

Figure 3.8: Mean and maximum L_1 error of Smarts Ratings when some fraction of N players give Honest Guesses, and the remaining players are split evenly among Random Guessing, Minimum Guessing, Mean Guessing, and Quantile Guessing strategies. The relationship between error and honest player proportion is linear.

An approximate answer to the right problem is worth a good deal more than an exact answer to an approximate problem.

John Tukey

4

Interview Question Generation

4.1 Introduction

Applications of artificial intelligence are often reactive: the human poses a question, and the machine attempts an answer. The beginning of a Luna Game may be confounding for such a responsive machine, since the Interview Phase requires each player to provide questions without any prior input. As a test for AI, the Luna Rating System does not demand that a machine be able to ask "good" questions; Smarts Ratings are presumably only a function of responses. Researchers who wish to use LRS purely as a test may choose a simple interviewing scheme, such as selection from a fixed set of questions. However, for those who want to take a more holistically AI approach to the Luna Game, the Interview Phase offers

an application of the intriguing Question Generation (QG) task.

Question Generation demands more than the creation of arbitrary linguistically valid questions. The defining feature of QG is that the questions produced must pertain to a given topic, which is often formalized as a collection of natural language samples. A slightly stronger requirement, sometimes included to narrow the scope of the task, is that the generated questions must be directly answered within the provided text. These text samples may be relatively short, like a section of a Wikipedia article, or long, like the full Wikipedia corpus. They may be factual, like the encyclopedia examples, or fictional, like a short story derived from a simulated world. A method for QG must accept these samples as input and produce questions related to the text. The metric used to assess this output may also vary. A straightforward approach is to count the number of unique questions generated from the input text. This metric avoids the difficulty of defining question quality but subsequently provides no guarantees about the question's content. An alternative method relies on manual human input to determine the quality of each question. While potentially more meaningful, this metric can be costly, subjective, and vulnerable to human error. The challenge of defining a metric that is both semantically meaningful and efficiently evaluated is arguably the greatest barrier between QG and mainstream NLP research.

In this chapter, I present a novel approach to QG that can provide quality questions without human labeling. The method extends the overgenerating transformations and ranking paradigm, which begins with the generation of a vast surplus of questions, and then refines the crop by ranking the best questions and selecting from the top. In this context, the limiting factor for progress is the ranking part of the problem; methods for generating semantics-agnostic questions from templates are abundant. Previous work on ranking

has relied on human-labelled questions and thus has been limited by the size of available annotated datasets. Taking inspiration from C&W and their principle of coherence, I provide a simple method for converting a dataset of unlabeled questions into an augmented dataset of real coherent questions and fake incoherent questions. With this dataset in hand, one can proceed with binary classification methods to train a classifier for question coherence. In particular, methods that require large amounts of training data, such as deep neural networks, become available to the QG task for the first time.

4.2 Related Work

Question Generation is often motivated by the prospect of automated intelligent educational tools. A system capable of generating a set of reasonable questions from an academic text would be invaluable for learning and assessment, especially in the era of MOOCs and other online educational resources. Heilman's work, which I describe in more detail below, is primarily motivated by this educational potential. In the specific context of English language learning, Kunichika et al. (2004) create an interactive system that generates questions from a novice textbook and then adaptively presents a series of questions to students based on their previous responses. Other motivations for QG include assisting human questioning, participating in general dialogue, and providing synthetic datasets for the related task of Question Answering (see Chapter 5). Piwek et al. (2008) make a broad case for QG as a task of interest for AI and Computational Linguistics, advocating an "open-minded approach... [towards] a new and hopefully soon burgeoning area of research."

If surface-level questions suffice for an application, and questions need not be diverse in style, there are several simple QG options available. One coarse approach is the Cloze

procedure, in which words from the input text are randomly replaced with blanks for the responder to complete. For example, the input text "George Washington was born in Virginia" might generate the question "George Washington was born in _____". The next level of sophistication involves applying templates to text in search of specific syntax patterns, and then applying one of several predefined transformations accordingly. For example, the previous input text might be matched and transformed to the question "Where was George Washington born?" or "Who was born in Virginia?" A variant of this template-based strategy is employed by ELIZA, the therapist-like chatterbot developed in 1966 by Weizenbaum. More recent work by Ali et al. (2010) relies on sentence classification and syntax parsing before applying transformations to generate questions. Wang et al. (2008) use a similar transformation-based procedure in the domain of medical texts. While template methods produce syntactically correct questions that are more appropriate for applications like dialogue, it is not clear that they produce deeper semantics than the simpler Cloze procedure.

In 2010, a Question Generation Shared Task and Evaluation Challenge (QGSTEC) was announced, leading to an uptick of interest on the problem and a set of more sophisticated methods. QGSTEC included two tasks: QG from sentences, and QG from paragraphs. In both cases, the questions generated by the five contestants were evaluated manually on the basis of relevance, question type, correctness, ambiguity, and variety. This human-dependent evaluation unfortunately restricts the long term impact of QGSTEC, since new methods cannot be reliably compared with the baselines set in the contest. Nonetheless, the challenge succeeded in bolstering work on QGSTEC. Ali et al. entered a straightforward syntax-based method using Part of Speech tagging and Named Entity analysis. Pal et

al. took a similar approach with an additional focus on clausal boundaries. Yao & Zhang broached semantics directly by converting inputs into Minimal Recursion Semantics representations and then applying transformations into questions. Varga & Ha modified an existing multiple choice question generator to fit the specification of the task. Mannem et al., the only team to attempt the paragraph to question subtask, used predicate argument structures to generate many more questions than required, and then ranked the candidate questions to select the final output.

