Pipelines for Computational Social Science Experiments and Model Building

Vanessa I. Cedeno

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Madhav V. Marathe, Chair
Chris J. Kuhlman, Co-Chair
Anil K. Vullikanti
Naren Ramakrishnan
Noshir S. Contractor
Joshua M. Epstein

May 16, 2019
Blacksburg, Virginia

Keywords: Online Social Experiments, Agent Based Models, Abductive Loop, Collective Identity, Pipelines

Pipelines for Computational Social Science Experiments and Model Building

Vanessa I. Cedeno

(ABSTRACT)

There has been significant growth in online social science experiments in order to understand behavior at-scale, with finer-grained data collection. Considerable work is required to perform data analytics for custom experiments. In this dissertation, we design and build composable and extensible automated software pipelines for evaluating social phenomena through iterative experiments and modeling. To reason about experiments and models, we design a formal data model. This combined approach of experiments and models has been done in some studies without automation, or purely conceptually.

We are motivated by a particular social behavior, namely collective identity (CI). Group or CI is an individual's cognitive, moral, and emotional connection with a broader community, category, practice, or institution. Extensive experimental research shows that CI influences human decision-making. Because of this, there is interest in modeling situations that promote the creation of CI in order to learn more from the process and to predict human behavior in real life situations.

One of our goals in this dissertation is to understand whether a cooperative anagram game can produce CI within a group. With all of the experimental work on anagram games, it is surprising that very little work has been done in modeling these games. Additionally, we use abduction as an inference approach that uses data and observations to identify plausibly (and preferably, best) explanations for phenomena. Abduction has broad application in robotics, genetics, automated systems, and image understanding, but have largely been devoid of human behavior. We use these pipelines to understand intra-group cooperation and its effect on fostering CI. We devise and execute an iterative abductive analysis process that is driven by the social sciences.

In a group anagrams web-based networked game setting, we formalize an abductive loop, implement it computationally, and exercise it; we build and evaluate three agent-based models (ABMs) through a set of composable and extensible pipelines; we also analyze experimental data and develop mechanistic and data-driven models of human reasoning to predict detailed game player action. The agreement between model predictions and experimental data indicate that our models can explain behavior and provide novel experimental insights into CI.

This material is based on work partially supported by DARPA Cooperative Agreement D17AC00003 (NGS2), DTRA CNIMS (Contract HDTRA1-11-D-0016- 0001), NSF DIBBS Grant ACI-1443054, NSF BIG DATA Grant IIS-1633028, NSF Grants DGE-1545362 and IIS-1633363, and ARL Grant W911NF-17-1-0021.

Pipelines for Computational Social Science Experiments and Model Building

Vanessa I. Cedeno

(GENERAL AUDIENCE ABSTRACT)

To understand individual and collective behavior, there has been significant interest in using online systems to carry out social science experiments. Considerable work is required for analyzing the data and to uncover interesting insights. In this dissertation, we design and build automated software pipelines for evaluating social phenomena through iterative experiments and modeling. To reason about experiments and models, we design a formal data model. This combined approach of experiments and models has been done in some studies without automation, or purely conceptually.

We are motivated by a particular social behavior, namely collective identity (CI). Group or CI is an individual's cognitive, moral, and emotional connection with a broader community, category, practice, or institution. Extensive experimental research shows that CI influences human decision-making, so there is interest in modeling situations that promote the creation of CI to learn more from the process and to predict human behavior in real life situations.

One of our goals in this dissertation is to understand whether a cooperative anagram game can produce CI within a group. With all of the experimental work on anagrams games, it is surprising that very little work has been done in modeling these games. In addition, to identify best explanations for phenomena we use abduction. Abduction is an inference approach that uses data and observations. Abduction has broad application in robotics, genetics, automated systems, and image understanding, but have largely been devoid of human behavior.

In a group anagrams web-based networked game setting we do the following. We use these pipelines to understand intra-group cooperation and its effect on fostering CI. We devise and execute an iterative abductive analysis process that is driven by the social sciences. We build and evaluate three agent-based models (ABMs). We analyze experimental data and develop models of human reasoning to predict detailed game player action. We claim our models can explain behavior and provide novel experimental insights into CI, because there is agreement between the model predictions and the experimental data.

Dedication

I dedicate this dissertation to my family for their, inspiration, support, and love.

Acknowledgments

I owe thanks to many people that made this dissertation possible.

First, thanks go to my advisors and committee members. Dr. Madhav Marathe, for believing in my work and for your mentoring that has inspired my career. Dr. Chris Kuhlman, for being so patient and teaching me how to approach problems, your way of looking at things has improved my decision making on many aspects in life. Dr. Anil Vullikanti and Dr. Naren Ramakrishnan for sharing their time and important feedback. Dr. Noshir Contractor and Dr. Joshua Epstein for providing the social science perspective and valuable insights. My committee guidance has helped me made this work possible and I will be always grateful.

I would like to express my deepest appreciation to Xinwei Deng, and Zhihao Hu for their hard work and statistical knowledge. Special thanks to Yihui Ren for his excellent input and cheerful aptitude which was a delight working with.

I thank the Network Dynamics and Simulation Science Laboratory for allowing me to interact with excellent researchers: Jose Cadena, Sandeep Gupta, Saliya Ekanayake, Dustin Machi, Gizem Korkmaz, Abhijin Adiga, S. S. Ravi, and Christopher Barrett. Special thanks to Erin Raymond for her support and direction. I hope that we continue collaborating for many years to come.

I thank my collaborators at the Virginia Tech Discovery Analytics Center: Brian J. Goode, Nathan Self and Parang Saraf. Their work in the group anagram web application is an essential part of this dissertation.

I am also grateful to ESPOL in Guayaquil, Ecuador where I will be working. Their support has helped Javier, Emma and me through this challenging process. I would like to thank my collaborators in ESPOL that guided me with their example: Katherine Chiluiza, Guido Caicedo, Cristina Abad, and Carmen Vaca. Special thanks to ESPOL president Cecilia Paredes who attended my graduation party because her niece Thessa also graduated from Virginia Tech. Thank you Thessa for babysitting Emma when we needed it.

I thank my dad Manuel for teaching me a strong work ethic and guiding me with his example. I thank my mom Loly for always being there for her family. I really appreciate all that you have done for me. I thank my brothers Jaime and Victor for their support and loyalty.

Finally, I am extremely grateful for my family and their support without whom none of this would have ever been possible. My husband Javier and my daughter Emma are my incentive, my biggest supporters, and this is all for them. Blacksburg became our home for almost five years, we are now Hokies for life.

Contents

1 Introduction **1**

1.1 Background, Motivation, and Summary 1

1.2 Research Questions 3

2 Pipelines and their Compositions for Modeling and Analysis of Controlled Online Networked Social Science experiments **5**

2.1 Abstract 5

2.2 Introduction 6

2.2.1 Background and Motivation 6

2.2.2 Technical Challenges 8

2.2.3 Novelty Of Our Work 9

2.2.4 Contributions 10

2.2.5 Additional Perspective 12

2.3 Data Model for Networked Experiments and for Modeling and Simulation . . 13

2.3.1 Formal Data Model 13

2.3.2 Illustrative Instances of Data Model Parameters 15

2.3.3 From Abstract Data Model to Software Specification 15

2.3.4 Data Common Specification 16

2.4 Graph Dynamical System Model: a Formal Framework for NESS Experiments and Agent-Based Models 17

2.4.1 Formal Model 17

2.4.2 Use of GDS in This Work 19

2.4.3 Example GDS and Resulting Dynamics: Threshold Systems 19
2.5 Hierarchical Pipeline Conceptual View 20
2.5.1 Pipeline Compositions 20
2.5.2 Pipelines 21
2.5.3 Functions Within Pipelines 21
2.5.4 Microservices 22
2.6 Formal Pipeline Model 23
2.6.1 Model 23
2.6.2 Execution of a Pipeline 24
2.6.3 Mapping of Model onto the Software System 26
2.7 Pipeline Implementation 26
2.7.1 Pipeline Configuration File 26
2.7.2 Pipelines 27
2.7.3 Functions Within Pipelines 29
2.8 Case Studies 29
2.8.1 Study 1: Full System Execution for Collective Identity Experiments . 29
2.8.2 Study 2: Data Model for Online Experiment in [47] 32
2.8.3 Study 3: Data Model for a Simulation Study in [146] 33
2.9 Related Work 34
2.9.1 (Networked) Experiments in the Social Sciences 34
2.9.2 Graph Dynamical Systems 34
2.9.3 Workflow Systems 34
2.9.4 Workflow and Scripting Languages 35
2.9.5 Specialized Pipelines 35
2.9.6 Microservices 36
2.9.7 Data Models 36
2.9.8 Formal Models of Pipelines 37
2.9.9 "-Ilities;" reproducibility; interoperability; composability; extensibility; scalability; reusability; and traceability 37

2.10 Summary and Future Work 37

3 Social Networked Experiments and Modeling for Producing Collective Identity in a Group of Human Subjects Using an Iterative Abduction Framework 50

3.1 Abstract 50

3.2 Introduction 51

3.2.1 Background and Motivation 51

3.2.2 Summary of Work Scope 53

3.2.3 Situating Our Work On Anagram Game Experiments and Modeling With Other Research 53

3.2.4 Overview of Our Experiment and Modeling Approach: Abductive Iterations 53

3.2.5 Novelty of Our Work 55

3.2.6 Contributions 55

3.2.7 Extensions from the Conference Paper 57

3.2.8 Paper Organization 58

3.3 Overview of Abductive Loop 58

3.4 Related Work 59

3.4.1 Overviews of CI 59

3.4.2 Individual Anagram Games: Experiments 59

3.4.3 Individual Anagram Games: Modeling 61

3.4.4 Individual Anagram Games: Experiments and Modeling 61

3.4.5 Collective Identity-Based Experiments: Formation of CI 61

3.4.6 Collective Identity-Based Experiments: Implications of CI 64

3.4.7 Measurement of CI 65

3.4.8 Combined Group Anagram and CI Experiments 66

3.4.9 Modeling of CI 67

3.4.10 Agent Based Models of Anagram Games and Formation of CI 67

3.4.11 Studies of Phenomena Related to CI 67

3.4.12 Data-Driven: Combining Experiments and Data-Driven Modeling . . 69
3.4.13 Modeling of Time Sequences of Actions 70
3.4.14 Evaluation of Model Predictions 71
3.4.15 Abduction and Abductive Loop 72
3.5 Experiments 72
3.5.1 Experiment Description 72
3.5.2 Experimental Data 76
3.6 Agent-Based Models (ABMs) of the Group Anagram Game and Modeling Results 82
3.6.1 Discrete-Time Stochastic Process 83
3.6.2 KL-Divergence 83
3.6.3 Overview of the Three Agent Based Models 84
3.6.4 Baseline Agent-Based Model M0 86
3.6.5 Agent-Based Model M1 89
3.6.6 Agent-Based Model M2 95
3.7 Model Evaluation 102
3.8 Abductive Loop Analyses and Results 107
3.8.1 Overview 107
3.8.2 Abductive Iterations with Hypotheses 108
3.8.3 Abductive Loop 1 (AL-1) 109
3.8.4 Abductive Loop 2 (AL-2) 111
3.8.5 Abductive Loops: Role of Analyst and Bigger Picture 112
3.9 Limitations and Additional Work 114
3.10 Summary 114

4 Mechanistic and Data-Driven ABM's in Group Anagrams Games 115
4.1 Abstract 115
4.2 Introduction 115
4.2.1 Background and Motivation 115

4.2.2 Our Work Scope and Differentiators from Previous Work 117
4.2.3 Novelty of Our Work 117
4.2.4 Contributions 118
4.3 Related Work 120
4.4 Online Social-Networked Group Anagram Game 121
4.5 Data Analysis and Model Development 122
4.5.1 Preliminaries 122
4.5.2 Player Action: Form Word 123
4.5.3 Player Action: Request Letter 126
4.5.4 Player Action: Reply to Letter Requests 128
4.6 Agent-Based Simulations and Results 130
4.7 Summary and Future Work 131

5 Conclusions **133**

Bibliography **134**

A Appendix: Pipelines **155**
A.1 Data Common Specification 155
A.2 Appendix: Mapping of Model onto the Software System 162
A.3 Appendix: Examples of the Software System 165
A.4 Appendix: Pipeline Functions 168
A.5 Appendix: Microservices 173
A.5.1 Characteristics 173
A.5.2 Benefits 174
A.5.3 Microservices as a Type of Service Oriented Architecture 174

B Appendix: Iterative Abduction Framework **176**
B.1 Experimental Data 176
B.1.1 Timestamp for Letter Request 176

B.1.2 Timestamp for Letter Reply 177
B.1.3 Timestamp delta between Reply Received and Request Sent 178
B.1.4 Timestamp for Word Formed 179
B.1.5 Temporal Comparisons of Distributions Between Model M0 and Experiments for Individual Player Actions 180
B.1.6 Temporal Comparisons of Distributions Between Models M0, M1 and Experiments for Individual Variables 186
B.1.7 Comparisons of Distributions Between Models M1, M2 and Experiments for Individual Variables at the End of the Anagram Game. . . 192
B.1.8 Temporal Comparisons of Distributions Between Models M1, M2 and Experiments for Individual Variables. 192
B.1.9 Comparisons of KL Divergence Distributions Between Models M1, M2 and Experiments for Individual Variables at the End of the Anagram Game. 214
B.1.10 Temporal Comparisons of KL Divergence Distributions Between Models M1 and M2, and Experiments for Individual Player Actions. . . . 216
B.1.11 Temporal Comparisons of KL Divergence Values Between Models M0, M1 and M2, and Experiments by k. 216

List of Figures

2.1 Five pipelines (in gray) for NESS experiments. In this analysis loop, experiments (conducted with an experimental platform, upper left) are performed. Experimental data are transformed, by the Experimental Data Transformation Pipeline, into a data common specification (in blue) that conforms to our data model (see Section 2.3). The Data Analytics Pipeline analyzes data and generates and prepares data for property inference. The Property Inference Pipeline determines properties for probabilistic agent-based modeling (ABM) and simulation (ABMS). These simulations are performed in the Modeling and Simulation Pipeline. The Model Evaluation and Prediction Pipeline generates comparisons between experimental data and model predictions using statistical and logical testing. We can then specify the parameters for a next set of experiments (experiment specification). Our data model provides a single specification that enables any experiment whose data can be cast in terms of the model, to be analyzed in the system (e.g., a classroom experiment). These pipelines are the focus of this work. This composition of pipelines is one of several possibilities. Our system uses human-in-the-loop. 7

2.2 The three types of models described in this work: (abstract) data model, graph dynamical system model, and pipeline model. Data model enables rigorous reasoning about both (*i*) experiments and experimental data specifications (requirements) and (*ii*) modeling and simulation (MAS) specifications. It, along with the GDS model, help to ensure consistency and correspondence between experiments and MAS. We use graph dynamical systems to model system dynamics. Specific data sources and modeling approaches are shown. These are used within our pipeline model. 9

2.3 Sequence of data models for reasoning about experiments and modeling and simulation. We advocate for pre-pending the abstract data model to the front end of the model process, as shown here. Table 2.1 shows our abstract data model and Figure 2.4 shows this data model translated into a entity-relationship diagram in unified modeling language (UML) form. The table and figures in Appendix A.1 (which support Section 2.7) show the Data Common Specification for our software design. 16

2.4 Data model of Table 2.1 translated into a entity-relationship diagram in unified modeling language (UML) form. This illustrates that the abstract data model can be translated to customary forms of data models more amenable for software development. 16

2.5 Network $G(V, E)$ for a GDS example, with $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. Thresholds θ_i are provided for nodes v_i, in blue, by the respective nodes. The local functions f_i are threshold functions for $v_i \in V$, $1 \leq i \leq 6$; see text for details. The discrete system dynamics are given by the configurations at successive times from 0 to 4, at the right in the figure. Each configuration is given by $C(t) = (s_1^t, s_2^t, s_3^t, s_4^t, s_5^t, s_6^t)$. The system reaches a fixed point at time $t = 3$, as evidenced by no change in the configuration in going from $t = 3$ to $t = 4$. 20

2.6 Functions, or h-function, h_i, $1 \leq i \leq 3$ (implemented as software) within a pipeline. Pipelines control the execution order of functions and the inputs and outputs for each function, through a pipeline job specification. Circles in the figure denote input and output digital objects, such as ASCII files of database tables. 22

2.7 An arbitrary software function h. Input data instances D may have to be transformed by transformation code τ to conform to required inputs I. Inputs and outputs are subjected to verification through comparisons with specified schema (not shown here). 22

2.8 Steps of the Algorithm Pipeline Execution. 25

2.9 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. Here we show how function h_1 is executed in a generic Pipeline 1 and how h_4 is executed in Pipeline 2. Input files are validated against their corresponding JSON schema (Configuration File Verification). The Pipeline Infrastructure Code invokes the corresponding functions. If necessary, the Pipeline Infrastructure Code invokes a function transformation procedure that transforms the file contents into a function input file format. When the contents of all inputs are in the function h_1 input file formats, the files are validated against their corresponding JSON schema (Configuration File Verification). After verification of formats by the corresponding JSON schemas, the function is executed and output files are generated (these digital object outputs may be, e.g., plot files, ASCII data files, and binary data files). There may be additional functions, indicated by the ellipsis below Pipeline 1 Function h_1 Execution. In this example, outputs from the generic pipeline 1 are inputs for the generic pipeline 2. Function h_4 in Pipeline 2 is executed in a similar fashion to function h_1 in Pipeline 1. See the text for descriptions of these various components. Note: the pipeline infrastructure code is the same code for all pipelines. 39

2.10 The anagram game screen, phase-2, for one player. This player has own letters "R," "O," and "L" and has requested an "E" and "A" from neighbors. The "E" is green, so this player's request has been fulfilled and so "E" can be used in forming words; but the request for "A" is still outstanding so cannot be used in words. Below these letters, it shows that Player 2 has requested "O" and "L" from this player. This player can reply to these requests, if she so chooses. Below that is a box where the player types and submits new words. 40

2.11 Case study 1. Partial representation of the data model for the online experiment composed of 3 phases with a set of V players ($n = |V|$). The phase 1 DIFI measure, a proxy for CI, uses a null (i.e., empty) network on n players; i.e., there are no edges in the graph because players play individually. In phase 2, a team-based CI-priming game, edges E are communication channels. Initial conditions B^v include letter assignments to players. The individual DIFI measure is repeated in phase 3. The action set A and illustrative action tuples T_i are given for each phase. 41

2.12 The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_3 plots the time series of number of words formed by player for experiment #2. 42

2.13 The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_5 generates the histogram for the number of actions "letter request" for three experiments. The x-axis is time in the group anagram game, binned in 30 seconds intervals. 42

2.14 The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_7 generates the discrete time actions for all three experiments. This latter output will inform the Property Inference pipeline for computing parameters for simulation models. Time (in seconds) is shown in the first row as 1, 2, 3, ..., and counts of the z vector components, per player and per experiment are given. 43

2.15 The Property Inference pipeline receives the input from h_7 of the Data Analysis Pipeline (DAP). The parameters in this figure were generated to inform an ABM model for the Modeling and Simulation Pipeline (MASP). The transitions in the figure are from from i to j, where $a_i \in A$ is the action at time t and $a_j \in A$ is the action at $(t + 1)$. Rows not shown mean there are no such transitions in the data. 44

2.16 The Modeling and Simulation pipeline (MASP) and Model Evaluation and Prediction pipeline (MEAPP) were executed to generate simulation results and model predictions, and to compare experimental data to model predictions. All three plots contain model predictions and use results from h_1 of the MASP. Function h_1 of MEAPP plots corresponding experimental and model output data (top plot) and compares experiment and model output using KL-divergence (center plot) for six parameters. Function h_2 of MEAPP uses h_3 of the Data Analysis pipeline (DAP) to plot model predictions from h_1 of the MASP (bottom plot) where now $n = 15$ (in experiments, $n = 6$). 45

2.17 Elements of the data model (Table 2.1), for the online social network experiment in [47]. 46

2.18 Data model of Table 2.4 translated into a entity-relationship diagram in unified modeling language (UML) form. 46

2.19 Data model of Table 2.5 translated into a entity-relationship diagram in unified modeling language (UML) form. 49

3.1 Conceptual view of three dimensions of this work, illustrating how our group anagram game study is situated. Our experiments consist of online web-based human subjects experiments. Our modeling component consists of model and algorithm development, and agent-based modeling. We study groups of interacting individuals. To our knowledge, this combination of study features is unique. These dimensions are used in Figure 3.2 to compare our work with others. 54

3.2 A hierarchy of different collective identity (CI) experiments in the literature, which puts the uniqueness of our work on anagram game experiments and modeling into the context of the works of others. The internal nodes refer to classifications between in-laboratory and online experiments, and between works with and without modeling. We distinguish between studies of individuals, with no interaction between subjects, and studies of groups, with interactions among subjects. Leaf nodes refer to representative works. Our work studies online experiments of collective identity with modeling and interaction among subjects. (These references are not exhaustive; more detail is included in the related work of Section 4.3.) 55

3.3 Steps in our iterative abductive analysis/loop. 59

3.4 Steps for the overall online game include: recruitment of players from Amazon Mechanical Turk (AMT), directions for the use of the platform, DIFI1 score procedure, anagram game, and DIFI2 score procedure. 73

3.5 Anagram game configuration with a $k = 2$ regular graph on $n = 4$ players (v_1, v_2, v_3, v_4) with number of initial letters $n_L = 3$ assigned to each player, as shown in the boxes next to the players. Requests for letters and replies are sent across the channel links (red to request letters, green to reply with letter). Request-Sent-Buffer keeps track of player v_i's letter requests. To-Reply-Buffer contains letter requests from other players to v_i. Key (#) shows the sequence of actions by all the players during a game. Table 3.2 shows a detailed description of these actions. 74

3.6 The anagram game screen of the web app for one player. This player has own letters "S," "O," and "L" and has requested an "E" and "A" from neighbors. The "E" is green, so this player's request has been fulfilled and so "E" can be used any number of times in forming words. But the request for "A" is still outstanding so cannot be used in words. Below these letters, it shows that player 2 has requested "O" and "L" from this player; this player has to reply to these requests, if she so chooses. Below that is a box where the player types and submits new words, like "SEE." 76

3.7 DIFI game where player v_i moves the smaller circle, representing herself, either over (partially), or away from, the bigger circle that represents the team. The team circle is stationary. The distance δ between centroids of circles is measured. The distance is such that $\delta = 0$ corresponds to the small and large circles just touching; $\delta < 0$ means that the two circles are disjoint; and $\delta > 0$ means the two circles overlap. The distance δ is transformed into a DIFI value. The range in DIFI value is: $-100 \leq \delta \leq 125$. The DIFI score is a proxy for CI. This is an individual player game. 77

3.8 Probability density distribution for time of request sent over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used estimate the probability density function. Letter requests are made throughout the game, rather than solely at the outset. 79

3.9 Probability density distribution for time of reply sent over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Letter replies are made throughout the game, rather than solely at the outset. 80

3.10 Probability density distribution for time duration between requesting a letter and replying to the request, over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Players generally respond relatively quickly to their neighbors letter requests, with replies typically made within 30 seconds of the request. 81

3.11 Probability density distribution for time of forming words over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Word submissions are made throughout the game, and the number of neighbors and available letters, does not affect this type of action. 82

3.12 ABM baseline M0 predictions of the $k = 2$ experiments (in green) and experimental data (in gray), over the entire 5-minute group anagram game. The probability density function is show for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 2$ experiments. The Baseline M0 predictions are from 100 simulations of a $n = 10$ player game. It is clear from visual inspection that model M0 predictions are in better agreement with the experiment data for the requests received and requests sent variables. We make this comparison more precise using KL-divergence below in Figure 3.13. 88

3.13 KL-divergence values for the baseline Model M0 across the five parameters of x: lower values are better. M0 does a better job predicting the number of Requests Received and Requests Sent. Analyses are based on the data of Figure 3.12, over five minutes, at the end of a game. 89

3.14 KL-divergence values for the baseline M0 model across the five parameters of x at one-minute intervals: lower values are better. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. The data are for conditions $(n = 10, k = 2)$. These plots show that request-related predictions are better than reply-related predictions over all time intervals. The reply-related predictions are better in the second half of the five-minute anagram games, but Figure 3.9 shows that in experiments, there are fewer replies in the second half of the games. 90

3.15 Distribution of KL divergence values for comparing distributions of model output with corresponding distributions of experimental data, for the group anagram game. The model is the $(n = 10, k = 2)$ baseline M0 ABM. The data sets used in comparison are experiments: $(n = 10, k = 2)$. The data sets used in comparison are experiments: $(n = 10,\ k = 2)$. There are 30 values in the distribution, with five values for the variables of x at the end of the game (over five minutes of the game), and 25 values for each of the five variables of x over five intervals of one minute duration. It shows that for Model M0, some KL divergence values are high, indicating that the model is in poor agreement with data. As we see in Figure 3.13, M0 does not do a good job predicting the number of replies received, replies sent, and words formed. 91

3.16 ABM M0 and M1 predictions of the $k = 2$ experiments, along with the experimental data. The probability density function is show for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. It is clear from visual inspection that model M1 predictions are in better agreement with the experiment data than are M0 predictions. We make this comparison more precise using KL-divergence in Figure 3.17. 93

3.17 KL-divergence values for the baseline M0 and M1 models across the five parameters of x: lower values are better. The modeling conditions are experiment with $k = 2$. This figure shows that M1 greatly improves a weakness of model M0 in poorly representing RplR (number of replies received), RplS (number of replies sent), and Wrds (number of words formed). 94

3.18 KL-divergence values for the baseline M0 and M1 across the five parameters of x: lower values are better. The modeling conditions are experiment with $k = 2$. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. While Model M0 has good predictions for the minute 3 and minute 5 (with the exception of the words formed), Model M1 has better predictions for the minute 3 and minute 5 for all five x variables. 95

3.19 Distribution of KL divergence for comparing distributions of model output with corresponding distributions of experimental data for the anagram game. Models are $(n = 10, k = 2)$ M0 and M1. The data sets used in comparison are experiments: $(n = 10, k = 2)$. There are 30 values in each distribution, with five values for variable x at the end of the game over all five minutes, and 25 values for the five variables x over five intervals of one minute increment each. These results shows that for model M1, the KL divergence values are frequently low, indicating that it is in better agreement with experimental data. 96

3.20 M1 model distributions predicted for the number of replies received at the end of game ($n = 10$, 100 simulations), for different regular degrees k of the game network G. This partially motivated our development of ABM M2, since model M1 predictions do not vary significantly with the number of a player's neighbors. 96

3.21 ABM M1 and M2 predictions of the $k = 2$ experiments, and experimental data, over all five minutes of the group anagram games. The probability density distributions are shown for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that models M1 and M2 generate similar predictions, in agreement with the experiment data, as M1 is learned solely from $k = 2$ experimental data. We make this comparison more precise using KL-divergence in Figure 3.23. 98

3.22 ABM M1 and M2 predictions of the $k = 4$ experiments, and experimental data, over all five minutes of the anagram games. The probability density distributions are shown for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that model M2 predictions are generally in better agreement with the experiment data than are M1 predictions. We make this comparison more precise using KL-divergence in Figure 3.24. 99

3.23 The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 2$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M1 and M2 generate similar predictions to the experimental data. 100

3.24 The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 4$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M2 gives much better performance than M1 predicting the time to generate an action for an agent. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. 100

3.25 KL-divergence values for the Models M1 and M2 predictions of the $k = 2$ experiments across the five parameters of x: lower values are better. Each plot contains data over a 1-minute time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute intervals, of the 5-minute anagram game. This figure shows that M1 and M2 generate similar predictions to the experimental data, as M1 is learned from $k = 2$ experimental data and M2 is developed from $2 \leq k \leq 8$ data. 101

3.26 KL-divergence values for the Models M1 and M2 predictions of the $k = 4$ experiments across the five parameters of x: lower values are better. Each plot contains data over a 1-minute time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute intervals, of the 5-minute anagram game. M2 gives much better performance than M1 predicting the time to generate an action for an agent after the minute one. M2 gives much better performance, as expected, as it explicitly accounts for agent degree for $2 \leq k \leq 8$. 102

3.27 Each plot shows the distribution of KL divergence for comparing distributions of model output with corresponding distributions of experimental data for the group anagram game. There are 30 KL-divergence values represented by each distribution in each plot, with five values for variable x at the end of the game, and 25 values for each of the five variables x over five intervals of one minute increment each. The plots correspond to different k values: (a) $k = 2$; (b) $k = 4$; (c) $k = 6$; and (d) $k = 8$. The y-axis range in (a) is different because of the high concentration of the x-axis KLD values. A model improves as the density is greater at lesser KL-divergence values. Hence, Model M1 is better for $k = 2$, and Model M2 is better for $k > 2$, consistent with the data used to build these models. 103

3.28 A scatter plot of KL-divergence for M1 (x-axis) and M2 (y-axis) for four k values and five x variables. For $k > 2$, M2 performs better than M1, as M2 incorporates experimental data with $2 \leq k \leq 8$. Interestingly, M1 and M2 perform equally well (highlighted) for $k = 2$ as M1 is learned from $k = 2$ experimental data (M1 is slightly better). These are data over the total 300 seconds anagram game. 104

3.29 KLD values from the comparison of models M0 (green box), M1 (red box), M2 (blue box), versus the experimental data. On the x axis we show the anagram game by the minute of the five minute game (i.e. [0-1),[1-2),[2-3),[3-4),[4-5)). Each box, by type of model, contains 100 values of KLD corresponding to $k = 2$, 4, 6, and 8, and the five x variables at the end of each minute. Model M0 shows the highest median values throughout the 5-minute game, except for minute 5 when it performs better than M1. The Model M1 median is lower at minute [0,1), while the Model M2 median is lower for the minutes two through five. Figure B.41 in Appendix B.1.11 shows the boxplots grouped by type of $k = 2, 4, 6, 8$ and shows the highest median values on the first two minutes of the game. 105

3.30 This shows that the model fits good for this category. The transitions in this category only contains intial status 1. Comparing empirical probabilities with probabilities estimated from model, they are very close to each other. 105

3.31 This shows that the model fits good for this category. The transitions in this category contains all intial status. Comparing empirical probabilities with probabilities estimated from model, they are very close to each other. 106

3.32 This shows that the model fits bad for this category. However, the model fits good for initial 1 and initial 4, which have 112 counts and 49 counts. While the model fits bad for initial 2 and initial 3. Thus, the count for each initial status determines the probabilities fitting for that status. 106

3.33 This shows that the model fits bad for this category. The transitions in this category only contains intial status 1, which has only 3 counts. 107

3.34 Scatter plot of RMSE against total count. The x-axis is the total count of a category, the y-axis is the RMSE for that category. The plot shows that a category with less count tends to have larger RMSE. 107

3.35 Scatter plot of RMSE against min.count. The x-axis is the minimum count of a category, the y-axis is the RMSE for that category. The plot shows that a category with less min.count tends to have larger RMSE. Furthermore, it shows that the minimum count has a sharp delineation between small and large RMSE. 108

3.36 Scatter plot of RMSE against Min.Count in different settings of covariates in Table 3.10. See Equation (3.8) for RMSE and text for Min.Count. 108

3.37 Abductive tree representing candidate abductive loops with dependencies. Hypotheses are nodes, and are provided in Table 3.11; edges are outcomes of ALs. The orange colored nodes correspond to abductive iterations presented herein. The red node is a candidate next loop. This tree is not unique; different analysts can devise different trees. 109

3.38 Statistical analysis correlation results of the anagram game parameters and DIFI2 score. (a) Probability density of replies received change markedly from $k = 2$ to $k = 4$, but relatively little for further increasing k. (b) Probability density of DIFI2 score moves dramatically to larger DIFI2 score with increasing k. Each of these results is novel. All the more novel is the combination of the two: while game measurables saturate (other data besides replies received), the DIFI2 score does not. 113

4.1 Simplified view of a networked group anagram game (GrAG), with illustrative actions among $n = 3$ players that communicate and share letters through the gray channels. Each player is initially given $n_l = 3$ letters. Letters that a player has "in-hand" to form words are shown in boxes. Player actions are shown in blue. At time t, v_2 requests a "u" from v_1 and v_3 forms the word "cot." At the next time, v_2 receives a "u" from v_1, forms the word "bug," and receives a request from v_3. 116

4.2 Structure of the **composite (agent-based) model**. At each time in a group anagram game, a player takes one of four actions, consistent with the online game: "form word," "request letter," "reply to letter request," or "think." The selection of each action is determined by a multinomial logistic regression model from [202], which we call the **action *type* and time** (ATAT) model. Each of the first three actions requires a **component model** (and software module) that simulates human reasoning and outputs the *specifics* of an action. These are expanded on in Figure 4.3 below. These *component models* in this figure are the focus of this work. The algorithms for these three actions are in Figure 4.7, Figure 4.9, and in [45], Figure 16. "Thinking" is an idling action, no model is needed. The common theme across these models is given in blue. Aptitude is described in Section 4.5. 118

4.3 **Component models** (i.e., combined *mechanistic* and *data-driven* models) for the three player actions in the GrAG. These are models of human reasoning, which output specific player actions in the game. The particulars of the mechanistic and data-driven models are given in the respective boxes under the actions and are detailed in Section 4.5. Mechanistic models are built first, and then augmented with data-driven models. The player actions and component models map onto those in Figure 4.2. 120

4.4 Comparison of ***mechanistic*** model predictions against data for the *form word* model. Mechanistic predictions are the values on the x-axis (d^L_{min}); data are on the y-axis ($d^L_{i,act}$). We use the $|C^W| = 5000$ word corpus. Each plot corresponds to a grouping of players by 20% bins of player performance in forming words according to d^L, and represents, in turn, P_j, $j \in \{1, 2, 3, 4, 5\}$, moving left to right. Numbers are numbers of observations in the data. If $d^L_{i,act}(w_1, w_2) = d^L_{min}$, then the experimental data correspond exactly with the mechanistic model. 124

4.5 For $(C^W_i, b^{wf}_i, d^L_{min}) = (5000 \text{ words}, P_1, 1)$, the distribution $\mathcal{D}^{d^L}$ of $d^L_{i,act}$ from experiments is shown. For a given d^L_{min} computed for optimal behavior, the appropriate distribution is sampled to obtain $d^L_{i,act}$ for v_i. These distributions are formed from the data in Figure 4.4 and they are part of the ***data-driven***model of form word. 125

4.6 Experimental data for $|C^W_i| = 5000$. Log-log scale plot (inset) of the distribution of ranks of words formed by players from the word set $W^{ih}_i(w_1, d^L_{i,act})$. Lesser rank means higher word frequency from corpus. Players most often choose words with lesser rank (i.e., greater frequency). 126

4.7 Steps of the Algorithm Form Word. This algorithm returns a word that an agent forms. 127

4.8 Comparison of ***mechanistic*** model predictions (in green) against data (the distributions) for the *request letter* model. Our mechanistic model predicts all letter requests will be of rank-1 in each of the four plots. (LEFT) Experimental data are for the 5000-word corpus, aptitude $b^{req}_i = Q_1 = 20\%$ for letter requests (plots for Q_j, $j \in \{2, 3, 4, 5\}$ are not shown). For aptitude Q_1, the frequency of the rank of the chosen letter is plotted. These data show that players most often choose letters with lower rank, meaning that they choose letters that can form relatively more words. (RIGHT) These three plots break down the left-most plot by showing distributions for different request numbers r_{num} by v_i. These distributions $\mathcal{D}^{\ell r}(|C^W_i|, b^{req}_i, r_{num})$ are used to sample $r_{i,act}$ based on $(C^W_i, b^{req}_i, r_{num})$. 128

4.9 Steps of the Algorithm Request Letter. This algorithm returns a letter that an agent requests. 129

4.10 (Left) Simulation results for Sim. nos. 1 through 5 of Table 4.2. The average number of words formed per player drops in going from $b_i^{wf} = P_1$ to P_5, $b_i^{req} = Q_1$ to Q_5, for fixed b_i^{rpl} =FB. (Right) Simulation results for Sim. nos. 5, 6, and 7 of Table 4.2. Using $b_i^{wf} = P_5$ and $b_i^{req} = Q_5$ as a baseline, these results show a precipitous drop-off in replies to letter requests, and to words formed, in going from b_i^{rpl} =LTFB to b_i^{rpl} =NR. Results in counts for b_i^{rpl} =LTFB are slightly less than those for b_i^{rpl} =FB. 131

4.11 Simulation for players v_i ($1 \leq i \leq 4$), arranged in a star. All players have the following conditions $b_i^{wf} = P_1$, $b_i^{req} = Q_1$, and b_i^{rpl} =FB. Player v_1 is at the center with three neighbors. v_1 is assigned the four most popular vowels in the alphabet; v_2, v_3 are assigned the six most popular consonants, and v_4 is assgned the five least popular consonants. See text for discussion of results. 132

A.1 JSON schema for the "Experiment" of the Data common specification. . . . 157

A.2 JSON schema for the "Phase" of the Data common specification. 158

A.3 JSON schema for the "Phase Description" of the Data common specification. 159

A.4 JSON schema for the "Player" of the Data common specification. 160

A.5 JSON schema for the "Action" of the Data common specification. 161

A.6 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This figure shows a portion of the schema for a configuration file that specifies the experiment JSON schema file location. . 162

A.7 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the phase description JSON schema file location. 162

A.8 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the phase JSON schema file location. 162

A.9 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the action description JSON schema file location. 163

A.10 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the player description JSON schema file location. 163

A.11 To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. In this configuration file there are five possible functions that can be executed in any order. This Figure shows a portion of the schema for a configuration file that specifies how to compose and execute one or more functions of a simple pipeline. For example, here it defines that a parameter called "actionId" is only necessary for functions h_2 through h_5. . 164

A.12 This is an example of the (1) Experimental Data Transformation pipeline execution to transform raw experimental data into the data common specification. Here we show how function h_1 is executed. Here we show an input CSV file as an example for the "Completed Session Summary" input file. If necessary, file contents are transformed to obtain the direct input for a function in the correct format. Here we show how the "Completed Session Summary" CSV input file is transformed into a "Completed Session Summary" json file that becomes the input for the function. After verification of formats by the corresponding JSON schemas, the function is executed and output files are generated. Here we show the output json file for the "Experiment" data common specification. 166

A.13 This is an example of the (2) Data Analytics pipeline execution to analyze files of data in the common specification. Here we show how function h_7 is executed. Input files are validated against their corresponding JSON schema. Here we show an example of a json schema file for the "Experiment" description input file. Figure A.1 contains the whole file. After verification of formats by the corresponding JSON schemas, if necessary, file contents are transformed to obtain the direct input for a function in the correct format. After verification of formats by the corresponding JSON schemas, function h_7 is executed and output files are generated. In this example the output file is an input for the (3) Property Inference pipeline. 167

B.1 Probability density distribution for requests sent over the 300 second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. (a) shows 1 experiment, for 28 experiments with k=2; (b) shows 1 experiment with k=3; (d) shows 9 experiments with k=4; (a) shows 2 experiments with k=5; (e) shows 3 experiments with k=6; (f) shows 4 experiments with k=8. A kernel-density estimation with Gaussian kernels is used. Letter requests are made throughout the game, rather than solely at the outset. However, if there are few neighbors (k=2) and consequently fewer available letters (3 letters per neighbor), there are fewer letter requests and letter replies near the end of the game. 177

B.2 Probability density distribution for reply sent over the 300 second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. (a) shows 28 experiments with k=2; (b) 1 experiment with k=3; (c) 9 experiments with k=4; (d) 2 experiments with k=5; (e) 3 experiments with k=6; (f) 4 experiments with k=8. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Letter replies are made throughout the game, rather than solely at the outset. 178

B.3 Probability density distribution for the time duration between Reply Received and Request Sent over the 300 second anagram game. (a) shows 28 experiments with k=2; (b) shows 1 experiment with k=3; (c) shows 9 experiments with k=4; (d) shows 2 experiments with k=5; (e) shows 3 experiments with k=6; (f) shows 4 experiments with k=8. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Players generally respond relatively quickly to their neighbors letter requests with replies typically made within 30 seconds of the request, the number of neighbors doesn't affect this type of action. 179

B.4 Probability density distribution for the time of words formed over the 300 second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. (a) shows 28 experiments with $k = 2$; (b) shows 1 experiment with $k = 3$; (c) shows 9 experiments with $k = 4$; (d) shows 2 experiments with $k = 5$; (e) shows 3 experiments with $k = 6$; (f) shows 4 experiments with $k = 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Word submissions are made throughout the game, and the number of neighbors and available letters, doesn't affect this type of action. 180

B.5 ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Replies Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, replies received predictions are better in the second half of the five-minute anagram games. 181

B.6 ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Replies Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, replies sent predictions are better in the second half of the five-minute anagram games. 182

B.7 ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Requests Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, requests received predictions, compared to the other variables, are good through the five minutes of the game. 183

B.8 ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Requests Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, requests sent predictions, compared to the other variables, are good through the five minutes of the game. 184

B.9 ABM M0 predictions of the $k = 2$ experiments for the distributions of Words Formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, words formed predictions are better in the first two minutes of the five-minute anagram games. 185

B.10 ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Replies Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, replies received predictions are better in minute 3, and minute 5 of the five minute anagram games. 187

B.11 ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Replies Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, replies sent predictions are better in minute 1, minute 3, and minute 5 of the five minute anagram games. 188

B.12 ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Requests Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, requests received predictions are good throughout the five minute anagram games. 189

B.13 ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Requests Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, requests sent predictions are good throughout the five minute anagram games. 190

B.14 ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of Words Formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, words formed predictions are good only in the first minute of the five minute anagram games. 191

B.15 ABM M1 and M2 predictions of the $k = 6$ experiments and experimental data. (a) Distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that model M2 predictions are in better agreement with the experiment data than are M1 predictions (with the exception of the words formed variable). We make this comparison more precise using KL-divergence in Figure B.37. . . 193

B.16 ABM M1 and M2 predictions of the $k = 8$ experiments and experimental data. (a) Distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that model M2 predictions are in better agreement with the experiment data than are M1 predictions (with the exception of the words formed variable). We make this comparison more precise using KL-divergence in Figure B.38. . . 194

B.17 ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received, M1 and M2 perform equally well for k = 2 throughout the five minute anagram games. 195

B.18 ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies sent Model M1 predictions are slightly better in minute 1, minute 2, minute 3, and minute 5 of the five minute anagram games. 196

B.19 ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M2 predictions are slightly better than M1 throughout the five minute anagram games. 197

B.20 ABM M2 and M1 predictions of the $k = 2$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M1 predictions are slightly better than M2 throughout the five minute anagram games. 198

B.21 ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed Model M1 predictions are slightly better than M2 throughout the five minute anagram games. 199

B.22 ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received Model M2 predictions are better than M1 throughout the five minute anagram games. 200

B.23 ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies sent Model M2 predictions are better than M1 (except for minute 3) throughout the five minute anagram games. 201

B.24 ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M2 predictions are better than M1 throughout the five minute anagram games. 202

B.25 ABM M2 and M1 predictions of the $k = 4$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests sent Model M2 predictions are slighlty better than M1 throughout the five minute anagram games. 203

B.26 ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed Model M2 predictions are better than M1 for minute 2, minute 4 and minute 5 of the five minute anagram games. 204

B.27 ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received most Model M2 predictions are better than M1 throughout the five minute anagram games. 205

B.28 ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies sent Model M2 predictions are better than M1 throughout the five minute anagram games. 206

B.29 ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M2 predictions are better than M1 throughout the five minute anagram games. 207

B.30 ABM M2 and M1 predictions of the $k = 6$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests sent Model M2 predictions are better than M1 throughout the five minute anagram games. 208

B.31 ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed, Model M2 predictions are better than M1 after the first two minutes of the five minute anagram game. 209

B.32 ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received, Model M2 predictions are better than M1 throughout the five minute anagram games. 210

B.33 ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies received Model M2 predictions are better than M1 throughout the five minute anagram games. 211

B.34 ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received, Model M2 predictions are better than M1 throughout the five minute anagram games. 212

B.35 ABM M2 and M1 predictions of the $k = 8$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests sent Model M2 predictions are better than M1 throughout the five minute anagram games. 213

B.36 ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed, Model M2 predictions are better than M1 throughout the five minute anagram games. 214

B.37 The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 6$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M2 gives much better performance than M1 predicting the time to generate an action for an agent. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. 215

B.38 The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 8$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M2 gives much better performance than M1 predicting the time to generate an action for an agent. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. 215

B.39 KL-divergence values for the Models M1 and M2 predictions of the $k = 6$ experiments across the five parameters of x: lower values are better. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. M2 gives much better performance than M1 predicting the time to generate an action for an agent after the minute two. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. 216

B.40 KL-divergence values for the Models M1 and M2 predictions of the $k = 8$ experiments across the five parameters of x: lower values are better. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. M2 gives much better performance than M1 predicting the time to generate an action for an agent after the minute three. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. 217

B.41 KLD values from the comparison of models M0 (Baseline), M1, M2, versus the experimental data. On the x axis we show the anagram game by the minute of the five minute game (i.e. [0-1), [1-2), [2-3), [3-4), [4-5)). Each box, by type of k, contains five values of KLD corresponding to the five x variables at the end of each minute. Our models show highest median values on the first two minutes of the game. 218

List of Tables

2.1 Definition of our abstract data model. The experiment schema describes experiment parameters. The phase schema structure describes parameter types for an experimental phase; an experiment can have any number n_p of phases. Particular instance variables within the phase schema structure can vary across phases. We use *experiment* throughout in the table and text for ease of exposition, but the data model is also used for *(simulation) models*. 14

2.2 Configuration input file description. See Appendix A.2 for details. 27

2.3 Sections and files from the execution of a generic Pipeline. Figure 2.9 describes how these elements interact, here we define and describe them. 28

2.4 Online social network experiment in [47], defined with our data model. One experiment has two independent phases, one with a clustered-lattice network and another with a random network; each with population size n=98 and number of health buddies per person d=6. 47

2.5 How the structure of communication networks among actors can affect system-level performance is studied in [146]. Here we define this model with our data model. 48

3.1 Topics described in Section 4.3 of Related Work. 60

3.2 Action table detailing the sequences of actions by all players during the anagram game example from Figure 3.5. The first column defines the number of the sequence of actions during the game. For this example, the duration of the game is 10 actions. The second column shows the player initiating the action. The third column shows the name of the action. The fourth column provides a description of the action. 74

3.3 Description of anagram game configurations played with players recruited from Amazon Mechanical Turk. 75

3.4 Summary of the analyses in the Experimental Data Section 3.5.2, and the questions we answer. Section 3.5.2 presents histograms for the timestamps for letter request. Section 3.5.2 presents histograms for the timestamps for letter reply. Section 3.5.2 presents histograms for the timestamps of the time duration between reply received and request sent. Section 3.5.2 presents histograms for the timestamps for word formed. 78

3.5 Actions of players in the model. The set A of actions is $\{a_1, a_2, a_3, a_4\}$. . . . 83

3.6 Progressively sophisticated models of the anagram game developed in this work. The incremental improvements in models are given, starting with model M0. 84

3.7 Variables that are measured in experiments for each player, and predicted with models for each agent, where vector $x = (x_1, x_2, x_3, x_4, x_5)$. All x_i, $1 \leq i \leq 5$, are time dependent. 85

3.8 Summary of the model comparison plots applied to each of the models M0, M1, and M2. For each model, we collect the data into the five groups shown. See the text for details and justification. The fifth column indicates the time period, in minutes, over which experimental data are compared to model predictions. 86

3.9 The feature vector $z = (z_L(t), z_W(t), z_B(t), z_C(t))$ used in the models M1 and M2. These capture history effects in determining the next action of a player. 92

3.10 Three bins and ranges of values for the z variables from Section 3.6.5. These bins are created for each of the four values of $k \in \{2, 4, 6, 8\}$. 104

3.11 Candidate hypotheses to evaluate in abductive iterations. Not all of the hypotheses are evaluated herein. The goal of these hypotheses, coupled with Figure 3.37, is to illustrate that there are many possible hypotheses that can be formulated, and it is up to analysts to decide which ones to pursue. An analyst will be guided by the results of completed iterative abductive analyses. 110

3.12 Constants in the regression of Equation (3.9) to predict DIFI2 score from outputs of the team anagram game. 111

3.13 Results of linear regression of variables in x against dependent variable DIFI2 score, indicating that interactions are more significant than number of words formed in producing CI. 111

4.1 Study 1 initial letter assignments to players in simulations for six players arranged as 2-regular graph. 130

4.2 Parameters that are systematically varied in the simulations of Study 1. These aptitude (b_i^{wf}, b_i^{req}, b_i^{rpl}) settings are the same for all agents in a simulation. . 130

A.1 Data Common Specification. 156

A.2 Listing of types of functions as microservices for the (1) Experimental Data Transformation Pipeline (EDTP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model. 168

A.3 Listing of types of functions as microservices for the (2) Data Analytics Pipeline (DAP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model. 169

A.4 Listing of types of functions as microservices for the (3) Property Inference Pipeline (PIP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model. . . . 170

A.5 Listing of types of functions as microservices for the (4) Modeling and Simulation Pipeline (MASP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model. 171

A.6 Listing of types of functions as microservices for the (5) Model Evaluation and Prediction pipeline (MEAPP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model. 172

Chapter 1

Introduction

1.1 Background, Motivation, and Summary

Online social science experiments/games are increasingly used to study social behaviors at-scale, with finer-grained data collection [47, 48, 51, 128, 131, 132]. These types of experiments explore phenomena such as collective identity [51], coordination [143], exploration versus exploitation [146], and diffusion and contagion [47, 164]. These experiments have subjects or participants interact via a network thus defining interaction patterns among subjects and constraining the information available to subjects during an experiment. Also, experiments are carried out until a certain condition is met or for a specified amount of time (as opposed to one shot games). Modeling of experiments is also becoming more common. Validated models can be executed far more quickly and at far less cost than experiments. Furthermore, iterations of the experiment-analysis-modeling (EAM) process enable incremental improvements in each of these three EAM tasks.