The overgenerating transformations and ranking of Mannem et al. has been championed by Heilman, who completed his dissertation on the subject in 2011. The paradigm is a promising approach to QG, since it separates the task of generating all syntactically possible questions from the task of selecting semantically valid and useful questions. In Heilman's work, the overgeneration step is accomplished through syntax pattern matching and manually encoded transformation rule applications. Ranking is then performed using a variety natural language features. The ranking step is shown to double the acceptability of generated questions, as defined by human judges. The dataset tested is larger than that of QG-STEC, but still strictly limited by the dependency on human input.

4.3 Question Generation as Binary Classification

4.4 Experiments

4.5 Results

4.6 Discussion

We assume that a computer cannot deal reasonably with language unless it can understand the subject it is discussing.

Terry Winograd

5

Learning to Respond

5.1 Introduction

5.1.1 The Response Problem

The true test for intelligence lies in the Response phase of the Luna Game. Players must be prepared to respond to prompts of virtually any variety. The prompts will likely be in natural language. They may be related to one another or not. The goal of the players to produce Responses that are convincingly intelligent, thereby inducing the opponent to provide a high Guess of the player's Smarts Rating. I call this general problem The Response Problem.

Definitions and Notation

The definitions and notation here were introduced in Chapter 4. Let P be the set of all players in the LRS. Let SR $\subseteq \mathbb{R}^+$ be the domain of Smarts Ratings. I use Σ^* to denote the set of all possible alphanumeric strings. Each player $p \in P$ is equipped with a Response Function $r_p : \Sigma^* \to \Sigma^*$ that gives the player's Response to any prompt. Each player is also equipped with a fixed finite set of prompts or questions, written $Q_p \subset \Sigma^*$. These questions are used by the player during the Interview Phase. A Guess Function, written as $g_p : \mathcal{R}_Q \to$ SR, gives the player's Guess of an opponent's Smarts Rating based on the opponent's Responses to a set of questions Q.

The Response Problem

The Response Problem: Learn a Response Function $r : \Sigma^* \to \Sigma^*$ that maximizes $E_{p \in P}[g_p(r(Q_p))]$, the expected Guess of an opponent.

Scope of the Problem

The Response Problem is extremely difficult. Given the range of potential questions, the only way to truly solve the problem is to solve all of AI. Consider just a few possible question types:

- Trivia: What is the name of Ron Swanson's first wife in Parks and Recreation?
- Commonsense: Why not wear black while biking at night?
- Creativity: Write an original poem about ducks.
- Opinion: Is it really better to have loved and lost?

- Sensorimotor: Turn left, left, left, and then left. Where are you now?
- Vision: What does ;) look like?
- Emotional Intelligence: John gives Jane a large holiday present. Jane does not give John one. How might John feel?
- Analysis: A journalist writes an article subtitled "First Sign of President's Radical Shift to the Right?" What biases might the journalist have?
- Philosophy: What does it mean to be virtuous?
- Open Problems: Does P = NP?

In practice, the boundaries of the Response Problem will emerge from the collective behavior of LRS players. The question types that carry the most weight will be those repeatedly asked by large factions of players. As players continue to adapt and refine their questions, any overfitted solutions, i.e. Response strategies based on only a shallow understanding of the particular questions in the system, will quickly be exposed as fraudulent. Thus the Response Problem strongly incentivizes machine player developers to focus on general AI.

5.1.2 Related Work

The Response Problem bears immediate resemblance to a subfield of Natural Language Processing known as Question Answering (Q&A). Q&A is sometimes defined as the general problem of automatically responding to any question posed in natural language (e.g.[??]), which suggests that Q&A and the Response Problem are identical. However, the more common definition distinguishes Q&A as the task of retrieving relevant information from

a natural language corpus in response to a factual question ([?][?][?][?][?]). The practical scope of Q&A is much more narrow than that of the Response Problem; in contrast with the ten question type examples given in section 1.1.3, Q&A deals almost exclusively with Trivia (see[?] for an exception). Most existing Q&A systems can only hope to address a small subset of the Response Problem.

Nonetheless, researchers of the Response Problem can take inspiration and guidance from the substantial work done on Q&A. Perhaps most enlightening is the manner in which Q&A is decomposed into subproblems. The 2003 review by Burger et al. defines a roadmap for Q&A that has often been followed by subsequent research[?][?]. The subtasks include[*] Question Classes, Question Processing, Context, Data Sources, Answer Extraction, and Answer Formulation. These six subtasks are sufficiently general that they may transfer to the Response Problem. Written two years later, the review by Andrenucci & Sneiders distinguishes three main approaches to Q&A: a Natural Language approach, an Information Retrieval approach, and a question template approach[?]. The three differ mainly in emphasis; the Natural Language approach focuses on the semantics of the question and answer, the Information Retrieval approach is chiefly concerned with the representation and access of information, and the question template approach deals almost exclusively with pattern matching questions and annotated information. Each of these approaches could in theory be generalized for the Response Problem, but the Natural Language approach, with its emphasis on semantics, is likely most appropriate.

Figure 5.1, reproduced from[?], provides an example of how an end-to-end system for

[*] There are twelve subtasks defined in[?], but the most notable are the first six, as the remaining six — Real-Time Q&A, Multilingual Q&A, Interactive Q&A, Advanced Q&A, User Profiling, and Collaborative Q&A — go beyond the basic objective of Q&A.