These iterations require several classes of operations: (1) design and conduct experiments, (2) acquire data, (3) fuse and integrate data, (4) analyze experimental data, (5) develop and verify models, (6) infer model parameters, (7) run simulations, (8) compare experimental data against model output (e.g., for validation), (9) exercise models beyond the ranges of experimental data, and (10) iterate (this is one ordering).

In this work we focus on EAM tasks and refer to this subject as *networked temporal social science (NESS)*. There is significant interest in using online systems to carry out NESS experiments to understand individual and group behavior. Such systems allow researchers to record fine grained information pertaining to the social experiment for further analysis. Considerable work is required for analyzing the recorded fine grained data to uncover interesting insights and to generate/refute hypothesis. This tedious yet important work is usually carried out by an analyst who develops tailor-made custom programs and analytical scripts that pertain to the experiment. This often leads to inefficiencies and duplication of effort.

This work takes a step towards overcoming this problem by focusing on the NESS EAM cycle.

We propose a formal model to which NESS experiments and models must conform. The formal specification makes the class of participant actions, states and interactions precise. Furthermore the formal specification is developed with an agent based modeling (ABM) approach in mind that is used for analysis and hypothesis generation and refinement. An explicit correspondence is developed between subjects in online experiments and computational agents in models. Computational modeling is useful in understanding and reasoning about the NESS experiments behaviors [92, 93].

Also, we design and build five composable and extensible automated software pipelines for (*1*) experimental data transformation; (*2*) data analytics; (*3*) model property inference; (*4*) modeling and simulation; and (*5*) results analysis and comparisons between experimental data and model predictions. The pipelines are generic and can be used to study any NESS experiment that confirms to the formal specification.

In this thesis, we are motivated by a particular social behavior, namely collective identity (CI). One of our goals is to understand whether a cooperative anagram (i.e., word construction) game can *produce* CI within a group. CI is an individual's cognitive, moral, and emotional connection with a broader community, category, practice, or institution [200]. While CI is our initial driver, we also purposely seek to generalize our work to enable study of other social phenomena, such as contagion and diffusion.

Our framework for the study of CI is abduction. Abduction is an inference approach that uses data and observations to identify plausible (preferably, best) explanations for phenomena [197]. Abduction has broad application in robotics, genetics, automated systems, and image understanding [12, 127, 216, 258]. In the artificial intelligence (AI) community, many abduction works have focused on topics such as producing explanations for different logic settings (e.g., [78]); determining the computational complexity of abduction problems (e.g., [265]); and generating solutions for special problems (e.g., [196]). Here, we extend the notion of abduction by developing an abductive looping process and execute it for the problem of CI. While there has been work on abductive iterations, we formalize this process, and make modeling a much more prominent component of an iteration.

Individual anagram games have been extensively studied for more than 60 years to analyze problems such as effects of goal-setting [144, 156, 213, 257], effects of compensation types [40, 41, 52], internal-external attributions [70, 84, 85, 86, 172, 227, 250], test anxiety [205, 208, 209], collective identity [104] and anagram game performance [75, 99, 166, 263]. This work involves a broad range of fields like sociology [41, 213, 257], economics and [40, 51, 52], (social) psychology [70, 84, 85, 86, 144, 156, 172, 205, 208, 209, 250]. With all of this experimental work on anagram games, it is surprising that very little work has been done in modeling these games.

The majority of work on anagram games involves individual players. Recently, work has

expanded to group anagram games where players cooperate by sharing letters [51, 202]. In this work, we analyze experimental data from novel web-based, networked experiments of group anagram games. We design, construct, and evaluate data-driven networked ABM of experiments. First, we develop a CRF model for predicting time sequences of types of actions that players execute during the group anagram games. Then, we develop models for agents that execute details of player actions (e.g, what particular word a player forms). For this latter effort we develop a process for combining mechanistic and data-driven (defined below) approaches to build models of human reasoning to predict detailed game player actions. We compare model predictions against experimental data, which enables us to provide explanations of human reasoning and behavior.

1.2 Research Questions

The work described above, and detailed in subsequent chapters, attempts to answer the following questions:

1. Can we devise a data model and apply a computational model that together form abstract representations of experiments and modeling and simulation (MAS) and ensure correspondence between experiments and MAS?
2. Can we design and implement an extensible software system to study social phenomena through iterative experiments and modeling?
3. We know that Collective Identity (CI) can be fleeting. Can CI form over a short time period with people that not know each other?
4. Can we build time sequence models that predict the types of actions that players take, in time, in a group anagram game?
5. Can we build purely mechanistic and data driven models of the details of player actions in the group anagram game? Can we devise a process for combining mechanistic and data-driven approaches to build models of human reasoning?
6. Can we implement time sequence, mechanistic, and data driven agent-based models to explain human behavior beyond that illuminated by experiments alone?

To explore an abstract representation of experiments and modeling and simulation (MAS) (Question 1) and study social phenomena through iterative experiments and modeling (Question 2), we describe in Chapter 2 a formal abstract data model for networked social science experiments to provide a common representation for both experiments and modeling, thus producing a correspondence between experiments and MAS. Also, in Chapter 2, we describe extensible pipelines for (1) experimental data transformation, (2) data analysis, (3)

model property inference, (4) modeling and simulation, and (5) model of evaluations and predictions against experiment results. These pipelines are developed and used for iterative modeling and simulation, and analysis, of networked experiments via abductive looping. In Chapter 3, using the pipelines, we study the social phenomenon of collective identity through iterative experiments and modeling (Question 3). We devise and evaluate a time sequence model of player actions in the group anagram game (Question 4). Finally, in Chapter 4, we construct mechanistic and data-driven models (defined in Chapter 4) that predict detailed player actions in the group anagram game. We develop a process for combining mechanistic and data-driven approaches to build models of human reasoning (Question 5). Finally, we combine the time sequence, mechanistic, and data driven models in forming an agent-based modeling and simulation (ABMS) platform for simulating group anagram games (Question 6).

Chapter 2

Pipelines and their Compositions for Modeling and Analysis of Controlled Online Networked Social Science experiments

2.1 Abstract

There has been significant interest in online social science experiments in order to understand behavior at-scale, with finer-grained data collection. To uncover interesting insights and to generate/refute hypotheses, considerable work is required to perform data analytics for custom experiments. Furthermore, such experiments are increasingly being done in an iterative loop that comprises four broad steps: (i) propose/modify hypothesis; (ii) carry out experiments; (iii) analyze to confirm/disconfirm a hypothesis; and (iv) build models from the data. This tedious yet important work is usually carried out by an analyst who develops tailor-made custom programs and analytical scripts that pertain to the experiments and modeling. This often leads to inefficiencies and duplication of effort.

This paper takes a step towards overcoming this problem by focusing on a *networked temporal social science (NESS)* experimental modeling cycle. In such a cycle, participants interact via a social network and carry out the needed tasks over a period of time via interactions with neighbors, exchanging appropriate information as specified by a NESS experiment. The loop allows for iterative modeling and experimental refinement. First, we propose a formal model to which such NESS experiments and models must conform, for reasoning about them. The formal specification makes the class of actions, states and interactions precise. Furthermore, the formal specification is developed with an agent based modeling approach in mind that is used for analysis and hypothesis generation and refinement. An explicit correspondence is

developed between the subjects in an online experiment and the computational agent-based model.

As a second step, we design and build five composable and extensible automated software pipelines for (*i*) experimental data transformation; (*ii*) data analytics; (*iii*) model property inference; (*iv*) model/simulation; and (*v*) results analysis and comparisons between experimental data and model predictions.

The pipelines are generic and can be used to study any NESS experiment that confirms to the formal specification. Our data model is for scenarios where subjects can repeat actions (from a set) any number of times over the game duration. Because the types of interactions and action sets are flexible, this class of experiments is large. Three case studies, on collective identity, complex contagion, and explore-exploit behaviors illustrate use of the system. The case studies show the generality of these pipelines and improving human productivity for carrying out online NESS experiments and modeling.

2.2 Introduction

2.2.1 Background and Motivation

Online controlled networked social experiments/games (henceforth referred to as NESS experiments or experimental loop) are increasingly used to study social behaviors [47, 48, 51, 128, 131, 132] and explore phenomena such as collective identity [51, 202], coordination [143], and diffusion and contagion [47, 51, 164]. NESS experiments have the following distinguishing features: (*i*) the experiments and analysis are performed in a loop; (*ii*) subjects or participants interact via a network, thus constraining the information available to them during an experiment; and (*iii*) the experiment is carried out until a certain condition is met or for a specified amount of time (as opposed to one shot games). Computational modeling is useful in understanding and reasoning about these behaviors [92, 93]. Several studies [81, 147, 162] motivate much of our work. A model that is validated using experimented data can be run much faster, more economically, and over a greater range of conditions than can be accomplished with experiments (e.g., for parametric studies, sensitivity analyses); e.g., see [146] for a pure modeling approach that examines a wide range of conditions. Combining experiments and modeling enables each to inform and guide the other. This combined approach has been accomplished in some studies without automation [5, 44, 217] or purely conceptually [253]. Reference [253] takes a combined experiment/modeling approach by defining a framework for *conceptual* modeling for simulation-based serious gaming. Often, there is emphasis on one or the other with no experiment/modeling iterations; e.g., experiments are emphasized, and there are no iterations [164].

NESS experiments require several classes of operations: (1) design experiments, (2) conduct experiments and collect data, (3) fuse and integrate data, (4) analyze experimental data,

(5) design, develop, and verify models (data-driven models if models are based on the data), (6) infer model parameters, (7) run simulations, (8) compare experimental data against model output, (9) exercise models beyond the ranges of experimental data (e.g., to explore counterfactuals), and (10) iterate. This is one ordering; see Figure 2.1.

Currently, these steps are done in an adhoc manner by individual scientists, leading to inefficiencies, duplication and an overall decrease in human productivity. Automating these steps can not only lead to improved productivity, but also to improved reproducibility, and scalability. Although there are software systems that address some of these operations [126, 203], these approaches do not take the semantics of social experiments into account and largely focus on providing a generic data schema. An automated and extensible system for evaluating social phenomena through iterative experiments and modeling that addresses all of these issues is lacking.

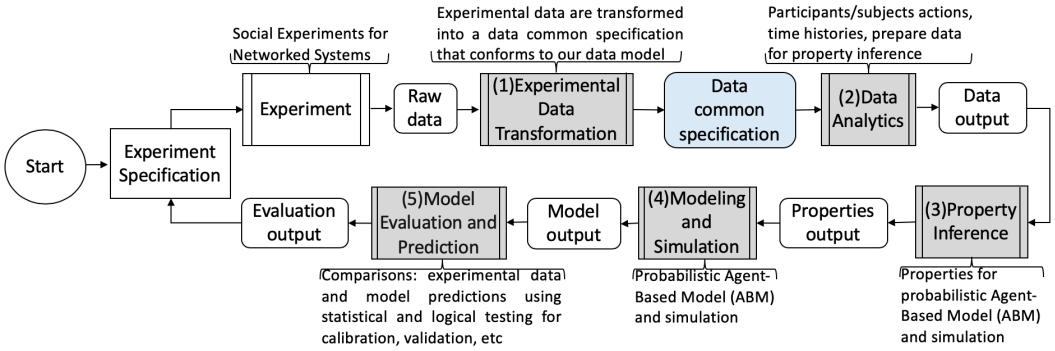


Figure 2.1: Five pipelines (in gray) for NESS experiments. In this analysis loop, experiments (conducted with an experimental platform, upper left) are performed. Experimental data are transformed, by the Experimental Data Transformation Pipeline, into a data common specification (in blue) that conforms to our data model (see Section 2.3). The Data Analytics Pipeline analyzes data and generates and prepares data for property inference. The Property Inference Pipeline determines properties for probabilistic agent-based modeling (ABM) and simulation (ABMS). These simulations are performed in the Modeling and Simulation Pipeline. The Model Evaluation and Prediction Pipeline generates comparisons between experimental data and model predictions using statistical and logical testing. We can then specify the parameters for a next set of experiments (experiment specification). Our data model provides a single specification that enables any experiment whose data can be cast in terms of the model, to be analyzed in the system (e.g., a classroom experiment). These pipelines are the focus of this work. This composition of pipelines is one of several possibilities. Our system uses human-in-the-loop.

One often thinks of software as a tool for increasing the speed of some sort of processing. However, in the loop of Figure 2.1, some of these steps can take months to complete, so that work is "slow" in some sense. For example, modifications in experimental procedures can take weeks to specify, implement, and verify before new experiments are performed. Similarly,

multiple models using fundamentally different methodologies may require several weeks to develop, even though they use the same experimental data. Indeed, this has happened in our work. These cases make documentation, data storage, and provenance critical, so that work done in the past can be recalled and traced.

In this work, we present a computational environment to support NESS experiments. In other words, NESS is an experiment/modeling/simulation/analysis environment composed of pipelines to study social behaviors. A **pipeline** is a sequence of operations, each of which performs a useful task by taking one or more inputs and producing one or more outputs. Our use of pipeline is motivated by the Pipes and Filters architecture pattern [38] and [91]. A pipeline combines operations in analyst-specified ways. We distinguish our work from *workflows* because, while pipelines have many common features with workflows, here we do not account for provenance of digital objects under a data management system; this work is in progress. See Related Work. Figure 2.1 provides an overview of our system that is described further below. We also provide multiple formalisms, for a data model, a dynamical systems computational model, and a pipelines model that underpin our environment. See Figure 2.2.

Throughout this work, *experiment* or *game* means having human subjects interact to achieve some objective, while the actions are recorded for later analysis. *Modeling* refers to building mathematical representations (i.e., models) of experiments. *Simulation* means running software implementations of models, e.g., of ABMs. For clarity, we purposely avoid ambiguous terms like *computational experiment*.

2.2.2 Technical Challenges

Technical challenges can be broken down into two categories: those pertaining to pipelines in general, and those that are more specific to social sciences. For large and complex scientific applications, abstractions that capture data processing and computation are important [69]. A system is easier to understand and reuse with high-level abstractions [94]. General challenges include: identifying the correct levels of abstraction for systems and applications (e.g., formal data models and computational models help with this), automation, reproducibility, interoperability, composability, extensibility, scalability, and traceability [97]. A persuasive argument for a similar approach to study experimental quantum mechanics is presented in [169]. In the case of NESS experiments, we have three unique challenges to address.

1. Greater range is required in modeling functionality; modeling in social sciences can be different from that in engineering disciplines because often a "model" is a qualitative textual description that is open to different interpretations due to lack of details. Hence a social science "model" can lead to different interpretations and algorithmic models to build and evaluate.

2. Experiments in the social sciences can vary widely, depending on the phenomena being

studied [244]. Hence, data analytics for these varying experiments, including data exploration, requires custom analyses.

3. Different classes of problems require different data and computational models.

We now address the novel aspects of our work. The three types of models described in this work: (abstract) data model, graph dynamical system model, and pipeline model. A system similar to Figure 2.1 is being pursued in [169] for quantum studies.

2.2.3 Novelty Of Our Work

The novelty of this work is:

1. Devising an **abstract data model** that is a representation of experiments and simulation models so that we can determine whether an experiment or simulation can be analyzed with our system. Furthermore, we incorporate a second model called **graph dynamical systems** (GDS) [6]. GDS and the abstract data model provide foundations to ensure correspondence between experiments and computational models. See Figure 2.2, where we have an experimental platform and a modeling and simulation (MAS) platform, and we need these two to interoperate through our data and GDS models. It shows specific data sources and modeling approaches.

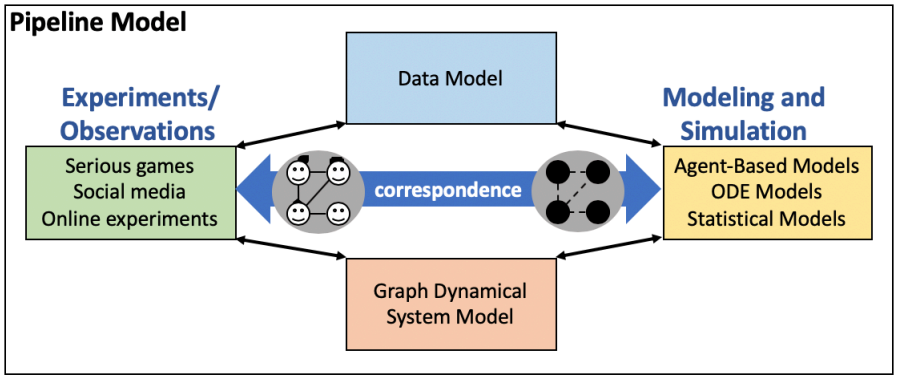


Figure 2.2: The three types of models described in this work: (abstract) data model, graph dynamical system model, and pipeline model. Data model enables rigorous reasoning about both (*i*) experiments and experimental data specifications (requirements) and (*ii*) modeling and simulation (MAS) specifications. It, along with the GDS model, help to ensure consistency and correspondence between experiments and MAS. We use graph dynamical systems to model system dynamics. Specific data sources and modeling approaches are shown. These are used within our pipeline model.

2. Our pipelines take a *microservices* conceptual approach [49, 206, 230] wherein the components of a pipeline—which we call *functions* or *h-functions*—have a narrow scope. (Functions are described below, but basically represent the software that provides the functionality that the pipelines orchestrate.) This way, new functions can be added for new experiments and models in a targeted way, fostering reuse without introducing redundant capabilities. Our contributions below elaborate on these points. In effect, then, our pipelines individually and collectively act as *gateways* into services. We are moving toward X-as-a-service (XaaS).

3. The range of functionality that we provide (and this range is expanding) is broad. Implied by the foregoing discussion and Figure 2.1, we seek a broader set of capabilities that *is* an experiment, modeling and simulation, and analysis environment (EMSAE). Our objective is well beyond the current focus of many systems, which is social network analysis [94].

4. The range of experiments that we can evaluate is large. We demonstrate this with the three cases studies of Section 2.8. This is particularly relevant in a field like social sciences (like others), where the range of phenomena under study is so large. Note, our system cannot deal with *any* experiment; we specify the class of experiment in Section 3.2.6 below.

2.2.4 Contributions

1. Development of conceptual views, formal models, and implementations for each of a data model and a pipeline model. For each of the data and pipeline models, we provide conceptual views, formal models, and implementation descriptions. This approach demonstrates the power of modeling to inform software system implementations. We also describe the GDS formalism which is used to provide a correspondence between experiments and (agent-based) models for simulating experiments. The GDS formalism is not our contribution, but its use in this setting is a contribution. Thus, taking the data, GDS, and pipeline systems each in turn, this contribution is specifically that we provide a consistent (and unified) view of, and approach to, pipeline systems building for social experiments and for modeling them. Specific contributions within this context follow.

2. Formal data model for NESS experiments and simulation modeling. We develop a formal abstract data model for networked social science experiments. The model provides a common representation for both experiments and modeling, thus producing a correspondence between experiments and modeling and simulation (MAS). It also provides a needed level of abstraction per Section 2.2.2. Our data model has the following five characteristics: (*i*) an experiment may be composed of multiple phases (i.e., sub-games); (*ii*) each phase may have a different finite duration; (*iii*) each phase may have a different interaction structure among players (i.e., different networks); (*iv*) each phase may have a different set of actions (and

interactions) among players; and (v) these actions may be repeated by players any number of times within the duration of a phase (i.e., temporal interactions). Experiments with our five characteristics represent a significant class of experiments, e.g., [47, 51, 128, 164], and experiments based on [146]. The data model, with our computational model (Section 2.4), provides a formal specification for experiments and models. The data common specification in Figure 2.1 is based on the data model.

3. Pipeline formalism and implementation. We provide a conceptual view of pipelines. This view is used to construct a formal description of our pipelines. From these, we design and construct a pipeline system and infrastructure to execute software pipelines. The pipeline software infrastructure is the same (common) among all five pipelines that we introduce in this paper to study social science experiments and modeling of them. It can be used for additional pipelines, as we have demonstrated in our work that it is extensible (i.e., our particular pipelines have been constructed over time and use the same infrastructure). Pipeline operations are: (i) read and parse the pipeline configuration file which specifies the pipeline tasks to complete; (ii) control accessing input files, JSON schema files, transformation codes, functions, etc.; (iii) check files against their JSON schema and terminates gracefully if a verification fails; (iv) invoke the proper transformation functions (if applicable), (v) invoke the proper h-functions in their proper order (and any other operations), and (vi) error handling.

4. Five extensible pipelines for modeling and simulation, and analysis, of controlled networked experiments. We design and construct pipelines for (1) experimental data transformation, (2) data analysis, (3) model property inference, (4) MAS, and (5) model evaluation against experiments results, and prediction. Pipeline functionality is based on a formal pipeline model. Each pipeline consists of an extensible collection of *functions* that can be composed to accomplish computational goals. Moreover, the pipelines themselves can be composed in several ways (Figure 2.1 is one way). Syntactic data validation of function inputs and outputs ensures robust software execution. The ten operations in Section 2.2.1 are embedded in these pipelines (note: generating software verification cases is not automated and model design is a human task). The Figure 2.1 caption explains why we emphasize *controlled* experiments; however, use of the pipelines does not require this (e.g., they can be used with social media data). The steps in Figure 2.1, while automated, are often executed with a human-in-the-loop to inspect results. The pipelines also satisfy the reproducibility, composability and other "ilities" of Section 2.2.2.

5. Pipeline functions. Our pipelines take a microservices conceptual approach [49, 206, 230] wherein the components executed by a pipeline—which we call *functions*—have a narrow scope. We provide 29 implemented functions within the five pipelines (See Appendix A.4). New functions can be added for new experiments and models in a targeted way (as we have done), fostering reuse without introducing redundant capabilities.

6. Case studies. We provide three case studies to illustrate the use of the NESS system. Case study 1 combines experiments and modeling. Case study 2 addresses experiments only.

Case study 3 focuses on modeling only. In case study 1, we describe social experiments that we conducted to produce collective identity (CI) within a group of human subjects [202]. Experiments and all five pipelines in Figure 2.1 are used. In a second case study, we take the experiment in [47] and demonstrate how it maps onto our data model, thus showing that we can evaluate such experiments with our system. The experiment objective is to study the effects of network structure on complex contagion diffusion. In a third case study, we take the explore-exploit model in [146] and demonstrate how it maps onto our data model. The objective in [146] is to investigate how the structure of communication networks among actors can affect explore and exploit actions. Although the number of subjects in experiments is limited to tens of people, our pipelines have been successfully run with millions of artificial subjects for evaluating scalability.

2.2.5 Additional Perspective

Reproducibility in science is of considerable interest, owing to the fact that several studies indicate that much of science is not reproducible [22, 186, 211]. A pipeline system such as this is a contribution to that goal. Pipeline operations can be recorded and rerun for reproduciblity. When we connect this pipelined system with our infrastructure, we will have provenance, too.

Approaches to rectify this lack of reproducibility include triangulation and consilience [176]. Triangulation uses multiple approaches to answer a single question, so that if these approaches produce the same/complementary results and hence similar conclusions, one may put more credence in the results. Our pipelines can help in this regard in at least two ways. First, for a given type of experiment (resp., model), multiple models (experiments) can be incorporated into the pipeline. Second, the experiments and models may both be changed to investigate a problem from multiple perspectives. This paper is a full treatment of, and an extension of a preliminary version that appears as [46].

Organization. We first present formalisms for the data model (Section 2.3) and the computational models of discrete dynamical systems (Section 2.4) because these underpin the pipelines. Next, we present a conceptual view of the pipelines and their components (Section 2.5), followed by a formal specification for the pipelines in Section 2.6. Next, selected implementation issues are presented in Section 2.7. We provide case studies in Section 2.8 that emphasize both the data model and the pipelines. Related work comprises Section 2.9, and is followed by conclusions. We provide a set of appendices.

2.3 Data Model for Networked Experiments and for Modeling and Simulation

2.3.1 Formal Data Model

One of the challenges is to formally represent social experiments and models. Here, we propose a general adaptive abstract data model for networked social experiments with the five characteristics itemized in Contribution 2 of Section 3.2.6. The purpose of the data model, provided in Table 2.1, together with the computational model of Section 2.4 and the pipeline model in section 2.6, is to provide formal representations for experiments and MAS, and their iterative interactions. Here, we focus only on the data model. Given a description of an experiment or model, one can determine whether our system can be applied; and in reverse, given a phenomenon of study, these models can be used to formulate experiments and models. The "data common specification" in Figure 2.1 (blue) is produced from this model. For ease of exposition, we describe the data model in terms of an experiment, but it is equally valid for modeling and simulation.

Experiment Schema. Per Table 2.1, each experiment has the following elements: a unique id *exp_id*, a number n_p of phases, a number n of players, a *t_begin* timestamp for the beginning of the game, a *t_end* timestamp for the end of the game. Each player has a unique id v_i for identification. A set of players in an experiment is defined by $V = \{v_1, ..., v_n\}$. An experiment has n_{sa} player attributes defined for each player. Player attributes Ω are invariant across phases (e.g., age and education level that might be solicited through a survey).

Phase Schema. An experiment may be composed of any number of phases. All phases have a common schema, per Table 2.1, but particular phase schema instances (e.g., variables) may be different across phases.

Each phase schema has the following elements, a unique id *ph_sch_id*, the number $i_{n_p} (1 \leq i_{n_p} \leq n_p)$ of the phase in the sequence of phases, a *t_ph_begin* timestamp at the beginning of the phase, number of time increments in the phase t_p and the unit of time u_p of one time increment. Each phase represents the interaction structure among players as a network $G(V', E')$ with meanings of edges Λ. Node attributes Γ and edge attributes Ψ over all nodes and edges capture attribute changes in time. Players and edges may have initial conditions B^v and B^e, respectively. A is the set of permissible player actions. An action tuple T_i, which captures pair-wise interactions between players, may be intimately tied to the attribute sequences Γ and Ψ of a phase because action tuples, for example, may cause or be caused by changes in node and edge attributes. In essence, Γ and Ψ can be viewed as sequences of node and edge states. Items 8 through 11 and 13 of the phase schema in Table 2.1 follow the same basic pattern, to capture features by node or edge, and by time. There is a sequence of values for a particular node v_j or edge e_j (e.g., Γ_j, Ψ_j, B^v_j, B^e_j, and T_j). Each entry in these sequences can be scalars, sequences, sets, or other structures. Then, these entries are

Table 2.1: Definition of our abstract data model. The experiment schema describes experiment parameters. The phase schema structure describes parameter types for an experimental phase; an experiment can have any number n_p of phases. Particular instance variables within the phase schema structure can vary across phases. We use *experiment* throughout in the table and text for ease of exposition, but the data model is also used for *(simulation) models*.

#	Parameters	Symbols	Description
Experiment Schema			
1	Experiment id	exp_id	Unique id for an experiment.
2	Number of phases	n_p	Number of phases in the experiment.
3	Number of players	n	The number of unique players over all phases in the experiment.
4	Begin time	t_begin	Timestamp of experiment beginning.
5	End time	t_end	Timestamp of experiment ending.
6	Set of player IDs	V	$V = \{v_1, \ldots, v_n\}$. Set of players over all phases; $v_i \in V$ is a unique id for player.
7	Player attributes	Ω	$\Omega = \cup_{j=1}^{n} \Omega_j$. $\Omega_j = (\omega_{j1}, \omega_{j2}, \ldots, \omega_{j,n_{sa}})$ is the sequence of n_{sa} attributes for $v_j \in V$.
Phase Schema *Structure*			
1	Phase schema id	ph_sch_id	Unique id for phase schema.
2	Sequence	i_{n_p}	$1 \leq i_{n_p} \leq n_p$. Element of the sequence of phases of the experiment.
3	Phase begin	t_ph_begin	Timestamp of phase beginning.
4	Phase duration	t_p	Number of time increments in the phase.
5	Unit of time	u_p	Time unit of one time increment (e.g., seconds, days).
6	Network definition	$G(V', E')$	Node set $V' = \{v_1, ..., v_\eta\}$ and edge set $E' = \{e_1, \ldots, e_m\}$, where $V' \subseteq V$ may not be all nodes (players) in the system, and edge $e_i = \{v_j, v_\ell\}$ with $v_j, v_\ell \in V'$. Note that E' may be empty.
7	Meaning of an edge.	Λ	Set Λ of string representations $\lambda \in \Lambda$ stating the meaning(s) of an edge (e.g., $\lambda =$ "communication channel" or "influence").
8	Node attributes for a phase.	Γ	$\Gamma = \cup_{t=0}^{t_p}\big(\cup_{j=1}^{\eta} \Gamma_j(t)\big)$. $\Gamma_j(t) = (\gamma_{j1}(t), \gamma_{j2}(t), \ldots, \gamma_{j,\eta_v}(t))$ is the sequence of η_v attributes for $v_j \in V'$ in the phase i_{n_p} at time t. Γ is a triple nested sequence in attributes, player ID, and time.
9	Edge attributes for a phase.	Ψ	$\Psi = \cup_{t=0}^{t_p}\big(\cup_{j=1}^{m} \Psi_j(t)\big)$. $\Psi_j(t) = (\psi_{j1}(t), \psi_{j2}(t), \ldots, \psi_{j,\eta_e}(t))$ is the sequence of η_e attributes for $e_j \in E'$ in the phase i_{n_p} at time t. Ψ is a triple nested sequence in attributes, edge ID, and time.
10	Initial conditions for nodes	B^v	Nodes: $B^v = \cup_{j=1}^{\eta} B_j^v$. $B_j^v = (b_{j1}, b_{j2}, \ldots, b_{j,\mu_v})$ is the sequence of μ_v initial conditions for the phase, for $v_j \in V'$; $\mu_v \geq 0$.
11	Initial conditions for edges	B^e	Edges: $B^e = \cup_{j=1}^{m} B_j^e$. $B_j^e = (\beta_{j1}, \beta_{j2}, \ldots, \beta_{j,\mu_e})$ is the sequence of μ_e initial conditions for the phase, for $e_j \in E'$; $\mu_e \geq 0$.
12	Action set	A	$A = \{a_1, a_2, ..., a_{n_a}\}$. Set of n_a actions that each player can execute, over time, any number of times, during a phase, where $n_a \geq 0$.
13	Action sequence	T	$T = \cup_{t=0}^{t_p}\big(\cup_{k=1}^{\eta} T_k\big)$. $T_k = (\sigma_i, a_j, v_k, v_\ell, t_o, py_q)$ is the schema for an *action tuple*. σ_i is a string that is a unique identifier for an action sequence. Action $a_j \in A$ is initiated by node $v_k \in V'$, and v_ℓ is the target node of the action, with edge $e = \{v_k, v_\ell\} \in E'$. $t_o \in \mathbb{R}$ is the time of the action ($0 \leq t_o \leq t_p$); py_q is the payload represented as a JSON schema.

sequenced over time through the union of entries over time, from time 0 through t_p. The exceptions are the initial conditions B_j^v and B_j^e, because by definition, they are specified only at time 0.

2.3.2 Illustrative Instances of Data Model Parameters

We provide a few illustrative examples of data model elements. A 3-phase game is described in Section 2.8, Case Study 1. Phase 2 is a group anagram (word construction) game. In phase 2, a network $G(V', E')$ is imposed on the players, where the meaning λ of a edge is a communication channel to request letters and reply to requests. A node initial condition b_{j1} for a game is the number of alphabet letters a player receives at the beginning of the phase to use in forming words, and b_{j2} is the set of letters. Each player can execute any action from the action set A, such as request a letter from a neighbor.

We now provide an example of an action tuple of an action sequence. If player v_i requests letter "z" (a request is action $a_\ell \in A$) from player v_j at time t_o, which initiates a sequence of actions (because there may be a subsequent letter reply from v_j) then the action tuple is $T_i = (\sigma_i, a_\ell, v_i, v_j, t_o, \text{“}z\text{”})$. Here, $\sigma_i = v_i + \text{“}-\text{”} + \textit{counter}$ (e.g., a string) is a concatenation of the initiator's (v_i's) ID with a player-specific counter to form a unique ID for the sequence of actions that is initiated with the letter request. If v_j responds with "z," then this (second) action tuple will use the same σ_i as the first element of the tuple, consistent with T_i. This is how action tuples are defined and identified in data processing, in forming action sequences T for a phase.

2.3.3 From Abstract Data Model to Software Specification

Ours is an abstract mathematical data model. There are several reasons for our choice of model representation. First, a mathematical representation is more abstract (which means, among other things, more versatile and flexible) in its use. Second, it corresponds much more closely to the information required for pipeline capabilities, and enables compact representations of simulation models. Third, it is naturally amenable to translation into other data model representations that are more common in software. We elaborate on each of these.

1. Abstract representation. An element of a sequence can *abstractly* represent any type of data, including scalars, vectors, sets, tensors, and complicated data structures (that may be implemented via a JSON schema). For example, consider γ_{j2} of Γ_j of Γ in Table 2.1, which is an attribute for node or player $v_j \in V'$. This variable might represent a 2-D matrix or a set. Furthermore, if the representation needs to be changed, it is much easier to do so with an abstract representation.

2. Compactness. Consider a capability for a simulation model, as part of a pipeline: multiplying two matrices, M_1 and M_2. A mathematical representation is simply $M_1 \cdot M_2$ or $M_1\ M_2$. A pseudo code representation for this functionality would require some five lines of code including three FOR loops. Clearly, $M_1 \cdot M_2$ is far more compact.

3. Principled transitions (progression) among software artifacts. The steps in

progressing from a mathematical data model to a software model are shown in Figure 2.3. Experiment and phase schemas in Table 2.1 contain data structures. Instances of our abstract data model (generated from the execution of an experiment) can be represented as entity-relationship diagrams, which are *conceptual* or *logical* data models. Examples are relational models [56], object-oriented models like Object Definition Language (ODL) [252] or Unified Modeling Language (UML) [14], or data structure diagrams [21], among others. A UML representation of an entity-relationship diagram for our abstract data model is presented in Figure 2.4. UML is the industry-standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems [14]. All of the structures from the abstract data model of Table 2.1 are translated into a entity-relationship diagram in unified modeling language (UML) form, demonstrating that the abstract data model can be translated into standard forms of data models more amenable for software development.

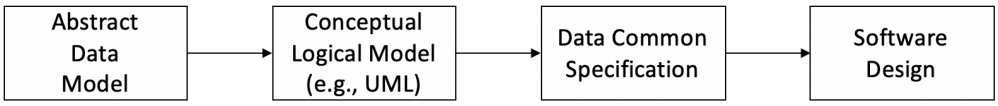


Figure 2.3: Sequence of data models for reasoning about experiments and modeling and simulation. We advocate for pre-pending the abstract data model to the front end of the model process, as shown here. Table 2.1 shows our abstract data model and Figure 2.4 shows this data model translated into a entity-relationship diagram in unified modeling language (UML) form. The table and figures in Appendix A.1 (which support Section 2.7) show the Data Common Specification for our software design.

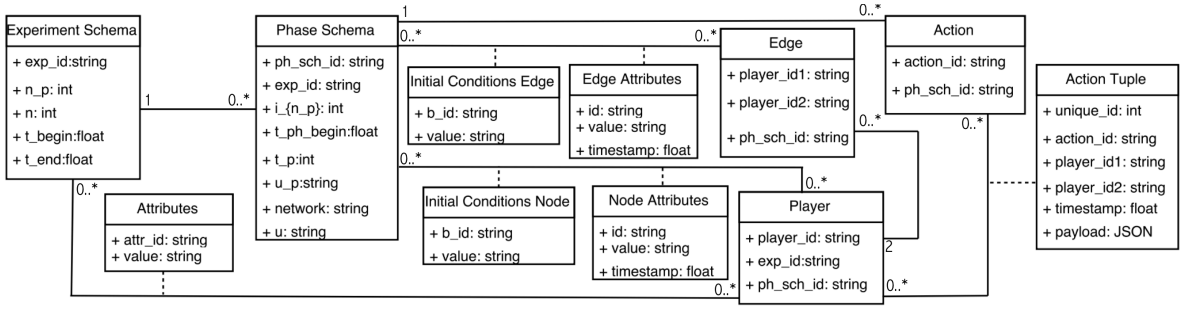


Figure 2.4: Data model of Table 2.1 translated into a entity-relationship diagram in unified modeling language (UML) form. This illustrates that the abstract data model can be translated to customary forms of data models more amenable for software development.

2.3.4 Data Common Specification

Every JSON input file in the pipelines needs a corresponding JSON schema for the verification of formats. For our Data Common Specification there are five classes of input every

experiment needs to define. The formal data model in Section 2.3.1 specifies that an experiment can have any number n_p of phases and a different set of players with an action set for each phase. Appendix A.1 defines through JSON schemas the formats and compositions of the elements of the Data Common Specification. These are implementation aspects of our pipelines. These are also the types of files we use in the case studies in Section 2.8.

2.4 Graph Dynamical System Model: a Formal Framework for NESS Experiments and Agent-Based Models

In this section, we present a computational model known as discrete **graph dynamical systems** (GDS). This model formalizes experiments and MAS by capturing the interactions between pairs of players. That is, we use GDS to specify, build, and execute experiments and simulators of experiments (and of other conditions). We use this GDS model because it is correspondent with the data model of Section 2.3 and is a general model of computation [23, 24]. We also achieve correspondence between experiments and MAS, per Figure 2.2. A number of other formal models could have been used; we find GDS to be a natural model for specifying NESS.

2.4.1 Formal Model

A synchronous **Graph Dynamical System** (GDS) [174] S is specified as $S = (G, W, F, U)$, where we define each in the following. (a) $G \equiv G(V, E)$ is an undirected graph with $n = |V|$, and represents the underlying graph of the GDS, with node set V and edge set E. The nodes represent agents in a system or test subjects in our experiments, and the edges represent pair-wise interactions between agents. (b) W is the state space, which is the union of the state space W^v for nodes and the state space W^e for edges; i.e., $W = W^v \cup W^e$. These are the states that nodes and edges can take during the dynamics. Each undirected edge $\{v_i, v_j\} \in E$, with $v_i, v_j \in V$, can be represented by two directed edges: v_i to v_j, $e_{ij} = (v_i, v_j)$, and $e_{ji} = (v_j, v_i)$. (c) $F = (f_1, f_2, \ldots, f_n)$ is a collection of functions in the system. Function f_i denotes the **local function** associated with node v_i, $1 \leq i \leq n$, that describes how v_i updates its state. (d) U is the method which describes how the local functions are ordered at each discrete time. Here, we use the synchronous update scheme where all f_i execute in parallel.

Each node of G has a state value from W^v. Each edge of G has a state value from W^e. Each function f_i specifies the local interaction between node v_i and its neighbors in G. The inputs to function f_i are the state of v_i, the states of the neighbors of v_i, and the states of the edges outgoing from v_i in G. Function f_i maps each combination of inputs to $s'_i \in W^v$

for v_i, and to $s'_{ij} \in W^e$ for each directed edge e_{ij}. s'_i becomes the next state of node v_i, and s'_{ij} becomes the next state of e_{ij} edge. These functions are executed in parallel at each time step t.

Here, we provide a more formal description of GDS, based on the overview above. We assume here that only nodes have vertex states. Let $G(V, E)$ be a graph with node set V and edge set E, and where $n = |V|$. Each node v_i has a state s_i Let $N(v_i)$ be the sequence of vertices adjacent to v_i in G, including v_i itself, so that $1 \leq |N(v_i)| \leq n$ for each $v_i \in V$. That is,

$$N(v_i) = (v_{v_i,1}, v_{v_i,2}, \ldots, v_{v_i,d(v_i)+1}) \ , \tag{2.1}$$

where $d(v_i)$ is the degree of v_i in G. Let $s(v_i)$ be the sequence of vertex states of the vertices in $N(v_i)$, so that $1 \leq |s(v_i)| \leq n$ for each $v_i \in V$, i.e., and $d(v_i) = |N(v_i)| - 1$.

$$s(v_i) = (s_{v_i,1}, s_{v_i,2}, \ldots, s_{v_i,d(v_i)+1}) \ . \tag{2.2}$$

We call $s(v_i)$ the *restricted state* of v_i. The **system state** or **configuration** C of a GDS is the n-vector $C = (s_1, s_2, \ldots, s_n)$.

A local function $f_i \colon (W^v)^{d(v_i)+1} \to W^v$ quantifies the dynamics of node v_i by computing v_i's next state s'_i using the states of nodes in its closed 1-neighborhood as

$$s'_i = f_i(s(v_i)) \ . \tag{2.3}$$

Updating the entire set of nodes in G at some time t is accomplished with the **GDS mapping**

$$\mathbf{F} \colon (W^v)^n \to (W^v)^n \ . \tag{2.4}$$

For the *synchronous* update scheme, where all f_i, $i \in \{1, 2, \ldots, n\}$, execute in parallel, the GDS mapping is defined by

$$\mathbf{F}(s_1, s_2, \ldots, s_n) = (f_1(s(v_1)), f_2(s(v_2)), \ldots, f_n(s(v_n))) \ . \tag{2.5}$$

In a simulation, we compute successive system states using this last equation, as $C(t+1) = \mathbf{F}(C(t))$, where $C(t)$ is the system state or configuration at time t, and $C(t+1)$ is the next system state.