Q&A can be separated into modules. The first section of the system is dedicated to linguistic processing. At the lowest level, syntax rules and a lexicon are used to parse the natural language input a standard format that a semantic interpreter can process. Semantic rules, such as first-order logic, and a world model, which encodes relationships and constraints for elements of the semantic domain, are integrated to produce a logical query into a database. After the database processes the query, a final module converts the database output into a human-readable response. This architecture offers an example of how a system for the Response Problem could be structured. The syntax rules and lexicon may be already be ready for more general use. With a sufficiently advanced collection of semantic rules, world model, and database, the system could perform quite well. Of course, the level of advancement in each of these areas required for the Response Problem seems far beyond the reach of current AI.

A system for Q&A offers a solution to the Trivia-type questions in the Response Problem. Several other possible question types have their own corresponding areas of research. Commonsense reasoning[?], creativity[?], spatial reasoning[?], vision[?], and emotional intelligence[?] have all been formalized and are actively studied[†] as subproblems in AI. Questions of opinion, and other questions that rely on a human identity, may be addressed with research in Dialogue, e.g. through a chatbot[?]. One possible approach to the Response Problem is to detect the type of a question and then parcel it out to the appropriate sub solution. This method is in line with the "bottom-up" approach to AI. The analogous method, the development of one general mechanism for answering all questions in the Response Problem, aligns with the "top-down" approach. To gain insight into the current state of AI,

[†]In fact, each of the preceding citations came from 2015!

each paradigm should be used for the Response Problem; the highest Smarts Rating of a single mechanism machine and the highest Smarts Rating of a machine that uses a combination of subproblems will both serve as illuminating indicators of progress.

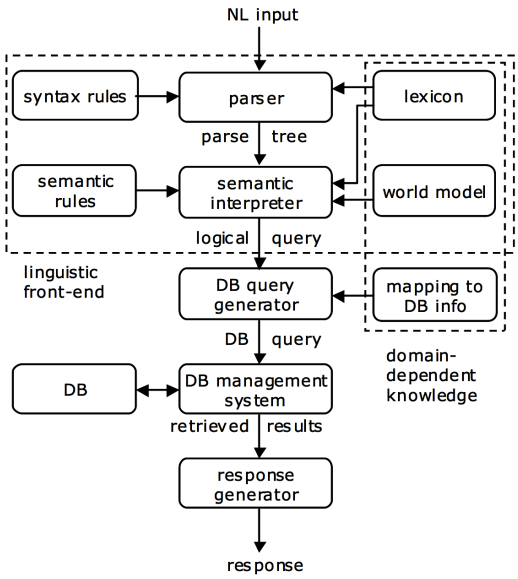

Figure 5.1: Architecture of a typical Natural Language system for Question Answering, from[?]. The structure could be used in full or in part as a template for approaching the Response Problem.

DIALOGUE

EACH TYPE OF QUESTION HAS BEEN STUDIED IN ITS OWN RIGHT

5.2 Methods

5.2.1 Approaching the Response Problem

TODO

5.2.2 Example Solution

TODO

5.2.3 Datasets

TODO

5.3 Results

TODO

5.4 Discussion

TODO

Searle may only be behaving as if he were thinking deeply about these matters. But, even though I disagree with him, his simulation is pretty good, so I'm willing to credit him with real thought.

Nils Nilsson

6

Luna Rating Prediction

6.1 Introduction

6.1.1 The Luna Rating Prediction Problem

In this chapter, I consider Luna Rating Prediction (LRP): the problem of predicting a respondent's Smarts Rating from their Responses to a fixed set of questions. I submit that LRP is not only critical for an AI player of the Luna Game, but also interesting more broadly as a machine learning question that has not been previously examined. Thus I treat the problem generally and offer several avenues for approaching the problem in practice.

Definitions and Notation

Let P be the set of all players in the LRS. Let SR $\subseteq \mathbb{R}^+$ be the domain of Smarts Ratings. Recall that $S : P \to SR$ gives the Smarts Rating of a player. I use Σ^* to denote the set of all possible alphanumeric strings.

Definition 5. Response Function

Each player $p \in P$ is equipped with a Response Function $r_p : \Sigma^* \to \Sigma^*$ that gives the player's Response to any prompt.

Definition 6. Questions

Each player $p \in P$ is also equipped with a fixed finite set of prompts or questions, written $Q_p \subset \Sigma^*$. These questions are used by the player during the Interview Phase.

For a finite set of questions $Q \subset \Sigma^*$, let $\mathcal{R}_Q = \{r_p(Q) | p \in \mathcal{P}\}$ be all possible Responses.

Definition 7. Guess Function

A Guess Function, written as $g : \mathcal{R}_Q \to SR$, gives a player's Guess of an opponent's Smarts Rating based on the opponent's Responses to a set of questions Q.

The LRP Problem

The LRP Problem: Let $Q \subset \Sigma^*$ be a finite set of questions. Learn a Guess Function g that minimizes expected error $E_{p \in P}[|g(r_p(Q)) - S(p)|]$.

LRP avoids some of the difficulties of natural language understanding, but also introduces a host of new challenges. For example, it is theoretically possible to achieve very good results on LRP without a complete understanding of the natural language responses being analyzed. On the other hand, a complete understanding of the responses is not sufficient

to solve LRP, since ratings must still be predicted. Machines may excel at mapping representations of responses to ratings, while humans may excel at representing the responses. A winner of the Luna Game must excel at both tasks.

6.1.2 Related Work

LRP can be viewed as a relaxed version of a problem that has been previously considered: Automatic Short Answer Grading (ASAG)[????]. Motivated by standardized testing, ASAG attempts to grade student responses to a fixed set of questions. While LRP is concerned only with the cumulative rating of all responses for a particular respondent, ASAG requires scores for each question. If player ratings are assumed to be completely determined based on their responses to the given set of questions, then LRP reduces to ASAG. In this scenario, a solution to ASAG would imply a solution to LRP, but LRP may be solved without solving ASAG. For example, if one question is much more predictive of a respondent's overall rating than other questions, LRP would be able to leverage this correspondence, while ASAG would be forced to grade all questions, each contributing equally to overall error.

Previous work on ASAG provides a baseline for work on LRP. A recent review by Burrows et al. summarizes advances in ASAG, drawing upon over 80 papers with 35 distinct systems[?]. These systems differ in their natural language processing techniques, model building, grading models, and effectiveness. Each of these systems could be potentially adapted for LRP, offering several possible avenues for future research. Here I focus on the state-of-the-art system developed by Education Testing Services[?]. This system serves both as a representative example of ASAG systems and as a starting point for LRP.

The latest ASAG system created by ETS performs logistic regression to classify responses as correct or incorrect based on two categories of features: word and character n-gram features, and text similarly features between response and answer key or response and other responses. The system also groups questions into different problem domains and then uses a "domain adaptation" technique, which introduces three separate copies of each feature for generic, domain-specific, and question-specific weighting. This straightforward logistic regression approach outperformed all other existing techniques on the two tested datasets. The general approach of defining a wide variety of natural language features and training a regression model is common among competing systems.

6.2 Methods

6.2.1 Datasets

I use two datasets to evaluate various approaches to LRP. Both datasets are comprised of responses to standardized tests with short answer questions. The first dataset was introduced by Basu et al. (2013) in their work on computer-assisted grading[?] . They created the dataset by posing 20 questions from the 2012 United States Citizenship Exam to workers on Amazon Mechnical Turk, collecting a total of 698 responses. 10 of the questions were selected for grading by three human judges, who marked each answer as each correct or incorrect. I refer to this dataset as the "Powergrading" dataset, derived from the name of the system that Basu et al. developed.

The second dataset I use for LRP was introduced by Mohler et al. (2011) in their work on ASAG[?] . These data consist of responses to short answer problem set and exam questions in a Data Structures course at the University of North Texas. In total, the dataset contains

24 complete responses to 87 questions. Responses are graded by two human judges on a scale from 0 to 5, and these two grades are averaged together per question. I refer to this dataset as the "Mohler 11" dataset. Responses in this dataset are typically longer and more involved than responses in the Powergrading dataset. This level of natural language, combined with the relatively small number of samples, makes the Mohler dataset significantly more challenging for LRP than the Powergrading dataset.

Neither of these datasets are immediately in the format required for LRP. In particular, each question is graded individually, as opposed to each respondent possessing an overall rating. To convert to LRP, I simply add all grades for each respondent and discard the per-question grades. Next I normalize these overall respondent ratings so that all ratings are on a scale of 0 to 1. The final rating preprocessing step is splitting into training and test sets at a ratio of 4 to 1 respectively. To preprocess text responses, I remove all punctuation, stop words, and convert all remaining characters to lowercase.

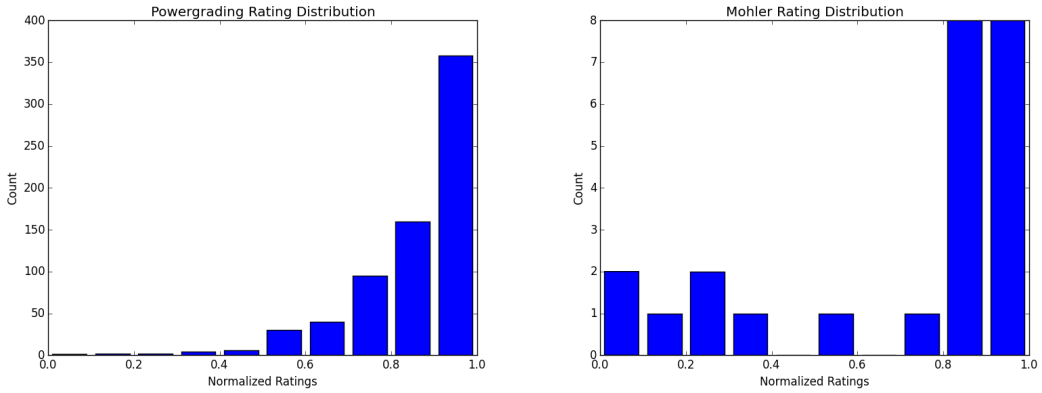

Figure 6.1: Normalized ratings for two datasets used for evaluating approaches to the Luna Prediction Rating problem. The Powergrading dataset, comprised of responses to a subset of the 2012 United States Citizenship Exam, was introduced by Basu et al. (2013)[?]. The Mohler data, comprised of responses to problem set and exam questions in a University of North Texas Data Structures course, was introduced by Mohler et al. (2011)[?]. Ratings were computed by summing individual question grades and normalizing.

6.2.2 Features

Beyond stating LRP as a new problem for the machine learning community, the main contribution of this chapter is a thorough exploration of approaches to the problem. Following the recent work done on ASAG, our strategy is to associate several features with each set of responses, which can be subsequently used for regression. I divide the feature definitions into two categories: question-based features and response-based features. The former takes into account expected responses, while the latter focuses more on data provided by previous rated respondents. I then experiment with several types of regression to combine the features for an optimal model.