To make this explicit, we now cast the preceding formalism into a pseudo-algorithm in computing the dynamics of a GDS. Let us assume for simplicity that only nodes possess state, and edges do not. At any time t, the **configuration** $C(t)$ of a GDS is the n-vector $C(t) = (s_1^t, s_2^t, \ldots, s_n^t)$, where $s_i^t \in W^v$ is the state of node v_i at time t $(1 \leq i \leq n)$. In a synchronous GDS, all nodes compute and update their next state *synchronously*. A GDS transition from one configuration $C(t)$ to a next configuration $C(t+1)$ in parallel at each time t can be expressed as follows,

for each node $v_i \in V$ **do in parallel**

(i) Compute the value of f_i (Equation (2.3)) using states in $C(t)$ and assign it to s'_i.

(ii) Assign s'_i as the next state of v_i in $C(t+1)$.

end for

Note that if the f_i are stochastic, $C(t+1)$ may not be unique. The extension to the update of edge states s'_{ij} is natural.

2.4.2 Use of GDS in This Work

For our purposes, the benefits of GDS are multi-fold. First, this computational model is correspondent with the data model in Section 2.3. For example, $G(V', E')$, per phase, in Table 2.1 corresponds to the graph $G(V, E)$ of the GDS. Node W^v and edge W^e state spaces in the model are represented as (subsets of) the node (Γ) and edge attributes (Ψ), respectively. Attributes may have additional parameters that are not part of the node or edge state, such as gender and age. Action tuples may be part of the state. Second, GDS is also useful in generating agent-based models (ABMs) and other types of models that can be generated from experimental data and/or first principles. We do this, and show illustrative results from these models in Section 2.8. Third, the unified framework for experiments and models enables us to reason more effectively about system behavior; e.g., using this framework, properties about different types of dynamical systems have been proven (e.g., [6, 23, 24, 174]).

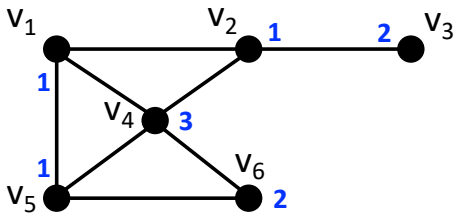
2.4.3 Example GDS and Resulting Dynamics: Threshold Systems

We provide an example of a GDS and the dynamics that it generates. We use a threshold contagion system, motivated by the work [47, 107, 162] in the social sciences. Also, we use this model in the second case study of Section 2.8. A progressive threshold system works as follows. The network $G(V, E)$ is provided at the left in Figure 2.5. The valid state set W for a node is $W = W^v = \{0, 1\}$, where state 0 means that a node does not possess a contagion and state 1 means that a node possesses the contagion and will assist in transmitting it. The threshold local function works as follows. Each node v_i is assigned a threshold $0 \leq \theta_i \leq d_i+1$, where d_i is the degree of v_i in G. If the state s_i of node v_i at time t is 1 (i.e., $s_i^t = 1$), then the output of f_i is 1 (that is, a node in state 1 at t remains in state 1 at $(t+1)$). If $s_i^t = 0$, then $s_i^{t+1} = f_i = 1$ if at least θ_i of v_i's neighbors are in state 1 at t; otherwise, $s_i^{t+1} = f_i = 0$. That is,

$$s_i^{t+1} \;=\; f_i(s^t(v_i)) = \begin{cases} 1 & \text{if } s_i^t = 1\ , \\ 1 & \text{if } s_i^t = 0 \text{ and } n_1 \geq \theta_i\ , \text{ or} \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

where $s^t(v_i)$ is the sequence of states in the closed neighborhood of v_i at time t, and n_1 is the number of nodes in state 1 in $s^t(v_i)$. This is a deterministic GDS.

The dynamics evolve as follows; see Figure 2.5. We specify as initial conditions that v_1 has the contagion at $t = 0$, i.e., $s_1^0 = 1$; all other nodes do not have it. See $C(0)$ in Figure 2.5, where only $s_1^0 = 1$. At $t = 1$, $s_2^1 = f_2(s_1^0, s_2^0, s_3^0) = 1$ because $\theta_2 = 1$ and $s_1^0 = 1$, and $s_2^0 = s_3^0 = 0$. So, the threshold for v_2 is just met by v_1. For the same reason, $s_5^1 = f_5(s_1^0, s_4^0, s_5^0, s_6^0) = 1$ (because $s_1^0 = 1$; v_5 and all other neighbors of v_5 are in state 0). No other node will change state at $t = 1$ and therefore $C(1)$ has three nodes in state 1 at $t = 1$. At $t = 2$, v_4 will change state, even though its threshold is large ($\theta_4 = 3$) because three of v_4's neighbors (v_1, v_2, and v_5) are now in state 1. This is the only node that changes state at $t = 2$ and so $C(2)$ is as shown in Figure 2.5. The same reasoning applies to the transitions of other node states. Note that v_3 will never transition because its threshold (2) is greater than the number of its neighbors (1). Also note that the system reaches a *fixed point* at $t = 3$ because no further state changes are possible.



Time, t	Configuration, C(t)
0	(1, 0, 0, 0, 0, 0)
1	(1, 1, 0, 0, 1, 0)
2	(1, 1, 0, 1, 1, 0)
3	(1, 1, 0, 1, 1, 1)
4	(1, 1, 0, 1, 1, 1)

Figure 2.5: Network $G(V, E)$ for a GDS example, with $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$. Thresholds θ_i are provided for nodes v_i, in blue, by the respective nodes. The local functions f_i are threshold functions for $v_i \in V$, $1 \leq i \leq 6$; see text for details. The discrete system dynamics are given by the configurations at successive times from 0 to 4, at the right in the figure. Each configuration is given by $C(t) = (s_1^t, s_2^t, s_3^t, s_4^t, s_5^t, s_6^t)$. The system reaches a fixed point at time $t = 3$, as evidenced by no change in the configuration in going from $t = 3$ to $t = 4$.

2.5 Hierarchical Pipeline Conceptual View

2.5.1 Pipeline Compositions

Our system is composed of all elements of Figure 2.1. We specifically separate the experimental platform from the pipelines so that the system can be used with different experimental software platforms, through the Data Common Specification, via the data model of Section 2.3. The full system is shown with five pipelines in gray: "Experimental Data Transfor-

mation," "Data Analytics," "Property Inference," "Modeling and Simulation," and "Model Evaluation and Prediction." An iteration of the loop, as shown in Figure 2.1, may use any number of the five pipelines, and any number of functions within them, for flexible composability, consistent with data dependencies [133]. Case studies demonstrating the use of the pipelines are provided in Section 2.8.

2.5.2 Pipelines

The five pipelines of Figure 2.1 now described. (1) The Experimental Data Transformation Pipeline transforms experimental raw data into a data common specification. (2) The Data Analytics Pipeline analyzes temporal interactions among players to identify patterns and phenomena in the data. Direct and derived data are used as input for (3) the Property Inference Pipeline. This pipeline generates property values for parameters of simulation models, often by combining data from multiple experiments. The simulation models (e.g., ABMs) are built off-line and software implementations of these models are part of (4) the Modeling and Simulation Pipeline. This pipeline invokes the code to run simulations, using the generated property values, as well as network descriptions, initial conditions, and other inputs. Simulations may model completed or contemplated experiments, or other scenarios beyond the scope of experiments. (5) The Model Evaluation and Prediction Pipeline combines simulation results across multiple (stochastic) executions and performs comparisons among sets of data. In one case, experimental data and model predictions may be compared. In another case, results from two models may be compared. One objective may be to predict beyond game data (counter-factuals) and propose further investigations suggested by analysis findings.

Each pipeline is currently a sequential composition of functions. This composition is specified by an analyst through a job definition. Similarly, compositions of the pipelines of Figure 2.1 are specified by an analyst. The pipeline process takes care of file dependencies between functions. Also, it validates the input and output data of functions, described below. The structure of a pipeline is shown in Figure 2.6, where function h_1 takes two inputs and generates three outputs (two are inputs to function h_2 and one is an input to function h_3); function h_2 generates two outputs, one of which is an input to h_3. Note that the pipelines control execution of functionality. Execution control consists of a pipeline invoking functions sequentially, as illustrated in Figure 2.6. Additional details are in Sections 2.6 and 2.7. Other control structures are being added.

2.5.3 Functions Within Pipelines

Functions are designed as microservices (i.e., modular software with limited scope) within pipelines. They provide a range of capability from simple plotting routines to cleaning and organizing, storing and accessing data sets, and inferring properties and running simulations.

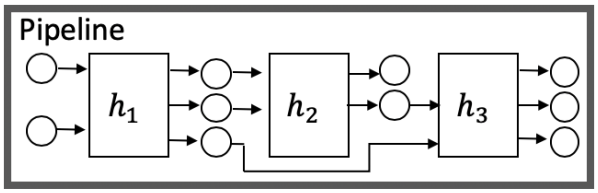


Figure 2.6: Functions, or h-function, h_i, $1 \leq i \leq 3$ (implemented as software) within a pipeline. Pipelines control the execution order of functions and the inputs and outputs for each function, through a pipeline job specification. Circles in the figure denote input and output digital objects, such as ASCII files of database tables.

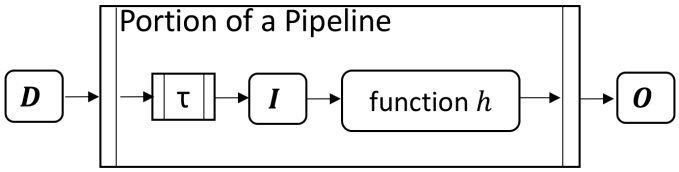


Figure 2.7: An arbitrary software function h. Input data instances D may have to be transformed by transformation code τ to conform to required inputs I. Inputs and outputs are subjected to verification through comparisons with specified schema (not shown here).

These are not exhaustive; users may add other functions and continue community-based development. This concept is illustrated in Figure 2.6. Functions h_i ($1 \leq i \leq 3$) in Figure 2.6 take inputs and generate outputs. Currently, inputs and outputs are files, but may include other digital objects, such as database table entries. Figure 2.7 drills down to show details for a function.

Figure 2.7 shows execution details associated with each function h. Input data (D) (e.g., in the form of an ASCII data file that may be raw data or output from a preceding function) may need to be transformed into a format required by h. This transformation is performed by transformation code τ, which generates the input (I) in the required format. This input object (I) conforms to a JSON specification to ensure compliance for input to h.

2.5.4 Microservices

Our functions map directly to microservices. Appendix A.5 addresses characteristics, benefits, and comparisons of microservices. We provide details of microservices because they are the fundamental execution units within our pipelines.

2.6 Formal Pipeline Model

With the conceptual view in Section 2.5, we now provide a formal model for pipelines. In Section 2.7, we address pipeline implementation which is based on our model. First, we provide the model. Then, we describe the execution of a pipeline. Finally, we describe some benefits of the formal model.

2.6.1 Model

Let $\mathcal{P}$ be a **collection of pipelines** with **pipeline** $P \in \mathcal{P}$ represented as $P(Q, S_{ID}, S, T, H)$. Here, Q is a set of data elements $q \in Q$; S_{ID} is a set of mappings $s_{ID} \in S_{ID}$ of data to schema; S is a set of schema $s \in S$; T is a set of data transformations $\tau \in T$; and H is a sequence of h-functions $h \in H$.

Data Q include **input data representations** $K \subseteq Q$ and **output data representations** $L \subseteq Q$. A data element $k \in K$ is an input to some function and data element $\ell \in L$ is an output of some function. An element $q \in Q$ may be both an input data element k and an output data element ℓ. We have $Q = K \cup L$. Moreover, the intersection of K and L will almost always be non-empty, i.e., $K \cap L \neq \emptyset$, because in a pipeline, an output element of an h-function may be an input to a subsequent h-function. We use q to denote an input data element, an output data element, or both.

We now address data schema and data verification. To verify that an instance of a data representation q is valid, an analyst provides a **schema ID mapping** $s_{ID} \colon Q \to S$ defined by a mapping from each data representation q to a unique schema $s \in S$. That is, $s = S_{ID}(q)$. An element $s \in S$ is a **schema** $s \colon Q \to \{0, 1\}$ that takes as input an instance of a data representation q and outputs a 1 when the instance of q conforms to the schema s (i.e., q is successfully verified against s), and outputs a 0 otherwise. That is, $s(q)$ returns a 0 or 1.

A set T of data transformation functions transform data elements $q \in Q$. A data transformation function $\tau \in T$ takes as input one or more data elements and outputs precisely one data element. The role of a data transformation function is to operate on inputs and outputs from one or more h-functions (defined below) and produce a new data element that is in the required format for input to another h-function. Hence, a **data transformation function** $\tau \colon Q^{n_\tau} \to Q$ is defined as $q' = \tau(q_1, q_2, \ldots, q_{n_\tau})$ where $q' \in Q$ and $q_j \in Q$, $1 \leq j \leq n_\tau$.

An h**-function** (or **function**) $h \in H$ represents a microservice that performs some small unit of work in a pipeline. An h-function takes as input a sequence of n_i input data representation elements and computes a sequence of n_o output data representation elements. Each input data element $k_j \in K$, $1 \leq j \leq n_i$, has been verified through $s_{ID} \in S_{ID}$ and element $s \in S$, so that the inputs to h are valid (i.e., so that the appropriate $s \in S$ outputs a 1 for each instance of k_j). Also, each of these input data elements may have been generated by transforming data into the required format, using one data transformation function $\tau \in T$. Each h outputs

a sequence of instances of $\ell_j \in L$, $(1 \leq j \leq n_0)$ which are also verified through $s_{ID} \in S_{ID}$ and elements $s \in S$, so that the sequence of outputs from h are valid (i.e., so that the appropriate $s \in S$ outputs a 1 for each instance of ℓ_j). Thus, we have the following. An h-function is $h\colon K^{n_i} \rightarrow L^{n_o}$ defined by $(\ell_1, \ell_2, \ldots, \ell_{n_o}) = h(k_1, k_2, \ldots, k_{n_i})$, where $k_j \in K$, $1 \leq j \leq n_i$, and $\ell_j \in L$, $1 \leq j \leq n_o$.

It is useful to define the composition of all h-functions within a pipeline, because it identifies the order in which h-functions execute. It naturally identifies the (input) data files that must exist before the pipeline starts (e.g., some input files for some h-functions are not specified initially because they are generated by other [*preceding*] h-functions); and which output files are generated. As the preceding model description indicates, one data transformation function may need to be executed on each input before each h-function is invoked, to put each input data element k into the required format for h. If there are n_i inputs to h, then the number of data transformation functions is at most n_i. Hence, executing one h-function can be thought of as a composition of functions $(\tau^*, h) = (h \circ \tau^*)$, where τ^* represents zero or more transformation functions that are required to put all inputs for h into the proper formats for execution of h. A **composition of n_f h-functions** $\mathcal{H}\colon K^{n_{p,i}} \rightarrow L^{n_{p,o}}$ is defined by $\mathcal{H} = (h_{n_f} \circ \tau^*_{n_f}) \circ (h_{n_f-1} \circ \tau^*_{n_f-1}) \circ \cdots \circ (h_2 \circ \tau^*_2) \circ (h_1 \circ \tau^*_1)$, where $(\ell_1, \ell_2, \ldots, \ell_{n_{p,o}}) = \mathcal{H}(k_1, k_2, \ldots, k_{n_{p,i}})$. We define $K^* = K^{n_{p,i}}$ and $L^* = L^{n_{p,o}}$ as short-hand. Thus, the $n_{p,i}$ input files that must exist before the pipeline is invoked are represented by K^*. The $n_{p,o}$ pipeline outputs are represented by L^*. It is often convenient to represent $\mathcal{H}$ as the (ordered) sequence $((\tau^*_1, h_1), (\tau^*_2, h_2), \ldots, (\tau^*_{n_f-1}, h_{n_f-1}), (\tau^*_{n_f}, h_{n_f}))$, where the ordering gives the order of execution.

2.6.2 Execution of a Pipeline

With the formalism of Section 2.6.1, the dynamics of pipeline execution are now presented. Figure 2.8 contains the algorithm. The algorithm steps through each $h_i \in \mathcal{H}$ and for each input of h_i, determines whether it needs to be created by transforming one or more other data elements. If so, the transformation function is executed. At this point the required input data exist, and h_i is invoked and the output files are generated. These outputs are stored. Note that at various points, data file formats are verified by using schema verification functions s.

The description thus far in this section is focused on a single pipeline. However, the model is equally valid across pipelines. In fact, grouping sets of h-functions into multiple pipelines, as we do herein, is largely a matter of practicality, and aids in software system organization and in reasoning about such systems. However, from Section 2.6.1 and this Section 2.6.2, it should be clear that all data transformation functions and h-functions could be put into a single $\tau*$ large pipeline.

Algorithm 1: PIPELINE EXECUTION.

Input:
h-functions of $\mathcal{H}$ to execute for the pipeline P. Data transformation functions T to execute. The set K^* of input files for the pipeline. Identification of the n_i inputs $K^{n_i} = (k_1, k_2, \ldots, k_{n_i}) \subseteq K*$ and n_o outputs $L^{n_o} = (\ell_1, \ell_2, \ldots, \ell_{n_o}) \subseteq L^*$ for each h-function. Inputs $q_1, q_2, \ldots, q_{n_\tau} \in Q$ and output $q' \in Q$ for each data transformation function $\tau \in T$, for each $h_i \in \mathcal{H}$. The set S of schema $s \in S$ for verification of data elements q. The set S_{ID} of schema ID elements $s_{ID} \in S_{ID}$ for the mapping of data elements q to schema s.
Output: The output files L^* generated by the pipeline P, represented by $\mathcal{H}$.
Steps:

1. **for each** $h_i \in \mathcal{H}$ **do**
   (a) Obtain the inputs n_i inputs $(k_1, k_2, \ldots, k_{n_i})$ for h_i.
   (b) **for each** $k_i \in K^{n_i}$ **do**
      i. **if** k_i requires data transformation prior to input to h_i **then**
         A. Let $k'_1, k'_2, ..., k'_{n_\tau}$, be the inputs to the transformation function τ such that $k_i = \tau(k'_1, k'_2, ..., k'_{n_\tau})$.
         B. Obtain the schema to verify each k'_i, as $s = s_{ID}(k'_i)$.
         C. Compute each $s(k'_i)(1 \leq i \leq n_\tau)$. If $s(k'_i) = 1$, then k'_i is verified. If $s(k_i) = 0$, then k_i is not verified; an error is found, and the pipeline $\tau(k'_1, k'_2, ..., k'_{n_\tau})$ terminates.
         D. Use the data transformation function τ to compute the input k_i for h_i, in the proper format, according to $k_i = \tau(k_i)$.
      ii. Obtain the schema to verify k_i, as $s = s_{ID}(k_i)$.
      iii. Compute $s(k_i)$. If $s(k_i) = 1$, then k_i is verified. If $s(k_i) = 0$, then k_i is not verified; an error is found, and the pipeline terminates.
   (c) Invoke function h_i and compute $(\ell_1, \ell_2, \ldots, \ell_{n_o}) = h^{ubi}(k_1, k_2, \ldots, k_{n_i})$.
   (d) Check the format of each output $\ell_i(1 \leq i \leq n_0)$ by obtaining the schema $s = s_{ID}(\ell_i)$ and invoking $s(\ell_i)$. If $s(\ell_i) = 1$, then the output file format is verified. Else ℓ_I is not verified, which is an error, and the pipeline gracefully terminates.
   (e) Store the outputs $(\ell_1, \ell_2, \ldots, \ell_{n_o})$ in Q, which may be used as inputs for subsequent $h_j \in \mathcal{H}$, $(j \neq i)$.
   (f) Store the outputs $(\ell_1, \ell_2, \ldots, \ell_{n_o})$ in L^*, which is the set of outputs from the pipeline.
2. Return L^*.

Figure 2.8: Steps of the Algorithm PIPELINE EXECUTION.

2.6.3 Mapping of Model onto the Software System

One reason for the particular development in Section 2.6.1 above is that it parses the model into components that are the responsibility of the pipeline framework, software that users put into a pipeline, and user-supplied information regarding data. Input data K for a pipeline or a collection of pipelines must be supplied by an analysts, or come from some previous analysis. While we assume here that instances of the elements of K are ASCII format (via files and data), output instances of representations of L may be ASCII format or some other binary format (e.g., for files containing data plots). There are natural extensions to other data formats, such as database tables.

The schema ID mapping and schema themselves are provided by the analyst to ensure that input and computed results conform to specified formats and contain the proper types of information. The execution of schema to verify data representation instances is the responsibility of the pipeline (not the functions). Data transformation functions and h-functions are executable software, and may be stand-alone executables that form processes. They are provided by an analyst or software developer. It is the pipeline's responsibility to invoke the correct functions and in the correct order, and to access the proper input files and to store the resulting output files, all of which are specified in a human-generated **pipeline configuration** file (addressed below). Functions are responsible for generating correct outputs.

2.7 Pipeline Implementation

2.7.1 Pipeline Configuration File

To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. Table 2.2 overviews the entire pipeline configuration file with a definition for each parameter. In the Configuration File, the following parameters, experiment, phasedesc, phase, action, and player, specify the JSON schema files for each component in the data common specification from Section 2.3.4. The parameter functions defines the available functions to run in the pipeline; also defines the input values for each function. Appendix A.2 contains a detailed example of a configuration file.

Figure A.11 shows the schema for a configuration file that specifies how to compose and execute one or more functions of a pipeline. In Figure A.11, there are up to five functions available and the required parameters for each function are defined; the enumeration is the list of valid candidate values that can be specified for functions in a specific pipeline.

Table 2.2: Configuration input file description. See Appendix A.2 for details.

#	Parameter	Description
1	experiment	Experiment Schema JSON file location. See Figure A.6.
2	phasedesc	Phase Description Schema JSON file location. See Figure A.7.
3	phase	Phase Schema JSON file location. See Figure A.8.
4	action	Action Schema JSON file location. See Figure A.9.
5	player	Player Schema JSON file location. See Figure A.10.
6	functions	The parameters inside varies for every function. Figure A.11 shows a definition for five functions.

2.7.2 Pipelines

The pipelines software (written in Python) performs these operations: (*i*) reads and parses the configuration file; (*ii*) controls accessing input files, JSON schema files, transformation codes, and h-functions; (*iii*) checks files against their JSON schema and terminates gracefully if a verification fails; (*iv*) invokes the proper transformation functions (if applicable), and (*v*) invokes the proper *h*-functions in their proper order (and any other operations), and (*vi*) error handling.

JSON schema are used in various ways: (*i*) to verify the configuration file, (*ii*) to verify inputs to transformation functions, (*iii*) to verify the outputs of transformation functions (which are inputs to the *h*-functions), and (iv) to verify the outputs from the *h*-functions (we use the term *h*-function to differentiate them from transformation functions).

Figure 2.9 shows the pipeline structure and elements of the execution chain. Here we show two pipelines. Table 2.3 provides a list and description of the structures and elements from the execution of a generic pipeline in Figure 2.9. A pipeline is executed according to the user-supplied configuration file for it. In the left side of Figure 2.9, inside Pipeline i, there is the configuration input file JSON schema.

In Figure 2.9, the first inner dashed box (Pipeline i Control), describes the pipeline software that performs the operations stated above. First, the pipeline receives a configuration input file and the corresponding input files. All files are JSON files. The JSON schema for the configuration file in Figure A.11 was described in Section 2.7.1, so the pipeline is ensured to have proper data formats. Second, the pipeline performs the validation of the JSON instances from the input files against their corresponding JSON schemas. Third, the pipeline invokes the functions specified in the configuration input file. The heart of the pipelines is invoking the h-functions and the associated transformation functions (if appropriate). The second inner dashed box inside (Pipeline i Function Execution) controls all *h*-functions. There can be any number of functions per pipeline. In this example, we are only showing one *h*-function per pipeline. The corresponding input files for h_1 and h_4 go through a transformation to allow them to conform to the functions h_1 and h_4 direct input formats (if necessary). After

Table 2.3: Sections and files from the execution of a generic Pipeline. Figure 2.9 describes how these elements interact, here we define and describe them.

#	**Input File Name**	**File Type**	**Description**
Pipeline i**:** In this section the input files are specified for execution.			
1	Configuration input file	JSON	Specifies functions from pipeline i to be executed, and their order of execution.
2	Input file	JSON	Actual input files to execute functions in the pipeline (possibly outputs from upstream pipelines).
Pipeline i **Control:** In this section the functions are invoked, specifying the order.			
1	Configuration file verification	JSON	Input files are validated against their corresponding JSON schema.
2	Pipeline infrastructure code	Python	If needed, invokes a function transformation that transform input files in the correct format for the function input files. Otherwise it invokes the corresponding function, providing the input files.
Pipeline i **Function Execution:** In this section the functions are executed.			
1	Function transformation	Python	Input files are transformed into a valid input file for function h_i.
2	Direct input file	JSON	Input files with the corresponding format that function h_i receives as input for execution.
3	Configuration file verification	JSON	Input files are validated against their corresponding JSON schema.
4	Function Execution	Multiple Programming Languages formats	Function h_i code is executed.
5	Function Output Files	Multiple formats	Function h_i output files.

validation of the JSON instances against their corresponding JSON schemas, direct input data are used in the h_1 and h_4 function executions. Pipelines can run on desktops, laptops, and (Linux) clusters. The two pipelines are shown in Figure 2.9 to make it clear that each pipeline has its own configuration file, h-functions, and associated digital objects, but that the pipeline code/software is the same for every pipeline. All these structures and files in a generic pipeline execution are detailed in Table 2.3. Appendix A.3 provides examples of input files for the Experimental Data Transformation Pipeline (Figure A.12), and the Data Analytics Pipeline (Figure A.13). The system is programming language agnostic to particular functions.

All pipelines in the system have been developed on this project and for the work described herein. We have added pipelines and functions over the course of a year, demonstrating the extensibility of the system, without modifying the pipeline infrastructure code discussed in Section 2.7.2

2.7.3 Functions Within Pipelines

Each pipeline has a list of available functions. The functions can be written in any programming language. Currently we use C++, Python, and R. A function may use as input any combination of outputs from preceding functions in the same pipeline, functions in preceding pipelines, files from previous iterations, and data from experiments.

Currently there are 29 functions across five pipelines. Listings of functions implemented per pipeline are provided in Appendix A.4 (one table for each pipeline). The functions provide a range of capabilities from simple plotting routines to cleaning and organizing, storing and accessing data sets, and inferring properties and running simulations. These are not exhaustive; users may add other functions and continue community-based development.

2.8 Case Studies

2.8.1 Study 1: Full System Execution for Collective Identity Experiments

Collective identity (CI) is an individual's cognitive, moral, and emotional connection with an enclosing broader group such as a team or a community [200]. There are many applications and contexts in which CI is important and therefore makes it worthy of study. For example, CI is important in the formation and maintenance of teams, and team behavior [72, 138]. It is also important in the formation and enforcement of norms [72, 138].

Here, we seek to produce CI among team members playing a game cooperatively. A complete game seeks to produce CI and measure the amount of CI formed among team players in an experiment. The experiment consists of: phase-1–measure individual levels of CI using the DIFI index [125] (for a baseline); phase-2–produce CI among team members using a *collaborative* anagram game; and phase-3–measure individual levels of CI in players using the same index as in phase-1.

Here, we focus the Dynamic Identity Fusion Index (DIFI) score [125] as a proxy for CI. The DIFI score is measured individually as part of our online experiments in the following way. A small (movable) circle represents an individual player and a second (stationary) larger circle represents the team. A player moves the small circle along a horizontal axis, where the distance between circle centroids represents that player's sense of identity with the team; it

is their DIFI score. The range in DIFI distance value is, $-100 \leq DIFI \leq 125$; $DIFI = 0$ corresponds to the two circles just touching, $DIFI < 0$ means that the two circles are disjoint (an individual has no positive affinity for the team), and $DIFI > 0$ means that the two circles overlap (an individual identifies with the team).

As a priming activity to foster CI among team members, in phase-2, they play a *collaborative* word construction (anagram) game motivated by [51]. This Phase 2 is the focus of our case study.

Web-based Experiment Software Platform, Game Play and Data Collection

We built a web application to conduct experiments. The primary components of our platform are the oTree framework [54], Django Channels and the online web interface. We designed and developed software for each phase of the experiment that interfaces with oTree. Django Channels technology supports interactions among players through websocket communication between individual participants and the server. Figure 3.6 shows the web interface for each player of the anagram game. The experimental platform recruits players from Amazon Mechanical Turk and, for all phases of an experiment, records players' actions. The actions are clicks and their event times for specified HTML objects such as letters, and submit buttons.

In phase-2, players are initially given three letters, and are provided communication channels to d number of other players, with whom they can share letters to help each other form words. That is, based on the number n of players recruited, the experimental platform generates a graph on the n players, with a pre-defined regular degree d. Players can form words, request letters from neighbors, and reply to letter requests from neighbors, which are explained in detail in the caption of Figure 3.6.

The goal is for the *team* to form as many words as possible. Total earnings in this game are based on the total number of words formed by the team, and earnings are split evenly among players. The words formed by a player have to be unique, but different players can form the same word. Each player has, in effect, an infinite supply of each of her initial three letters so that she can use letters to form words, and also freely share these initial letters with her neighbors. These features are intended to foster cooperation.

Data Analysis, Modeling and Simulations, and Modeling Evaluations using the Pipelines

Several data model features from Table 2.1 are provided in Figure 2.11. For the DIFI measures (phases 1 and 3), the action set A, with its one element (submit DIFI score), is shown, and the action sequence T is the action tuple of submitting DIFI score for each agent. For phase 2, the word construction game, the edge set E for the four players is provided,

as is the action set A, containing four elements. The action "thinking" is a no-op in the model. Initial letter assignments to players, which are part of B_j^v for each node (player) v_j, are shown. So, too, is an illustrative sequence of action tuples. For example, T_3 states that v_i requests the letter "G" from v_3.

Several ABMs were built to model the phase 2 group anagram game. The one used here is based on a transition probability matrix where the transition probability from one action $a(t) = a_i$ at time t to the next action $a(t+1) = a_j$ for each agent v, $i, j \in [1..4]$ and $a(t) \in A$, is given by $\pi_{ij} = Pr(a(t+1) = j | a(t) = i)$ with $\sum_{j=1}^{4} \pi_{ij} = 1$. For clarity, we use i and j to represent the actions a_i and $a_j \in A$. Agent v executes a stochastic process driven by transition probability matrix $\Pi = (\pi_{ij})_{m \times m}$, where $m \equiv |A|$ (here, $= 4$). We use a multinomial logistic regression model for π_{ij}. Details are in [202]. In essence, the ABM predicts action tuples T_i for players v_i in the game, over the 5-minute game duration.

The complete system of Figure 2.1, and portions of it, were executed over many loops in this study. Here we focus on one iteration of three experiments, with $n = 6$ and number of neighbors $d = 5$, to analyze ***only the anagram game***. Figures 2.12, 2.13, 2.14 show results for the Data Analytics Pipeline (DAP). Figure 2.15 show results for the Property Inference Pipeline (PIP). Figure 2.16 show results for the Modeling and Simulation Pipeline (MASP) and Model Evaluation and Prediction Pipeline (MEAPP). See the figure captions for details. In this work: (*i*) output data from the DAP are inputs for the PIP; (*ii*) outputs from the PIP are inputs to the MASP; and (*iii*) outputs from the DAP and MASP are inputs to the MEAPP.

We now address some particular aspects of these results. The plot generated by h_3 in Figure 2.12 shows, for each player of one game, the time series of words formed. Each step in a curve indicates the time at which a new word is formed. "Form word" is $a_4 \in A$ in Figure 2.11. The time series for all actions can be formed with h_3. These data, like those for h_5 in Figure 2.13, are used to (*i*) understand player behaviors, (*ii*) assist in specifying the structure of ABMs, (*iii*) infer properties of ABMs, and (*iv*) help validate models by comparing model predictions with them. Function h_7 generates the data needed for property inference and showed in Figure 2.14.

The β coefficients in Figure 2.15 are parameters in the multinomial logistic regression model alluded to above. In the π_{ij} terms above, each transition is from action i to j. For example, the β coefficients at the bottom are for the transition from forming word (a_4 in Figure 2.11) to the next actions being a_2 through a_4; the probability that the next action is a_1 (thinking) is 1 minus the sum of other three transition probabilities.

In Figure 2.16, the Modeling and Simulation Pipeline is used to generate all three plots (the first two for simulating experiments, the third for predictions beyond the experiments). The Model Evaluation and Prediction Pipeline is used in the first two plots to compare experiments and model predictions.

2.8.2 Study 2: Data Model for Online Experiment in [47]

Overview

In [47], the effects of network structure on complex contagion diffusion are studied by the spread of health behavior through networked online communities. We represent this experiment with the data model from Section 2.3. Each experiment, *exp_id*, consists of two independent phases ($n_p = 2$), one with $G(V', E')$ being a clustered-lattice network and another $H(V'', E'')$ being a random network. $V = V' \cup V''$ is the set of all players with player $v_i \in V$, and $1 \leq i \leq n$. There are $n/2$ players in each of the two networks, assigned randomly. Γ_i contains variables for v_i's profile (i.e., avatar, username, health interests), ratings of the forum content, and the state of v_i in time, i.e., whether v_i has joined the forum. The meaning of an edge is λ = communication channel between pairs of subjects. B_i^v contains initial conditions for the game, including values for the elements of Γ_i. The set of actions is $A = \{a_1, a_2, a_3\}$, where a_1 is "send a message" to encourage a neighbor to adopt a health related behavior; a_2 is "join forum" which notifies a participant every time a neighbor adopts the behavior; and a_3 is "input rating content" in the forum. Figure 2.17 shows many of these variables, and examples of action tuples. Here we also provide detail of the action sequence from Figure 2.17. In T_1, v_1 sends a message to v_2, then in T_2, v_1 sends a message to v_3. All these are signals from v_1 to encourage health buddies to join the forum. In T_3, v_2 decides to join because of v_1's message. This is why the unique identifier σ_i for the action sequence is the same as in T_1. After this, the news is propagated to v_2's health buddy v_3 in T_4. v_2 sends a message to v_4 in T_5. In T_6, v_1's inputs rating content to the forum. This data model instance, coupled with a GDS formulation (not shown), means that the experimental data can be analyzed (and modeled) with the pipeline system.

Formal Data Model

Table 2.4 details the online social network experiment in [47], defined with our data model. We define one experiment with two independent phases, one with a clustered-lattice network and another with a random network. Each has a population size n=98 and number of health buddies per person d=6.

Figure 2.18 shows the model of Table 2.4 translated into a entity-relationship diagram in unified modeling language (UML) form. This data model instance, that represents an experiment instance, means that the experimental data can be analyzed (and modeled) with the pipeline system. We can perform similar mappings for other social experiments [132, 164, 207].

Formal GDS Model

The GDS model for this system and experiments is that given in Section 2.4.3.

2.8.3 Study 3: Data Model for a Simulation Study in [146]

Overview

In this case study, we evaluate research that is purely simulation-based. We cast their problem in terms of our data model. With this mapping, we then can reason that if we performed experiments according to this data model, we would have a correspondence between those experiments and the simulation system. Hence, in a sense, this case study demonstrates a process of going from modeling to experiments. Another note is that even with simulation models and no experiments, we can still use our pipeline system.

The model in [146] investigates how the structure of communication networks among actors can affect system-level performance. This is an agent-based computer simulation model of explore-exploit tradeoffs, with information sharing. [146] produces an arbitrarily large number of statistically identical "problem" for the simulated agents to solve (explore). Also, the less successful emulate the more successful (exploit). They state that solutions involve the conjunction of multiple activities, in which the impact of one dimension on performance is contingent on the value of other dimensions. For example, activities A, B, and C each actually hurt performance unless all are performed simultaneously, in which case performance improves dramatically. These are defined as synergies, and the presence of such synergies produces local optima.

Formal Data Model

Table 2.5 details the model in [146], defined with our data model. We define one experiment with one phase, with a population of 100, 20 human activities, and 5 synergies (i.e., activities that performed simultaneously improves dramatically the activity performance). Here we also provide an example of an action sequence. In T_1, v_1 posts a solution, then in T_2, v_2 posts a solution. All these are signals from v_1 to encourage health buddies to join the forum. In T_3, v_3 evaluates v_1 solution. In T_4 v_3 copies solution from v_1. The payload will have the information of how accurate agents copy the solution from other, (i.e.) if it was "mimic" or "adapt".

Figure 2.19 shows the model of Table 2.5 translated into a entity-relationship diagram in unified modeling language (UML) form.

This data model instance, that represents a modeling instance, means that the computational modeling results can be analyzed with the pipeline system.

2.9 Related Work

We address several different topics below.

2.9.1 (Networked) Experiments in the Social Sciences

There are several online and in-person experiments with individuals [84, 85, 172, 198, 207, 213, 257] and groups [47, 48, 51, 131, 132, 164]. Some include modeling of the experiment [164]. Also, none of these works appears to do iterative evaluations involving modeling and experiments. There is no platform, that we know of, that allows the iterative process of data analysis, design of data-driven model to simulate experiments, model validation and verification in order to predict behavior. In this work our focus is to formalize a general methodology, through a generic data pipeline, for online controlled experiments of human subjects aim to explain diverse phenomena.

2.9.2 Graph Dynamical Systems

GDSs have been used to produce a wide range of theoretical, modeling, simulation, and policy results on computational complexity, discrete dynamical systems, optimization, (agent-based) simulators, and simulation systems. See the review article [6] for an overview of GDS, research topics that employ GDSs, theoretical results, and practical applications.

2.9.3 Workflow Systems

There are many workflow systems. Here, we cite several popular workflow systems and then describe how they relate to social sciences and pipelines for computation. Examples include Taverna [271] for bioinformatics, chemistry, and astronomy; Pegasus [74] and CyberShake, built on Pegasus [42], for large-scale workflows in astronomy, seismology, and physics; Kepler [25, 157] for ecology and environmental workflows. Other workflow engines including Toil [260], and Rabix [130], were developed for computational biology. To the best of our knowledge, none of these systems addresses social sciences for modeling/experiments as we do here. For example, Taverna is used to analyze suicide data in [222] and Galaxy is used for genomic research [102]; neither has a modeling component.

Most workflows in the social sciences are for social network analyses [94]; we seek to go well beyond that. Also in [73], a taxonomy of features is defined from the way scientists make use of existing workflow systems; this provide end users with a mechanism by which they can assess the suitability of workflow to make an informed choice about which workflow system would be a good choice for a particular application. The importance of interoperability

between these systems is detailed in [80] and identifies three dimensions; execution environment, model of computation (MoC), and language. MoCs provide the semantic foundation, but a data model is a prerequisite. [69, 73, 94, 150] are among the works that overview several workflow systems. An overview and discussion of future directions is provided in [15]. Challenges and future directions for life science workflows are provided in [63]. Ontologies for workflow objects are discussed in [28].

2.9.4 Workflow and Scripting Languages

Workflow languages are usually represented in a textual manner, or through graphical interfaces. A textual representation is often employed for storing the workflows in files, even when a graphical representation is employed. For full interoperability, it is important to have the capacity to translate between workflow languages [80]. Wings [98] uses rich semantic representations to describe compactly complex scientific applications in a data-independent manner. Swift [267] and Swift/T [8, 273] are workflow languages built for executing parallel programs within workflows. NextFlow [248] is a domain specific language for computational workflow management systems. Workflow languages include Common Workflow Language (CWL) [11, 130] and Workflow Description Language (DWL) [13]. Script of Scripts [262] is a workflow system with an emphasis on support for different scripting languages.

2.9.5 Specialized Pipelines

Specialized pipelines are used in different fields. We address several as illustrative. In natural language processing (NLP) [62, 67], allows modular extensions that can be incorporated in a configurable pipeline, but they only focus on NLP components. Similarly, in computer vision [122, 155] use definitions of different models for sub-task classifiers. In computational biology, a structure-oriented pipeline is proposed, capable of detecting RNA motifs [275]. In economics, a web-based IT infrastructure for supply chains is in [187]. In social sciences, [126] develops a three-part pipeline for data analysis and student support in social learning; there is no modeling component. One popular approach is ABM [160] and simulations. In [203], an XML-based data pipeline for interactive simulation is implemented, but it does not integrate modeling with experiments, nor does it have facilities for comparing model predictions with data. In [253] a conceptual model for online games is developed to work with simulations but it does not provide a formal data model for online experiments nor an implementation. Workflows for statistical analysis of social science data are addressed in [251].

While several of these works address one or a couple of the aspects of our pipelines, none of these works provide formal data models and dynamics models (e.g., for ABMs) for pipelines, pipeline designs and implementations, pipeline functions, and case studies, as we do.

2.9.6 Microservices

Our pipelines take a microservices conceptual approach. First defined in 2012, Microservices [153] is an architectural style, addressing how to build, manage, and evolve architectures out of small, self-contained units [49, 189, 206, 230]. The function component of our pipelines have a narrow scope; this way, new functions can be added for new experiments and models in a targeted way, fostering reuse without introducing redundant capabilities.

Microservices Architecture (MSA) and Service-Oriented Architecture (SOA) both rely on services as the main component. But they vary greatly in terms of service characteristics. SOA divides applications into sets of business applications offering services through different protocols. This aims to solve the problem of complexity. SOA applications are costly and complex and are designed to support high workloads, and a large number of users. In [153] is stated that microservices keep services independent so that a service can be individually replaced without impacting an entire application.

In 2012 [152] defined microservices as a way to more swiftly build software by dividing and conquering, using Conway's Law to structure teams. Issues, advantages and disadvantages of microservices are identified in [237]. For example an issue identified is the system decomposition. Advantages include the increase in scalability and the clear boundaries. Disadvantages include the difficulty to learn. The microservice architectural style is largely used by several companies such as Amazon [139], Netflix [165], and many others.

2.9.7 Data Models

In [214], a data model is presented for supporting the modeling, execution and management of emergency plans before and during a disaster. In [231], aspects of a business data model are described. In [191], a data model is presented for capturing workflow audit trail data relevant to process performance evaluation. In [247], models for social networks that have mainly been published within the physics-oriented complex networks literature, are reviewed, classified and compared.

In [173], an object-relational graph data model is proposed for modeling a social network. It aims to illustrate the power of this generic model to represent the common structural and node-based properties of different social network applications. A multi-paradigm architecture is proposed to efficiently manage the system. In [121], a semantic model that can naturally represent various academic social networks is presented; it describes various complex semantic relationships among social actors.

2.9.8 Formal Models of Pipelines

The possibility of incorporating formal analytics into workflow design is investigated in [231]. It provides a model that includes data dependencies. The workflow design analytics they propose helps construct a workflow model based on information about the relevant activities and the associated data. Also, it helps determine whether the given information is sufficient for generating a workflow model and ensures the avoidance of certain workflow anomalies. A detailed treatment of data dependencies is found in [133].

In [219], to improve data curation process efficiency for biological and chemical oceanography data studies, pipelines are defined using a declarative language. The pipelines are serialized into formal provenance data structures using the Provenance Ontology (PROV-O) data model (defined in the paper).

2.9.9 "-Ilities;" reproducibility; interoperability; composability; extensibility; scalability; reusability; and traceability

Foreseeable and unforeseeable changes occur in a system, ilities are attributes that characterize a system's ability to respond to both. Ilities describe what a system should be, providing an enduring architecture that is potent and durable, yet flexible to evolve with the insertion of new systems.

The use of ilities for systems engineering of subsystems and components is investigated in [274]. They show how some ilities are passed and used as a non-functional property of electrical and structural subsystems in aircraft. They demonstrate that a useful practice for systems engineers, to ensure that customer needs are actually met by the system under design or service, is to flow ilities down to the subsystem level. The system ilities are passed down and translated from non-functional to functional requirements by subject matter experts.

Pipelines and workflows provide reproducibility [28], interoperability [145], reusability [28]. The microservices conceptual approach of our pipelines satisfy the reproducibility, interoperability and reusability properties. We show the pipeline composability feature, also it properties for extensibility, scalability, and traceability.

2.10 Summary and Future Work

Online social science experiments are used to understand behavior at-scale. Considerable work is required to perform data analytics for custom experiments. Furthermore, modeling is often used to generalize experimental results, enabling a greater range of conditions to be studied than through experiments alone. In order to transition from experiments to modeling, model properties must also be inferred. Consequently, our work presents an auto-

mated and extensible system for evaluating social phenomena through iterative experiments and modeling. Our work scope ranges from formal models through software design and implementation. Our models include a formal experimental data model (and data common specification), a network-based discrete dynamical systems model (graph dynamical system, GDS), and a formal model for pipeline composition. These models aid in reasoning about the design and construction of five composable and extensible software pipelines, which currently contain 29 functions. We provide three case studies, on collective identity, complex contagion, and explore-exploit behavior, respectively, to illustrate the successful use of the system. Ongoing work includes integrating the pipelines into a distributed data management system, and adding new functions to the pipelines.