Question-Based Features

I assume now that Luna Game players have a set of expected answers for every question, which I refer to as an "answer key". The first time that the Luna Game is played with these questions, the player's only option is to compare the opponent's responses with the answer key. The closer a response is to the expected response, the higher the player's guess of their rating will likely be. Framed in this way, the problem becomes one of measuring the similarity between two samples of natural language.

Measuring semantic similarity between two samples of natural language can be a difficult task. If the two sentences differ only in word order, or only in a few words, then simple approaches can reliably detect similarity. By considering synonyms of the words in both sentences, the similarity measure may be improved. However, in many cases, it is impossible to accurately capture similarity without a deeper semantic understanding of the two sentences. One common approach for representing semantic similarities is to infer vector represen-

tations for words in the vocabulary of the training corpus, and then to calculate similarity based on some geometric measure, such as cosine similarity. The problem of inferring these vectors, referred to as neural embeddings, is an active area of research.

In the list below, I present all question-based features considered. In all cases, if there are multiple answers in the answer key, I take the maximum value over all possible values. All question-based features result in a vector whose length is equal to the number of questions in the dataset.

1. Binary Word Overlap (BWO): The i^{th} entry of this feature vector is 1 if any words in the i^{th} response appear in the answer key and 0 otherwise.

2. Fraction Word Overlap (FWO): The i^{th} entry of this feature vector is the number of overlapping words in the i^{th} response with the corresponding answer key, divided by the number of words in either the response or the answer key.

3. Character Edit Distance (CED): The i^{th} entry of this feature vector corresponds to the edit distance (Levenshtein distance) between the i^{th} response and the corresponding answer key at the level of characters. The value is 1.0 minus the fraction of the edit distance and the length of the larger string.

4. Word Edit Distance (WED): The i^{th} entry of this feature vector corresponds to the edit distance (Levenshtein distance) between the i^{th} response and the corresponding answer key at the level of words. For example, the WED between "hello, world" and "goodbye, moon" is 2. The value is 1.0 minus the fraction of the edit distance and the greater number of words in either string.

5. Word2Vec Cosine Similarity (WCS): First I train a neural embedding of words using the answer key as a corpus. Then the i^{th} response and the corresponding answer key are compared, and the i^{th} entry of the feature vector is the maximum cosine similarity between the two. I use the library provided by Gensim here[?] .

6. Semantic Nets (Li et al.) (SNL): For a more sophisticated approach to short sentence similarity, I refer to the work of Li et al. (2006)[?] . They offer a comprehensive metric that takes into account lexical, semantic, and syntactical information about the two sentences being compared. Their system is trained on a corpora of English sentences separate from the training data. I use an existing Python implementation of SNL that uses the NLTK English corpora[?] . The output of their similarity measure is a value between 0 and 1. Thus the i^{th} entry of this feature vector contains the SNL similarity measure between the i^{th} response and the corresponding answer key. One notable downside to this algorithm is that it takes significantly more time (up to 20x) than any other feature described in this chapter.

One inherent challenge in the design of question-based features is the limited size of the answer key. At best, an answer key will document several responses that are considered completely correct. Unfortunately, for open-ended questions, the number of possible completely correct responses can be unbounded. Writing all possible correct responses becomes impractical and many responses may be incorrectly processed due to superficial differences with the answer key. Here I suggest two possible approaches to mitigating this problem. First, the answer key may be multiplied by applying various transformations, such as considering the synonyms of words in the original answer key or performing rearrangements in the syntax of the sentence. Second, as the training set increases, one may choose to include

the answers of highly rated players in the answer key. In both cases, one should be sure that growing the answer key does not add prohibitive computational cost.

For future work on developing question-based features for LRP, it may be worthwhile to adapt methods for two related problems: paraphrase detection and textual entailment. Paraphrase detection assesses whether one sample is a paraphrase of the other. Textual entailment is the task of detecting whether one sample of natural language logically implies another sample. For our purposes, I might like to give high ratings to players whose responses imply the answer key, even if they are not exactly the same. Both of these problems are active areas of research in natural language processing and new results will likely be relevant to LRP as well.

Response-Based Features

As a player asks the same questions over several rounds, learning the opponent's true rating at the end of each round, the player's rating model should be updated to take into account the new data. For example, if a new set of responses is very similar to those of a previous opponent, the player's guess of the new opponent's rating should be close to the known rating of the previous opponent. Here I ignore the answer key and consider features that may be extracted solely from responses. In the list below, I present all response-based features tested.

1. Bag of Words (BOW): Our simplest response-based feature is the standard natural language processing technique of Bag of Words. Given all the training data, I begin by creating a shared vocabulary list of all words that appear in any responses. I then associate a binary vector with each response in the training data, where entry i is 1 if

the response contains the i^{th} word in the vocabulary and 0 otherwise.

2. Bigrams (BIG): This feature is the same as BOW, except rather than building a vocabulary of single words, I build a vocabulary of two adjacent words.

3. Bag of Synsets (SYN): This feature is similar to BOW. I create a thesaurus for every word in the generated vocabulary. I use the WordNet database for finding synonyms[?] . Then I associate a vector for each response in the training data, where entry i is 1 if the response contains any synonyms (including the word itself) of the i^{th} word in the vocabulary.

4. Nearest Neighbor via Overlap (NNO): This feature is based on the premise that similar responses will result in similar contributions to the respondents' ratings. For a given response, the most similar response in the training set is identified according to the number of overlapping words. The rating of the corresponding respondent is put into the feature vector, so that the final output is a vector containing the ratings of the most similar respondents per question.

5. Nearest Neighbor via Character Edit Distance (NNC): This feature is the same as NNO, except similarity is defined by character edit distance rather than overlap.

6. Nearest Neighbor via Word Edit Distance (NNW): This feature is the same as NNO, except similarity is defined by word edit distance rather than overlap.

6.2.3 Regression

With features defined, the task then takes the standard form of a regression problem. For simplicity, I assume here that the rating is a linear combination of the features. I then con-

sider two models of regression. The first is Ordinary Least Squares Linear Regression, which minimizes the sum of squares between predicted and actual ratings. I then run Lasso Regression ($\alpha = 0.1$), which biases towards simpler models by penalizing the number of non-negative coefficients in favor of sparse solutions. In both cases, I use the implementations in the Scikit-Learn library for machine learning[7].

6.3 Results

I ran OLS Linear Regressions and Lasso Regressions on each of the 12 features for 100 trials. In addition, I included a combination of all 12 features (COM), and two controls. The first control predicts a random rating for a given set of responses (RAN). The second control, which I refer to as the "constant best guess", always predicts the average of all previously seen ratings (CBG). To consider a feature useful, it should at least outperform both of these controls on average. I define performance in terms of the Root Mean Square Error between predicted ratings and actual ratings in the test set. The results of these experiments are depicted in Figure 2. Lasso Regression yielded no meaningful differences from OLS, so I depict only the latter.

The relative performances of the approaches seem to be consistent between the two datasets tested. As expected, the Mohler 11 dataset proves far more difficult, likely due to the very limited size of the training set, the large number of responses per respondent, and the longer length of each response. Also unsurprising is the result that for OLS Linear Regression, a combination of all features outperforms any single feature. The combination does not perform as comparably well in Lasso Regression, presumably because the number of features actually included is small due to the penalty term. In both cases, any margin of improve-

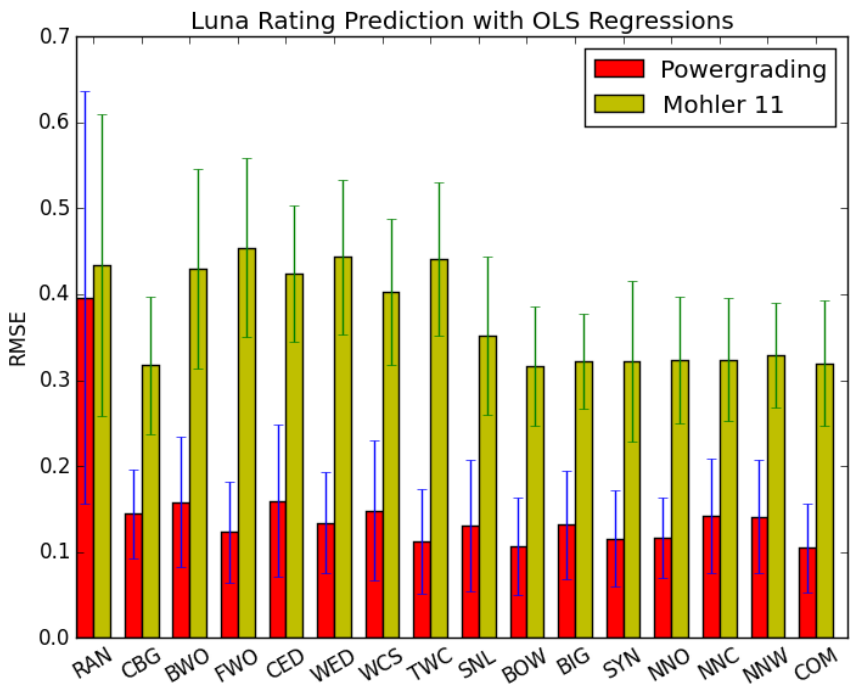

Figure 6.2: Fifteen approaches to the Luna Prediction Rating problem on two datasets using Ordinary Least Squares Linear Regression. From the left, the first two approaches represent baseline controls; the following six approaches are each OLS Linear Regressions on single question-based features; the following six are also OLS Linear Regressions, but on single response-based features instead; the last is an OLS Linear Regression that includes all 12 previous features. The Root Mean Square Error averaged over 100 trials is plotted for each approach. Refer to the text for the feature abbreviations.

ment offered by a combination of all features is slim. Simple response-based features such as Bag of Words or Nearest Neighbor via Overlap are competitive with the combination. In practice, it may be preferable to use one or both of these due to the savings in computational resources. Among the question-based features, Binary Word Overlap fares the best, but cannot outperform the Constant Best Guess baseline. In future work, the question-based features could likely be improved by augmenting the answer keys as discussed above.

6.4 Discussion

In this chapter, I defined the Luna Rating Prediction problem of guessing a respondent's rating based on previous observations of responses and ratings. I discussed the closely related problem of Automatic Short Answer Grading and demonstrated how datasets and methods for ASAG can be adapted for LRP. I then presented initial work on LRP, which centered around several features of response sets that I use to learn a model for predicting ratings. These features were presented in two categories: question-based features, which rely only on a preset answer key, and response-based features, which ignore the answer key but take advantage of seen responses from previous rated respondents. I combined these features using traditional regression techniques to arrive at a comprehensive system for LRP.

For a player of the Luna Game, the ability to solve LRP is critical. The outcome of a Game hinges on the difference between predicted ratings and actual ratings. A human player may take advantage of the machine learning approaches to LRP offered here, but more likely the human will learn to rate well after several rounds of play without any formalisms or explicit machine learning. This human ability is based on an understanding of natural language, a deep knowledge base, and the relative ease with which humans judge one another in everyday scenarios. As machine players build an understanding of natural language and a knowledge base from answering questions in the Luna Game, these advances should be transferrable to LRP as well. A truly intelligent player of the Luna Game is unlikely to consider LRP and question answering as completely separate problems, but rather as manifestations of one general underlying problem.