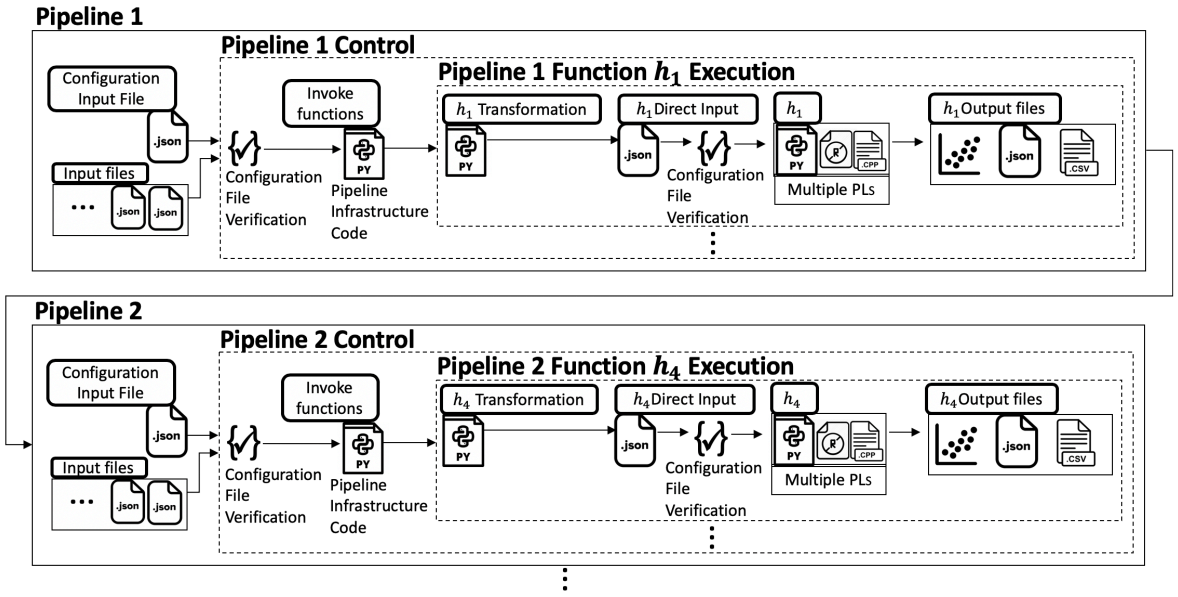


Figure 2.9: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. Here we show how function h_1 is executed in a generic Pipeline 1 and how h_4 is executed in Pipeline 2. Input files are validated against their corresponding JSON schema (Configuration File Verification). The Pipeline Infrastructure Code invokes the corresponding functions. If necessary, the Pipeline Infrastructure Code invokes a function transformation procedure that transforms the file contents into a function input file format. When the contents of all inputs are in the function h_1 input file formats, the files are validated against their corresponding JSON schema (Configuration File Verification). After verification of formats by the corresponding JSON schemas, the function is executed and output files are generated (these digital object outputs may be, e.g., plot files, ASCII data files, and binary data files). There may be additional functions, indicated by the ellipsis below Pipeline 1 Function h_1 Execution. In this example, outputs from the generic pipeline 1 are inputs for the generic pipeline 2. Function h_4 in Pipeline 2 is executed in a similar fashion to function h_1 in Pipeline 1. See the text for descriptions of these various components. Note: the pipeline infrastructure code is the same code for all pipelines.

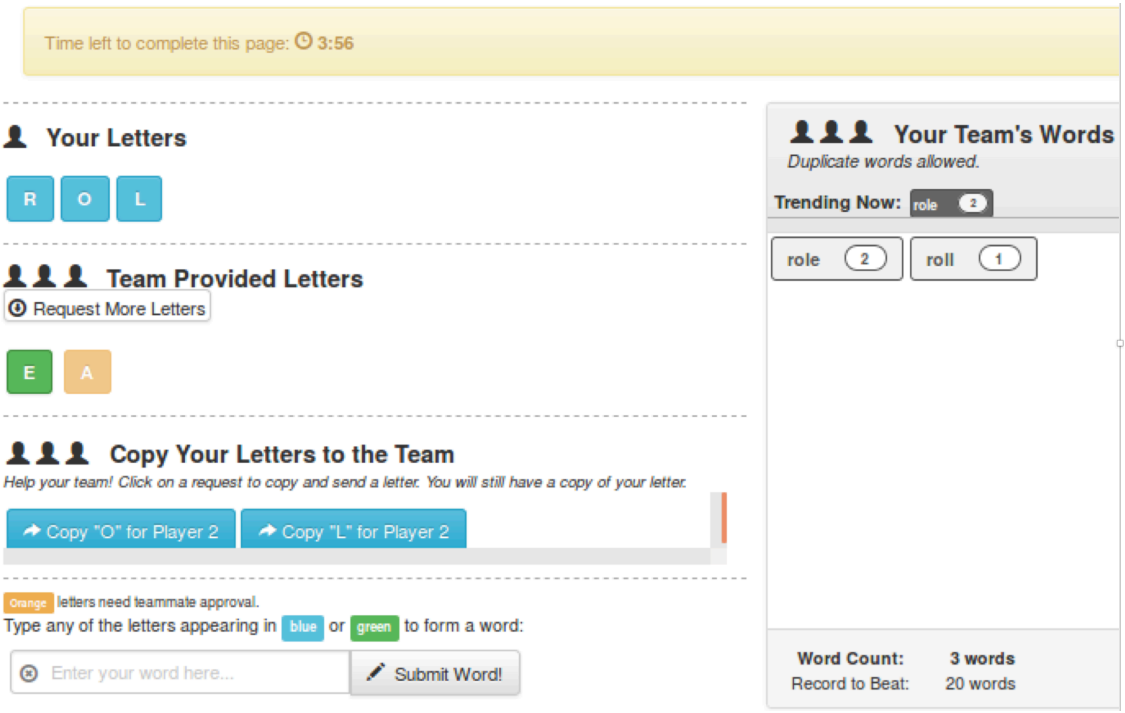


Figure 2.10: The anagram game screen, phase-2, for one player. This player has own letters "R," "O," and "L" and has requested an "E" and "A" from neighbors. The "E" is green, so this player's request has been fulfilled and so "E" can be used in forming words; but the request for "A" is still outstanding so cannot be used in words. Below these letters, it shows that Player 2 has requested "O" and "L" from this player. This player can reply to these requests, if she so chooses. Below that is a box where the player types and submits new words.

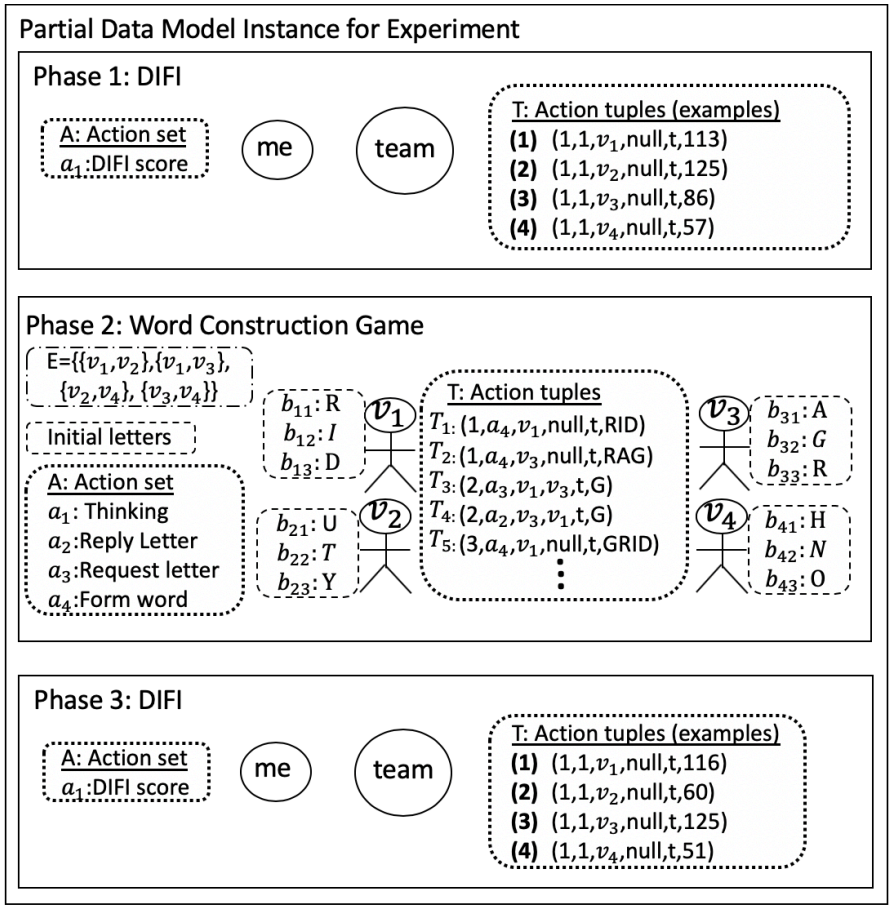


Figure 2.11: Case study 1. Partial representation of the data model for the online experiment composed of 3 phases with a set of V players ($n = |V|$). The phase 1 DIFI measure, a proxy for CI, uses a null (i.e., empty) network on n players; i.e., there are no edges in the graph because players play individually. In phase 2, a team-based CI-priming game, edges E are communication channels. Initial conditions B^v include letter assignments to players. The individual DIFI measure is repeated in phase 3. The action set A and illustrative action tuples T_i are given for each phase.

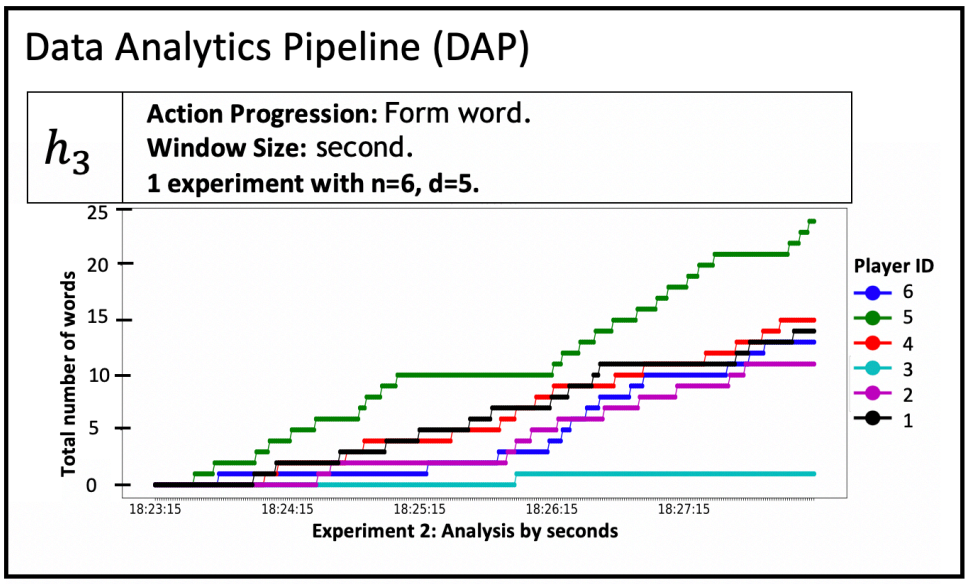


Figure 2.12: The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_3 plots the time series of number of words formed by player for experiment #2.

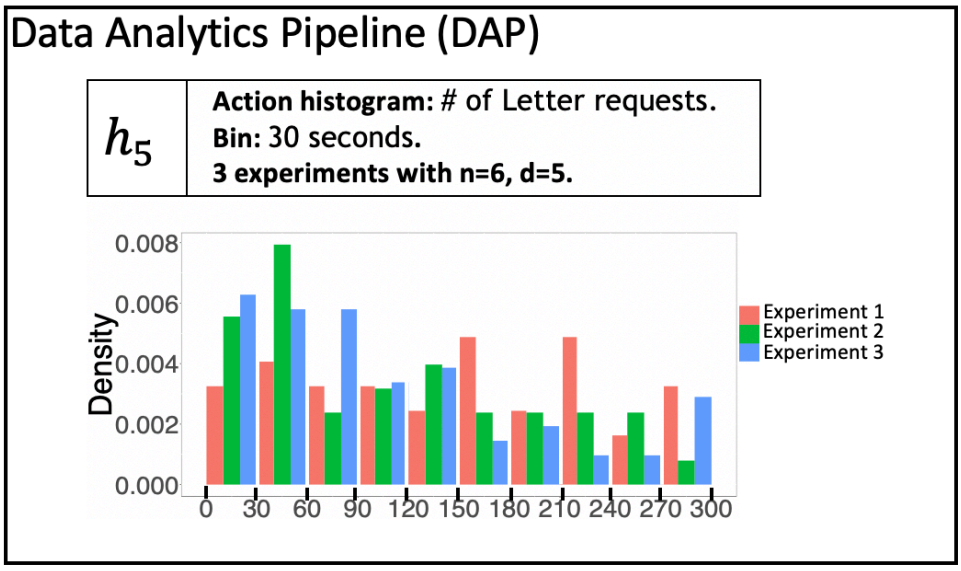
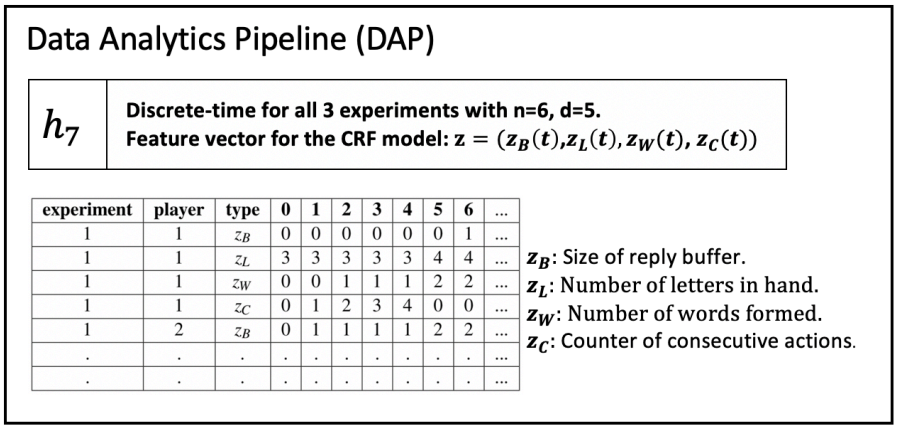


Figure 2.13: The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_5 generates the histogram for the number of actions "letter request" for three experiments. The x-axis is time in the group anagram game, binned in 30 seconds intervals.



experiment	player	type	0	1	2	3	4	5	6	...
1	1	z_B	0	0	0	0	0	0	1	...
1	1	z_L	3	3	3	3	3	4	4	...
1	1	z_W	0	0	1	1	1	2	2	...
1	1	z_C	0	1	2	3	4	0	0	...
1	2	z_B	0	1	1	1	1	2	2	...
.	.	.	.	.	.	.	.	.	.	...
.	.	.	.	.	.	.	.	.	.	...

Figure 2.14: The Data Analytics pipeline (DAP) was executed to analyze phase 2 of three experiments with $n = 6$ and $d = 5$. Function h_7 generates the discrete time actions for all three experiments. This latter output will inform the Property Inference pipeline for computing parameters for simulation models. Time (in seconds) is shown in the first row as 1, 2, 3, ..., and counts of the z vector components, per player and per experiment are given.

Property Inference Pipeline (PIP)

h_2

Beta coefficients for Model ABM 1 generated from experiment data with n = 6, d = 5. The parameters are for the given features in the column names and the β coefficient for computing the next action i that are the row state/action labels, from state i to j, i, j ∈ {1, 2, 3, 4}.

Transition 1 to j					
	(Intercept)	buffer z_B	letter z_L	Words z_W	constant z_C
2	-3.9240	0.2604	-0.0312	0.0061	-0.0172
3	-2.9071	-0.0895	-0.0406	-0.0111	-0.0126
4	-4.0571	0.0812	0.1796	0.0272	-0.018

Transition 2 to j					
	(Intercept)	buffer z_B	letter z_L	Words z_W	constant z_C
2	-2.8873	1.2164	0.2115	-0.1066	0
4	-6.5411	-6.3222	0.0799	-1.6579	-0.1185

Transition 3 to j					
	(Intercept)	buffer z_B	letter z_L	Words z_W	constant z_C
2	-5.5048	0.2570	0.2097	-0.0523	0
3	-4.1109	-67.1075	0.0425	-0.2558	0

Transition 4 to j					
	(Intercept)	buffer z_B	letter z_L	Words z_W	constant z_C
2	-5.2707	0.2285	0.1973	-0.0681	0
3	-1.3798	0.7187	-3.3517	0.7732	0
4	-3.4645	-0.4355	-0.0116	0.0769	0

Figure 2.15: The Property Inference pipeline receives the input from h_7 of the Data Analysis Pipeline (DAP). The parameters in this figure were generated to inform an ABM model for the Modeling and Simulation Pipeline (MASP). The transitions in the figure are from from i to j, where $a_i \in A$ is the action at time t and $a_j \in A$ is the action at $(t+1)$. Rows not shown mean there are no such transitions in the data.

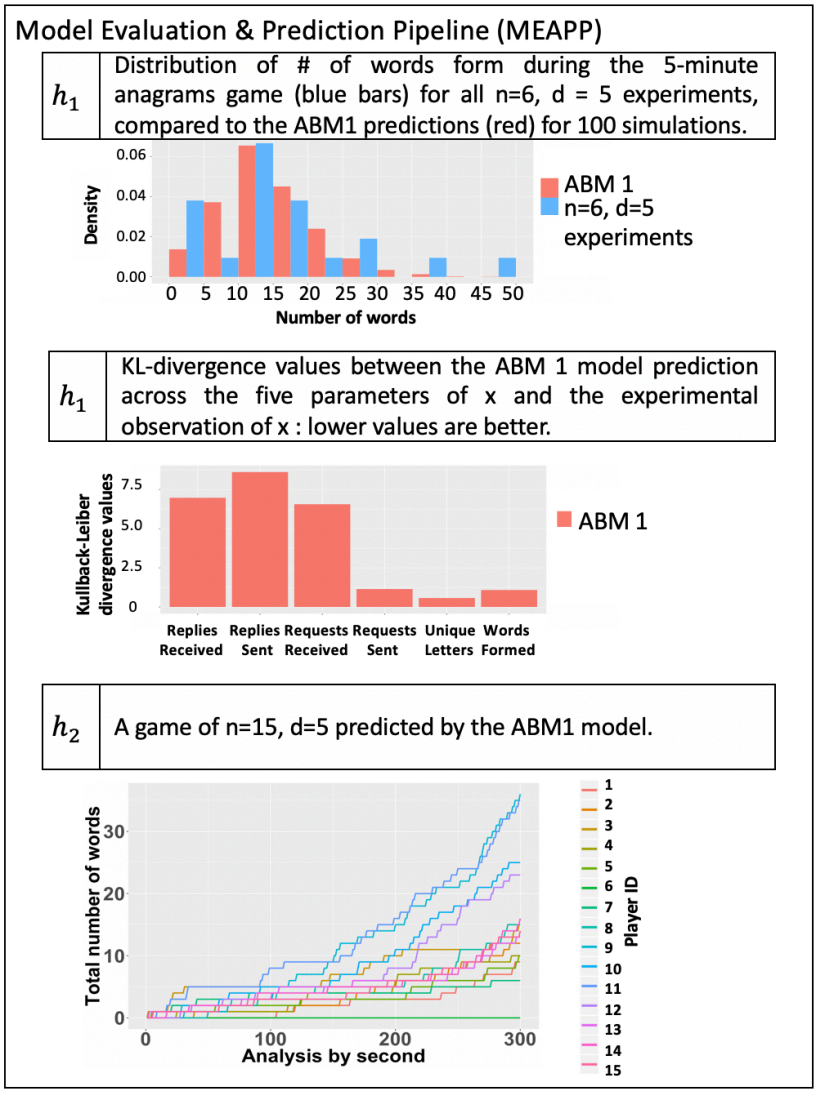


Figure 2.16: The Modeling and Simulation pipeline (MASP) and Model Evaluation and Prediction pipeline (MEAPP) were executed to generate simulation results and model predictions, and to compare experimental data to model predictions. All three plots contain model predictions and use results from h_1 of the MASP. Function h_1 of MEAPP plots corresponding experimental and model output data (top plot) and compares experiment and model output using KL-divergence (center plot) for six parameters. Function h_2 of MEAPP uses h_3 of the Data Analysis pipeline (DAP) to plot model predictions from h_1 of the MASP (bottom plot) where now $n = 15$ (in experiments, $n = 6$).

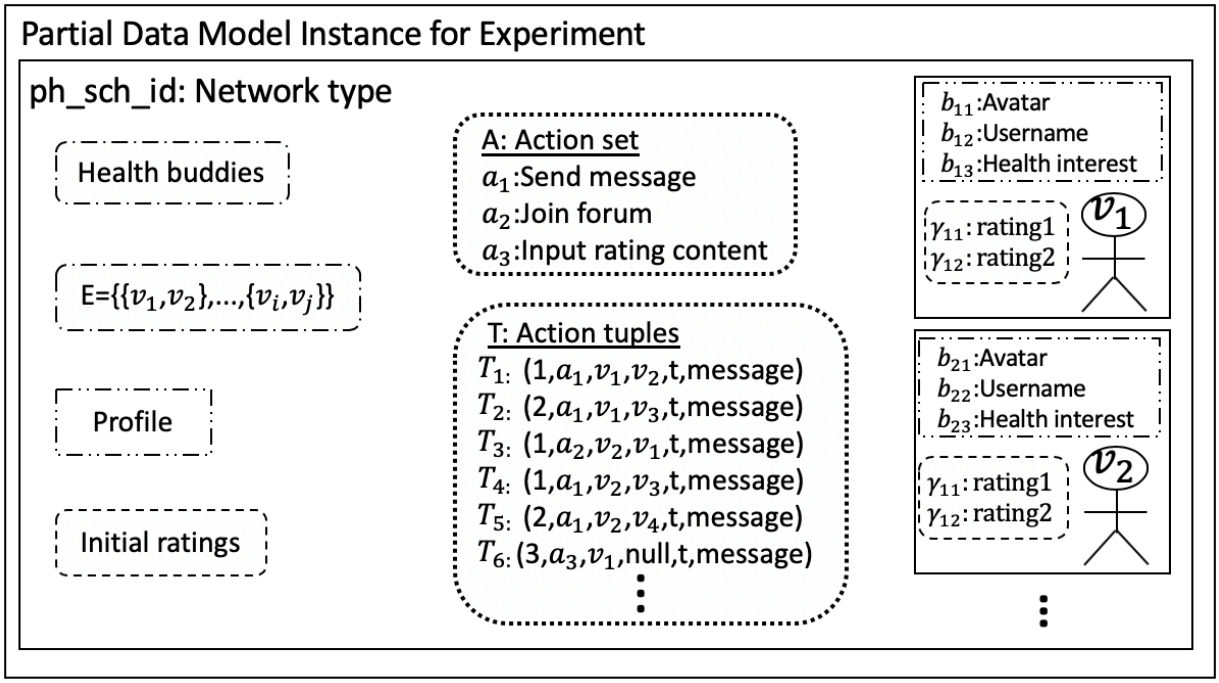


Figure 2.17: Elements of the data model (Table 2.1), for the online social network experiment in [47].

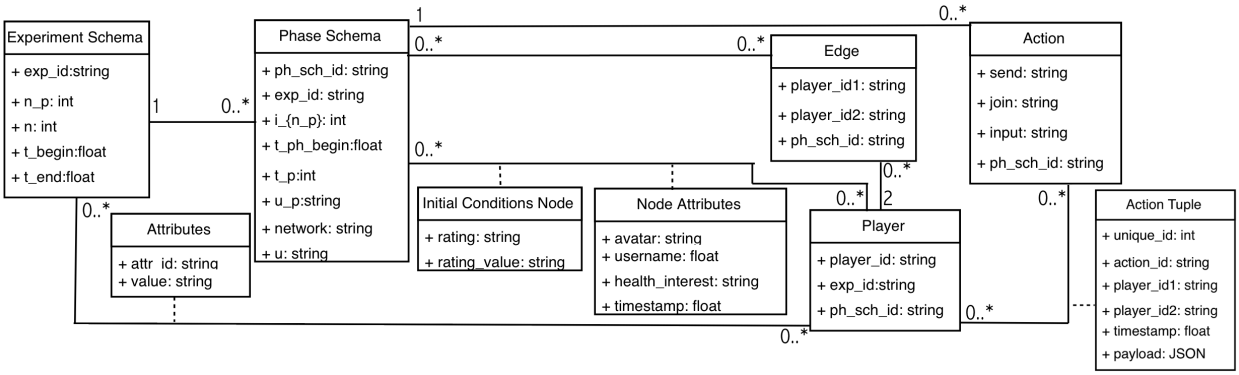


Figure 2.18: Data model of Table 2.4 translated into a entity-relationship diagram in unified modeling language (UML) form.

Table 2.4: Online social network experiment in [47], defined with our data model. One experiment has two independent phases, one with a clustered-lattice network and another with a random network; each with population size n=98 and number of health buddies per person d=6.

#	Parameter	Description
Experiment Schema		
1	$exp\_id = 1$	Experiment id for an experiment.
2	$n_p = 2$	Number of phases in the experiment.
3	$n = 196$	The number of unique players over all phases.
4	t_begin	Timestamp of experiment beginning.
5	t_end	Timestamp of experiment ending.
6	V	$V = \{v_1, \ldots, v_{196}\}$, set of players over all phases.
Phase Schema		
1	$ph\_sch\_id = 1$	Id for phase schema.
2	$i_{n_p} = 1$	Element of the sequence of phases of the experiment.
3	t_ph_begin	Timestamp of phase beginning.
4	$t_p = 13$	Number of time increments in the phase.
5	$u_p = days$	Time unit of one time increment.
6	$G(V', E')$	Clustered-lattice network, node set $V' = \{v_1, ..., v_{98}\}$ and edge set $E' = \{e_1, \ldots, e_{294}\}$, where the number of health buddies each person has is 6.
7	λ	λ = communication channel between health buddies. $\lambda \in \Lambda$
8	Γ	$\Gamma_j(t) = (\gamma_{j1}(t), \gamma_{j2}(t), \ldots, \gamma_{j,\eta_v}(t))$ is the sequence of η_v attributes for $v_j \in V'$. η_v= # of initial ratings in the forum to provide content for the early adopters.
10	B^v	$B_j^v = (\text{avatar}_{j1}, \text{username}_{j2}, \text{health\_interest}_{j3}, \ldots)$.
12	A	A=$\{a_1, a_2, a_3\}$ where a_1 is send message, a_2 is join forum, and a_3 is input rating content.
13	T	$T_1 = (1, a_1, v_1, v_2, t, message)$. v_1 "sends message" to $v2$. $T_2 = (2, a_1, v_1, v_3, t, message)$. v_1 "sends message" to $v3$. $T_3 = (1, a_2, v_2, v_1, t, message)$. v_1 "joins forum" after T_1. $T_4 = (1, a_1, v_2, v_3, t, message)$. v_2 "sends message" to $v3$. $T_5 = (2, a_1, v_2, v_4, t, message)$. v_2 "sends message" to $v4$. $T_6 = (3, a_3, v_1, null, t, message)$. v_1 "inputs rating content" to forum. ...
Phase Schema		
1	$ph\_sch\_id = 2$	Id for phase schema.
2	$i_{n_p} = 2$	Element of the sequence of phases of the experiment.
3	t_ph_begin	Timestamp of phase beginning.
4	$t_p = 13$	Number of time increments in the phase.
5	$u_p = days$	Time unit of one time increment.
6	$H(V'', E'')$	Random network, node set $V' = \{v_{99}, ..., v_{196}\}$ and edge set $E' = \{e_1, \ldots, e_{294}\}$, where the number of health buddies each person has is 6.
7	λ	λ = communication channel between health buddies. $\lambda \in \Lambda$
8	Γ	$\Gamma_j(t) = (\gamma_{j1}(t), \gamma_{j2}(t), \ldots, \gamma_{j,\eta_v}(t))$ is the sequence of η_v attributes for $v_j \in V'$. η_v= # of initial ratings in the forum to provide content for the early adopters.
10	B^v	$B_j^v = (\text{avatar}_j, \text{username}_j, \text{health\_interest}_j, \ldots)$.
12	A	A=$\{$send_message, join_forum, input_rating_content.$\}$.
13	T	$T_1 = (1, a_1, v_1, v_2, t, message)$. v_1 "sends message" to $v2$. $T_2 = (1, a_2, v_2, v_1, t, message)$. v_1 "joins forum" after T_1. ...

Table 2.5: How the structure of communication networks among actors can affect system-level performance is studied in [146]. Here we define this model with our data model.

#	**Parameter**	**Description**
Experiment Schema		
1	$exp\_id = 1$	Experiment id for an experiment.
2	$n_p = 1$	Number of phases in the experiment.
3	$n = 100$	The number of unique players over all phases.
4	t_begin	Timestamp of experiment beginning.
5	t_end	Timestamp of experiment ending.
6	V	$V = \{v_1, \ldots, v_{100}\}$, set of players over all phases.
Phase Schema		
1	$ph\_sch\_id = 1$	Id for phase schema.
2	$i_{n_p} = 1$	Element of the sequence of phases of the experiment.
3	t_ph_begin	Timestamp of phase beginning.
4	t_p =converge	The phase runs until it converges on a single solution.
5	$u_p = seconds$	Time unit of one time increment.
6	$G(V', E')$	Linear network, node set $V' = \{v_1, ..., v_{100}\}$ and edge set $E' = \{e_1, \ldots, e_{98}\}$.
7	λ	λ = influence channel between neighbors. $\lambda \in \Lambda$
8	Γ	$\Gamma_j(t) = (\text{density}_j(t), \text{average\_path\_length}_j(t), \text{score}_j(t))$
10	B^v	$B^v_j = (\text{human\_activities}_j, \text{synergies}_j, \ldots)$.
12	A	A={post_solution, evaluate, copy_solution.}.
13	T	$T_1 = (1, a_1, v_1, null, t, solution)$. v_1 "posts solution". $T_2 = (1, a_1, v_2, null, t, solution)$. v_2 "posts solution". $T_3 = (1, a_2, v_3, v_1, t, solution)$. v_3 "evaluates" v_1 solution. $T_4 = (1, a_3, v_3, v_1, t, solution)$. v_3 "copies solution" from $v1$. ...

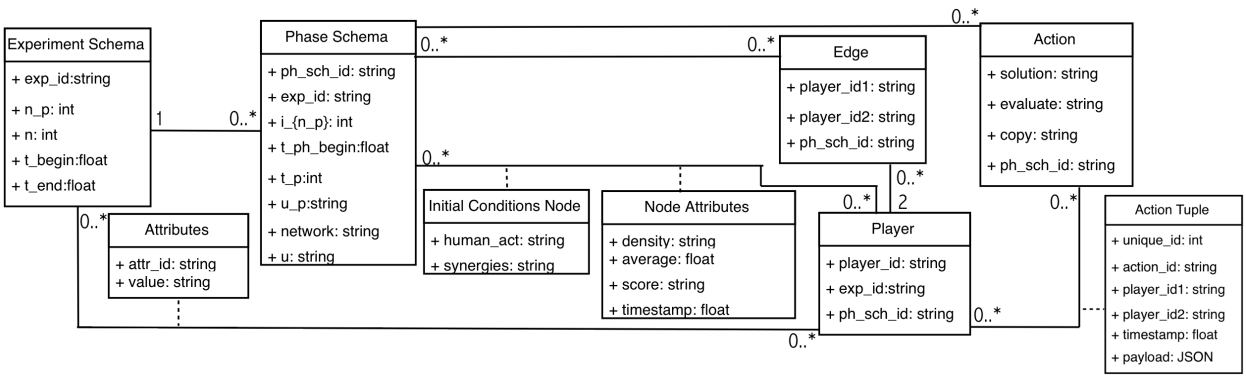


Figure 2.19: Data model of Table 2.5 translated into a entity-relationship diagram in unified modeling language (UML) form.

Chapter 3

Social Networked Experiments and Modeling for Producing Collective Identity in a Group of Human Subjects Using an Iterative Abduction Framework

3.1 Abstract

Group or collective identity is an individual's cognitive, moral, and emotional connection with a broader community, category, practice, or institution. There are many different contexts in which collective identity operates, and a host of application domains where collective identity is important. Collective identity is studied across myriad academic disciplines. As an example, extensive experimental research shows that collective identity influences human decision-making. Consequently, there is interest in understanding the collective identity formation process. In laboratory and other settings, collective identity is fostered through priming a group of human subjects. However, there have been no works in developing agent-based models for simulating collective identity formation processes. Our focus is understanding a game that is designed to produce collective identity within a group. In this work, we build an online game platform; perform and analyze controlled laboratory experiments; build, exercise, and evaluate network-based agent-based models; and form and evaluate hypotheses about collective identity. We conduct these steps in multiple abductive iterations to improve our understanding of collective identity as this looping process unfolds. Our work serves as an exemplar of using abductive looping in the social sciences.

3.2 Introduction

3.2.1 Background and Motivation

Group or collective identity (CI) is an individual's cognitive, moral, and emotional connection with a broader community, category, practice, or institution [200].[1] There are several themes of, and implications for, CI, including: (1) an individual's willingness to place the needs of the group above personal needs (e.g., contributions to Public Goods Games (PGGs) [34, 51]); (2) a person's susceptibility to positive social influence from group members (e.g., sensitivity to evaluations from a collective group [34, 51]); (3) one's desire to differentiate from others not in the collective (e.g., allocation between ingroup/outgroup [32]); (4) an individual's willingness to enforce conformity to group norms established by the collective identity [34, 72, 138, 167]; and (5) a person deriving self-esteem from the group [137, 238]. Hence, there are many behavioral and attitudinal manifestations as consequences of CI.

There are many types of, and contexts for, collective identity, including: (1) religious identity [29, 193], (2) philosophical identity [108, 175], (3) gender identity [39, 43], (4) (sports) fan identity [224], (5) labor movements [103], (6) social movements such as African American civil rights, women's suffrage, gay rights ([200, 225, 242]), (7) political identities [129, 199], (8) racial and ethnic identities [10, 82, 177, 241], (9) national and cultural identities [10, 161, 163], and (10) ideologies [254].

CI is a widely studied concept across academic disciplines. Extensive experimental research in social science, political science, psychology, biology, geography, anthropology, religion, criminology, philosophy, and economics shows that CI influences human decision-making [2, 29, 33, 34, 36, 68, 82, 83, 104, 159, 184, 190, 194, 198, 204, 217, 220, 232, 256, 272].

There is a host of applications for which CI is important, including team formation, maintenance, and behavior in organizations and communities [72, 138]. The ability to generate identity within (marginalized) groups, e.g., through sacred values, is an important aspect of violent group formation [17, 18, 220]. These are compounded by effects of culture and ethnicity [16, 100, 101]. International relations are affected by CI among independent states [266]. Political leaders of minority or marginalized groups may control identity narratives to persuade their constituents of posturing with governments [61]. Relatedly, CI is a cohesive force for groups fighting governments to secure rights and indigenous lands [36, 225]. Religious identity can be a source of stability for immigrants assimilating into a new country [193]. Language and preservation of culture are intimately tied to collective or group identity [36]. Ramifications of a lack of identity are studied in [228].

Individuals may possess several group identities, with different degrees of salience (strength

[1] There are other definitions for collective identity. For example, [168] state that collective identity means that members become more familiar and equal. [266] defines CI as the positive identification with the welfare of another, such that the other is seen as a cognitive extension of the self, rather than independent. See [188] for a discussion of various definitions of CI.

of affinity and association), such that multiple identities may be simultaneously operative [29, 193, 224]. There may be a hierarchy of identities, with different identities coming to the fore in different situations [229]. (The ability to use different identities in different situations has been referred to as *freedom* in a philosophical context [115].) Multiple identities may also be negatively correlated, e.g., religious and national identities [259]. Furthermore, identities and their saliences may be transient over short time scales, and may ebb and flow over longer time scales [29, 39, 188, 224, 261]. Consequently, a person's identity may include a combination of dynamically changing, hierarchical collective identities.

Relationships between CI and other phenomena can be intricate. We take collective action (CA), for which there is a massive literature (e.g., [107, 185, 210, 240]), as an example. Causal relations between CI and CA are very complicated, with the causal direction between the two changing for different circumstances [61, 90, 200, 224, 266].

These issues make the study of CI both interesting and challenging. It is the generality of the concept of CI, its application in a wide range of contexts, its many types and its ramifications for humans and their behaviors that have led to myriad CI studies since the term *collective identity* was first coined by Durkheim some 65-plus years ago [77]. *Our focus here is the CI formation process: how CI is formed among a group of people.*

CI *formation* is studied in several works [5, 34, 37, 51, 58, 61, 109, 193, 198, 235, 266]. See Related Work, Section 3.4.5. All of these works, except one, are empirical, examining events in the field. Surveys, questionnaires, and interviews with human subjects are used to establish, through expert judgment, whether CI has formed within a group.

The work by [51] also studies CI formation, but is quite different from these other works. They use controlled experiments to produce CI among human subjects through priming using team anagram games, wherein players work cooperatively to form words from a collection of letters that they are given. For example, letters t, c, a, and s can be used to form words such as *cat* and *cats*. There are many other aspects to their game. Group identity was then measured after the anagram game using a public goods game (PGG). The greater the PGG contributions of individuals to the team, the greater the collective identity of these individuals. It was found that the priming activity increased PGG contributions. To the best of our knowledge, these are the only controlled experiments that seek to produce CI through priming (in an anagram game) and measure CI quantitatively (through the proxy of PGG contributions). [51] influence our work herein.

We note in passing that priming tasks are central in social and economic experiments (e.g., [76, 87, 223]) and are therefore worthy of study for this reason alone.

In addition to the references above, CI formation is discussed and theorized about in [90, 170, 171, 200, 224, 225, 240]. We note that these theoretical works are descriptive and qualitative in nature, and are *not* concerned with computational modeling. Yet, despite all of the work on CI (described here and in Related Work, Section 4.3), we know of no works that

quantitatively model any CI formation process[2]. We investigate one CI formation process by modeling the priming process of a group anagram game.

3.2.2 Summary of Work Scope

Our work has three broad elements. First, we develop an online experiment, motivated by the work of [51], that is designed to produce CI within a group of participants, through priming, and then measure the CI produced. Specifically, the priming activity consists of players cooperating in a group anagram game, where participants share letters with their neighbors in order to help all players form more words. Specifically, the main player actions are: (1) requesting letters from neighbors, (2) replying to letter requests of neighbors, and (3) forming words. Players equally share in all earnings generated by the team. This priming activity is accompanied by a dynamic identity fusion index (DIFI) task that measures—individually—how much a person associates with a team or group (it is a proxy for CI). Second, we construct models of the CI priming process (the group anagram game) and compare predictions of player behavior against experiments. We develop and evaluate three agent-based models. Third, we use abduction as our framework for this study where both experimental work and modeling work take place within an abductive loop framework.

3.2.3 Situating Our Work On Anagram Game Experiments and Modeling With Other Research

We have mentioned other works in the preceding section. Here we explicitly situate our work relative to those of others. Figure 3.1 shows the context of our work along three dimensions of experiments, modeling, and types of experimental subjects. Figure 3.2 makes this more concrete by presenting representative works along various combinations of values along the three dimensions of Figure 3.1. It is clear that our work—identified at the bottom of the chart in Figure 3.2—is unique.

3.2.4 Overview of Our Experiment and Modeling Approach: Abductive Iterations

Abduction is an inference approach that uses data and observations to identify plausible (preferably, best) explanations for phenomena [89, 197]. That is, abduction is reasoning from effects to causes [50]. Effects are often generated by results from (laboratory) experiments

[2]We use the term *model* to mean a representation of equations and algorithms to compute some result. In contrast, in the social and some other sciences, *model* often refers to a qualitative (textual) description of some process that is much more conceptual and not computational. Our models that we present herein are of the first type. We use the term *model* in the former (quantitative) sense, unless otherwise specified.

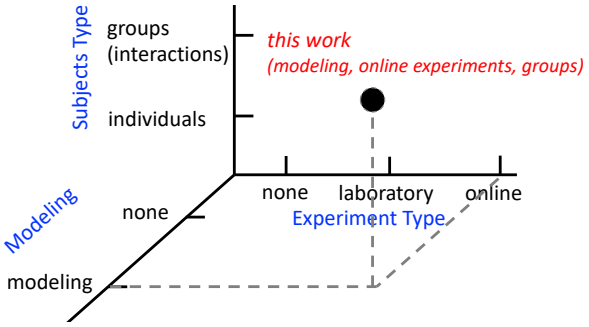


Figure 3.1: Conceptual view of three dimensions of this work, illustrating how our group anagram game study is situated. Our experiments consist of online web-based human subjects experiments. Our modeling component consists of model and algorithm development, and agent-based modeling. We study groups of interacting individuals. To our knowledge, this combination of study features is unique. These dimensions are used in Figure 3.2 to compare our work with others.

or in situ observations of systems. One then constructs hypotheses and identifies or develops theories that explain these observations.

Much of the work on abduction has focused on topics such as producing explanations for different logic settings (e.g., [78]); determining the computational complexity of abduction problems (e.g., [265]); and generating solutions for special problems or transformations that are useful in obtaining solutions (e.g., [196]). Abduction has broad application in robotics, genetics, automated systems, and image understanding [12, 127, 216, 258].

However, in contrast to the above notion of abduction, our focus is the specification and implementation of an abductive *looping* process, wherein abduction is executed in successive iterations. Every iteration builds off of all previous ones, so that explanations may evolve from accumulated data from experiments and observations. As a differentiator from previous work, our interests are behaviors and human interactions within networked groups in the social sciences. In particular, our exemplar is to understand whether a cooperative game can produce collective identity (CI) within a group.

The abductive loop (AL) process that we employ is described in Section 3.3, but among its components are experiments and modeling, and we make note of works on coupling experiments and modeling here. There have been several controlled experimental studies of comparable size to our experiments (e.g., [128, 131, 132]). Also, empirically grounded, data-driven modeling of human behavior is done [154, 164, 179, 277]. We combine these two ideas, in a particular way that is guided by abduction, and perform them iteratively. The proposed abductive analysis is to form hypotheses to evaluate theories as part of the looping process, and develop new insights about CI. Looping over abductive analyses is relatively rare (see the robotics work [216] as an exception), and the use of abduction and abductive

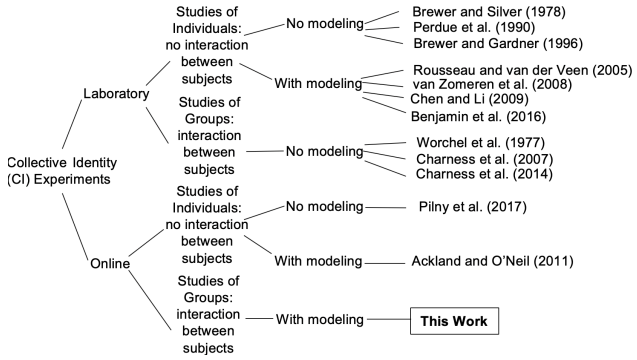


Figure 3.2: A hierarchy of different collective identity (CI) experiments in the literature, which puts the uniqueness of our work on anagram game experiments and modeling into the context of the works of others. The internal nodes refer to classifications between in-laboratory and online experiments, and between works with and without modeling. We distinguish between studies of individuals, with no interaction between subjects, and studies of groups, with interactions among subjects. Leaf nodes refer to representative works. Our work studies online experiments of collective identity with modeling and interaction among subjects. (These references are not exhaustive; more detail is included in the related work of Section 4.3.)

iterations in the social sciences is very rare. Our approach provides an exemplary case of coupling theory development/evaluation with real problems.

3.2.5 Novelty of Our Work

We summarize the novelty of our work based on the foregoing discussion. (1) Our *online* group anagram games are the first of their kind ([51] conducts face-to-face games and their game is different from ours in several respects). (2) We develop quantitative models and agent-based models of the priming process for producing CI. (3) We compare distributions of experimental data to distributions of model predictions. (4) We use an iterative abductive looping process to combine experimental and modeling work, which is quite novel in the social sciences. (5) Our approach provides an exemplary case for combining theory evaluation with an important subject of interest to social scientists (CI).

3.2.6 Contributions

Our major contributions follow.

1. Insights on the collaborative anagram game. We present novel experimental data

that illustrate how players interact in group anagram games. We focus on experimental data that are useful in modeling. We find that letter requests and letter replies are made throughout the game, rather than solely at the outset. However, if there are few neighbors ($k = 2$) and consequently fewer available letters (3 letters per neighbor), there are fewer letter requests and letter replies near the end of the game. Also, players generally respond relatively quickly to their neighbors' letter requests: replies are typically made within 30 seconds of the request. In the same way as letter requests and letter replies, word submissions are made throughout the game, but the number of neighbors and available letters, does not affect this type of action.

2. Data-driven networked agent-based models (ABMs) of experiments: design, construction, and evaluation. We design, construct, and evaluate three data-driven ABMs of the group anagram game experiment. We adapt a conditional random fields (CRF) [233] modeling approach with four parameters to flexibly incorporate history effects on agent actions that evolve in time. That is, our models predict time histories of player actions in the group anagram game. These actions are: (1) requesting letters from neighbors, (2) replying to letter requests of neighbors, (3) forming words, and (4) thinking (or idling). We capture these activities through a state transition matrix approach, where, in our most sophisticated model, the action at time $(t + 1)$ is based on the action at time t and on a feature vector that captures an individual's state. Our approach can alleviate the overfitting problem that would arise with, e.g., a static Markov model that would require capturing many more state transitions.