Throughout this chapter, I sought to treat LRP as a general problem beyond the scope

of the Luna Game, since I expect work on the problem to have several applications. Any situation in which a judge wishes to automatically evaluate a respondent can be thought of as an instance of LRP. The quantity being evaluated may be completely known to the respondent, as is the case in the Luna Game. However, in some applications, the quantity may be only partially known or unknown. For example, consider the case of a government official wishing to estimate the probability that a convict will repeat an offense based on questionnaire responses, or the case of an health insurance company wishing the estimate the amount of money that a customer will require for medical care. These real life applications combined with the motivation provided by the Luna Game make LRP a problem worthy of further study.

I rarely find it useful to distinguish between theory and practice; their interplay is already profound and will only increase as the systems and problems we consider grow more complex.

Michael I. Jordan

7

A Web Implementation of the Luna Rating System

7.1 Introduction

The Luna Rating System is designed to be a practical test for machine intelligence. It may offer some utility as a thought experiment, but its primary value can only be realized in practice. In this chapter, I describe the design and launch of the first web-based implementation of LRS. The application is meant to serve as a comprehensive proof-of-concept. For the application to fulfill its purpose, it must first be capable of recruiting and handling hun-

dreds of human players. The design should then persuade players to play several games, so as to refine their Smarts Ratings. Finally, the behavior of players should indicate a collective understanding of the Luna Game, and evidence should suggest their aspirations towards honest play.

The baseline success of the LRS implementation can be easily measured through traffic statistics: how many players play, and how long do they typically stay? Such statistics comprise the first section of results in this chapter. In addition to these standard metrics, the questions and responses traded by the players can be assessed qualitatively. I provide a representative sample of questions and discuss common themes from the full dataset. The final source of results is derived from human players' interactions with simple standardized machine players. I argue that the Guesses that human players make of the machine players should converge over time as a consensus on the semantics of Smarts Ratings is reached. The introduction of machine players also provides a demonstration for AI researchers who wish to introduce their own machines into the system.

7.2 Building the Web Interface

7.2.1 Design of the Web Interface

My implementation of the Luna Rating System is built on the open-source software stack consisting of MongoDB, Express.js, Angular.js, and Node.js, which is collectively referred to as the MEAN stack[7] . The latter three technologies are all written in JavaScript and MongoDB is a NoSQL database. At a very high level, MongoDB stores all player and game data, Angular.js controls and displays the client side, Express.js forms the foundation of the server side, and Node.js runs the code written on top of that foundation. Notable libraries used

include Mongoose.js, a Node.js library for interfacing with MongoDB; Passport.js, a Node.js library which provides middleware for user authentication; and Angular-UI, an Angular.js library that simplifies routing, i.e. navigation between different states of the website. To protect user anonymity, usernames are hashed on the client side using MD5. The front end of the website, i.e. the style, formatting, and organization of content, was modified from the "Material Admin" LESS/Angular.js template, for which I purchased a single application license[?]. The site was developed locally and then hosted on EC2 by Amazon Web Services under the domain name "luna-game.com".

The skeletal framework of the website was first paper-prototyped and then iterated upon[?]. The centerpiece of the website is the ongoing Luna Games, and a secondary component allows users to review their performance in previous Games. A single Luna Game seamlessly transitions from one phase to the next, while retaining cohesion as a whole Game. All opportunities for latency — between phases of a Game or between Games — are reduced as much as possible to encourage repeated play. To increase the probability that a first-time user explores the website beyond the landing page, the website has an option to play as a "guest." The only functional difference between a user that signs up and a guest is that signed up users may log out and log back in, while a guest account expires once the 60-day token is removed from the browser or the user signs out. User experience was further considered in the context of the results presented in Chapter 3 of this thesis, which demonstrates that all players must have a clear understanding of their goals during a Luna Game. While users should be able to start playing Luna Games as quickly as possible, they should first be fully aware of the Game's structure and their goals.

Weighing these considerations, the core features of the website are divided into 10 pri-

mary states and 4 secondary states. The secondary states include static pages — the contact page and two extended information pages — and one profile page that allows a user to review statistics related to her historical performance. The primary states are presented in Figure 7.1. They include four states corresponding to the phases of a Luna game, one home state, which contains games as nested states, a landing page, a guest state, a sign up state, a login state, and an information state. Note that first-time users must provide consent and review information about the structure of Luna Games before they are able to play. The eager first-time user is able to reach a new game with only three clicks from the home page, and a returning user can reach games with two clicks and a form submission. This simple state architecture ensures that users are informed and playing as soon as possible. Screen shots of a typical user experience are depicted in Figure 7.2.