ABM is used as our simulation modeling approach because of its fine granularity and for its generative properties [81]. That is, local interactions produce population-level dynamics. We use inductive and deductive inference in three ways, use KL-divergence to compare model predictions with experimental data, and compare results across multiple ABMs. For example, our KL-divergence evaluations are broken down by ABM, player action, and number of neighbors in a game. For each combination, we use overall data at the end of the 5-minute group anagram game and at 1-minute intervals during the game to evaluate temporal effects. All of these are used to demonstrate that the ABMs successively improve with the process of incorporating more data.

Our three successive ABMs are named M0, M1, and M2. Our work in evaluating the ABMs shows that ABM M1 reduces KL-divergence values by $4\times$ or more, over those for ABM M0, in many cases (smaller KL-divergence values are better; they indicate better agreement with experimental data). Our work also shows that in many cases, ABM M2 has KL-divergence values that are $4\times$ or more reduced from those of ABM M1. Interestingly, ABM M1 does slightly better than our most sophisticated model (ABM M2) for a small range of parameters that were used in generating M1, but M2 does much better over the remaining input parameter space.

3. Specification and demonstration of iterative abductive analysis process. We perform experiments (Contribution 1), and modeling and evaluation (Contribution 2), within

an iterative abductive process. Using [112, 246] as a starting point, we explicitly incorporate modeling and iterations into the abductive process. The latter necessitates specifying what is to be done in the next iteration. The iterative process is successfully demonstrated through the group anagram experiments, agent-based modeling, and hypothesis generation and testing. *The proposed abductive process can be considered as a general methodology for other social science researches.* For example, our method of model construction from data (see Contribution 4 below) can be used to capture other temporal human action sets among interacting agents.

4. Statistical analysis of numbers of samples required for modeling. We evaluate the quality of our state transition matrices of our ABMs using a root of mean squared errors (RSME) approach. Specifically, we are interested in how many test samples are required to achieve a specified small error in predicted transition probabilities as compared to measured transition probabilities. We use our feature vector from the ABM and break each element down into bins, and add to it the number of neighbors that a player has in a game. By evaluating all of the state transitions among the actions, within each of the resulting 324 distinct bins of data, we find that the minimum number of observations (samples) for each state transition clearly demarcates small from large RSME. The data show that small RSME values result when a state transition has at least 100 observations.

5. New experimental understanding of the formation of collective identity (CI). We discover three novel insights on the formation of CI by coupling the team anagram game and DIFI score. First, players' DIFI scores increase with increasing numbers of neighbors in the anagram game. Second, the number of interactions increases as number of neighbors (i.e., a player's network degree) increases from 2 to 4. However, the numbers of interactions, relatively speaking, saturate with further increases in degree. Third, despite this saturation, the DIFI score continues to increase with degree, suggesting complicated interactions among game parameters. Our analysis is a first work on quantifying the formation of CI since little work has been conducted on this subject in the literature. It is important to note that this experimental work (like the modeling work) takes place within the abductive loop framework.

3.2.7 Extensions from the Conference Paper

This paper was originally published as [202]. Extensions of that work, presented herein, are summarized as follows. (1) Introduction has been expanded to give fuller treatment of background, motivation, and problem context. (2) Related work is expanded with more detail and new topics. (3) Game description has more detail. (4) Experimental data from the game are given with new insights on player behavior. (5) Fuller treatment of the development of each of the three ABMs (M0, M1, and M2) and comparisons of model predictions with experimental data. This includes providing data for all of our experimentally measured and predicted quantities, and providing temporal variations of these data and predictions.

(6) Additional model evaluation and data, comparing model predictions to experimental results across games. (7) Enhanced description and results in error analysis, comparing experiments and models.

3.2.8 Paper Organization

An overview of the abductive loop process is presented in Section 3.3, providing a framework for the rest of the paper. Related work is in Section 4.3. The group anagram experiments are described in Section 3.5. Models of the experiments are developed in Section 3.6. Section 3.7 contains error analyses of the models. Sections 3.5 through 3.7 contain the major technical components of the abductive loop that is overviewed in Section 3.3. These analyses and results enable a more streamlined description of the abductive loop for CI in Section 3.8. Limitations of this work are presented in Section 3.9. Section 3.10 summarizes. Sections 3.5 and 3.6 are substantial in size. Consequently, we provide tables within these sections to organize the work and guide the reader, and we present many of the results in an appendix.

3.3 Overview of Abductive Loop

Figure 3.3 illustrates our iterative abductive process, which includes inductive and deductive steps and hypothesis testing. All work in this paper takes place within this framework. This structure follows that of [112, 246], which are based on Piercian abduction [197], but augments it in key areas. Note that in contrast to confirmatory (deductive) analyses, where theories, hypotheses, and models are developed *first*, and used to predict results of future candidate experiments, one-step abduction first generates data through experiments or observations. (Abduction uses data to drive the scientific discovery process.) Then, data analysis consists of searching for *patterns* and generalizing these into *phenomena*, which is an inductive step. These results are used to formulate hypotheses based on theories whose purpose is to explain the data. Hypotheses may exist (e.g., from a previous loop) or may be proposed in this step, and can be removed (e.g., via falsification). Multiple candidate theories may be posed for a given phenomena. Models are developed from the data, with the objective of generating outputs that help evaluate hypotheses and theories, and/or help guide experiments for the next loop. The best explanation, or hypothesis/theory appraisal, is the process of identifying the best explanation for the phenomena [243]; this includes hypothesis falsification. Finally, the last step in an iteration is to determine what to do next, in terms of designing new experiments. The iterative process may terminate for any number of reasons; e.g., a best explanation has been found.

This description provides the structure for the rest of the paper. The experimental work of Figure 3.3 is described in Section 3.5, after related work. The modeling work in Figure 3.3 is presented in Sections 3.6 and 3.7. We provide the experimental and modeling methodologies,

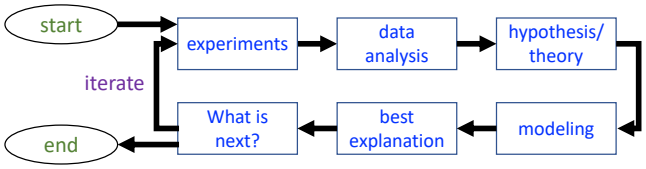


Figure 3.3: Steps in our iterative abductive analysis/loop.

data and results in these sections because they are too large to fit within a discussion of results from the abductive iterations. Following these sections, we return to the abductive loop and reference experimental and modeling results as appropriate, to make the looping process and results more streamlined and cogent (and provide additional results).

3.4 Related Work

Related work topics are provided in Table 3.1, along with each topic's relevance to our work. Each subsection below provides research for one row in the table.

3.4.1 Overviews of CI

Overviews of CI are provided in [4, 88, 119, 188, 224, 238, 261]. [193] provides an interesting view of CI as a combination of social structure (through roles) and processes (via perceptions and interactions) [171].

3.4.2 Individual Anagram Games: Experiments

Over 20 experimental works use anagram games—with *individual* players (e.g., [40, 41, 70, 75, 84, 85, 86, 99, 104, 144, 156, 166, 172, 205, 208, 209, 213, 227, 250, 257, 263]). An individual game means no interactions (e.g., sharing letters) between subjects playing a game at the same time.

We review anagram game studies that are purely experimental. In [227], experiments of anagram games are used to test player's specification of causality for their performance (e.g., did a player attribute good performance to skill or luck?). It was found that people more likely to be responsible for their own actions attributed success or failure to their own behavior, versus assigning outcomes to chance. [172] analyzed how individuals engage in attributions of causality. Situational factors were studied through anagram games in [70].

Effects of goal-setting are analyzed with anagram tasks in [144, 156, 213, 257]. In [257] players played the anagram game and their assigned goals became increasingly difficult. For example, for each goal trial, subjects were assigned a goal for the number of words they have to form.

Table 3.1: Topics described in Section 4.3 of Related Work.

Section of Related Work	**Name**	**Relevance**
3.4.1	Overviews of CI	CI is a broad topic. These are surveys of CI for the interested reader.
3.4.2	Individual Anagram Games: Experiments	Individual anagram games are precursors to group anagram games and have been extensively studied for more than 60 years to analyze the effects of goal-setting, compensation types, internal-external attributions, and test anxiety. It includes a broad range of disciplines like sociology, economics, management science, and (social) psychology. For our work, anagram games are priming activities.
3.4.3	Individual Anagram Games: Modeling	With all of the experimental work on anagram games, it is surprising that very little work has been done in modeling and simulating these games.
3.4.4	Individual Anagram Games: Experiments and Modeling	Few works combining experiments and modeling of individual anagram games exist [84, 85, 86].
3.4.5	Collective Identity-Based Experiments: Formation of CI	Our work is motivated by CI, and in particular the CI formation process. These works study different methods from ours in generating CI.
3.4.6	Collective Identity-Based Experiments: Implications of CI	Along with the Introduction, this section provides works that demonstrate the implications of CI, thus motivating why we study it.
3.4.7	Measurement of CI	Methods used in research to measure (quantify) CI are important.
3.4.8	Combined Group Anagram and CI Experiments	This section emphasizes that there is only one work on group anagram game. That work motivated our work. However there are differences between that work and ours.
3.4.9	Modeling of CI	Demonstrates that there are few modeling studies of CI, and no works like ours.
3.4.10	Agent Based Models of Anagram Games and Formation of CI	This puts our preliminary results into context. The first and only work, to our knowledge, in modeling human group anagram games is our work [202].
3.4.11	Studies of Phenomena Related to CI	As described in the Introduction, CI is relevant for and closely related to, many other phenomena like cooperation and collective action. These works provide some background on these works.
3.4.12	Data-Driven: Combining Experiments and Data-Driven Modeling	Demonstrates that combined experimental and modeling studies, as we do here, are used for other phenomena besides CI.
3.4.13	Modeling of Time Sequences of Actions	These are studies that investigate time series models. Our modeling and ABMs are essentially time series models.
3.4.14	Evaluation of Model Predictions	Methods for comparing experimental and model prediction distributions, as we do here, are presented.
3.4.15	Abduction and Abductive Loop	We use abductive iterations as a framework for our experimental and modeling work. We survey other abductive works.

After each goal trial, subjects recorded their performance (i.e., the number of words formed) as well as their assigned goal for the next trial. Difficulty of assigned goal was increased by two words per trial. Before beginning the next trial, subjects completed a form on which they calculated their GDF (goal discrepancy feedback: performance minus assigned goal) and PDF (performance discrepancy feedback: performance this trial minus performance last trial). Assigned goals were rejected when GDF became sufficiently negative. GDF and PDF differed both in sign and magnitude of effects on acceptance and personal goals, indicating that subjects used these feedback discrepancies differently in the goal evaluation process. Unusually, personal goals and performance remained high even after assigned goals were rejected. In [144, 156], theories of goal settings are developed. In [213], it was found that people with unmet goals were more likely to engage in unethical behavior than people

attempting to do their best.

[104] use the anagram task to examine three factors and their effects on group performance: intergroup competition or cooperation, intragroup competition or cooperation, and task means interdependence. In [205, 208, 209], studies look at anxiety generated from performing a task, where the task is the anagram game. In [40], pay-for-performance and fixed-salary compensation were compared using an anagram task. In [41], an anagram game was employed as the experimental task to evaluate a target-based compensation system, a linear piece-rate system and a tournament-based bonus system. Larger amounts of cheating occurred under target-based compensation. In [75, 99, 166, 263], the effects of letter order and word frequency on anagram game performance are analyzed.

3.4.3 Individual Anagram Games: Modeling

In [250], problem solving and verbal cues are analyzed with an anagram game. [250] modifies the [111] mediational model of problem-solving behavior (introducing word length and letter position), to understand anagram problem solving. This is a theoretical model of individual anagram games.

3.4.4 Individual Anagram Games: Experiments and Modeling

These works combine experiments and modeling. In [84], it was found that subjects who were initially confident of passing an anagram game test tended to attribute success to ability and failure to bad luck. However, subjects who were initially not confident tended to attribute success to good luck and failure to lack of ability. Results are discussed in terms of Heiderian theory and a valence-difficulty model. In [85, 86], two individuals played anagram games simultaneously but independently to test whether a person attributed her success (if she performed better) to skill versus good fortune, and failure to inferior skill or bad luck. Attributions were found to be dependent on expectations of players. Results are discussed in terms of models involving Heider's principle of balance and his analysis of the causes of action, in terms of positivity biases in social perception, and as indicating effects of the social context of performance upon attribution and valence.

3.4.5 Collective Identity-Based Experiments: Formation of CI

The following references study or theorize on the CI formation process. That is, they study processes by which a group of individuals that does not possess CI can form CI by, for example, interacting or undergoing a priming task.

In [34], laboratory experiments of CI with no interactions between subjects are performed

using priming. They argue that the personal, relational, and collective levels of self-definition (shift from personal to collective) represent distinct forms of self-representation with different origins, sources of self-worth, and social motivations. They suggest the concept "we" primes social representations of the self that are more inclusive than that of the personal self-concept. In a preliminary investigation of the implications of different levels of the social self-concept, a set of three experiments were conducted to explore the effects of priming various "we" schemas on individual judgments and self-descriptions. In the priming task, participants read a descriptive paragraph with instructions to circle all the pronouns that appeared in the text, as part of a proofreading and word search task. After completing this word search task, participants were escorted to another room and asked to judge, as quickly as possible, whether the statements were similar or dissimilar to their own views by pressing a number key on the keyboard, ranging from 1 (very dissimilar) to 4 (very similar). They found that individuals primed with "we" would entail an expanded sense of self that would lower thresholds for agreement and assimilation.

In [58], laboratory experiments with no interactions between subjects measure the effects of induced group identity on participant social preferences. They show that participants are more altruistic towards an ingroup match. They evaluate different ways of creating group identity in the laboratory, to explore the formation of groups and to investigate the foundation of what group identity is. When participants are matched with an ingroup member (as opposed to an outgroup member) they show a 47-percent increase in charity concerns when they have a higher payoff and a 93-percent decrease in envy when they have a lower payoff. Also, participants are 19 percent more likely to reward an ingroup match for good behavior, but 13 percent less likely to punish an ingroup match for misbehavior. Participants are significantly more likely to choose social-welfare-maximizing actions when matched with an ingroup member.

In [198], online experiments with no interactions between subjects are performed. To expand upon perspectives on the commons dilemma (e.g., do I contribute to the common resource or do I free ride), [198] developed an online experiment grounded on group decision-making. They create manipulations based on three modalities of structure: dense versus sparse networks (domination), collective versus individual identity (signification), and social sanction versus non-social sanction (legitimation). The online experiments reveal that modalities of signification positively influence contribution rates on the commons dilemma, when participants were provided information meant to stimulate a CI. This is analogous to the findings of [51]; see Section 3.4.8. They mention how challenging it is for an online experiment to create CI, because the individual is sitting alone playing the game on a computer. In their experiments they try to stimulate CI by communicating three additional pieces of information regarding collective outcomes: (1) total collective score, rather than just an individual collective score, (2) collective rank compared to previous sessions, and (3) the score of the highest collective score from previous sessions. By including more collective, rather than individual, information, the user may come to behave more in a collective fashion and contribute to the public good.

In [5], an online experiment using data collected from the websites of over 160 environmental activist organizations is developed. A model is presented where social movement actors exchange practical and symbolic resources in the guise of website text content and hyperlinks, as part of a process of online CI formation. The hyperlink and online frame networks are compared on three measures of centralization: degree, betweenness and closeness.

[266] argues that international cooperation among independent states can be fostered through CI. He describes different mechanisms that may lead to CI, and takes examples from past events or general ideas. For example, he states that trade relations among states can foster CI through the emergence of the feeling of a common fate, but there are no experiments nor historical observations. It focuses more directly on identities and interests as the dependent variable and investigates whether, how, and why identities change.

[109] evaluates self and recognition theories—recognition theory states that an individual or group places recognition of itself by others as a very high-priority goal—to determine whether these two ideas can combine to produce CI. The reasoning is that as the self acknowledges others, and this process is replicated by all participants, a collective identity is formed. However, social identity theory-based experiments do not support this line of reasoning. This work is more akin to a meta-study, summarizing existing results.

[193] studies CI generation among Muslims in the United States. It is an empirical study of the formation of religious CI among 127 subjects, using focus groups, individual interviews, and participant observations. She presents three consecutive steps to form CI: religion as an ascribed identity; religion as chosen identity; and finally religion as declared identity.

[61] studies the relationships among specific (poor) constituent groups and governments, and how these groups use their shared (collective) identity to position themselves. She also uses observations (of group meetings) and interviews of group leaders to produce a model of CI formation and its effect on collective action.

[235] uses small groups of music students (sizes of 2 to 5 students) to study the formation of CI. Again, as with several previous works, surveys, interviews, and observational studies are used to document the CI formation processes as students work together.

[37] examines South Africa and the current fracturing of the nation among different societal groups. Factors contributing to the lack of a national CI (e.g., a lack of trust among sub-groups and misunderstandings) are also discussed. Finally, the article posits that one way to heal these divisions and form a national CI is through religious understanding.

A final work in CI formation are experiments with interactions among subjects performed in [51]. This work, in a general way, motivated our anagram game experiments (although there are many differences between our work and that in [51]). Consequently, we address this work separately below.

Dismissing for the moment this last reference, it is clear that none of the above works on CI formation are like ours. In contrast, our work uses controlled online laboratory experiments

to produce CI through priming groups of subjects using a cooperative anagram game.

3.4.6 Collective Identity-Based Experiments: Implications of CI

The effects of religious (group) identity on individual behavior is studied in [29]. Subjects (self-identified as Protestant, Jew, Catholic, or agnostic/atheist) were primed or not primed with respect to religion. Priming consisted of having players unscramble a set of words that form a sentence, and that sentence has religious content. The unprimed subjects unscrambled words to form a sentence with no religious content. The purpose of priming is to make salient the religious identities of players, if they exist. Subjects then played a number of games, including public goods games, risk aversion games, discount rate elicitation games (i.e., delayed gratification games), among others. In a public goods game, players are given some amount of money. They have the option of contributing a portion of their money to the group. The pooled money that is contributed to the group by all members is then typically multiplied by some factor and redistributed to the players. Hence, there may be some incentive to contribute to the group. There are several interesting results. Among them is that religious identity salience (i.e., priming) produced an increase in Protestant subjects' contributions to Public Goods Games (PGG), while it generated a decrease in Catholic subjects' contributions.

In related experimental economics work using Indian caste and other nonreligious identities, [79], [116, 117], [53], [57], [64], [59], [65] find that group identity effects on behavior strengthen with the salience of group membership. [55] manipulate the norms (expressed by legal rulings) that subjects are exposed to and study how these norms affect their self-identification.

The following works study the implications of *identity fusion*, where individuals may feel fused with (i.e., strongly connected to) a group [105, 106, 234, 268, 269, 270]. We interpret identity fusion to be synonymous with, or very similar to, CI.

In [270], the authors use online experiments to test the notion that fusion represents a distinctive form of allegiance to groups. They propose that when people become fused with a group, their personal and social identities become functionally equivalent. To measure identity fusion they used a modified version of a fusion scale developed by [212]. They prove that activating either personal or social identities of people who were fused with their group increased the extent to which they were willing to fight or even die for the group. Thus, even when people become deeply aligned with a group, their personal identities remain potent.

In [234], using an intergroup version of the trolley problem, the authors explored participants' willingness to sacrifice their lives for their group. Studies showed that nonfused participants expressed reluctance to sacrifice themselves, and identification with the group predicted nothing. To measure identity fusion they used the same scale as in [270].

In [269], they assume that autonomic arousal will increase agency (i.e., the capacity to ini-

tiate and control intentional behavior) for fused and nonfused persons. In four experiments, increasing autonomic arousal through physical exercise elevated heart rates among all participants. Fused participants, however, uniquely responded to arousal by translating elevated agency into endorsement of pro-group activity. To measure identity fusion they used the same scale as in [270].

In [106], online experiments showed that when people are ostrasized (i.e., rejected and excluded) by either an outgroup or an ingroup, they may either withdraw or engage in compensatory activities designed to reaffirm their social identity as a group member. The authors proposed that individual differences in identity fusion (an index of familial orientation toward the group) would moderate the tendency for people to display such compensatory activity. Four experiments showed that irrevocable ostracism increased endorsement of extreme, pro-group actions (fighting and dying for the ingroup) among fused persons but not among nonfused persons. To measure identity fusion they used the same scale as in [270].

In [105], the authors determine what fusion is and the mediating mechanisms that lead fused individuals to make extraordinary sacrifices for their group. For measure of group identification, they proposed a seven-item verbal scale with greater fidelity than the earlier pictorial measure of identity fusion from [270].

In [268], online experiments explored the cognitive and emotional mechanisms that underlie the endorsement of self-sacrifice. Using participants responses to moral dilemmas, they found that only those who were strongly fused with the group preferentially endorsed self-sacrifice. Identity fusion was measured using the seven item verbal fusion scale from [105].

3.4.7 Measurement of CI

Researchers measure or declare the existence of CI in different ways. This is in part because there are many definitions for, and types of, CI (see Section 3.2.1).

Many references on CI formation [61, 109, 193, 235, 266], presented in Section 3.4.5, pronounce that CI has been formed based on expert evaluation of textual comments of participants, survey responses, and interviews. These are subjective approaches for determining the existence of CI. They require an expert to interpret the data, and multiple experts may arrive at different conclusions.

In PGGs [148], players are given some amount of money. They have the option of contributing a portion of their money to the group. The pooled money that is contributed to the group by all members is then typically multiplied by some factor and redistributed to the players. Hence, there may be some incentive to contribute to the group. [51, 58] use PGG contributions as a proxy for CI. In [51], the percentage of a persons money that they contribute to the group is taken as their identification with the group: those with greater group identity contribute more of their money to the team.

In [270] a modified version of a fusion scale developed by [212] is proposed. To capture fusion in a manner that emphasized perceived overlap and nothing else, participants choose from five pictures which best represented the way they perceived their relationship with the group. Each figure in the scale shows two circles of different sizes. The small circle represents "the self", the big circle represents "the group". When participants need to choose from the scale, five figures with symmetrical degrees of overlap (0%, 25%, 50%, 75%, and 100%) are presented. For example, the first figure shows the two circles not intercepting, the second figure show a 25% interception and the fifth figure show a 100% interception with the small circle. To measure identity fusion, the following works use this scale [105, 234, 269, 270].

In [105], a seven-item verbal scale is proposed to obtain greater fidelity in the measurement of identity fusion, compared to the pictorial measure from [270]. The levels in the verbal scale are represented with the following sentences (1)"I am one with my group", (2)"I feel immersed in my group", (3)"I have a deep emotional bond with my group", (4)"My group is me", (5)"I'll do for my group more than any of the other group members would do", (6)"I am strong because of my group", (7)"I make my group strong". [268] uses this scale to measure identity fusion.

In [125] the DIFI is introduced to combine the simplicity of the single pictorial item [270] with the higher predictive fidelity of the verbal scale [105]. The scales presented in [105, 270] are not dynamic. In [125] the DIFI is defined as a continuous measure of identity fusion, introducing a dynamic behavior for web-based questionnaires. The DIFI shows a figure formed by two circles of different sizes in the screen of the computer. The small circle represents "the self", the big circle represents "the team". The player can move the small circle by clicking and dragging with the mouse to measure the degree to which the player feels part of the team.

3.4.8 Combined Group Anagram and CI Experiments

A group anagram game entails cooperation in requesting and receiving letters, with the goal of forming more words with additional letters received from teammates. The only *face-to-face* cooperative team-play of an anagram game is reported in [51]. Their goal, like ours, is to foster CI among teammates. While this motivated our experiment, there are several differences in procedures and context. Major differences include (*i*) the game setup: we used larger fixed team compositions, while in [51], the four-person team composition varied in time (by people voting themselves and others onto and off of teams); (*ii*) in [51], games were played face-to-face among participants in the same room cooperatively manipulating Scrabble-like tiles on a table, while we used remote players interacting in a game through a web application; and (*iii*) in [51], they measure CI with the proxy of PGG contributions, while we use DIFI score.

3.4.9 Modeling of CI

[159, 204] use ABMs to study identity diffusion. An agent adopts (changes) her type of identity to that of a neighbor with a stronger (higher valued) type of identity. Hence, these are contagion processes and are implemented much like voter models [71, 195]. Other works modeling collective identity [5, 29, 58, 256] are presented in Section 3.4.5.

3.4.10 Agent Based Models of Anagram Games and Formation of CI

The [51] work in Section 3.4.8 has no modeling for the group anagram game. This motivated the *online* experiments and ABMs in [202]. This article is an expansion of [202]. In this work, we model the priming process of producing CI, which is the group anagram game. There are no ABMs (or models of any kind) of group anagram games, to our knowledge, other than ours.

3.4.11 Studies of Phenomena Related to CI

Many phenomena, such as in-group and out-group effects are related to CI. In [35, 194], laboratory experiments with no interactions between subjects are performed. In [35], it was found that bias in favor of the ingroup on a reward allocation task was unaffected by the arbitrariness of classification into groups. An effort was made to assure that subjects in the arbitrary condition would not perceive the outgroup as dissimilar. They found that similarity-dissimilarity of the outgroup did not affect allocation bias as long as the ingroup was perceived as similar to the subject. Subjects were divided clearly into groups labelled "dark" and "light." Subjects then were asked to indicate their ratings first of "the other members of my group" and then of "the members of the other group" on a series of six-point bipolar scales (friendly-unfriendly; trustworthy-untrustworthy; cooperative-competitive; intelligent-stupid; weak-strong; generous-stingy; likeable-unlikeable). In [194], classical conditioning in-group and out-group descriptors (e.g., "us" and "them") are used to establish evaluative responses to novel, unfamiliar targets. Nonsense syllables unobtrusively paired with in-group designating pronouns (e.g., "we") were rated as more pleasant than syllables paired with out-group designators (e.g, "they").

[190] studies how the anticipated interaction between groups determines the representations that groups have of each other. When students are categorized into groups, discrimination occurs such that the ingroup is more favorably represented than the outgroup before interaction takes place and also when no interaction is anticipated. Such discrimination is stronger when competitive interaction is anticipated in an important situation. In this condition, intergroup differences are also more easily projected on physical traits. Categorization is

shown to be not only an independent variable but also a dependent variable in intergroup relations.

In [2], Own Group Bias (OGB) was measured by differences in pre and postgame scores on the evaluative scales of the Semantic Differential (SD).

In [217], an experiment on Amazon Mechanical Turk was used to develop an agent-based simulation to understand how people's motivations and behaviors within public goods dilemmas interact with the properties of the dilemma to lead to collective outcomes. They predict how the public good's benefit and size, combined with controlling individual versus group properties, produce different levels of cooperation in public goods dilemmas.

In [215], a simple model of collective action is presented as a framework for empirical research into the issue of when collective action in the commons will be successful.

In [256], an integrative social identity model of collective action (SIMCA) is developed that incorporates three socio-psychological perspectives on collective action. Instructions for coders were to answer different questions like "Does the measure of identification (used in this study) refer to a disadvantaged group or a social movement?", "Is this group incidentally disadvantaged or structurally disadvantaged?". Coders also rated the extent to which collective disadvantage was structural on a 5-point Likert-type scale ranging from 1 (not at all) to 5 (very much).

In [207], new insights into the role of individual behavior on collective outcomes are obtained using a multiple-worlds experimental design in a web-based experiment in which 2,930 participants listened to, rated, and download 48 songs by up-and-coming bands.

In [232], laboratory experiments with interactions between subjects are performed. Web-based experiments are conducted where 24 individuals played a local public goods game arranged on one of five network topologies that varied between disconnected cliques and a random regular graphs. It was found that although players did generally behave like conditional cooperators, they were as likely to decrease their contributions in response to low contributing neighbors as they were to increase their contributions in response to high contributing neighbors. They also found that positive effects of cooperation were contagious only to direct neighbors in the network.

In [44], online experiments using Amazon Mechanical Turk were used to develop a predictive model of human cooperation able to organize a number of different experimental findings that are not explained by the standard model.

In [204], an agent-based computer simulation of identity change explores how changes in the attributes of the individual and/or elements of the environment influence the dependent variable: the degree of shared identity in a population.

There is a host of other studies that investigate phenomena such as cooperation and a person's affinity for a group that are closely related to CI. In [53, 272] laboratory experiments with interactions between subjects are performed. They study concepts such as group at-

traction and salience, respectively, which are related to CI. In [272], study groups worked cooperatively on two tasks and results were interpreted as showing that both previous interaction and success of combined effort are important variables in determining when intergroup cooperation will increase intergroup attraction. In [53], groups perform two stage games as priming tasks, the Battle of the Sexes and Prisoner's Dilemma. Results show that the salience of the group affects behavior of members, as well as the behavior of people in another group, and that participants anticipate these effects.

3.4.12 Data-Driven: Combining Experiments and Data-Driven Modeling

This section reports on works that combine experiments with data-driven modeling. These works cover explore-exploit networked experiments with limited modeling [164]; individual models of single-choice (i.e., one-shot) evacuation decisions [179]; ABM of emotion and information contagions spreading on a network and comparisons with a single event [154]; and ABM of solar panel adoption and comparisons with data in San Diego county [277]. See [276] for a review of innovation diffusion models. None of these works use ABMs to model networked experiments where individuals take a series of actions (that may be repeated) over time, to study CI, as we do.

In [158], small-scale laboratory experiments and an ABM were used to analyze the dynamics of collaborative inhibition. In [95], the model in [158] was tested against human data collected in a large-scale experiment to find that participants demonstrate non-monotonicities not evident in the predictions. These unexpected results motivate more recent work in elucidating the algorithms underlying collaborative memory. In [192], using real-time online social experiments data, a statistical model is used to study interpersonal coordination in a "minimally interactive context" to explore how people become coupled in their perceptual and memory systems while performing a task together.

In contrast to the above works, where *controlled experiments* are used to produce data that are then used for modeling, there are many models based on *observational* data. We survey some of these works here.

In [135], the possibility of predicting a social protest (planned, or unplanned) based on social media messaging is studied. In [180], to help increase the performance of retweet prediction, a flexible model under the framework of Random Forest classifier captures a number of behavior signals affecting user's retweet decision. In [121], a semantic model that can naturally represent various academic social networks, especially various complex semantic relationships among social actors, is presented. In [201], the proposed method integrates topology and content of networks, and introduces a novel adaptive parameter for controlling the contribution of content with respect to the identified mismatch degree between the topological and content information. In [19], data-driven multi-agent models

predict Twitter trends. In [255], a method that implements, validates, and improves an individual behavior model is proposed. The multi-agent model contains the social network structure, individual behavior parameters, and the scenario that are obtained from empirical data. In [149], emergence and propagation of reputations in social networks is modeled with a distributed algorithm. In [60], using several Twitter data sets, focusing in particular on the tweets sent during the soccer World Cup of 2010, a model of how users switch between producing information or sentiments and sharing others news or sentiments is developed. In [136], a theoretical analysis is developed for how social-chatter quantitatively relates to action via a superlinear scaling law.

Other works include using data from geotagged social media messages and data from mobile health applications [142, 249] In [249], to understand citizen reactions regarding Ebola, a large-scale data-driven analysis of geotagged social media messages is performed. In [142], data from mobile health applications is used to develop a statistical model, called TIPAS (Time-varying, Interdependent, and Periodic Action Sequences). This approach is based on personalized, multivariate temporal point processes that model time-varying action propensities through a mixture of Gaussian intensities. Their model captures short-term and long-term periodic interdependencies between actions through Hawkes process-based self-excitations.

Clearly, much of the modeling of observational data is motivated by social media.

3.4.13 Modeling of Time Sequences of Actions

We review modeling of time sequences because our ABMs are essentially in this class of models.

Many complex action sequences from human behavior are being collected from different environments, like sensors [9, 30, 110, 239] or computer-based applications [60, 134, 142]. Sequence mining techniques to model and predict human behavior in the real world can be used in different types of applications to improve a person's life (e.g., mobile health [142], education patterns [134], smart-home optimization [9, 110]).

Sequence analysis is an important task to understand human behavior [3]. The sequential pattern mining problem was first introduced by [7], where the main focus is on the patterns present in the sequential order of different transactions. But the complexity of human behavior with time-varying, interdependent and periodic action sequences [142] makes accurate analysis and predictions a challenging task.

[142] use activity data from logging applications to model the task of predicting future user actions and their timing through a mixture of Gaussian intensities. The model captures short-term and long-term periodic interdependencies between actions through Hawkes process-based self excitations [114]. Accurate recommendations could improve a person's health through the personalization of these applications. In [134], a combination of sequence

mining techniques uses data from computer-based learning environments to model students learning behavior patterns. [110] use sequential pattern learning to model an agent-based system to aid elderly people in living longer in their homes. [9] uses pervasive home sensors, like motion sensors, door close sensors, and floor pressure pads, to model and predict discrete human actions with smoothed n-grams.

We are not modeling specific actions in our work. Rather, we are modeling the sequencing of actions during an anagram game. The above works use primarily data from in situ environments, while our data come from human subjects experiments.

3.4.14 Evaluation of Model Predictions

Predictive models can have many forms. For example, simple classifier algorithms try to predict discrete class labels. Another technique used in predictive modeling is regression analysis, which tries to predict the mean value of a quantitative response variable. Also, the factor analysis approach, tries to predict the distribution of a set of correlated quantitative variables (i.e., predicts the values of some variables from knowing the values of others). The evaluation of prediction models can be developed using a variety of different methods and metrics. For classification, the usual measure of error is the fraction of cases mis-classified, called the mis-classification rate or the error rate. For linear regression, the measure of accuracy is R^2 and the measure of error is the sum of squared errors or $1 - R^2$. For the method of factor analysis, when a model predicts a whole distribution, the negative log-likelihood is the usual measure of error, but sometimes a direct measure of the distance between the predicted and the observed distribution is used [113].

In this work, we are primarily concerned with using well-known measures to characterize the difference between two statistical distributions. In our work, one distribution is generated from experimental data, and one distribution is generated from predictions of models from Section 3.6. [96] list ten metrics on probability measures: (1) Discrepancy, (2) Hellinger distance, (3) Relative entropy (or Kullback-Leibler divergence), (4) Kolmogorov (or Uniform) metric, (5) Lévy metric, (6) Prokhorov metric, (7) Separation distance, (8) Total variation distance, (9) Wasserstein (or Kantorovich) metric, and (10) χ^2 distance.

It is clear that there are many measures for comparing two probability distributions, and different ones are used in different settings. For our needs, we have chosen to use KL-divergence (also called relative entropy). The KL-divergence was introduced by Solomon Kullback and Richard Leibler in 1951 as the directed divergence between two distributions [141].

The most important measure in information theory is called entropy and measures the uncertainty associated with a random variable. The entropy of a random variable X denoted $H(X)$ is a lower bound on the average length of the shortest description of the random variable [66]. The concept of information entropy was introduced by [218]. The Shannon entropy,

defined in [218], measures how close a random variable is to being uniformly distributed. Shannon entropy estimates the average minimum number of bits needed to encode a string of symbols based on an alphabet size and the frequency of the symbols. Is calculated using the following formula $H(X) = -\sum_{x \in X} P(x) \log P(x)$. The KL-divergence measures the discrepancy between two probability distributions, and from which Shannon entropy can be constructed. For discrete probability distributions P and Q defined on the same probability space, the KL-divergence between P and Q is defined to be

$$D_{KL}(P||Q) = -\sum_{x \in X} P(x) \log \Big(\frac{Q(x)}{P(x)}\Big).$$

In the simple case, a KL divergence of 0 indicates that the two distributions in question are identical. The KL-divergence is not symmetric. The evaluations of our models are described in Section 3.7.

3.4.15 Abduction and Abductive Loop

Works on constructive procedures for implementing abductive analyses include [112, 246]. We extend those works for abductive looping by making modeling a first-class process, and by adding the task of determining what to do in the next iteration. In addition to the applications cited in the Introduction, abduction was used to understand emergency room personnels' efforts to save injured people in terms of "social viability" [245]. Perhaps the work closest to ours is [221] in that they develop models and make predictions based on data. However, their data are either artificially generated or address isolated individuals, and they use abduction rather than abductive iterations. Several additional works are provided in Section 3.2.4.

3.5 Experiments

In Section 3.5.1, we provide a description of the experiment and overview the web application (web app) software system for running games. An experiment consists of an anagram game and two executions of the dynamic identity fusion index (DIFI) procedure. We present analyses of the experimental data that illustrate how players interact in the anagram games in Section 3.5.2.

3.5.1 Experiment Description

The elements of an experiment, as specified in Figure 3.4, are:

1. Players are recruited from Amazon Mechanical Turk, to play our anagram game.

2. Players receive directions on how to use the platform, including a description of the game and how to play it, and information about remuneration at the end of the game.
3. Players play the Dynamic Identity Fusion Index (DIFI) 1, DIFI1, procedure individually.
4. Players play the anagram game in a cooperative group setting.
5. Players play the DIFI2 procedure individually.

The terms DIFI1 and DIFI2 are used to indicate the first and second uses of the DIFI procedure (Figure 3.4). The two DIFI procedures are the same.

Figure 3.4: Steps for the overall online game include: recruitment of players from Amazon Mechanical Turk (AMT), directions for the use of the platform, DIFI1 score procedure, anagram game, and DIFI2 score procedure.

Group Anagram Game Description

The group anagram game is a word construction game, where n players cooperate in sharing letters to form and submit words of length ≥ 3 letters. Communication channels between pairs of agents mean that they can request and share letters with each other. An edge between nodes (players) v_i and v_j means that v_i and v_j can share letters with each other; v_i and v_j are neighbors. We use random regular graphs of degree k on the n players so that everyone has the same number of neighbors. Over all abductive loops, experiments are run in groups with nominal values of $10 \leq n \leq 20$ and with regular degrees $2 \leq k \leq 8$.

An example game configuration and system states are provided in Figure 3.5. The game configuration can be represented as a graph $G(V, E)$ where V is the set of nodes that represent players and E is the set of edges that are communication channels between pairs of nodes. Red channels are for letter request and green channels are for letter replies. The number of players is $n = 4$ with players v_1, v_2, v_3, and v_4, the degree of each player is $k = 2$, and the number of initial letters per player is $n_L = 3$. The players have the following initial letters: $L_{v_1}^{init} = \{RID\}$, $L_{v_2}^{init} = \{AGR\}$, $L_{v_3}^{init} = \{HNO\}$, and $L_{v_1}^{init} = \{UTY\}$. Key (#) shows the sequence of actions by all the players during a game. In Figure 3.5, the sequence of actions is detailed in Table 3.2, which narrates the actions. The To-Reply-Buffer and the Request-Sent-Buffer of Figure 3.5 are buffers, per player, that contain outstanding requests-to-be-fulfilled and requests of letters, respectively. For example, in step (5) of Table 3.2, v_2 has a request from v_3 for the letter A. Therefore, v_3 has an entry A in its Request-Sent-Buffer and v_2 has an entry A in its To-Reply-Buffer. If/when v_2 fulfills that request (in the example

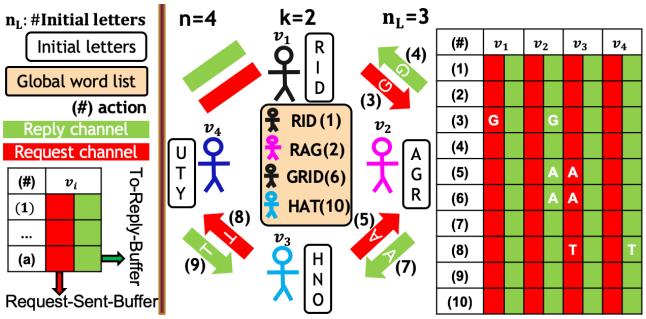


Figure 3.5: Anagram game configuration with a $k = 2$ regular graph on $n = 4$ players (v_1, v_2, v_3, v_4) with number of initial letters $n_L = 3$ assigned to each player, as shown in the boxes next to the players. Requests for letters and replies are sent across the channel links (red to request letters, green to reply with letter). Request-Sent-Buffer keeps track of player v_i's letter requests. To-Reply-Buffer contains letter requests from other players to v_i. Key (#) shows the sequence of actions by all the players during a game. Table 3.2 shows a detailed description of these actions.

(#)	Player	Action	Description
(1)	v_1	form word	v_1 forms word "RID"
(2)	v_2	form word	v_2 forms word "RAG"
(3)	v_1	request letter	v_1 requests v_2 for letter "G"
(4)	v_2	reply letter	v_2 replies v_1 with letter "G"
(5)	v_3	request letter	v_3 requests v_2 for letter "A"
(6)	v_1	form word	v_1 forms word "GRID"
(7)	v_2	reply letter	v_2 replies v_3 with letter "A"
(8)	v_3	request letter	v_3 requests v_4 for letter "T"
(9)	v_4	reply letter	v_4 replies v_3 with letter "T"
(10)	v_3	form word	v_3 forms word "HAT"

Table 3.2: Action table detailing the sequences of actions by all players during the anagram game example from Figure 3.5. The first column defines the number of the sequence of actions during the game. For this example, the duration of the game is 10 actions. The second column shows the player initiating the action. The third column shows the name of the action. The fourth column provides a description of the action.

this happens in step (7)), v_3's "received letters" will contain an A, A will be removed from v_3's Request-Sent-Buffer, and v_2's To-Reply-Buffer will become empty.

Team members earn money by forming as many words as possible. Players are told that the total team earnings e_t are split evenly; each player receives e_t/n, so that it is in their interests to assist their neighbors. Players must form words with at least three letters. A single letter can be used any number of times in a word, e.g., a player can form the word TOT if she has a T and an O among her current letters (own letters and those received from neighbors) because the T can be used twice. Moreover, players do not lose letters that they use. Hence, a player has infinite multiplicity of each letter they possess so that letters can be reused any number of times. This means that a player only has to request a letter (and receive it) one time. Therefore if a player forms "TOT", she still possesses T and O with which to form more words. A player can only share their initial letters with her neighbors; letters received from neighbors cannot be shared with others. These rules were designed to foster word construction, to increase earnings potential, and to foster team cohesion.

A total of 105 players participated in 47 games. The anagram game is played for five minutes. Table 3.3 shows all the game configurations played.

Table 3.3: Description of anagram game configurations played with players recruited from Amazon Mechanical Turk.

Degree, k	**No. Players,** n	**No. Games**
2	10	18
2	20	10
3	15	1
4	15	9
5	15	2
6	15	3
8	15	4

We provide an overview of the web application (app) game platform that we built. The web app software platform consists of the oTree infrastructure [54] for recruiting players from Amazon Mechanical Turk (AMT) and interactions during the game; Django Channels for player interactivity; and JavaScript and HTML for generating the screens for a consent form, instructions, information, a survey, and game interactions. Experiments and data analyses *are part of* the abductive loop of Sections 3.3 and 3.8 and Figure 3.3. This game platform was constructed as part of our work.

A screen shot of one player's screen at one point in time is shown in Figure 3.6. Each player is given $n_L = 3$ letters that she can use to form words and that she can share with others. She has an infinite supply of letters so that sharing letters does not inhibit her own use of letters. A player can also request letters from her neighbors and if the neighbors provide

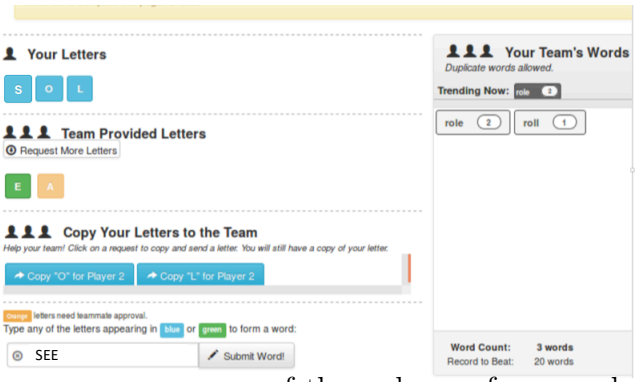


Figure 3.6: The anagram game screen of the web app for one player. This player has own letters "S," "O," and "L" and has requested an "E" and "A" from neighbors. The "E" is green, so this player's request has been fulfilled and so "E" can be used any number of times in forming words. But the request for "A" is still outstanding so cannot be used in words. Below these letters, it shows that player 2 has requested "O" and "L" from this player; this player has to reply to these requests, if she so chooses. Below that is a box where the player types and submits new words, like "SEE."

those letters, then she can use those letters in words, but she cannot pass on the received letters.

Initially, a player sees her n_L own letters and those of all of her neighbors, but has access only to her own letters. Over the 5-minute anagram game duration, players can form words, request letters from their neighbors and reply to requests.

DIFI Description

The DIFI procedure precedes and follows the anagram game. Each player executes individually the DIFI procedure [269], to measure the degree to which a player feels part of a team (i.e., associates their identity with that of a team). Each player does this individually by moving a circle in a browser, representing herself, relative to a fixed team circle. The DIFI score is in the range [-100,125], with a score < 0 representing no overlap of circles, and therefore indicating no CI; $= 0$ representing the circles just touching; and > 0 indicating overlap of the two circles and hence formation of some level of CI. See Figure 3.7. There are screens in the web app that also step each player through the steps in the DIFI game/procedure.

3.5.2 Experimental Data

In this section we present an analysis of the experimental data that illustrates how players interact in the anagram games. We focus on experimental data that are useful in modeling. We identify three main actions for a player during the game: (1) letter request, (2) letter

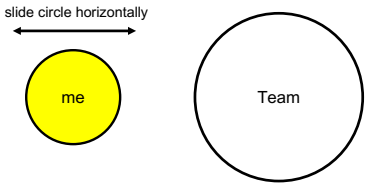


Figure 3.7: DIFI game where player v_i moves the smaller circle, representing herself, either over (partially), or away from, the bigger circle that represents the team. The team circle is stationary. The distance δ between centroids of circles is measured. The distance is such that $\delta = 0$ corresponds to the small and large circles just touching; $\delta < 0$ means that the two circles are disjoint; and $\delta > 0$ means the two circles overlap. The distance δ is transformed into a DIFI value. The range in DIFI value is: $-100 \leq \delta \leq 125$. The DIFI score is a proxy for CI. This is an individual player game.

reply, and (3) word formation and submission.

We define the following variables for the actions in the game:

- When v_i sends a requests for a *letter* to v_j, a **request sent** occurs.
- When v_j receives the *letter* request from v_i, a **request received** occurs.
- When v_j replies with the *letter* requested from v_i, a **reply sent** occurs.
- When v_i receives the *letter* reply from v_j, a **reply received** occurs.
- When v_i uses its own letters to form a word, a **word formed** occurs.

Table 3.4 shows a summary of the section plots and the questions we answer with the analyses.

Timestamp for Letter Request

The number of letters a player can request through a game depends on the number of its neighbors. Each neighbor can share up to three letters (the initial three letters), so if a player has $k = 2$ neighbors, then six letters can be requested throughout the game. If a player has $k = 8$ neighbors, then 24 letters can be requested. We want to analyze the behavior of players with reference to the letter request action and answer the following questions. When do players request letters during the game? How does the number of neighbors affects the behavior of a player to request a letter in the game?

Figure 3.8 shows a histogram with 10 bins of 30-seconds each of timestamps for **request sent**, for 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian

Table 3.4: Summary of the analyses in the Experimental Data Section 3.5.2, and the questions we answer. Section 3.5.2 presents histograms for the timestamps for letter request. Section 3.5.2 presents histograms for the timestamps for letter reply. Section 3.5.2 presents histograms for the timestamps of the time duration between reply received and request sent. Section 3.5.2 presents histograms for the timestamps for word formed.

Section	**Histograms**	**Questions for Analysis**
3.5.2	Timestamps for letter request	When do players request letters during the game?
		How does the number of neighbors affect the behavior of a player to request a letter in the game?
3.5.2	Timestamps for letter reply	When do players reply to letter requests during the game?
		How does the number of neighbors affect the behavior of a player to reply a letter in the game?
3.5.2	Timestamps for time duration(reply received - request sent)	How long does it take players to reply to a letter request?
		How does the number of neighbors affects the time duration between the timestamps of the letter reply action and the letter request action?
3.5.2	Timestamps for word formed	When do players submit words during the game?
		How does the number of neighbors and the number of available letters affects the number of words formed by a player?

kernels is used to estimate the probability density function. It indicates that more letters are being requested during the first half of the 300-second anagram game. To analyze whether the number of neighbors affects the letter request, Figure B.1 in Appendix B.1.1 shows histograms with 10 bins of 30-seconds each for **request sent** for experiments with k= 2, 3, 4, 5, 6, 8. The same trends exist for each value of k. However, if there are few neighbors (k=2) and consequently fewer available letters (3 letters per neighbor), there are fewer letter requests and letter replies near the end of the game.

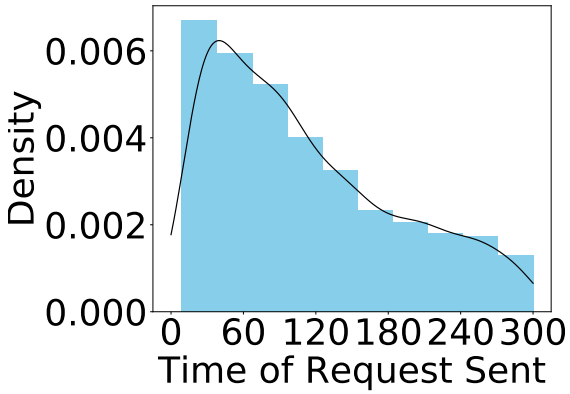


Figure 3.8: Probability density distribution for time of request sent over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used estimate the probability density function. Letter requests are made throughout the game, rather than solely at the outset.

Timestamp for Letter Reply Sent

The number of letters a player can reply with, in response to letter requests, through a game depends on the number of its neighbors. Each neighbor can share up to 3 letters, so if a player has $k = 2$ neighbors, then 6 letters can be replied (when requested) at any time through the game, since the number of letters assigned initially is three. We want to analyze the behavior of players with reference to the letter reply action and answer the following questions. When do players reply letters during the game? How do the number of neighbors affects the behavior of a player to reply a letter in the game?

Figure 3.9 shows a histogram with 10 bins of 30 seconds each, for **reply sent**, for 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. It indicates that letter requests are being replied to throughout the game, but moreso at the earlier stages of the game. To analyze whether the number of neighbors affects the letter request, Figure B.2 in Appendix B.1.2 shows histograms with 10 bins of timestamp for **reply sent** for experiments with $k = 2, 3, 4, 5, 6, 8$. Similar trends are obtained when data are broken down by k. We find that letter reply are made throughout the game, rather than solely at the outset.

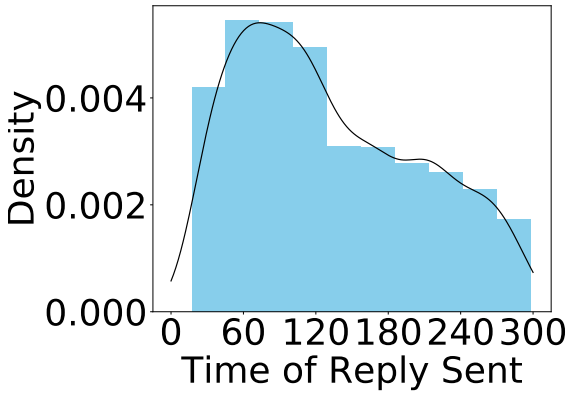


Figure 3.9: Probability density distribution for time of reply sent over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Letter replies are made throughout the game, rather than solely at the outset.

Time duration from Sending a Letter Request to Receiving the Requested Letter

When v_i requests a letter of v_j, it has to wait for v_j to respond. Once v_j replies with the letter, then v_i is allowed to use the received letter and form words to contribute to the team. This time duration between request sent and reply received reveals how long players take to reply to their neighbors' requests. A player only has to request a letter (and receive it) on one occasion to use it as any number of times in forming words. Remember that these rules were designed to foster word construction, to increase earnings potential, and to foster team cohesion. We want to analyze the behavior of players with reference to the time duration between the timestamps of the letter reply action and the letter request action, to answer the following questions. How long does it take for players to reply to a letter request? How does the number of neighbors affect the difference between the timestamps of the letter reply action and the letter request action?

Figure 3.10 shows a histogram with 10 bins of 30-seconds each, for the time difference between **reply received** and **request sent**, for 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used estimate the probability density function. Players generally respond relatively quickly to their neighbors letter requests with replies typically made within 30 seconds of the request.

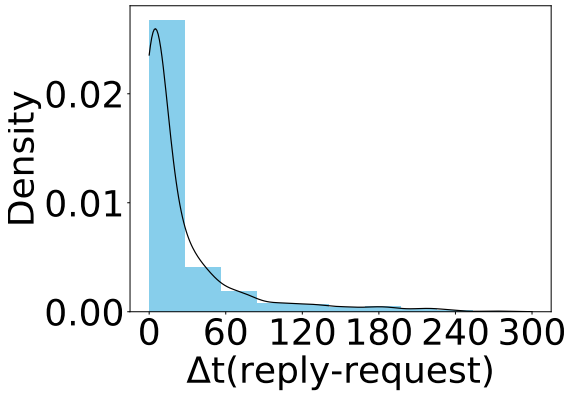


Figure 3.10: Probability density distribution for time duration between requesting a letter and replying to the request, over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Players generally respond relatively quickly to their neighbors letter requests, with replies typically made within 30 seconds of the request.

To analyze whether this behavior is common while increasing the number of k neighbors in a game, Figure B.3 in Appendix B.1.3 shows histograms with 10 bins of 30-seconds each of timestamp change between reply received and request sent for experiments with $k = 2, 3, 4, 5, 6, 8$. The number of neighbors doesn't affect this type of action, players generally respond relatively quickly to their neighbors letter requests with replies typically made within 30 seconds of the request.

Timestamp for Word Formed

At any time during a game, a player can form a word and submit it for validation to our web application. If a player possesses letters to form a valid word, then she forms and submits a word, the application validates it, and the word is added to the game screen. We want to analyze the behavior of players with reference to the action of word formed and answer the following questions. When do players submit words during the game? How does the number of neighbors and the number of available letters affects the number of words formed by a player?

Figure 3.11 shows a histogram with 10 bins of 30-seconds each for **word formed**, for 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used estimate the probability density function. It suggests that words are being formed throughout the game, and even up through the end of the game. This justifies a 5-minute anagram game duration. To analyze whether the number of neighbors affects the word formation, Figure B.4 in Appendix B.1.4 shows histograms with 10 bins of 30-seconds each for timestamp of **word formed** for experiments with $k = 2, 3, 4, 5, 6, 8$. Word submissions

are made throughout the game, and the number of neighbors and available letters does not affect this behavior.

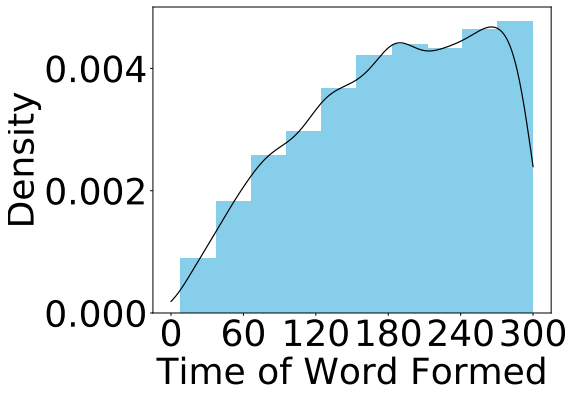


Figure 3.11: Probability density distribution for time of forming words over the 300-second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. It shows 47 experiments with $k = 2, 3, 4, 5, 6, 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Word submissions are made throughout the game, and the number of neighbors and available letters, does not affect this type of action.

3.6 Agent-Based Models (ABMs) of the Group Anagram Game and Modeling Results

We present three progressively more sophisticated ABMs of the anagram game that are used in the abductive loop analyses to follow in Section 3.8. All models were developed *as part of* the abductive loop process, but are presented here to emphasize their construction and evaluation, and to obviate the need for a large digression for the models in the description of the AL process in Section 3.8. Each model represents the behavior of one player or agent. The models are data-driven, and hence *inductive inference* is used with data in three ways: to inform model structure, model parameters, and to compute parameter values.

In all models, we represent the set V of players and the set E of their communication channels (edges) as an undirected graph $G\,(V, E)$. The game is modeled as a discrete-time stochastic process, where at each time step, a player performs one of the actions from the action set A, consisting of: (i) a_1: idling (i.e., thinking); (ii) a_2: replying to a neighbor with a requested letter, (iii) a_3: requesting a letter from a neighbor, and (iv) a_4: forming and submitting a word. Table 3.5 shows the actions.

Table 3.5: Actions of players in the model. The set A of actions is $\{a_1, a_2, a_3, a_4\}$.

Item	Variable	Name	Description
1	a_1	idling	thinking
2	a_2	reply	replying to a neighbor with a requested letter
3	a_3	request	requesting a letter from a neighbor
4	a_4	words	forming and submitting a word

3.6.1 Discrete-Time Stochastic Process

In all ABMs, actions are taken at integer numbers of seconds; that is, simulations of interacting agents take place as time advances in discrete 1-second increments from 0 to 300. This time increment is based on the experimental data where no player takes two or more actions in one second.

We chose ABMs for their generative properties, fine granularity, and ability to model temporal effects. These enable us to more readily quantify "what if" scenarios (counter factuals) as part of parametric studies and sensitivity analyses. Also, ABM maps well onto the actual experiments: players have connections in a network arrangement and they interact through their edges, taking actions at discrete times as in Figure 3.5.

The choice of discrete time or discrete event simulation arises. If we selected discrete event simulations, then we would also have to predict the time at which the next action for a player takes place (at some Δt into the future). However, with discrete time, we know we are always predicting for the next time unit (here, one second). We also used a multinomial logistic regression model; other approaches could have been employed.

3.6.2 KL-Divergence

To measure the performance of our models, we use *Kullback Leibler-divergence* between our model prediction on x and the experimental observation of x, *throughout this manuscript*. That is, we are comparing distributions of data: distributions of experimental data against distributions of model predictions. Most relevant for our work is Boltzmann's [20] concept of generalized entropy, where the entropy of a physical system is a measure of disorder related to it. [141] derived an information measure, now referred to as the KL divergence, the negative of Boltzmann's entropy. The motivation for Kullback and Leibler's work was to provide a rigorous definition of information. The Kullback-Leibler distance can be conceptualized as

a directed distance between two models, say a and b [140]. This is a measure of discrepancy. It is not a simple distance because the measure from a to b is not the same as the measure from b to a. It is a directed, or oriented, distance. The KL divergence $D_{KL}(a, b)$ is always positive, except when the two distributions a and b are identical (i.e., $D_{KL}(a, b) = 0$ if and only if $a(x) = b(x)$ everywhere). Entropy is zero if there is unit probability at a single point. If the distribution is widely dispersed over a large number of individually small probabilities, then the entropy is high (e.g., $D_{KL} > 1$).

3.6.3 Overview of the Three Agent Based Models

ABM M0 is a baseline model, where each player makes a probabilistic transition from action $a_i \in A$ to action $a_j \in A$. The transition matrix is time invariant and is the same for all players. Data from the experiments is used to infer the model parameters using a ring topology (degree of each node is 2) of player connectivity within an anagram game. Model M1 is similar to M0 but with the crucial difference that the transition matrix is time variant. Model M2 is similar to M1 but now instead of a ring topology, we used other topologies and infer model parameters (degree from 2 to 8). Models M0, M1 and M2 predict the actions of A for a player but are generic in that letter request a_3, letter reply a_2, and submit word a_4 are not associated with particular letters. For example, if the player action is a_4, then the model assumes that the player can form a word. Table 3.6 shows a description of the three progressively sophisticated models.

Models M0, M1, and M2 are presented in Sections 3.6.4, 3.6.5, and 3.6.6 respectively. In each of these subsections, model development and results are provided.

Table 3.6: Progressively sophisticated models of the anagram game developed in this work. The incremental improvements in models are given, starting with model M0.

Model	**Transition Probabilities**	**Degree** k
M0	fixed	2
M1	temporal	2
M2	temporal	2, 4, 6, 8

Throughout, we use k to denote the number of neighbors (degree) of an agent $v \in V$. Also, we evaluate five variables and their distributions, across all players in a set of games, in comparing models and experiments: $x = (x_1, x_2, x_3, x_4, x_5)$, where x_1 is the number of letter replies received ($RplR$); x_2 is the number of replies sent ($RplS$); x_3 is the number of letter requests received ($RqsR$); x_4 is the number of requests sent ($RqsS$); and x_5 is the number of words formed ($Wrds$). Table 3.7 summarizes these variables.

Table 3.7: Variables that are measured in experiments for each player, and predicted with models for each agent, where vector $x = (x_1, x_2, x_3, x_4, x_5)$. All x_i, $1 \leq i \leq 5$, are time dependent.

Item	Variable	Name	Description
1	x_1	RplR	Number of replies received
2	x_2	RplS	Number of replies sent
3	x_3	RqsR	Number of requests received
4	x_4	RqsS	Number of requests sent
5	x_5	Wrds	Number of words formed

In the results sections for each model, simulations are performed using ABMs that implement each of the described models. These simulations produce, for each player, time histories of the actions in Table 3.7. One hundred simulations are run and results are averaged across these simulations, i.e., are averaged across all players in each simulation. These data are post-processed to generate distributions of the variables in Table 3.7. These distributions from ABM predictions are compared against corresponding distributions generated from experiments.

Note that fixing $n = 10$ in all simulations does not introduce errors because the distributions that we use are density distributions, not counts. Thus, the number of players is normalized out of all comparisons of distributions of experimental data and model predictions.

Table 3.8 shows the structure of comparisons of results for each of the models M0, M1, and M2. First, comparisons are made between distributions of experimental results and model predictions, for each x_i of Table 3.7, at the end of a game (i.e., over all five minutes of an anagram game). Then, these data are broken down into one-minute intervals to assess temporally the distributions of data and predictions. Next, we compute KL-divergence values that provide a scalar representing how well the model predictions of the distributions of x_i compare with those of the experimental data. From Section 3.6.2, $D_{KL} = 0$ means the model distribution agrees very well with the corresponding experimental distribution. As D_{KL} increases from zero, model predictions worsen. Table 3.8 denotes that these comparisons are performed over all five minutes of the anagram game (number 3), corresponding to the end of the group anagram game, and for each one-minute interval over the game (number 4) of Table 3.8. Finally, we compare these sets of computed D_{KL} across all x_i of Table 3.7. The reason for the temporal breakdown is to examine model predictions over time. Temporal comparisons are hidden in numbers 1, 3 and 5 of Table 3.8, which examine aggregated data.

Table 3.8: Summary of the model comparison plots applied to each of the models M0, M1, and M2. For each model, we collect the data into the five groups shown. See the text for details and justification. The fifth column indicates the time period, in minutes, over which experimental data are compared to model predictions.

No	**Method**	**Plot**	**Variables, Player Actions**	**Time**
1	Comparisons of distributions at end of game	(a)	x_1	0-5
		(b)	x_2	0-5
		(c)	x_3	0-5
		(d)	x_4	0-5
		(e)	x_5	0-5
2	Temporal comparisons of distributions	(a)	x_1, x_2, x_3, x_4, x_5	0-1
		(b)	x_1, x_2, x_3, x_4, x_5	1-2
		(c)	x_1, x_2, x_3, x_4, x_5	2-3
		(d)	x_1, x_2, x_3, x_4, x_5	3-4
		(e)	x_1, x_2, x_3, x_4, x_5	4-5
3	Comparisons of KL divergence distributions at end of game		x_1, x_2, x_3, x_4, x_5	0-5
4	Temporal comparisons of KL divergence distributions	(a)	x_1, x_2, x_3, x_4, x_5	0-1
		(b)	x_1, x_2, x_3, x_4, x_5	1-2
		(c)	x_1, x_2, x_3, x_4, x_5	2-3
		(d)	x_1, x_2, x_3, x_4, x_5	3-4
		(e)	x_1, x_2, x_3, x_4, x_5	4-5
5	Comparisons of KL divergence distributions combining all vars.		x_1, x_2, x_3, x_4, x_5	0-5

3.6.4 Baseline Agent-Based Model M0

ABM M0 Development

The goal is to accurately quantify the transition probability from one action $a(t) = a_i$ at time t to the next action $a(t+1) = a_j$ for each agent $v \in V$, $i, j \in [1..4]$ and $a(t) \in A$. For clarity, we use i and j to represent the actions a_i and a_j. Agent v executes a stochastic process driven by transition probability matrix $\Pi = (\pi_{ij})_{m \times m}$, where $m \equiv |A|$ (here, $= 4$)

and

$$\pi_{ij} = Pr(a(t+1) = j | a(t) = i) \text{ with } \sum_{j=1}^{m} \pi_{ij} = 1 \, . \tag{3.1}$$

The transition matrix Π is formed from the data by using successive pairs of actions of players in experiments so that the 16 values of π_{ij} in Equation (3.1) are *constant*, i.e., time-invariant. The matrix in Equation (3.2) shows the transition probabilities for Model M0 (the baseline model) generated from experiment data with $n = 10, k = 2$. For example, given that the action of a player v_i at time t is a_2 (replying to a letter request), the probability that v_i's next action, at time $(t+1)$, is a_1 (thinking) is 0.93.

$$\Pi = \begin{array}{c|cccc} & a_1(t+1) & a_2(t+1) & a_3(t+1) & a_4(t+1) \\ \hline a_1(t) & 0.93 & 0.01 & 0.02 & 0.04 \\ a_2(t) & 0.84 & 0.16 & 0 & 0 \\ a_3(t) & 0.98 & 0.01 & 0.01 & 0 \\ a_4(t) & 0.93 & 0.01 & 0 & 0.06 \end{array} \tag{3.2}$$

ABM M0 (Baseline) Results

We address all of the results in Table 3.8 for model M0.

Comparisons of distributions between model and experiments for individual variables at the end of the anagram game. Figure 3.12 shows the ABM M0 predictions of the $k = 2$ experiments. Figure 3.12(a) shows the distribution of replies received, Figure 3.12(b) shows the distribution of replies sent, Figure 3.12(c) shows the distribution of requests received, Figure 3.12(d) shows the distribution of requests sent, and Figure 3.12(e) shows the distribution of words formed, each at the end of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. It is clear from visual inspection that model M0 predictions are in better agreement with the experimental data for the requests received and requests sent variables. We make this comparison more precise using KL-divergence in Figure 3.13.

Temporal comparisons of distributions between model and experiments for individual player actions. Appendix B.1.5 shows the figures resulting from the temporal analysis by minute of distributions between Model M0 and experiments for $k = 2$. Each plot contains data over a time window for each variable of x from Table 3.7. Often, but not always, the largest discrepancies between the model predictions and experiments occur in the first minute of the game.

Comparisons of KL divergence values between model and experiments for individual variables at the end of the anagram game. Figure 3.13 shows the KL-divergence

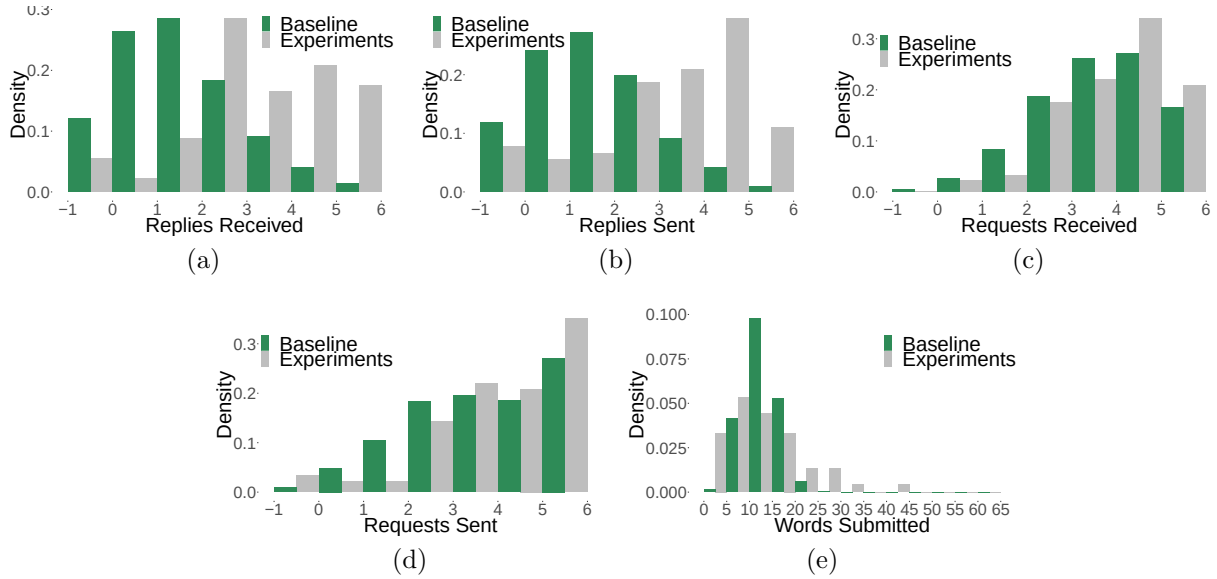


Figure 3.12: ABM baseline M0 predictions of the $k = 2$ experiments (in green) and experimental data (in gray), over the entire 5-minute group anagram game. The probability density function is show for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 2$ experiments. The Baseline M0 predictions are from 100 simulations of a $n = 10$ player game. It is clear from visual inspection that model M0 predictions are in better agreement with the experiment data for the requests received and requests sent variables. We make this comparison more precise using KL-divergence below in Figure 3.13.

values for the baseline M0 across the five parameters of x: lower values are better. M0 does a better job predicting the number of requests received and requests sent at the end of a game. These data span the entire five-minute game. That is, the request-related operations are better predicted than reply operations.

Temporal comparisons of KL divergence values between model and experiments for individual player actions. Figure 3.14 shows the temporal KL-divergence values for the baseline M0 across the five parameters of x, at one-minute intervals: lower values are better. Each Figure contains data over a time window: Figure 3.14(a) shows the 0-1 minute, Figure 3.14(b) shows the 1-2 minute, Figure 3.14(c) shows the 2-3 minute, Figure 3.14(d) shows the 3-4 minute, and Figure 3.14(e) shows the 4-5 minute results of the 5-minute anagram game. These plots show that request-related predictions are better than reply-related predictions for the first three minutes, but are worse for the last two minutes, based on KL-divergence. Reply-related predictions are better in the second half of the five-minute anagram games, but Figure 3.9 shows that in experiments, there are fewer replies in the

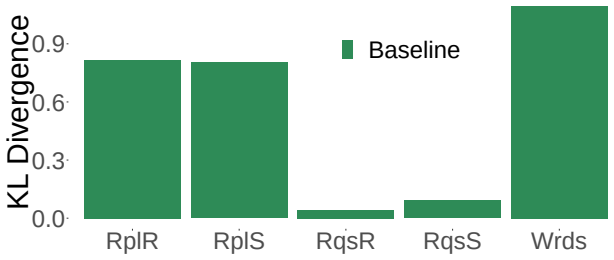


Figure 3.13: KL-divergence values for the baseline Model M0 across the five parameters of x: lower values are better. M0 does a better job predicting the number of Requests Received and Requests Sent. Analyses are based on the data of Figure 3.12, over five minutes, at the end of a game.

second half of the games.

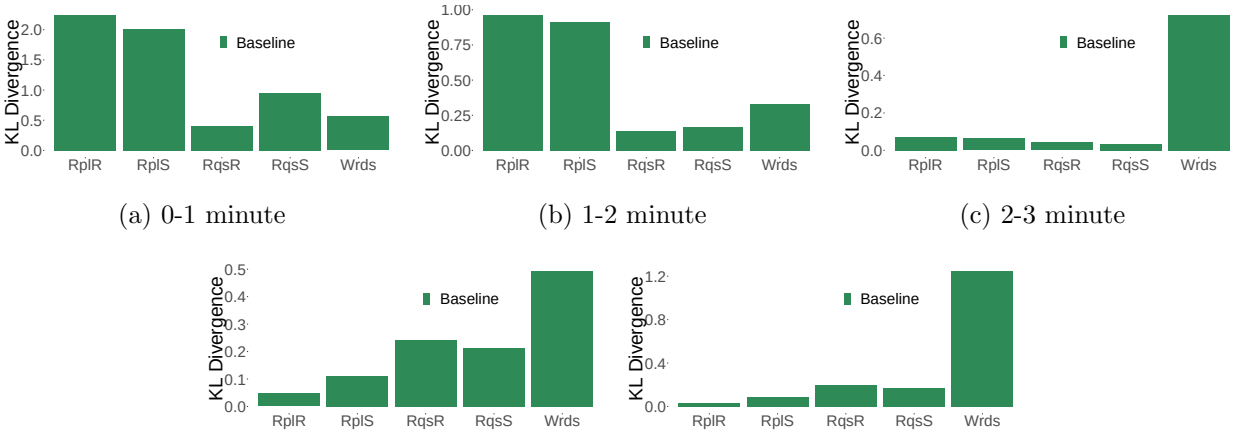
Comparisons of KL divergence values between model and experiments for combining all variables. Figure 3.15 shows the distribution of KL divergence values for comparing distributions of model output with corresponding distributions of experimental data for the anagram game. The model is the ($n = 10$, $k = 2$) baseline. The data sets used in the comparison are ($n = 10$, $k = 2$). There are 30 values in the distribution, with five values for variables x_i over the five-minute game, at the end of the game; and 25 values for the five variables of x over five intervals of one minute duration. It shows that for model M0, some KL divergence values are high (e.g., > 0.5), indicating poor agreement between model predictions and the experiment data. As we see in Figures 3.13 and 3.14, M0 does not do a good job predicting the number of replies received, replies sent, and words formed.

3.6.5 Agent-Based Model M1

Model M1 is similar to M0 but with the important enhancement that the transition matrix Π is time variant.

ABM M1 Development

To make Π (and its components π_{ij} in Equation (3.1)) dynamic in time and account for history effects, four variables are introduced in Equation (3.3): number $z_L(t)$ of letters that v has available to use (i.e., in hand) at t; number $z_W(t)$ of valid words that v has formed; size $z_B(t)$ of the buffer of letter requests that v has yet to reply to; and number $z_C(t)$ of consecutive time increments that v has taken the same action. See Table 3.9. Thus, letting $z = (1, z_L, z_W, z_B, z_C)_{(m+1)\times 1}$, we can model π_{ij} as a function of these covariates, among other variables.



(a) 0-1 minute (b) 1-2 minute (c) 2-3 minute

(d) 3-4 minute (e) 4-5 minute

Figure 3.14: KL-divergence values for the baseline M0 model across the five parameters of x at one-minute intervals: lower values are better. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. The data are for conditions $(n = 10, k = 2)$. These plots show that request-related predictions are better than reply-related predictions over all time intervals. The reply-related predictions are better in the second half of the five-minute anagram games, but Figure 3.9 shows that in experiments, there are fewer replies in the second half of the games.

We use multinominal logistic regression to model π_{ij} as

$$\pi_{ij} = \frac{\exp(\boldsymbol{z}'\boldsymbol{\beta}_j^{(i)})}{1 + \sum_{h \neq i} \exp(\boldsymbol{z}'\boldsymbol{\beta}_h^{(i)})}, \tag{3.3}$$

where $\boldsymbol{\beta}_j^{(i)} = (\beta_{j1}^{(i)}, \ldots, \beta_{j,m+1}^{(i)})'$, $\boldsymbol{\beta}_i^{(i)} = \mathbf{0}$, and prime indicates transpose. For a given i, the parameter set can be expressed as

$$\boldsymbol{B}^{(i)} = \begin{pmatrix} \beta_{11}^{(i)} & \beta_{12}^{(i)} & \cdots & \beta_{1,m+1}^{(i)} \\ \beta_{21}^{(i)} & \beta_{22}^{(i)} & \cdots & \beta_{2,m+1}^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{41}^{(i)} & \beta_{42}^{(i)} & \cdots & \beta_{4,m+1}^{(i)} \end{pmatrix}. \tag{3.4}$$

Parameters in Equation (3.4) are inferred from the $k = 2$ experimental data using the framework of maximum likelihood estimation for the multinomial distribution.

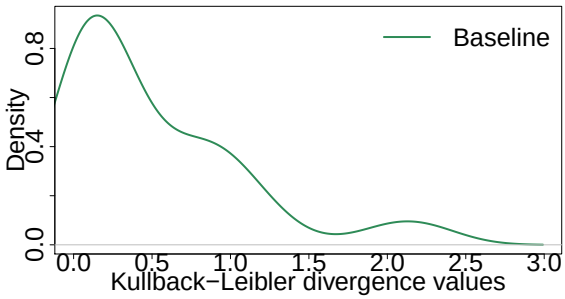


Figure 3.15: Distribution of KL divergence values for comparing distributions of model output with corresponding distributions of experimental data, for the group anagram game. The model is the $(n = 10, k = 2)$ baseline M0 ABM. The data sets used in comparison are experiments: $(n = 10, k = 2)$. The data sets used in comparison are experiments: $(n = 10,\ k = 2)$. There are 30 values in the distribution, with five values for the variables of x at the end of the game (over five minutes of the game), and 25 values for each of the five variables of x over five intervals of one minute duration. It shows that for Model M0, some KL divergence values are high, indicating that the model is in poor agreement with data. As we see in Figure 3.13, M0 does not do a good job predicting the number of replies received, replies sent, and words formed.

Inductive Inference

We address the three dimensions of inference stated above: (i) model structure; (ii) model parameters; and (iii) parameter values. First, the model structure is informed by the $k = 2$ data, by design, as described above. Second, the parameters identified in the feature vector z are described and justified in Table 3.9. In fact, we claim that identifying this feature vector has elements of art. Third, parameters in Equation (3.4) are inferred from the $k = 2$ experimental data using the framework of maximum likelihood estimation for the multinomial distribution.

The reason to emphasize inductive inference is because this is an integral part of the abductive looping process, and of abduction itself: ***the data drive the model and theory development and hypothesis identification****, and not the other way around.*

ABM M1 Results

Results for Model M1 are provided according to Table 3.8 as was done for Model M0. In many cases, we compare KL-divergence values for M0 and M1 to show improvements in performance. These results, like those for model M0, are compared against the $k = 2$ data in Table 3.3.

Comparisons of distributions between models and experiments for individual

Table 3.9: The feature vector $z = (z_L(t), z_W(t), z_B(t), z_C(t))$ used in the models M1 and M2. These capture history effects in determining the next action of a player.

Variable	Name	Description
z_B	Size of reply buffer.	Number of current letter requests to which this player may reply. Captures the notion that the more letter requests that have not been replied to, the more likely v is to reply.
z_L	Numer of letter in hand.	Number of unique letters in hand to form words. Captures the idea that the more letters v has in-hand, the more likely the agent is to form words.
z_W	Number of words formed.	Number of words formed. Captures the notion that the more words that have been formed, the larger the vocabulary of the player.
z_C	Number of consecutive actions.	Number of consecutive time steps at which player takes the same action. Captures the notion that the more time v is idle (thinking), the more likely v will take some other action at the next timestep.

variables at the end of the anagram game. Figure 3.16 shows M0 and M1 model predictions and experimental data distributions for all variables in Table 3.7. These data are over all five minutes of the anagram game for all $k = 2$ experiments. Model predictions are averages over 100 simulations with $n = 10$ players. Figure 3.16(a) shows the distributions of replies received, Figure 3.16(b) shows the distributions of replies sent, Figure 3.16(c) shows the distributions of requests received, Figure 3.16(d) shows the distributions of requests sent, and Figure 3.16(e) shows the distributions of words formed. It is clear from visual inspection that model M1 predictions are in better agreement with the experiment data than are M0 predictions. We make this comparison more precise using KL-divergence in Figure 3.17.

Temporal comparisons of distributions between models and experiments for individual variables. Appendix B.1.6 shows the figures resulting from the temporal analysis by minute of distributions between Models M0, M1 and Experiments for $k = 2$. Each plot contains data over a 1-minute time window for each variable of x from Table 3.7. It is clear from visual inspection that model M1 predictions are in better agreement with the experiment data than are M0 predictions.

Comparisons of KL divergence values between models for individual variables at the end of the anagram game. Figure 3.17 shows KL divergence values for comparing distributions of model outputs with corresponding distributions of experimental data for the anagram game. The models are (baseline) M0 and M1 for the ($n = 10$, $k = 2$) experiments. The comparisons are at the end of the game, i.e., at $t = 5$ minutes, over the entire game. For each experiment/model combination, the variables (and hence distributions) compared

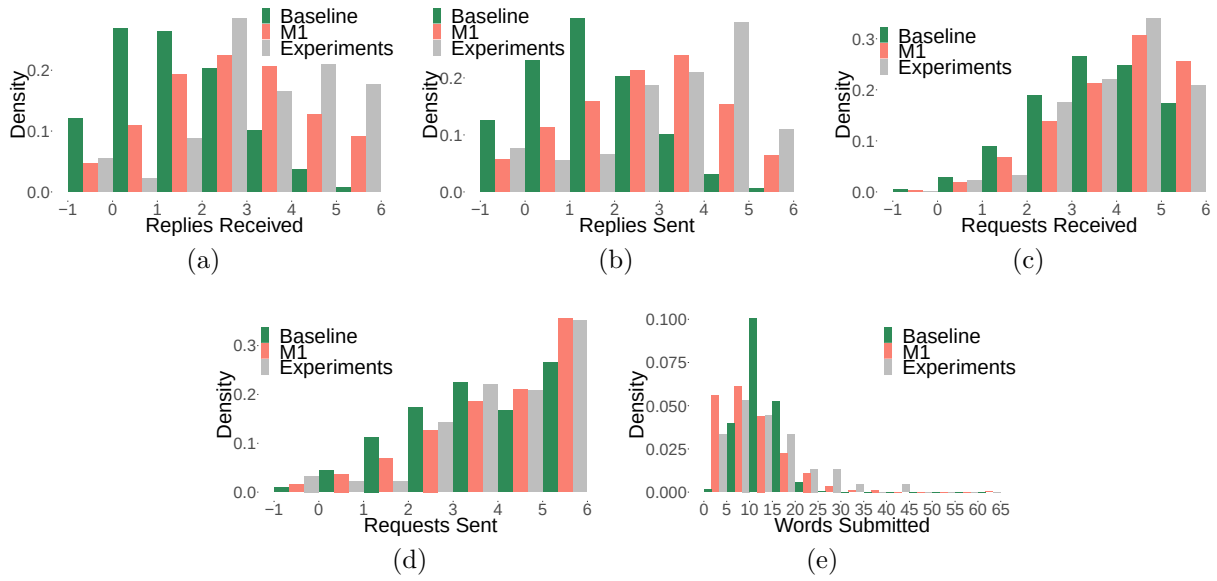


Figure 3.16: ABM M0 and M1 predictions of the $k = 2$ experiments, along with the experimental data. The probability density function is show for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. It is clear from visual inspection that model M1 predictions are in better agreement with the experiment data than are M0 predictions. We make this comparison more precise using KL-divergence in Figure 3.17.

are: number of replies received, number of replies sent, number of requests received, number of requests sent, and number of words formed. Lower values are better. This figure shows that M1 generates predictions much closer to the experimental data than does M0. For example, M1 significantly reduces the reply-related and words formed KL-divergence values (weaknesses of model M0 as shown in Figure 3.13).

Temporal comparisons of KL divergence values between models for individual player actions. Figure 3.18 shows the temporal KL-divergence values for the baseline M0 and M1 across the five parameters of x: lower values are better. Each plot contains data over a time window: Figure 3.18(a) for 0-1 minute, Figure 3.18(b) for 1-2 minute, Figure 3.18(c) for 2-3 minute, Figure 3.18(d) for 3-4 minute, and Figure 3.18(e) for 4-5 minute time intervals of the 5-minute anagram game.

The plots demonstrate that KL-divergence values for the model M1 predictions are closer to the experimentally-determined data distributions than are those from model M0. While

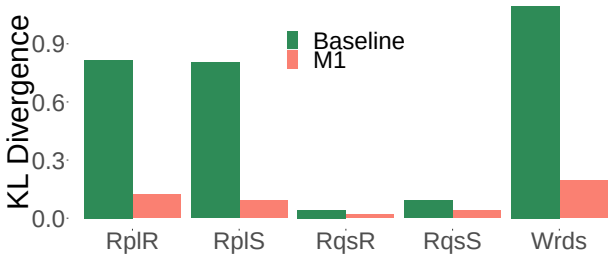


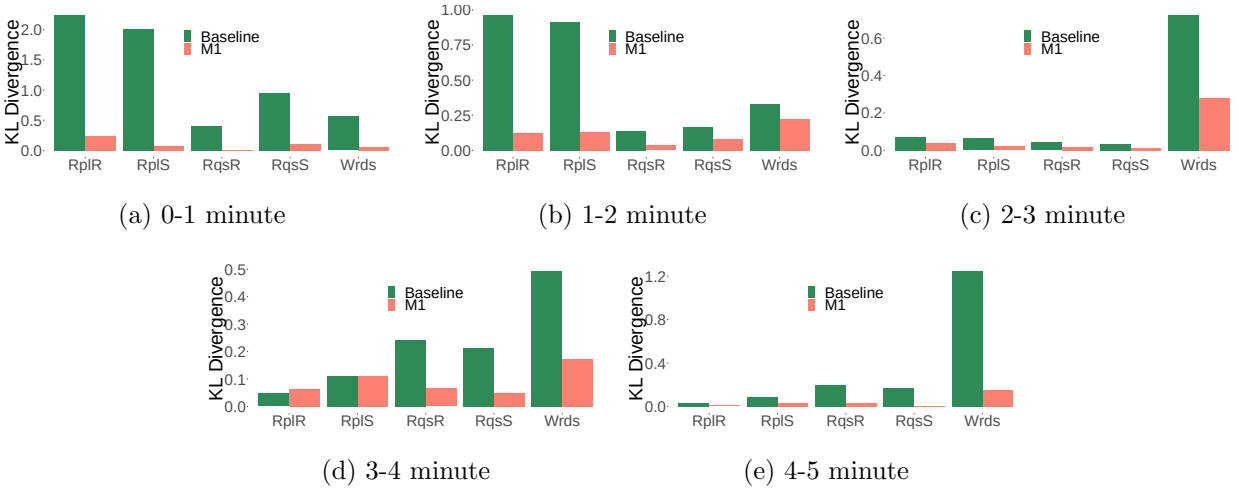
Figure 3.17: KL-divergence values for the baseline M0 and M1 models across the five parameters of x: lower values are better. The modeling conditions are experiment with $k = 2$. This figure shows that M1 greatly improves a weakness of model M0 in poorly representing RplR (number of replies received), RplS (number of replies sent), and Wrds (number of words formed).

Model M0 has good predictions for the minute 3 and minute 5 (with the exception of the words formed), Model M1 has better predictions for the minute 3 and minute 5 for all five x variables of Table 3.7. These data are significant because they evaluate the quality of the models to predict behavior temporally. That is, just because a model can produce predictions at the end of some scenario, this does not mean that it can capture the trajectory (or time evolution) of phenomena. With these types of plots, we demonstrate that our models do capture temporal behavior.

Comparisons of KL divergence distributions between models and experiments for combining all variables. Figure 3.19 shows the distribution of KL divergence for comparing distributions of model output with corresponding distributions of experimental data for the anagram game. The models are $(n = 10, k = 2)$ M0 and M1. The data sets used in comparison are experiments: $(n = 10, k = 2)$. There are 30 values in the distribution, with five values for each variable x at the end of the game, and 25 values for the five variables x over five intervals of one-minute increment. It shows that for Model M1, the great majority of KL-divergence values are less than 0.2, while they can be much greater for Model M0.

Summary of M0 and M1 model comparisons. Clearly, ABM M1 is in better agreement with the experimental data compared to the baseline model. From KL-divergence values in Figures 3.17 through 3.19, it is clear that the predictions of M1 represent the experimental data better than those of the baseline model.

In addition, we use M1 to make predictions for anagram games with $k > 2$, resulting in more interactions. Counterintuitively, as shown in Figure 3.20, the number of replies does not change as k increases. These results call for more experiments at larger k. Note that we exercise M1 learned from experiments with $k = 2$. The results in Figure 3.20 indicate that M1 predicts no changes in the number of letter replies received as k increases, which seems counter intuitive. One would expect more letter requests and replies with increasing numbers of neighbors. These types of data lead us to construct model M2.



(a) 0-1 minute (b) 1-2 minute (c) 2-3 minute

(d) 3-4 minute (e) 4-5 minute

Figure 3.18: KL-divergence values for the baseline M0 and M1 across the five parameters of x: lower values are better. The modeling conditions are experiment with $k = 2$. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. While Model M0 has good predictions for the minute 3 and minute 5 (with the exception of the words formed), Model M1 has better predictions for the minute 3 and minute 5 for all five x variables.

We remark that we also fitted M1 using experimental data with $k = 4$ (call this Model M1b), and consequently made predictions for the case of $k = 2$. We compared the distributions of x between prediction and experimental results using KL-divergence, and determined values in the range 0.11 to 0.46, indicating good predictions. Note that Model M1b is interpolating when it predicts $k = 2$ experimental data, while Model M1 is extrapolating to predict $k = 4$ experimental data.

3.6.6 Agent-Based Model M2

ABM M2 Development

Model M1 was developed with data where all game players have the same degree $k = 2$. To generalize M1 to incorporate various k, we conducted additional experiments with $2 < k \leq 8$ as a part of the second AL (Section 3.8.4 below).

We build a hierarchical model to incorporate the effect of agent degree k. For different values of k, the parameter coefficients in $\boldsymbol{B}^{(i)}$, used in Equation (3.3), are now a function of k, denoted as $\boldsymbol{B}^{(i)}(k)$. We use an orthogonal polynomial basis to construct a continuous

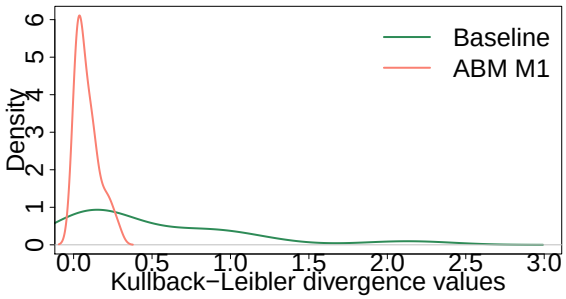


Figure 3.19: Distribution of KL divergence for comparing distributions of model output with corresponding distributions of experimental data for the anagram game. Models are $(n = 10, k = 2)$ M0 and M1. The data sets used in comparison are experiments: $(n = 10, k = 2)$. There are 30 values in each distribution, with five values for variable x at the end of the game over all five minutes, and 25 values for the five variables x over five intervals of one minute increment each. These results shows that for model M1, the KL divergence values are frequently low, indicating that it is in better agreement with experimental data.