7.2.2 An API for Machine Players

To allow AI researchers to enter machines as players on the Luna Rating System website, I implemented a RESTful API. Machine players must first register separately from human players to receive an API token. This process is currently done by hand, since the system is not yet protected against malicious machine players (e.g. there is no limit on the number of API calls that one player may make). With an API token obtained, the experience of machine players is very similar to that of humans. The machine may request to start a new game, provide questions during the interview phase of an ongoing game, receive questions and provide answers during the response phase, provide a guess during the guess phase, and receive updates of its own Smarts Rating and total wins. Notably, the guess of a machine is not factored into the Smarts Rating of its opponent. A Python template, which may be

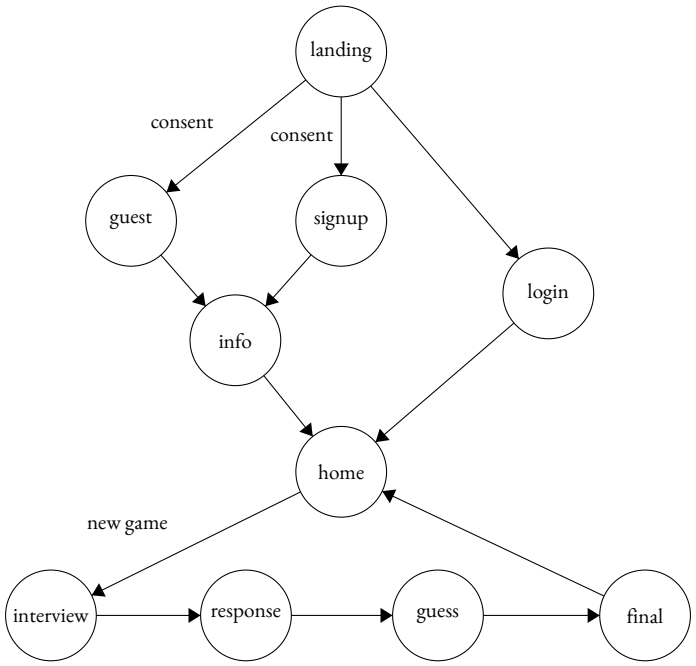

Figure 7.1: Schematic diagram depicting the primary states of the Luna Rating System web interface. Nodes represent website states and edges indicate how users typically navigate between states. A first-time user arrives at the *landing* state. Upon providing consent, the user may play as a *guest* or *sign up*, both which lead to the *info* state. The info state explains the Luna Game and then directs users to the *home* state. The user can then create a new Luna Game, launching the *interview* state. As the user progresses through the Luna Game, she transitions from interview to *response*, to *guess*, and to *final*. They then return to the home state to start another game. Returning signed up users may *login* from the landing state to reach the home state.

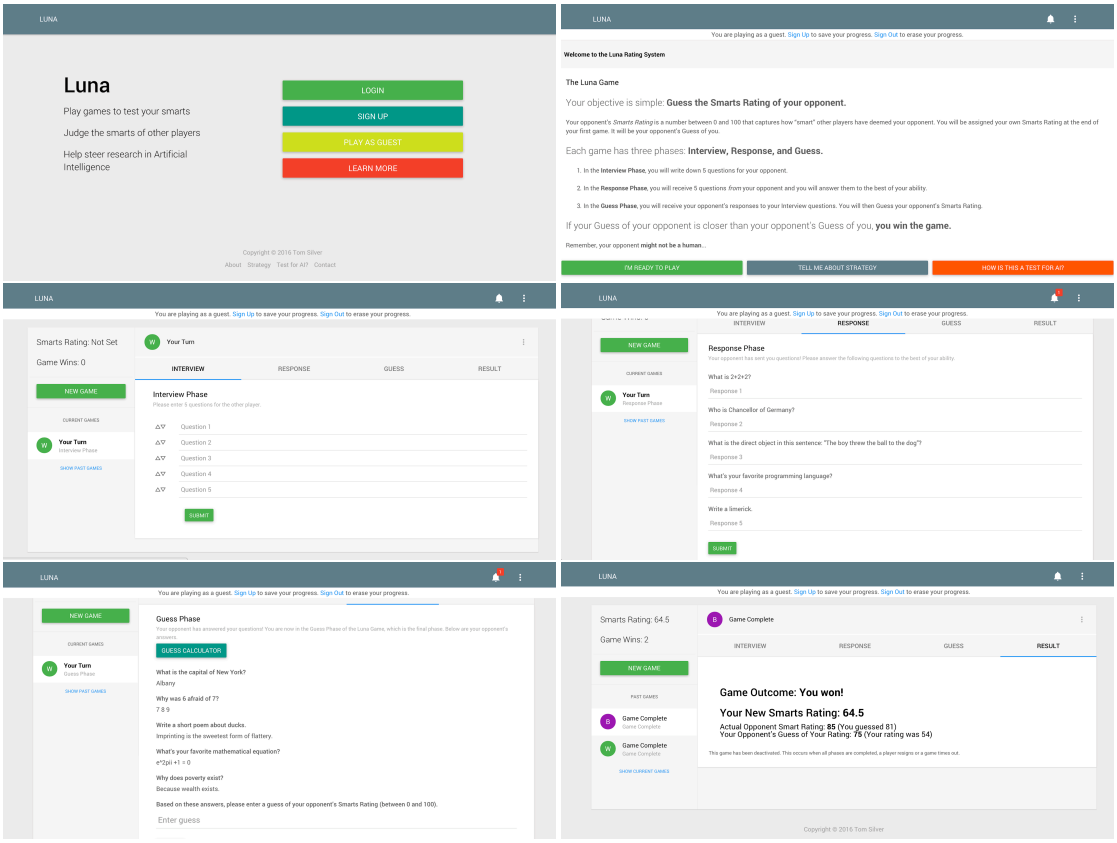

Figure 7.2: Exemplary screenshots of the Luna website. Clockwise from top left: the landing page, the information page for first-time users, interview phase, response phase, guess phase, and final phase.

used directly or as a blueprint for other languages, is provided for researchers.

7.2.3 Launching the Website

MARKETING

7.3 Results

TRAFFIC - FORTHCOMING

7.4 Analysis

ANALYSIS OF USER BEHAVIOR

7.5 Discussion

TWEAKS FOR FUTURE WEBSITE

References