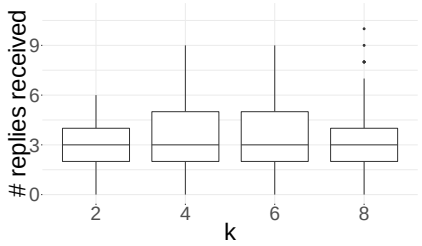


Figure 3.20: M1 model distributions predicted for the number of replies received at the end of game ($n = 10$, 100 simulations), for different regular degrees k of the game network G. This partially motivated our development of ABM M2, since model M1 predictions do not vary significantly with the number of a player's neighbors.

and smoothing function for $\beta_{jh}^{(i)}(k)$ for any given i, j, h, as

$$\beta_{jh}^{(i)}(k) = \alpha_0^{(i,j,h)} + \alpha_1^{(i,j,h)}\xi_l(k) + \alpha_2^{(i,j,h)}\xi_q(k), \tag{3.5}$$

where ξ_l and ξ_q are the linear and quadratic functions of the orthogonal basis in terms of k. We have

$$\boldsymbol{B}^{(i)}(k) = \boldsymbol{C}_0^{(i)} + \boldsymbol{C}_1^{(i)}\xi_l(k) + \boldsymbol{C}_2^{(i)}\xi_q(k), \tag{3.6}$$

where

$$\boldsymbol{C}_r^{(i)} = \begin{pmatrix} \alpha_{11}^{(i,r)} & \alpha_{12}^{(i,r)} & \dots & \alpha_{1,m+1}^{(i,r)} \\ \alpha_{21}^{(i,r)} & \alpha_{22}^{(i,r)} & \dots & \alpha_{2,m+1}^{(i,r)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{41}^{(i,r)} & \alpha_{42}^{(i,r)} & \dots & \alpha_{4,m+1}^{(i,r)} \end{pmatrix}, r = 0, 1, 2, \tag{3.7}$$

with $\alpha_{ih}^{(i,r)} = 0$ for any r and h.

Inductive Inference

We address the three dimensions of inference, as for M1: (i) model structure; (ii) model parameters; and (iii) parameter values. In this case, the model structure we employ to capture the effect of k was identified a priori. However, if the model structure was found lacking, we would have tried another approach. The model parameters given in Equations (3.6) and (3.7) were also anticipated owing to the development of ABM M1. Hence, these first two steps were not solely driven by the data. To estimate the parameters sets $\boldsymbol{C}_0^{(i)}, \boldsymbol{C}_1^{(i)}, \boldsymbol{C}_2^{(i)}$, we use maximum likelihood estimation across the experimental observations for $k = 2$, 4, 6, and 8. For a given i and k, denote the corresponding observational data as $\mathcal{D}_k^{(i)}$. Then we conduct parameter estimation by

$$\hat{\boldsymbol{C}}_0^{(i)}, \hat{\boldsymbol{C}}_1^{(i)}, \hat{\boldsymbol{C}}_2^{(i)} = \arg\max \sum_{k=d_{min}}^{d_{max}} \log L(\boldsymbol{C}_0^{(i)}, \boldsymbol{C}_1^{(i)}, \boldsymbol{C}_2^{(i)} | \mathcal{D}_k^{(i)}),$$

where $L(\boldsymbol{C}_0^{(i)}, \boldsymbol{C}_1^{(i)}, \boldsymbol{C}_2^{(i)} | \mathcal{D}_k^{(i)})$ is the likelihood function with respect to the data $\mathcal{D}_k^{(i)}$ collected under the setting of k neighbors in the experiments of Section 3.5.

ABM Model M2 Results

Results for model M2 are provided according to Table 3.8. Results are often compared to those for model M1.

Comparisons of distributions between models and experiments for individual variables at the end of the anagram game. Figure 3.21 shows data distributions at the end of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions of distributions (blue) for 100 simulations of an $n = 10$ player game. These results are over all five minutes of the group anagram game.

Figure 3.22 shows data distributions at the end of the 5-minute anagram game (gray bars) for all $k = 4$. In appendix B.1.7, Figure B.15 shows data distributions at the end of the 5-minute anagram game (gray bars) for all $k = 6$. Figure B.16 shows data distributions at the end of the 5-minute anagram game (gray bars) for all $k = 8$. Model M1 is shown in red for comparison. In all of these figures, Figure (a) shows the distributions of replies received, Figure (b) shows the distributions of replies sent, Figure (c) shows the distributions of requests received, Figure (d) shows the distributions of requests sent, and Figure (e) shows the distributions of Words Formed. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. As expected, M1 and M2 perform equally well for $k = 2$ as M1 is learned from $k = 2$ experimental data.

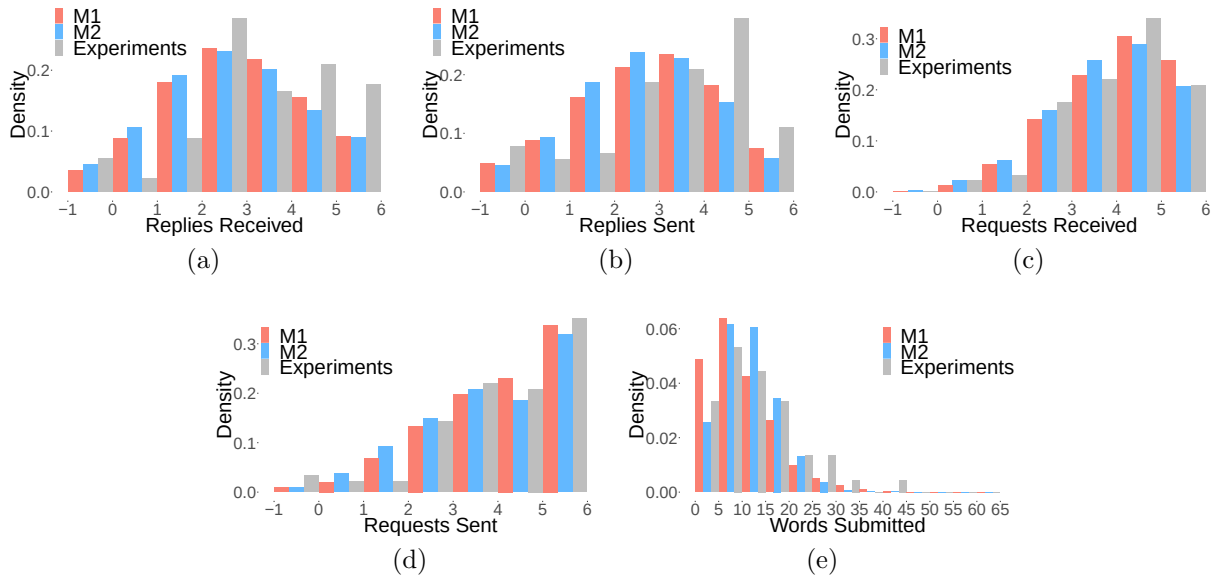


Figure 3.21: ABM M1 and M2 predictions of the $k = 2$ experiments, and experimental data, over all five minutes of the group anagram games. The probability density distributions are shown for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that models M1 and M2 generate similar predictions, in agreement with the experiment data, as M1 is learned solely from $k = 2$ experimental data. We make this comparison more precise using KL-divergence in Figure 3.23.

Temporal comparisons of distributions between models and experiments for individual variables. Appendix B.1.8 shows the temporal analysis by minute of distributions for models M1 and M2 and experiments for $k = 2$. Each plot contains data over a time window of one minute. For $k = 2$ experiments, Figure B.17 shows temporal analysis for the number of Replies Received at the end of each minute. Figure B.18 shows temporal analysis for the number of Replies Sent at the end of each minute. Figure B.19 shows temporal analysis for the number of Requests Received at the end of each minute. Figure B.20 shows temporal analysis for the number of Requests Sent at the end of each minute. Figure B.21 shows temporal analysis for the number of Words Formed at the end of each minute.

Collections of plots for each of $k = 4$, 6, and 8 are analogously provided in Appendix B.1.8. As expected, M1 and M2 perform equally well for $k = 2$, as M1 is learned from $k = 2$ experimental data. For $k > 2$, M2 performs better. We make this comparison more precise using KL-divergence below.

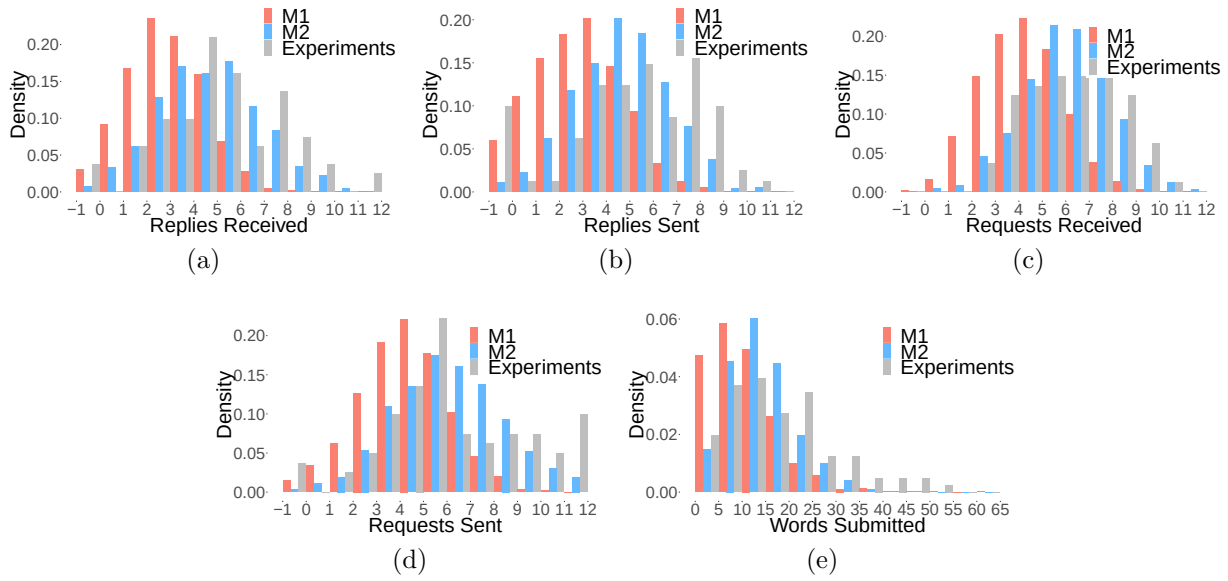


Figure 3.22: ABM M1 and M2 predictions of the $k = 4$ experiments, and experimental data, over all five minutes of the anagram games. The probability density distributions are shown for (a) distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that model M2 predictions are generally in better agreement with the experiment data than are M1 predictions. We make this comparison more precise using KL-divergence in Figure 3.24.

Comparisons of KL divergence distributions between models and experiments for individual variables at the end of the anagram game. Figures 3.23 and 3.24 in this section, and Figures B.37 and B.38 in Appendix B.1.9 show KL divergence values for comparing distributions of model outputs with corresponding distributions of experimental data, for the group anagram game. The figures are for, respectively, $k = 2$, $k = 4$, $k = 6$, and $k = 8$ experiments. The models are M1 (red) and M2 (blue). These four figures show clear and interesting behavior. Model M1 agrees better with experiments than does Model M2 for $k = 2$, since Model M1 was specifically developed with $k = 2$ data. However, for larger k ($4 \leq k \leq 8$), Model M2 does better than M1. This is because Model M2 was developed using data across all of these k values. Hence, to obtain a wider range in input space for simulations, our Model M2 does slightly worse for a particular k ($k = 2$).

Temporal comparisons of KL divergence distributions between models and experiments for individual player actions. This section shows the temporal KL-divergence

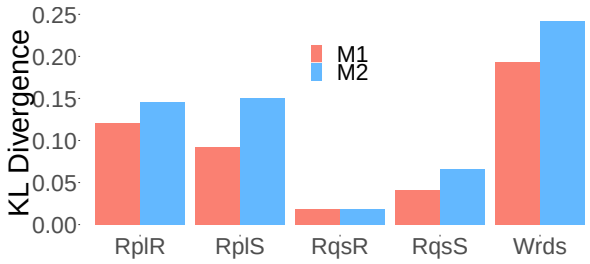


Figure 3.23: The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 2$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M1 and M2 generate similar predictions to the experimental data.

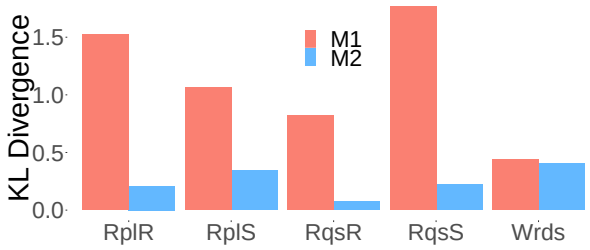
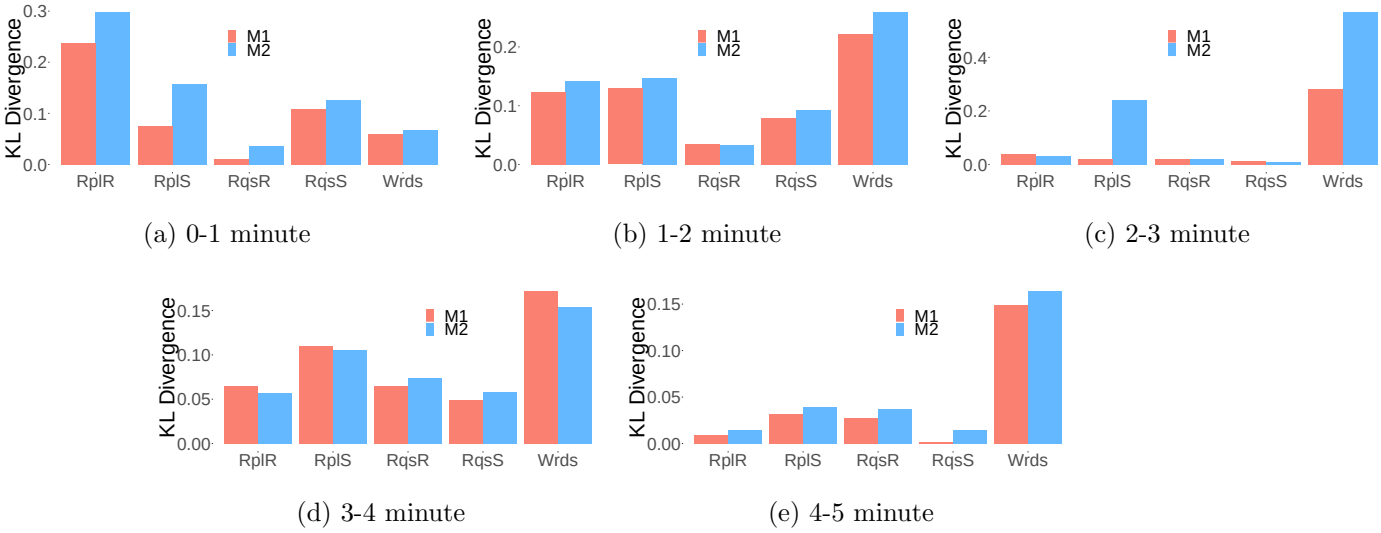


Figure 3.24: The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 4$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M2 gives much better performance than M1 predicting the time to generate an action for an agent. M2 gives much better performance, as expected, as it explicitly accounts for agent degree.

values for the model M1 and M2 predictions across the five parameters of x. Lower values are better. Figure 3.25 shows $k = 2$ experiments, and Figure 3.26 shows $k = 4$ experiments. In appendix B.1.10 Figure B.39 shows $k = 6$ experiments, and Figure B.40 shows $k = 8$ experiments. Each plot contains data over a 1-minute time window, as in previous analyses. It is clear from visual inspection that model M2 predictions are in better agreement with the experiment data than are M1 predictions for $k > 2$. As noted above, however, Model M1 does slightly better for $k = 2$. That is, the comparisons between models M1 and M2, for temporal variations in 1-minute time intervals over the 5-minute group anagram game, are similar to those comparisons when combining all data into one analysis over the entire five-minute game.

Comparisons of KL divergence distributions between models and experiments for combining all variables. Each plot in Figure 3.27 shows the distribution of KL divergence for comparing distributions of model output with corresponding distributions of



(a) 0-1 minute (b) 1-2 minute (c) 2-3 minute

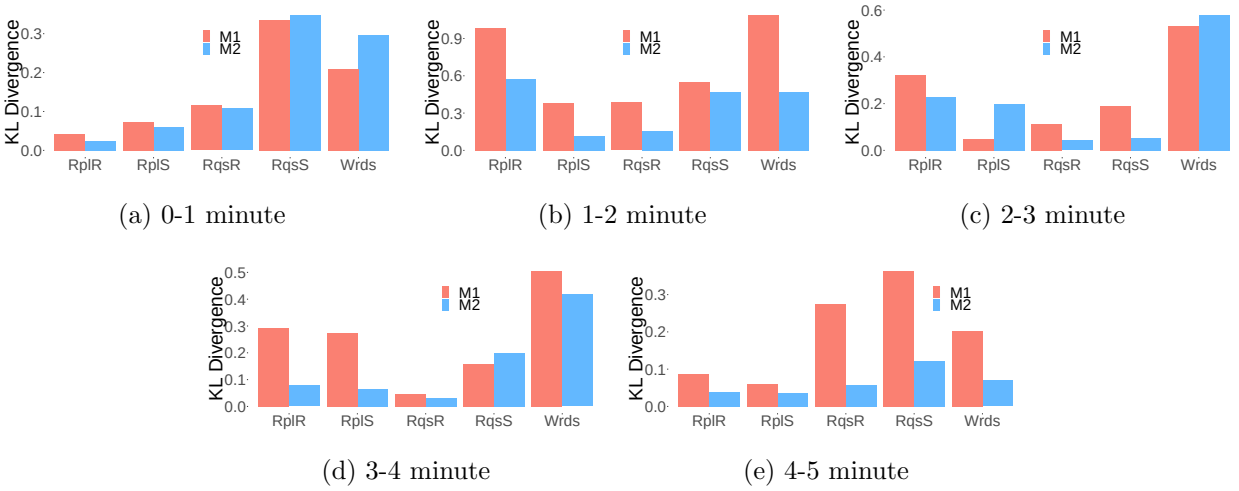
(d) 3-4 minute (e) 4-5 minute

Figure 3.25: KL-divergence values for the Models M1 and M2 predictions of the $k = 2$ experiments across the five parameters of x: lower values are better. Each plot contains data over a 1-minute time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute intervals, of the 5-minute anagram game. This figure shows that M1 and M2 generate similar predictions to the experimental data, as M1 is learned from $k = 2$ experimental data and M2 is developed from $2 \leq k \leq 8$ data.

experimental data for the anagram game. There are 30 values in each distribution, with five values for variable x at the end of the game over the 5-minute game duration, and 25 values for each of the five variables x over five intervals of one-minute increment each. These four plots, one for each value of k, summarize the previous trends: Model M1 is in slightly better agreement with $k = 2$ data, but Model M2 is better for $k > 2$.

Summary of M1 and M2 model comparisons. In addition to Figure 3.27 discussed immediately above, Figure 3.28 compares Model M1 and Model M2 for each of the five actions in x, accumulated through the 5-minute group anagram game. Model M2 does not perform quite as well as Model M1 for the $k = 2$ data, but does better than M1 for $k = 4, 6, 8$. Thus we sacrifice some quality for $k = 2$ and get in return capabilities over a range of k. Hence, Model M2 is of greater value, since it covers a broader range of inputs for simulations.

Figure 3.29 show how well the models M0, M1, and M2 work in time. We plot the KLD values from the comparison of models M0 (green box), M1 (red box), M2 (blue box) predictions versus the experimental data. On the x axis we show the group anagram game by minute of the five minute game (i.e. [0-1), [1-2), [2-3), [3-4), [4-5)). Each box, by type of model, contains 100 values of KLD corresponding to $k = 2$, 4, 6, and 8 and the five x variables at the end of each 1-minute time interval of the five minute game. Model M0 shows the greatest median values throughout the 5-minute game, except for minute 5, when it performs better



(a) 0-1 minute (b) 1-2 minute (c) 2-3 minute

(d) 3-4 minute (e) 4-5 minute

Figure 3.26: KL-divergence values for the Models M1 and M2 predictions of the $k = 4$ experiments across the five parameters of x: lower values are better. Each plot contains data over a 1-minute time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute intervals, of the 5-minute anagram game. M2 gives much better performance than M1 predicting the time to generate an action for an agent after the minute one. M2 gives much better performance, as expected, as it explicitly accounts for agent degree for $2 \leq k \leq 8$.

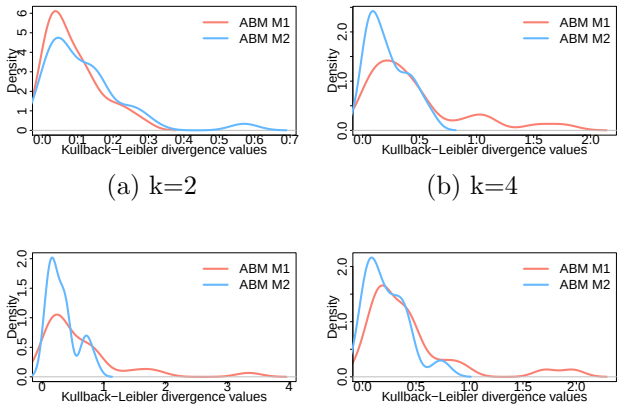
than M1. In comparing models M1 and M2, the Model M1 median at minute [0,1) is lower, while the Model M2 median is lower for the minutes two through five.

In Appendix B.1.11, Figure B.41 shows the boxplots grouped by type of $k = 2, 4, 6, 8$, where each box contains five values of KLD corresponding to the five x variables at the end of each minute. The plot show that our models show highest median values on the first two minutes of the game.

3.7 Model Evaluation

This section contains evaluations of Model M2 from Section 3.6. Our goal is to understand the conditions for which our estimated model transition probabilities π_{ij} are sufficiently accurate.

To evaluate the goodness of fitting for the proposed hierarchical model, we compare the estimated (model) transition probability matrix $\hat{\Pi} = (\hat{\pi}_{ij})$ with the empirical (data) transition probability matrix $\tilde{\Pi} = (\tilde{\pi}_{ij})$ under different settings of covariates (the z vector of Table 3.9). Here, the empirical transition probability matrix $\tilde{\Pi}$ is obtained under the settings by grouping the value of each covariate into three levels, as described in Table 3.10, to obtain comparable numbers of samples across bins.



(a) k=2 (b) k=4

(c) k=6 (d) k=8

Figure 3.27: Each plot shows the distribution of KL divergence for comparing distributions of model output with corresponding distributions of experimental data for the group anagram game. There are 30 KL-divergence values represented by each distribution in each plot, with five values for variable x at the end of the game, and 25 values for each of the five variables x over five intervals of one minute increment each. The plots correspond to different k values: (a) $k = 2$; (b) $k = 4$; (c) $k = 6$; and (d) $k = 8$. The y-axis range in (a) is different because of the high concentration of the x-axis KLD values. A model improves as the density is greater at lesser KL-divergence values. Hence, Model M1 is better for $k = 2$, and Model M2 is better for $k > 2$, consistent with the data used to build these models.

For each setting, there is a level combination of the four covariates. We compute a counting matrix $\mathcal{N} = (n_{ij})$, where n_{ij} is the number of data instances observed for the transition from action i to next action j across all players in group anagram games. (Here, actions i and j refer to actions a_i, $a_j \in A$ in Table 3.5.) We then calculate the empirical probability $\hat{\pi}_{ij} = \frac{n_{ij}}{\sum_j n_{ij}}$. There are 324 settings in total from the grouping of variables in Table 3.10 (three settings of each of four variables, and four k values), and 279 of them have valid empirical transition probability matrices. For the estimated transition probability matrix $\hat{\Pi} = (\hat{\pi}_{ij})$, the value of $\hat{\pi}_{ij}$ is estimated by the proposed model under each setting of covariates, where the averaged value at each level of the covariate is used in the estimated model.

The squared Root of Mean Squared Errors (RMSE) is used to quantify the difference between $\hat{\Pi} = (\hat{\pi}_{ij})$ and $\tilde{\Pi} = (\tilde{\pi}_{ij})$. RMSE is calculate as follows:

$$RMSE = \sqrt{\frac{1}{4|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{j=1}^{4} (\hat{\pi}_{ij} - \tilde{\pi}_{ij})^2} \tag{3.8}$$

where $\mathcal{I} = \{i : \min_j n_{ij} > 0\}$ is the index set of the rows where the empirical probability can be obtained.

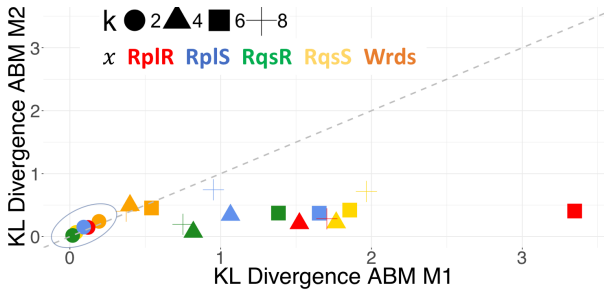


Figure 3.28: A scatter plot of KL-divergence for M1 (x-axis) and M2 (y-axis) for four k values and five x variables. For $k > 2$, M2 performs better than M1, as M2 incorporates experimental data with $2 \leq k \leq 8$. Interestingly, M1 and M2 perform equally well (highlighted) for $k = 2$ as M1 is learned from $k = 2$ experimental data (M1 is slightly better). These are data over the total 300 seconds anagram game.

Table 3.10: Three bins and ranges of values for the z variables from Section 3.6.5. These bins are created for each of the four values of $k \in \{2, 4, 6, 8\}$.

Level	Buffer (z_B)	Letter (z_L)	Word (z_W)	Consec. (z_C)
1	0	0-3	0-1	0-3
2	1	4-6	2-8	4-11
3	$\geq$2	$\geq$7	$\geq$9	$\geq$12

To illustrate this process, Figure 3.30 and Figure 3.31 contain results from good model fits. Each figure shows empirical and estimated transition probabilities π_{ij} where "initial" in the figures means action i (or a_i) and "final" in the figures means action j (or a_j). The values of the variables for the settings are provided above the matrix. For example, in Figure 3.30, the empirical probability in going from action 1 (a_1) to action 3 (a_3) is 0.0129.

Figure 3.32 and Figure 3.33 show analogous data for settings where the model fit is poor. These figures also provide an extra column of numbers of counts for each row of data. For example, in Figure 3.32, there are sufficient data for the transitions from a_1 and a_4, but not for a_2 and a_3, where the numbers of samples are 2 and 1, respectively. With data such as these, we perform the analysis below.

Under each setting of covariates from Table 3.10, we define the *Min.Count* as $n_{min} = \min_{i,j}\{n_{ij} : n_{ij} > 0\}$. That is, n_{min} is the smallest nonzero counts among transitions from state i to any state j ($1 \leq j \leq 4$) within one setting from from Table 3.10. Thus, n_{min} is determined as the minimum value of the number of observations among each of the four states i. The motivation for using n_{min} is that it is known that when it is small, the empirical probability $\tilde{\pi}_{ij}$ is not accurate. We also consider the total number of observations for a setting. Figure 3.34 and Figure 3.35 show RMSE as a function of total count and min.count, respectively. It is apparent that n_{min} does a better job in discriminating between small and

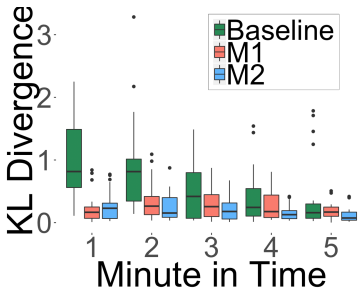


Figure 3.29: KLD values from the comparison of models M0 (green box), M1 (red box), M2 (blue box), versus the experimental data. On the x axis we show the anagram game by the minute of the five minute game (i.e. [0-1),[1-2),[2-3),[3-4),[4-5)). Each box, by type of model, contains 100 values of KLD corresponding to $k = 2$, 4, 6, and 8, and the five x variables at the end of each minute. Model M0 shows the highest median values throughout the 5-minute game, except for minute 5 when it performs better than M1. The Model M1 median is lower at minute [0,1), while the Model M2 median is lower for the minutes two through five. Figure B.41 in Appendix B.1.11 shows the boxplots grouped by type of $k = 2, 4, 6, 8$ and shows the highest median values on the first two minutes of the game.

obs	k	buffer	letter	word	constant
66	2	3	2	1	3

Empirical Probability	Final 1	Final 2	Final 3	Final 4
Initial 1	0.9381	0.0180	0.0129	0.0309
Initial 2	NA	NA	NA	NA
Initial 3	NA	NA	NA	NA
Initial 4	NA	NA	NA	NA
Estimated Probability	Final 1	Final 2	Final 3	Final 4
Initial 1	0.9393	0.0200	0.0151	0.0256
Initial 2	0.1237	0.8763	0.0000	0.0000
Initial 3	0.9707	0.0265	0.0028	0.0000
Initial 4	0.9575	0.0113	0.0000	0.0311

Figure 3.30: This shows that the model fits good for this category. The transitions in this category only contains intial status 1. Comparing empirical probabilities with probabilities estimated from model, they are very close to each other.

large RSME.

Figure 3.36 shows the scatter plot between the RMSE and n_{min} for the 279 settings for which there are sufficient data, where the plot is in log10-log10 scale. From the figure, the proposed method generally provides an accurate estimation of probability transition matrix in most of settings. Clearly, the value of RMSE decreases as the Min.Count n_{min} increases. When $n_{min} \geq 100$, the value of RMSE is smaller than 0.069, showing a very good model fitting.

obs	k	buffer	letter	word	constant
25	2	1	3	3	1

Empirical Probability	Final 1	Final 2	Final 3	Final 4
Initial 1	0.8807	0.0000	0.0034	0.1158
Initial 2	0.9500	0.0000	0.0000	0.0500
Initial 3	1.0000	0.0000	0.0000	0.0000
Initial 4	0.9129	0.0000	0.0000	0.0871
Estimated Probability	Final 1	Final 2	Final 3	Final 4
Initial 1	0.8832	0.0038	0.0040	0.1089
Initial 2	0.9344	0.0565	0.0000	0.0091
Initial 3	0.9909	0.0091	0.0000	0.0000
Initial 4	0.9352	0.0034	0.0000	0.0613

Figure 3.31: This shows that the model fits good for this category. The transitions in this category contains all intial status. Comparing empirical probabilities with probabilities estimated from model, they are very close to each other.

obs	k	buffer	letter	word	constant
52	2	3	3	1	2

Empirical Probability	Final 1	Final 2	Final 3	Final 4	Counts
Initial 1	0.8839	0.0268	0.0000	0.0893	112
Initial 2	0.0000	1.0000	0.0000	0.0000	2
Initial 3	1.0000	0.0000	0.0000	0.0000	1
Initial 4	0.7959	0.0204	0.0000	0.1837	49
Estimated Probability	Final 1	Final 2	Final 3	Final 4	
Initial 1	0.8837	0.0052	0.0038	0.1073	
Initial 2	0.8112	0.1888	0.0000	0.0000	
Initial 3	0.9882	0.0118	0.0000	0.0000	
Initial 4	0.9332	0.0039	0.0000	0.0629	

Figure 3.32: This shows that the model fits bad for this category. However, the model fits good for initial 1 and initial 4, which have 112 counts and 49 counts. While the model fits bad for initial 2 and initial 3. Thus, the count for each initial status determines the probabilities fitting for that status.

When n_{min} is small, the RMSE is relatively high. One explanation is that the empirical probabilities cannot be calculated accurately when n_{min} is small.

obs	k	buffer	letter	word	constant
206	6	2	2	3	2

Empirical Probability	Final 1	Final 2	Final 3	Final 4	Counts
Initial 1	0.3333	0.6667	0.0000	0.0000	3
Initial 2	NA	NA	NA	NA	0
Initial 3	NA	NA	NA	NA	0
Initial 4	NA	NA	NA	NA	0
Estimated Probability	Final 1	Final 2	Final 3	Final 4	
Initial 1	0.8976	0.0193	0.0284	0.0547	
Initial 2	0.9057	0.0943	0.0000	0.0000	
Initial 3	0.9934	0.0066	0.0000	0.0000	
Initial 4	0.9320	0.0056	0.0045	0.0580	

Figure 3.33: This shows that the model fits bad for this category. The transitions in this category only contains intial status 1, which has only 3 counts.

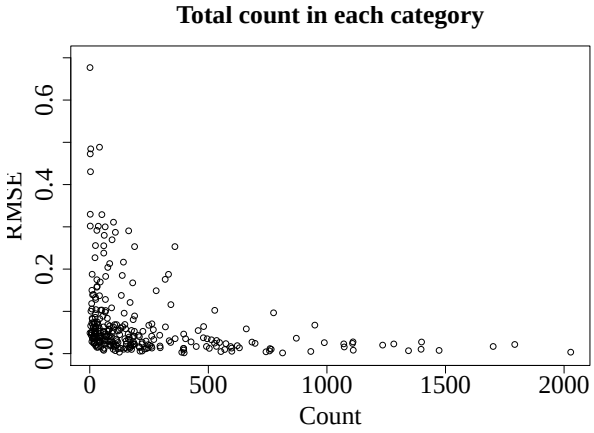


Figure 3.34: Scatter plot of RMSE against total count. The x-axis is the total count of a category, the y-axis is the RMSE for that category. The plot shows that a category with less count tends to have larger RMSE.

3.8 Abductive Loop Analyses and Results

3.8.1 Overview

In this section, we present the results of iterative abductive analyses, described in Figure 3.3. First, we "unroll" the abductive loop to illustrate several levels of abduction and different paths that can be taken, depending on results generated up to that point. Then, we present two ALs. At the end of loop-2, we describe how ABM M2 can be used in further loops. We note that the experiments (Section 3.5) and modeling (Section 3.6) are major components of the abductive looping process, and were separated out to make this section more streamlined.

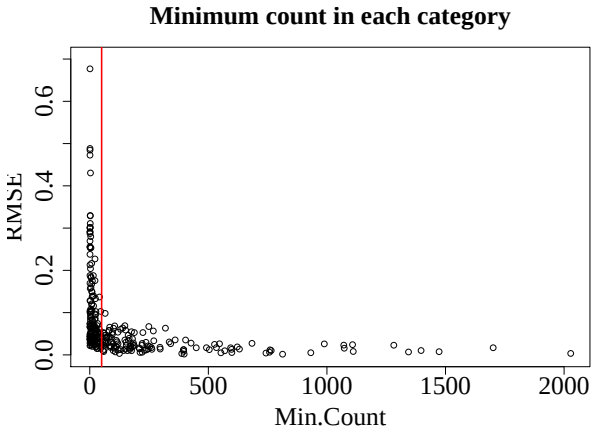


Figure 3.35: Scatter plot of RMSE against min.count. The x-axis is the minimum count of a category, the y-axis is the RMSE for that category. The plot shows that a category with less min.count tends to have larger RMSE. Furthermore, it shows that the minimum count has a sharp delineation between small and large RMSE.

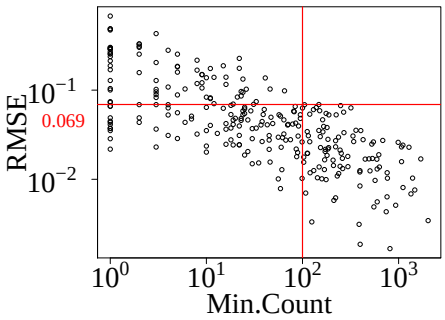


Figure 3.36: Scatter plot of RMSE against Min.Count in different settings of covariates in Table 3.10. See Equation (3.8) for RMSE and text for Min.Count.

3.8.2 Abductive Iterations with Hypotheses

Figure 3.37 provides a tree structure representation of several *candidate* abductive loops. Each node is a hypothesis, which is provided in Table 3.11. Each edge is labeled by outcome of the experiments that correspond to the hypothesis at the parent node. An edge and the child node of the edge represent one abductive iteration. The hypotheses are candidates because they depend on the data generated [112, 246].

We now overview the two abductive iterations detailed in subsequent sections. At the start, DIFI experiments are performed with and without the group anagram game. If we look at the "With Group Anagram Priming" edge, we form hypothesis H_{11}. Since it was not clear that CI was formed, we follow the "No CI detected" path to arrive at the node hypothesis H_{22}. Since we do obtain a CI signal from these experiments, we follow edge "CI detected"

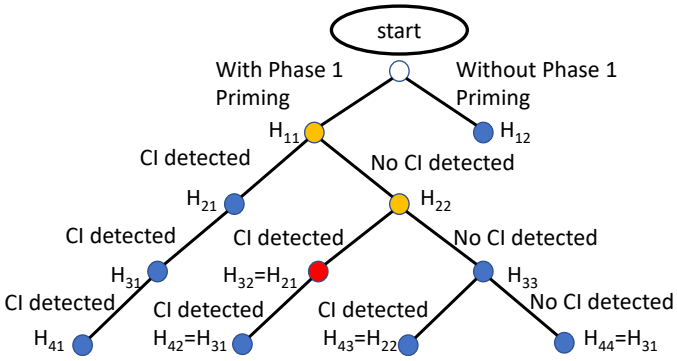


Figure 3.37: Abductive tree representing candidate abductive loops with dependencies. Hypotheses are nodes, and are provided in Table 3.11; edges are outcomes of ALs. The orange colored nodes correspond to abductive iterations presented herein. The red node is a candidate next loop. This tree is not unique; different analysts can devise different trees.

to hypothesis H_{32}. Details are provided below, and we note that modeling results guide decisions about what experiments to perform in the next abductive iteration, illustrating the value of modeling. This is one reason why our abductive loops promote modeling to a central role. We also note that a hypothesis can appear at multiple nodes within the abductive tree. Finally, a node need not have two children; e.g., fewer or more children are possible.

3.8.3 Abductive Loop 1 (AL-1)

We execute the steps of the abductive loop in Figure 3.3.

Experiments. A set of 18 experiments with a total of 87 players was completed where $k = 2$.

Hypothesis/Theory. Hypothesis H1 (H_{11}): *In the team-based anagram game, the sense of CI formed is driven more by the number of words a player forms than the number of interactions of a player (requests and replies).* Social Exchange Theory [120] focuses on the individual and suggests that the number of words resonates more in forming CI because they are directly related to reward in the game. Theory of Social Interactions [26] indicates that interactions are important for forming an interdependent organization. Reciprocity Theory suggests that v_i will respond to v_j's requests because v_i wants v_j to respond to hers, so that interactions are important.

Models. There are two types of models constructed. One is the models of the group anagram game. The other is a regression model to predict DIFI2 score as a function of outputs from the group anagram games (e.g., number of requests sent, n_{RqsS}, number of replies received n_{RplR}).

The group anagram game models M0 and M1 of Sections 3.6.4 and 3.6.5 were constructed

Table 3.11: Candidate hypotheses to evaluate in abductive iterations. Not all of the hypotheses are evaluated herein. The goal of these hypotheses, coupled with Figure 3.37, is to illustrate that there are many possible hypotheses that can be formulated, and it is up to analysts to decide which ones to pursue. An analyst will be guided by the results of completed iterative abductive analyses.

Hypothesis Number	**Description**
H_{11}	In the team-based anagram game, the sense of collective identity formed is driven more by the number of words a player forms than the number interactions of a player (requests and replies).
H_{12}	Playing the group anagram game will produce greater individual DIFI scores than not playing the group anagram game.
$H_{21} = H_{32}$	As the number and quality of letters assigned to a person decreases (i.e., as the letters assigned to a player occur less frequently in common words), collective identity of the player will increase.
$H_{22} = H_{43}$	(a) As the number of neighbors of a player increases in the anagram game, the level of CI of the player increases because there are more interactions. (b) However, beyond four neighbors (equivalently, for more than 12 neighbor letters) there is no benefit to a player, in terms of increased CI, of additional neighbors.
$H_{31} = H_{42} = H_{44}$	Playing the game with players face to face will produce greater individual DIFI scores (by enabling the players to communicate and pick up on visual and verbal cues).
H_{33}	Lesser payouts in the group anagram game means that players do not have enough incentive to engage their neighbors.
H_{41}	Having the group anagram game score of another team displayed during the game will increase CI because it will create a stronger in-group/out-group paradigm.

from the time histories of actions of players for experiments with $k = 2$. The results relevant to this iteration are provided in Figures 3.16 and 3.17. ABM M1 is much better at capturing the dynamics in the experiments than is baseline model M0.

From data on the actions of A from the games, and the measured DIFI2 scores after the group anagram games, a regression was performed to predict DIFI2 score as a function of number of actions of each kind. The DIFI2 score is given as

$$\begin{aligned}\widehat{DIFI2} =& c_1 + c_{RplR}\ n_{RplR} + c_{RplS}\ n_{RplS} + \\ & c_{RqsR}\ n_{RqsR} + c_{RqsS}\ n_{RqsS} + c_{Wrds}\ n_{Wrds}\end{aligned} \tag{3.9}$$

where Table 3.12 provides the equation coefficients and the defintions of variables.

Best Explanation. Results of a linear regression in Table 3.13 indicate that hypothesis H1 is

Table 3.12: Constants in the regression of Equation (3.9) to predict DIFI2 score from outputs of the team anagram game.

Coefficient	**Value**
Intercept c_1	102.7
c_{RplR} on number of replies received n_{RplR}	14.95
c_{RplS} on number of replies sent n_{RplS}	-12.99
c_{RqsR} on number of requests received n_{RqsR}	6.406
c_{RqsS} on number of requests sent n_{RqsS}	-16.43
c_{Wrds} on number of words formed n_{Wrds}	-0.2134

Table 3.13: Results of linear regression of variables in x against dependent variable DIFI2 score, indicating that interactions are more significant than number of words formed in producing CI.

Var.	Interc.	RplR	RplS	RqsR	RqsS	Wrds
est.	103.	15.0	-13.0	6.41	-16.4	-0.213
p-val.	0.001	**0.019**	**0.011**	0.332	**0.011**	0.735

falsified because Wrds, the number of words formed, is not significant, while RplR, RplS and RqsS (i.e., interactions) are significant. Thus, Social Exchange Theory can be eliminated as a theory of CI formation in this experiment. It is somewhat surprising that Wrds is not significant because it is the variable that is most closely associated with the reward (earnings). In the social sciences, and in many domains, eliminating candidate theories is a valuable result (that is, an analysis does not always have to identify the best theory). Thus, at this point, the best explanation is Reciprocity Theory and Theory of Social Interactions.

What is Next? Figure 3.20 indicates that the model predicts behavior that is invariant with respect to the degrees of players [and hence the number of letters that neighbors possess] (plots of other variables of x are similar). We want to determine whether there is an effect of k, and hence the next experiments are specified as using increasing k (i.e., $k > 2$). Thus, the ABM M1 (driven by the data) is guiding what to do next. While Social Exchange Theory was eliminated in this loop, Reciprocity Theory and Theory of Social Interactions are carried forward into the next loop(s), where they may be supported or refuted.

3.8.4 Abductive Loop 2 (AL-2)

We execute the steps of the abductive loop in Figure 3.3.

Experiments. A set of 16 experiments with a total of 137 players was completed where $k = 4$, 6, and 8 in turn.

Data Analysis. We continued the same types of analyses described in AL-1, but with the

added dimension of k.

Hypothesis/Theory. Hypothesis H2 ($= H_{22}$): *(a) As the number of neighbors of a player increases in the anagram game, the level of CI of the player increases because there are more interactions. (b) However, beyond four neighbors (equivalently, for more than 12 neighbor letters) there is no benefit to a player, in terms of increased CI, of additional neighbors.* The Theory of Social Interactions states that interactions with more neighbors creates more interdependence. Theory of Cognitive Load [236] suggests that cognitive load might be too great at some point, resulting in a player being unable to take advantage of more input.

Model. Model M2 of Section 3.6.6 was constructed from the time histories of actions of players, from the combined data from *both* iterations. Model results relevant to this iteration are provided in Figures 3.21 through 3.29. ABM M2 captures trends in degree k much more effectively than ABM M1, for all parameters of x.

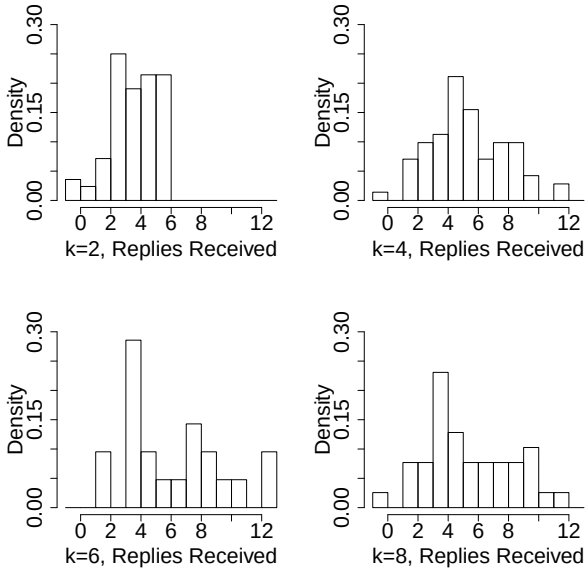
Best Explanation. Figure 3.38 provides results that address hypotheses H2(a) and H2(b). Figure 3.38a shows the frequency distributions for replies received, for the four values of k. Note the large change in distributions in going from $k = 2$ to $k = 4$, but relatively minor changes for further increases in k. Thus, the saturation in the distributions (and others are similar), supports hypothesis H2(b): the number of neighbors increases, but the number of interactions does not, for $k > 4$. This is consistent with Cognitive Load Theory.

Figure 3.38b shows that as k increases from 2 to 8, the probability density of DIFI2 shifts demonstrably to increasing DIFI2. That is, greater numbers of neighbors produce more CI, as measured by the DIFI2 score. This does not wholly support H2(a). While increasing k does correlate with increasing DIFI2 score, it is not because of the number of interactions, which does not increase appreciably for $k > 4$. These data falsify H2(b): there is additional benefit, in terms of increased DIFI2 score, with increasing number of neighbors. The applicability of the theory of social interactions is not clear, but the data suggest that it is the number of different people with whom one interacts that is important, rather than the total number of interactions. More experiments are needed.

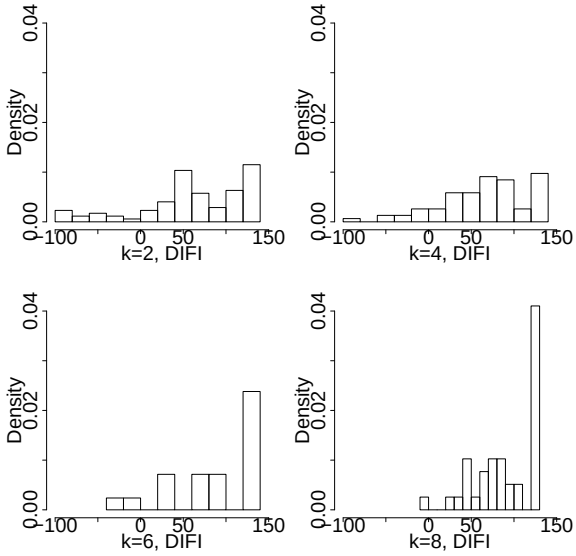
What is Next? At this point we halt the iterative abduction process for this paper. In a next iteration, we could (i) try to isolate the effects of number of interactions versus the number of neighbors in different experiments, or (ii) study the effects of different degrees of players and different numbers and qualities of letters initially assigned to players within the same experiment. We could also perform a deductive (confirmatory) analysis by making specific quantitative predictions for experiments using ABM M2 as part of AL-2, and running corresponding experiments in AL-3.

3.8.5 Abductive Loops: Role of Analyst and Bigger Picture

Two ALs have been demonstrated. Many additional loops are possible, as illustrated in Figure 3.37, which depicts several hypotheses, including the two addressed above (in orange).



(a)



(b)

Figure 3.38: Statistical analysis correlation results of the anagram game parameters and DIFI2 score. (a) Probability density of replies received change markedly from $k = 2$ to $k = 4$, but relatively little for further increasing k. (b) Probability density of DIFI2 score moves dramatically to larger DIFI2 score with increasing k. Each of these results is novel. All the more novel is the combination of the two: while game measurables saturate (other data besides replies received), the DIFI2 score does not.

These additional loops would require more experiments. Figure 3.37 and Table 3.11 make clear the important role of an analyst in this process, as she guides the direction of the looping. So, while a plan such as that in Figure 3.37 may be useful, the actual tree structure will evolve with analyst decisions as the looping progresses and as data are generated, because hypotheses are based on newly-generated data in abduction.

3.9 Limitations and Additional Work

More experiments, particularly at greater k would be useful. Also, we would like to alter the number of letters and to control the "quality" of letters that are assigned to players (e.g, e is a more desirable letter than q) in additional experiments, so that we can run experiments that will more stringently test the models. We would like to study more network structures (i.e., connectivity among players in a game), such as a clique structure. We attempted to correlate player behavior with survey information in the online experimental platform. For example, we tried to correlate DIFI score with player age, gender, nationality, ethnic group, and education level. We did not get a strong signal in any of these correlation studies. This would be a huge step forward if such correlations exist and can be found because it would relate macro-player features with player behavior. With respect to modeling, we can improve the models for the player actions (e.g., the process of forming words); this work is in progress. We can improve the modeling in translating results in the group anagram game to the DIFI scores, to better understand the connection between priming and CI formation.

3.10 Summary

We formalize an abductive loop, implement it computationally, and exercise it in an experimental setting (the group anagram game) designed to induce CI, as operationalized by Swann's DIFI score. However, our abductive looping process is not tied to CI. As part of the abductive iterations, we provide novel experimental insights into CI and build and evaluate three ABMs. This work establishes the potential of iterative abductive looping for the (computational) social sciences.

Chapter 4

Mechanistic and Data-Driven ABM's in Group Anagrams Games

4.1 Abstract

In anagram games, players are provided with letters for forming as many words as possible over a specified time duration. Anagram games have been used in controlled experiments to study problems such as collective identity, effects of goal-setting, internal-external attributions, test anxiety, and others. The majority of work on anagram games involves individual players. Recently, work has expanded to group anagram games where players cooperate by sharing letters. In this work, we analyze experimental data from online social networked experiments of group anagram games. We develop mechanistic and data-driven models of human reasoning to predict detailed game player actions (e.g., what word to form next). With these results, we develop a composite agent-based modeling and simulation platform that incorporates the models from data analysis. We compare model predictions against experimental data, which enables us to provide explanations of human reasoning and behavior. Finally, we provide illustrative case studies using agent-based simulations to demonstrate the efficacy of models to provide insights that are beyond those from experiments alone.

4.2 Introduction

4.2.1 Background and Motivation

In one form of an **individual anagram game**, a player is provided with a set of alphabetical letters to form as many words as possible in a prescribed time duration. The performance of a player is often quantified based on the number of words formed.

In a **group anagram game** (GrAG), multiple players collaborate. Each player is given letters and forms words with her own letters, and can share letters with her neighbors to enable everyone to form more words. Figure 4.1 provides a schematic of a 3-player GrAG. Each player (v_1, v_2, and v_3) is initially provided with $n_l = 3$ letters as shown. A player may form words, and through the communication channels in gray, may request letters and reply to letter requests.

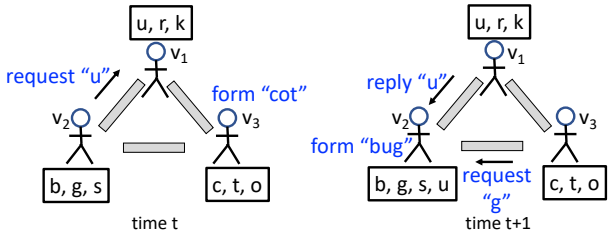


Figure 4.1: Simplified view of a networked group anagram game (GrAG), with illustrative actions among $n = 3$ players that communicate and share letters through the gray channels. Each player is initially given $n_l = 3$ letters. Letters that a player has "in-hand" to form words are shown in boxes. Player actions are shown in blue. At time t, v_2 requests a "u" from v_1 and v_3 forms the word "cot." At the next time, v_2 receives a "u" from v_1, forms the word "bug," and receives a request from v_3.

Overwhelmingly, research on anagram games considers the individual setting. It has been extensively studied (over 20 published works) for more than 60 years to analyze phenomena such as goal-setting, compensation types, internal-external attributions, and test anxiety (e.g., [52, 70]). Other names for anagram game are *word formation game* and *word construction game*.

There are several reasons to study GrAGs. A face-to-face GrAG has recently been played. In particular, [51] used them to study experimentally the formation of collective identity (CI), defined in social psychology as an individual's cognitive, moral, and emotional connection with a broader community, category, practice, or institution [200]. A second motivation is their relevance to other types of group dynamics, notably intergroup and intragroup cooperation and competition (e.g., [104]). A third motivation is that many of the phenomena listed above for the individual anagram game (e.g., goal-setting) could be studied in group settings with models of group behavior.

Overall, researches involving anagram games encompass a broad range of disciplines like sociology, economics, management science, and (social) psychology [40, 52, 70]. It is clear that using anagram games is valuable in various fields of research. With all of this experimental work on anagram games, it is surprising that very little work has been done in modeling and simulating these games. The first and only work on modeling GrAGs was recently completed [202]. We enumerate the differences between our work and [202] in Section 4.2.2 immediately below.

4.2.2 Our Work Scope and Differentiators from Previous Work

Work scope. Our work starts with data from online social network GrAGs. (The game platform and online experiments are *not* the focus in this work.) With these data: (*i*) data analytics are performed to support model development; (*ii*) different models for different player actions in the GrAG are developed; (*iii*) the models are evaluated against experimental data; and (*iv*) these models are then recast as agent-based models and executed within an agent-based modeling and simulation (ABMS) platform to produce computational results that go beyond the experiments.

Based on this work scope, all of the following are completely different in this work, compared to that in [202]: data analytics, the aspects of the game that are being modeled, the types of modeling techniques used, the models themselves, and the quantities that the models predict. We address particular differences between [202] and our work now.

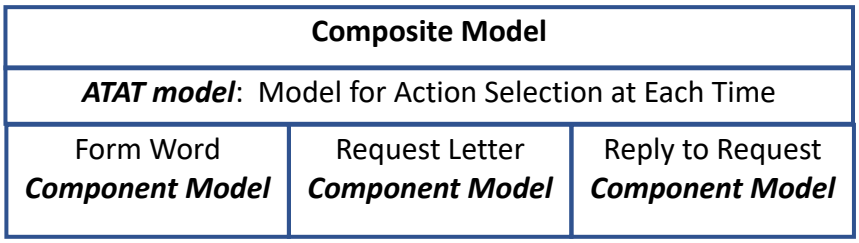
Work in Ref. [202]. Figure 4.2 serves to emphasize our models and to differentiate our work from that in [202]. The *action **type** and time (ATAT) model* of the figure is the subject of [202], which builds the model using multinomial logistic regression. In that work, the goal was to develop models to predict the *type* of action taken in time, e.g., predictions of the form: player v_i takes action type "form word" at time t. Also, if a player action is form word, and the player has letters that cannot form a word (e.g., letters q, z, and r) then that model will nevertheless form an unspecified (unrealistic) word from these letters. Moreover, the models of [202] do not consider the particular letters assigned to players in a game and hence have no heterogeneity. Consequently, all player behaviors will tend toward the same mean behavior in agent-based simulations (ABSs).

Our work. In contrast, our work focuses on the three *component models* of Figure 4.2. Different models are developed for the actions "form word," "request letter," and "reply to (letter) request." Our models account for network structure, letter assignments and letters in-hand (i.e., letters that a player has to form words), and particular player parameter assignments (detailed below)—all of which can vary among players—so results will remain distinct across agents. That is, we capture heterogeneity in several ways.

Per Figure 4.2, our ABMS framework uses a **composite model**: a combination of the ATAT model (to determine what action types players take in time) and the three component models developed herein (to predict the specifics of each action). The composite model is our agent-based model (ABM). This ABMS system simulates GrAG scenarios beyond those of the experiments.

4.2.3 Novelty of Our Work

First, our work is an exemplar of a detailed procedure for combining mechanistic and data-driven models to form single models of human *reasoning* and *decision-making* that output



Unifying theme across three models for the actions *form word, request letter, reply to request*:

(player action specifics) = (human reasoning) x (aptitude) x (data objects)

Figure 4.2: Structure of the **composite (agent-based) model**. At each time in a group anagram game, a player takes one of four actions, consistent with the online game: "form word," "request letter," "reply to letter request," or "think." The selection of each action is determined by a multinomial logistic regression model from [202], which we call the **action *type* and time** (ATAT) model. Each of the first three actions requires a **component model** (and software module) that simulates human reasoning and outputs the *specifics* of an action. These are expanded on in Figure 4.3 below. These *component models* in this figure are the focus of this work. The algorithms for these three actions are in Figure 4.7, Figure 4.9, and in [45], Figure 16. "Thinking" is an idling action, no model is needed. The common theme across these models is given in blue. Aptitude is described in Section 4.5.

human *actions* in a game. *Mechanistic models*, for our purposes, have the following characteristics: (*i*) the models are based on first principles and are not tied to any particular domain; and (*ii*) the models are specified, implemented, and executed without any experimental data. To augment mechanistic models by accounting for variability in player behaviors, *data-driven models* are constructed from analyses of experimental data. Second, because the mechanistic models capture player behavior, these models *explain* behaviors, as described in our contributions below. Third, our mechanistic models are novel: Levenshtein Distance (LD) [151] (see Section 4.5.1) and a greedy optimization procedure describe human decision-making and have not been used in anagrams contexts (we could not find LD used in any modeling of human behavior, as we do here). Fourth, with these models, we develop an ABMS platform to model the detailed actions of players in GrAGs beyond the experiment conditions.

As called for in the social sciences, our focus is on model construction and predictions, and explanations of human behavior [118, 264].

4.2.4 Contributions

1. A process for combining mechanistic and data-driven approaches to build models of human reasoning. We provide the details of our process in Section 4.5. See

Figure 4.3. First, mechanistic models are conjectured and evaluated by comparing their predictions to experimental data. This does three things: (*i*) enables comparisons of model predictions with experimental data, and if these comparisons are favorable (which they are), then (*ii*) the structures of the models provide *explanations* for human decision-making [31, 124], and (*iii*) the mechanistic models form the basis of the ABMs. Second, because the mechanistic models can be improved by including data from experiments, we use data-driven modeling approaches to introduce stochasticity to account for variability across human subject game players. Hence we utilize these two modeling approaches in a well-defined process.

2. Mechanistic models. We use concepts such as LD, word corpora, word proximity networks (WPNs), and a greedy optimization algorithm (all defined in Section 4.5) to develop mechanistic models for two of the three player actions (see Figure 4.3). The LD model, used for word formation, could be used within any agent that is required to form words, and the greedy optimization algorithm, used for requesting letters, could be used by agents to make a choice from among a finite set of options. That is, these models are not tied to our GrAG. But the next contribution presents their utility within the GrAG.

3. New experimental findings and explanations of player behaviors based on cognitive and economic theories. The analyses focus on data for three types of player actions: (1) form a word; (2) request a letter; and (3) reply to a letter request. A summary of some explanations follows. A word w_2 that a player forms is explained by considering (*i*) the letters that the player has in-hand (i.e., in her possession) and (*ii*) LD [151] between the most recently formed word w_1 and the next word to be formed w_2 from a candidate set of words (Section 4.5.2). This is motivated by, and consistent with, cognitive load theory [236] in that people try to reduce cognitive load during learning. Here, the closer the next word formed is to the previously formed word—as measured by LD—the lesser the cognitive load in forming a new word. For letter requests, we use the idea that player action is based on rational choice theory [27]. Our analyses (Section 4.5.3) demonstrate that the letter that a player requests from her neighbors is explained by identifying the letter that maximally increases the number of words that the player can form, when also considering the letters that the player has in-hand (greedy optimization algorithm). This behavior is consistent with rational choice theory. This is because players' earnings in games are proportional to the number of words formed, so it is rational for a player to choose a letter to maximize the size of their candidate word set. It is interesting that our explanation means that players are reasoning beyond more naive approaches, such as simply requesting some "most frequently" used letter (e.g., preferring e over z). (We have modeled this naive approach—results not shown here—and this model's results are not consistent with the data.) Finally, we also show that there are four types of behavior in replying to letter requests (Section 4.5.4).

4. Agent-based models and results. A family of ABMs are developed, yielding a composite model, where each ABM is comprised of a distinct model for each of the three actions, with user-specified parameter values for player/agent characteristics, such as the

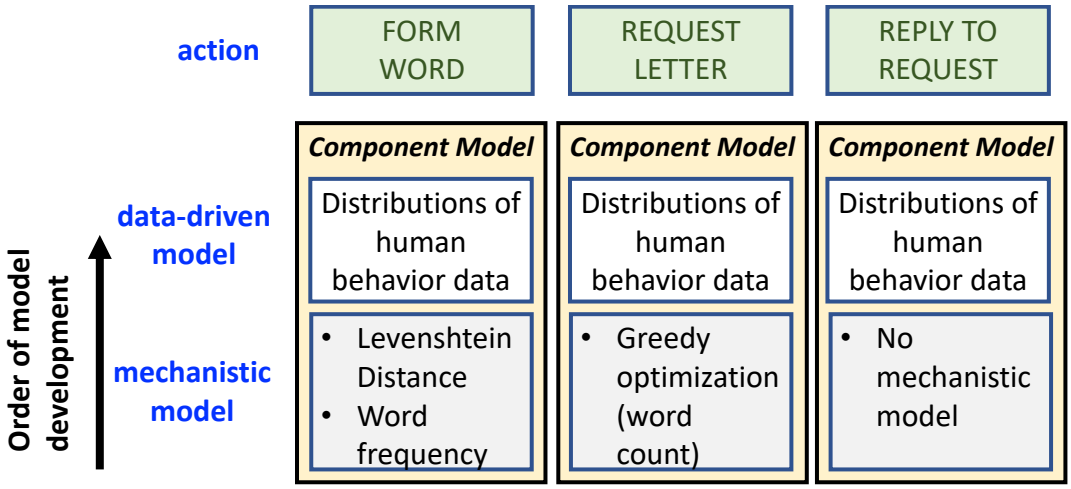


Figure 4.3: **Component models** (i.e., combined *mechanistic* and *data-driven* models) for the three player actions in the GrAG. These are models of human reasoning, which output specific player actions in the game. The particulars of the mechanistic and data-driven models are given in the respective boxes under the actions and are detailed in Section 4.5. Mechanistic models are built first, and then augmented with data-driven models. The player actions and component models map onto those in Figure 4.2.

agent's vocabulary and their aptitude, i.e., the degree to which they perform optimally. See Figure 4.2. The multi-logit regression model based on [202] is adopted to determine which action type each agent selects at each discrete time in a simulation (time granularity is seconds). The selected action type then determines the appropriate model developed herein to predict details of the action. Note that there is a fourth action, a no-operation (no-op), where the agent does nothing at particular times, which represents agent thinking and requires no model. We also provide new insights from exercising the ABMs (see Section 4.6), such as demonstrating how player performance decreases with decreasing player aptitude and the effects of heterogeneous initial letter assignments to players.

4.3 Related Work

By far, the most relevant study to our work is the modeling in [202], which is **agent-based modeling of anagram games**. To the best of our knowledge that is the only work prior to ours that models the GrAG [202]. That work was discussed in detail in relation to our work in Sections 4.2.1 and 4.2.2. We now address other topics related to our work.

Anagram experiments. Over 20 experiment works (e.g., [52, 70]) use *single player* anagram games. The only cooperative GrAG, which is *face-to-face*, is reported in [51]. The game is used to foster CI among teammates. Multiple simultaneously-playing teams can change composition (4-player teams) as players vote others into and off the team. This motivated

the *online* experiments in [202] where one team of fixed size plays the game through a web browser.

Networked experiments and modeling. There are several other online (e.g., [164]) and in-person (e.g., [51]) experiments with interacting participants that can be represented as networks, and analyses of network populations (e.g., [181, 182]), where edges represent interaction channels.

Mechanistic and data-driven modeling. Several works use AI methods and data to model behavior (e.g., tutoring and learning [278]). Also, neuroscientists are using neuroimaging to understand human decision-making; [183] discusses optimization methods, such as the one we use in the model for requesting letters.

Explanatory modeling. There are many works (e.g., [31, 124]) that describe different definitions of *explanations*, different types of explanations that models provide, and procedures for arriving at explanations. We follow ideas from [31, 124]: that the structure of mechanistic models that adequately predict human behavior can be used to explain behavior. As a counter example, [123] discusses the need for visual representations for simulation explanation when models are not well understood. However, it is precisely our goal in this paper to describe our modeling approach, our models, and our experimental justification for them. We present well-motivated, well-defined, formal models in Section 4.5 that are also used in ABM.

4.4 Online Social-Networked Group Anagram Game

We built a customized web application (web app) for an online GrAG. Players are recruited through Amazon Mechanical Turk (MTurk), are provided game instructions, participate in the GrAG through their web browsers, and are paid based on their performance. A total of 48 experiments were performed using a total of 367 players, with numbers of players per game ranging from 3 to 17. The game duration is 5 minutes. In the following, we describe the GrAG/experiment.

Figure 4.1 provides a description of the game setup and actions. A game begins with n players, v_1 through v_n. Each player has a degree d that specifies the number of connections to other players. A connection (edge) between two players denotes a communication channel where a letter ℓ can be requested and sent (sending a letter is a reply). Thus, an experiment configuration is a graph $G(V, E)$ with player set V and communication channels E. In experiments, G is a k-regular random graph ($k \equiv d$), with uniform degree $2 \leq k \leq 8$. Each player starts the game with n_ℓ initial letters, which they can use to form words or share among their neighbors, when requested. At the beginning of a game, a word corpus C^W is defined with a list of words a player can form during the game. The three major player actions in a game are now described.

Player action: forming a word. At any point during a game, a player v_i can form a word w_i. All letters in the word w_i must come from the set of letters v_i has in-hand L_i^{ih} (superscript ih). A single letter ℓ in L_i^{ih} can appear any number of times in a word. For a word submission to be accepted in the game, the word has to be in the game word corpus C^W. A player can submit a word only once; multiple players can form the same word.

Player action: requesting a letter. At any point during a game, a player v_i can request a letter ℓ_{ij}^{req} from a neighbor v_j's set of n_ℓ *initial* letters L_j^{init}. The anagram game screen shows all neighbors' *initial* letters as available for request. A letter received by v_i is put into the set L_i^{ih}.

Player action: replying with a letter. At any point during a game, a player v_i can reply with a letter ℓ_{ij}^{rep} to a neighbor v_j's request (ℓ_{ij}^{rep} must be in L_i^{init}). The anagram game screen for v_i shows all of the letters requested of v_i.

To encourage cooperation, any letter in L_i^{ih} can be used any number of times in forming words, and the letter is not lost; the letter bestows an infinite supply of use. Similarly, if v_i requests a letter ℓ from v_j, and v_j replies with it, v_j still retains a copy of the letter and can use it. Also, earnings for the team are based on the total number of words formed, and all players receive $(1/n)$ of the total earnings. Typical player earnings are \$7 to \$10 per game.

4.5 Data Analysis and Model Development

Figure 4.3 provides the roadmap for building the models for the three player actions, which is the focus of this section. Ultimately, our goal is to use these models as ABMs (see Figure 4.2) in an ABMS framework to study GrAGs well beyond those of experiments.

For each action—which is a component model of the ABM—we provide: (i) our premise for understanding player behavior and the key concepts for this premise, (ii) experimental analyses and results for these key ideas that construct and justify (i.e., give evidence for) the component model of the composite ABM, and (iii) a formal algorithm for the component model for the action in Figure 4.3. Note that the steps of algorithms that we specify below are not focused on efficient implementation, but rather on conveying the steps of the algorithms as they relate to the data analyses. First, we address preliminaries.

4.5.1 Preliminaries

We introduce two concepts used in data analysis and modeling. **Levenshtein distance** (d^L) [151], an edit distance, is prominent in our work and the work's novelty, and is motivated by work in linguistics and bioinformatics [226]. It quantifies the difference in letters of two words. In starting with one word to obtain a second word, a letter substitution counts as one, as does each of letter insertion and letter deletion. Hence, going from *had* to *hats* requires

$d^L = 2$: one to substitute t for d and one for inserting an s.

A **word proximity network** (WPN) is a clique graph $H(V_H, E_H)$ where vertices V_H are words that can be formed, according to a word corpus C^W, with the letters that a player currently has in-hand and E_H is the set of edges between pairs of words, labeled with the d^L between the two words.

Each player is assigned a **word corpus** C^W. For this we use a list of the top 5000 words from the 450 million word Corpus of Contemporary American English, the only large and balanced corpus of American English [1].

4.5.2 Player Action: Form Word

Basic premise and key concepts. We seek to identify a method that explains the process of players selecting words to form. Our premise is that given the last word w_1 that v_i has formed, the next word w_2 that v_i will form will be one with minimal d^L from w_1 because this requires a minimal number of letter manipulations (i.e., lesser cognitive load [236]). For the first word, v_i selects the most frequent word from the corpus that can be formed with its letters in-hand L_i^{ih}. We note that for each player v_i, there is a set L_i^{ih} of letters that she has in-hand and a corresponding set $W_i^{ih} \subseteq C^W$ of words that v_i can form from the entire corpus C^W of words, based on the letters in L_i^{ih}. As v_i requests and receives more letters from her neighbors, the cardinalities of L_i^{ih} and W_i^{ih} will (typically) increase. Also note that for a given word w_1 formed by v_i in a game, W_i^{ih} can be partitioned based on $d^L(w_1, w_2)$ for each $w_2 \in W_i^{ih}$ using the WPN. Let $W_i^{ih}(w_1, d^L) \subseteq W_i^{ih}$ be the set of words at d^L from w_1 that v_i can form.

Our data analysis is based on two central ideas, for each player v_i. First, we compare d^L values between two consecutive words formed (w_1 and then w_2), both the actual value $d^L_{i,act}(w_1, w_2)$ measured from experiments and the optimal (i.e., minimal) value of d^L, denoted $d^L_{min}(w_1, w^*)$, for some w^* in W_i^{ih} that is at a minimum LD from w_1. Both d^L values are based on v_i's set L_i^{ih}. (We drop the arguments when they are obvious from context.) Second, for a given set of words at some d^L from w_1, denoted $W_i^{ih}(w_1, d^L)$, we select w_2 based on the popularity of words as provided by the rank (frequency of use) from [1]. All of these parameters are either inputs (e.g., C^W), measured in experiments, or computed from experimental data. These high-level steps enable us to understand players' behavior in forming words, as described next.

Data analysis. Analysis step 1. For each player v_i in the game, we consider pairs of consecutive words formed, (w_1, w_2). From this, we compute $d^L_{i,act}(w_1, w_2)$, the actual d^L. Also from these data and from L_i^{ih} at the time w_2 was formed, we can compute d^L_{min} and the word set $W_i^{ih}(w_1, d^L_{min})$. We compute $\Delta d^L = d^L_{i,act} - d^L_{min}$. A value of zero means that the player is performing optimally according to our premise; a value > 0 means that v_i is performing suboptimally—v_i is making more letter edits (expending greater effort) than is required by

the data.

We rank the players by their average Δd^L, Δd^L_{ave}, over all pairs of words (w_1, w_2) that they form in a game. We partition the ranking of players into five equi-sized bins, P_1 through P_5, such that players in P_1 (resp., P_5) have the smallest (resp., largest) values of Δd^L_{ave}. That is, the players in P_1 perform closest to optimal. A player v_i's aptitude b_i^{wf} in forming words takes a value from P_1 through P_5. We take this player-centric approach because we want to produce agent models based on individual player and groups of players' behaviors.

Analysis step 2. For each of the five groups of players P_j $(1 \leq j \leq 5)$, we plot all data points $(x, y) = (d^L_{min}, d^L_{i,act}(w_1, w_2))$ for each person in that group, in Figure 4.4. In each plot, for each d^L_{min} on the x-axis (the *mechanistic model prediction*), there is a range of $d^L_{i,act}(w_1, w_2)$ (from the data) for all v_i in a particular 20% bin. If we break the players down into 10% bins (instead of the 20% bins), the top 30% of players perform such that the median value of $d^L_{i,act}(w_1, w_2)$ equals d^L_{min}. That is, in a median sense, these top 30% of players form words w_2 such that $d^L_{i,act}(w_1, w_2) = d^L_{min}$, and hence w_2 is formed optimally (i.e., according to the mechanistic model). Moreover, if we look at the top 80% of players, then $d^L_{min} \leq d^L_{i,act}(w_1, w_2) \leq d^L_{min} + 1$. *These data for* $|C^W| = 5000$ *substantiate our premise that players form word* w_2 *based on* d^L. Although not shown, similar results are generated for $|C^W| = 1000$, 2000, 3000, and 4000, if we take these sets as the 1000, 2000, 3000, and 4000 most frequently used words in the original corpus of 5000 words.

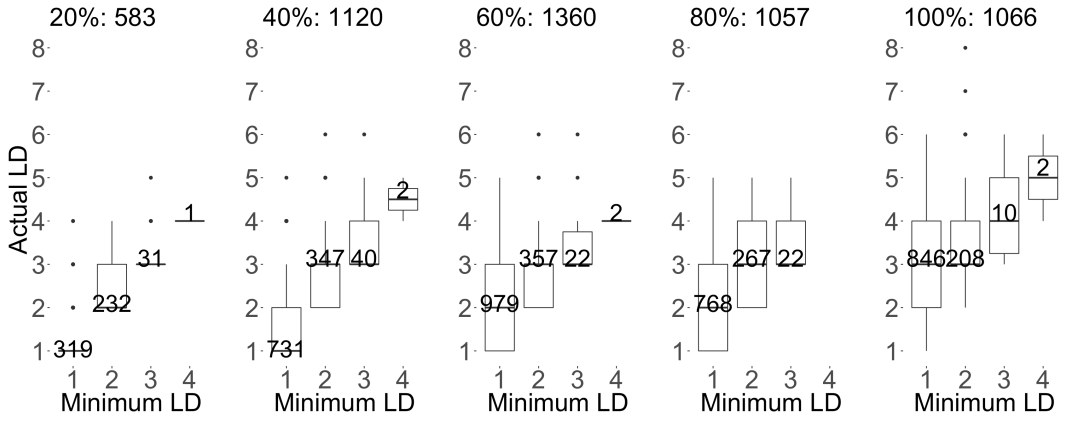


Figure 4.4: Comparison of ***mechanistic*** model predictions against data for the *form word* model. Mechanistic predictions are the values on the x-axis (d^L_{min}); data are on the y-axis ($d^L_{i,act}$). We use the $|C^W| = 5000$ word corpus. Each plot corresponds to a grouping of players by 20% bins of player performance in forming words according to d^L, and represents, in turn, P_j, $j \in \{1, 2, 3, 4, 5\}$, moving left to right. Numbers are numbers of observations in the data. If $d^L_{i,act}(w_1, w_2) = d^L_{min}$, then the experimental data correspond exactly with the mechanistic model.

Analysis step 3. For each box plot in Figure 4.4, we form a frequency distribution $\mathcal{D}^{d^L}$ as a function of the triple $(C_i^W, b_i^{wf}, d^L_{min})$. Figure 4.5 provides one such distribution. In this way,

given a C_i^W, an aptitude b_i^{wf} for forming words, and a d_{min}^L, one can sample an actual LD, $d_{i,act}^L$, in forming w_2 from w_1.

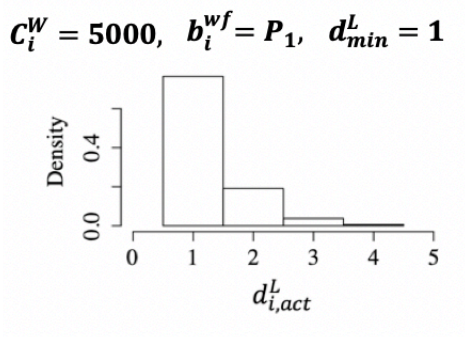


Figure 4.5: For $(C_i^W, b_i^{wf}, d_{min}^L) = (5000 \text{ words}, P_1, 1)$, the distribution $\mathcal{D}^{d^L}$ of $d_{i,act}^L$ from experiments is shown. For a given d_{min}^L computed for optimal behavior, the appropriate distribution is sampled to obtain $d_{i,act}^L$ for v_i. These distributions are formed from the data in Figure 4.4 and they are part of the ***data-driven*** model of form word.

Analysis step 4. For a given w_1 and $d_{i,act}^L$, $W_i^{ih}(w_1, d_{i,act}^L) \subseteq W_i^{ih}$ is the candidate set of words that v_i can form as w_2. The issue is how players extract a particular word from $W_i^{ih}(w_1, d_{i,act}^L)$ as w_2. Figure 4.6 provides the answer. For each v_i, we rank the words in $W_i^{ih}(w_1, d_{i,act}^L)$ in decreasing order of frequency of occurrence (which is obtained from the word corpus itself), such that the first ranked word is the most frequently used word. This plot shows the number of times the chosen word w_2 is of a particular rank. It is clear that players select w_2 based on the frequency of the word's use, e.g., the top-ranked word is selected almost 700 times from the corpus. This result also holds over different corpus sizes from 1000 to 5000 words. These data support our use of a mechanistic model of selecting the word with highest frequency of use in a word corpus from the candidate set of words.

Remark: These data analyses substantiate our claim that our models are explanatory. The data are consistent with the explanation that humans reason about what word to form using LD and word frequency (familiarity), consistent with cognitive load theory [236].

Remark: It is emphasized that players in the experiments are not given a word corpus, frequency of letter use, d^L concepts and values, etc. Our construction and procedures presented here are our representation of the mental reasoning processes that players engage in, resulting in human behavior in the form of detailed actions. In experiments, players are only given letters and the ability to share them. This remark holds for the next two models, too.

Algorithm for form word. The algorithm is in Figure 4.7, and follows directly from the above data analysis. This is cast as the *agent* model in the ABMS.

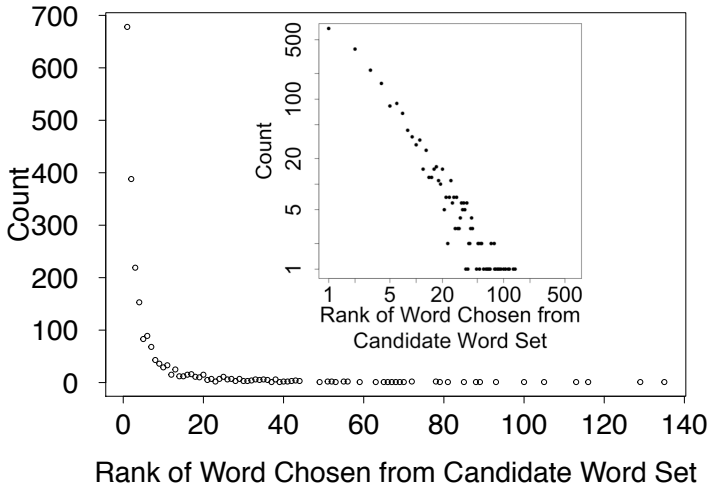


Figure 4.6: Experimental data for $|C_i^W| = 5000$. Log-log scale plot (inset) of the distribution of ranks of words formed by players from the word set $W_i^{ih}(w_1, d_{i,act}^L)$. Lesser rank means higher word frequency from corpus. Players most often choose words with lesser rank (i.e., greater frequency).

4.5.3 Player Action: Request Letter

Basic premise and key concepts. Our goal is to uncover a process that explains how players select the next letter to request from their neighbors. Our premise is that player v_i will select the next letter to request as the letter from the set of candidate neighboring letters L_i' that produces the greatest increase in the number of words that v_i can form. The key idea is to examine each candidate letter ℓ and determine the number of *new* words $|W_i^{ih\ell}|$ that can be formed with existing letters in L_i^{ih} and the requested letter combined (this word set is $W_i^{ih\ell}$), rank these letters in decreasing order of $|W_i^{ih\ell}|$, and select the letter to request based on this ranking. This is a greedy process—in the sense of selecting the best letter (i.e., the letter that ranks first), one at a time—and is our mechanstic model. This is a rational choice approach [27] because players are incentivized to form as many words as possible, so it is rational to select a letter that maximally increases the number of words that can be formed. Note that as more letters have been requested and received, the number of letters to request, $|L_i'|$, decreases because once a player has a letter, she can use it any number of times. We now provide the evidence for behavior that is aligned with this premise.

Data analysis. Analysis step 1. We rank all players by their performance in requesting letters in the GrAG, as follows. For each v_i, and for each actual letter request, we rank the candidate letters to request in L_i' according to our greedy model (given immediately above), and then identify the rank $r_{i,act}$ of the letter $\ell_{i,act}$ actually requested. Then we compute an average rank of letter requests $r_{i,ave}$ for each v_i, over the first 1/2 of all v_i's requests. We use only the first 1/2 of requests in computing $r_{i,ave}$ because as $|L_i'|$ decreases, the selected

Input: Agent $v_i \in V$. Agent word-forming aptitude b_i^{wf}. Word corpus or vocabulary C_i^W for v_i. Letters in-hand L_i^{ih}. Most recent word formed by v_i, w_1. Words W_i^f formed up to now by v_i. Distribution $\mathcal{D}^{wr}$ of word frequencies from C_i^W and distribution $\mathcal{D}^{d^L}$ of $d_{i,act}^L$ frequency as a function of tuple $(C_i^W, b_i^{wf}, d_{min}^L)$.
Output: Next word w_2 that v_i forms, if any.
Steps:

1. From letters in-hand L_i^{ih}, construct the set W_i^{ih} of words that v_i can form (and that v_i has not yet formed). Set $V_H = W_i^{ih}$ and let H be the WPN network induced by V_H. Let the edge set be E_H, with edge labels of d^L.
2. If V_H is empty, terminate algorithm and return no word.
3. From the values of the edge labels $d^L(w_1, w_j)$, for all edges $\{w_1, w^*\} \in E_H$ of WPN H, where $w_1, w^* \in V_H$, determine the minimum LD, d_{min}^L.
4. For the triple $(C_i^W, b_i^{wf}, d_{min}^L)$, sample from the distribution $\mathcal{D}^{d^L}$ to obtain the actual LD, $d_{i,act}^L$, that v_i will use to form the next word. (Example provided in Figure 4.5.)
5. From the set $W^{d_{i,act}^L} \subseteq V_H$ of words at $d_{i,act}^L$ from w_1, order the words from most frequently used word to least (C^W provides this ranking).
6. From the frequency distribution $\mathcal{D}^{wr}$ of words in $W^{d_{i,act}^L}$, draw a rank r_i of a word. Select the unique word w_2 that corresponds to rank r_i. Return w_2.

Figure 4.7: Steps of the Algorithm Form Word. This algorithm returns a word that an agent forms.

rank and the top-ranked letters will be more closely aligned because there are so few letters left; hence, in order to not bias the results, we use only the first 1/2 of letter requests. The players v_i are ranked by $r_{i,ave}$, smallest to largest value, and the players are partitioned into five equi-sized bins Q_1 through Q_5, where players in Q_1 (resp., Q_5) select letters to request that are most (resp., least) conformant to our mechanistic model. A player v_i's aptitude b_i^{req} in requesting letters takes a value from Q_1 through Q_5. This partitioning is to ensure a sufficient number of observations for each bin. Again, we partition based on players because we want to develop agent behaviors based on player behavior.

Analysis step 2. We analyze each Q_j, $j \in \{1, 2, 3, 4, 5\}$, separately, as follows. We take each $v_i \in Q_j$, note each $r_{i,act}$ corresponding to each letter request in the first 1/2 of requests, count the number of occurrences of the ranks of each requested letter, and sum the counts over all players. Results are shown in the left-most plot of Figure 4.8 for $b_i^{req} = Q_1 = 20\%$. (Note that player v_i's aptitude b_i^{req} in requesting letters may take values Q_1 through Q_5.) These data are for comparison against our mechanistic model (in green), which predicts all letter requests will be of rank 1 in this plot. Note that for the $b_i^{req} = Q_1 = 20\%$ data, the number of occurrences of a selected rank generally increases as the rank decreases, though the effect is sometime less pronounced for some cases. See Figures 11 and 13 of [45] for more data. We claim that the data support our premise, i.e., our model explains the data. That

is, players select letters to request that generate the greatest increase in the number of words that they can form.

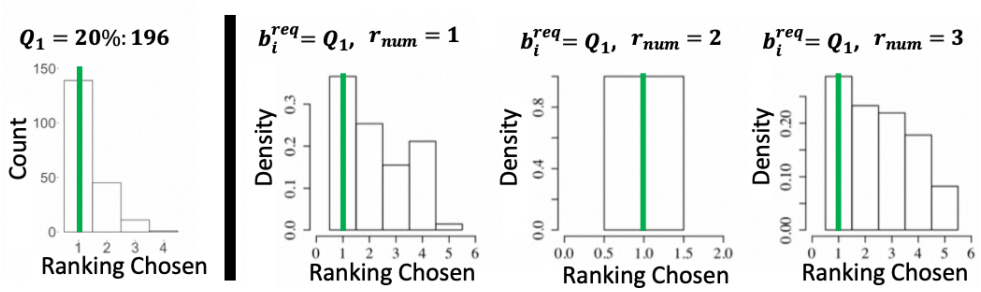


Figure 4.8: Comparison of ***mechanistic*** model predictions (in green) against data (the distributions) for the *request letter* model. Our mechanistic model predicts all letter requests will be of rank-1 in each of the four plots. (LEFT) Experimental data are for the 5000-word corpus, aptitude $b_i^{req} = Q_1 = 20\%$ for letter requests (plots for Q_j, $j \in \{2, 3, 4, 5\}$ are not shown). For aptitude Q_1, the frequency of the rank of the chosen letter is plotted. These data show that players most often choose letters with lower rank, meaning that they choose letters that can form relatively more words. (RIGHT) These three plots break down the left-most plot by showing distributions for different request numbers r_{num} by v_i. These distributions $\mathcal{D}^{\ell r}(|C_i^W|, b_i^{req}, r_{num})$ are used to sample $r_{i,act}$ based on $(C_i^W, b_i^{req}, r_{num})$.

Analysis step 3. We break down each plot of the type in Figure 4.8, at the left, to account for C_i^W, b_i^{req}, and the number r_{num} of the letter request in the three right-most plots of the figure. By sampling from frequency distributions $\mathcal{D}^{\ell r}(|C_i^W|, b_i^{req}, r_{num})$ based on $(C_i^W, b_i^{req}, r_{num})$ for v_i, we obtain the rank of the actual letter requested $r_{i,act}$ in the model. This provides finer modeling granularity by accounting for the number of the letter request.

Algorithm for request letter. The algorithm is in Figure 4.9 and follows directly from the data analysis just presented. This algorithm is presented in the form of an *agent* model.

Remark: These analyses and data provide evidence for our claim that this model is explanatory. Players generally request letters by (roughly) maximizing the increase in number of words that they can form, which follows rational choice theory [27].

4.5.4 Player Action: Reply to Letter Requests

Unlike the previous two models, this model is purely data-driven. A mechanistic-based model is under development. For space reasons, we provide an abbreviated description here; a fuller treatment is in [45].

Input: Agent $v_i \in V$. Agent letter requesting aptitude b_i^{req}. Word corpus C_i^W. Letters in-hand L_i^{ih}. The set L_i' of letters that v_i's neighbors were initially assigned that v_i has not yet requested; this is the candidate set of letters to request. The request number r_{num}. Distributions $\mathcal{D}^{\ell r}(|C_i^W|, b_i^{req}, r_{num})$ of letter ranks for triples $(C_i^W, b_i^{req}, r_{num})$.

Output: Next letter ℓ^* that v_i requests, if any.

Steps:

1. If L_i' is empty, terminate and return no letter.
2. For each candidate letter to request $\ell \in L_i'$ that has yet to be requested, determine the *new* words $W_i^{ih\ell}$ that can be formed from C^W with the letters in set $L_i^{ih} \cup \{\ell\}$ (include only words that have not yet been formed).
3. If every word set $W_i^{ih\ell}$ for all ℓ is empty, remove an arbitrary letter ℓ^* from L_i', terminate this algorithm and return ℓ^*.
4. Rank the letters in $\ell \in L_i'$ in decreasing values of $|W_i^{ih\ell}|$. Let $r(\ell)$ be the rank of ℓ.
5. Determine the rank $r_{i,act}$ of the letter to select for requesting by sampling from distribution $\mathcal{D}^{\ell r}(|C_i^W|, b_i^{req}, r_{num})$ using as input $(C_i^W, b_i^{req}, r_{num})$. (See Figure 4.8 for three examples.)
6. Select the letter ℓ^* such that $r(\ell^*) = r_{i,act}$. Break ties arbitrarily. Remove ℓ^* from L_i'. Return ℓ^*.

Figure 4.9: Steps of the Algorithm Request Letter. This algorithm returns a letter that an agent requests.

Basic premise, key ideas, and data analysis. The goal is to produce a model that explains how players respond to letter requests from their neighbors. The basic premise is that players can be partitioned into categories of behavior. We determined from the data these four categories: (1) those players that respond to all queued (pending) letter requests in their buffer (called FB for full buffer); (2) those that respond to some fraction of all pending letter requests in their buffer (called LTFB for less than full buffer); (3) those that sometime behave as FB and sometimes as LTFB (called Mixed); and (4) those that never reply to letter requests (called NR). The key ideas are that for each category, we need to determine: (*i*) how many replies to letter requests are made uninterrupted (i.e., contiguously) for categories LTFB and Mixed, and (*ii*) for each number of letter replies, the time duration over which these letter replies are made (for categories FB, LTFB, and Mixed). See [45] for results. These are the four values for a player v_i's aptitude b_i^{rpl} in replying to letter requests.

Algorithm for reply to (letter) request. Owing to space limitations, the algorithm is not provided here, but is provided in a web-accessible version in [45], Figure 16.

Remark: In these various algorithms, elements of sets are returned, or a distribution corresponding to particular inputs is sampled. In some cases, there are no data for specified conditions. For these types of situations, we implement a recursive search technique to sample from the distribution or set with the closest set of inputs.

4.6 Agent-Based Simulations and Results

Remark: *Model evaluation* is an important step and has been performed. Figures 4.4, 4.5, and 4.8 are part of this process. We refer the reader to Section VI of [45] for additional work.

Simulation model. We conduct discrete time agent-based simulations (ABSs) of the GrAG. Each time unit is one second of the 300-second GrAG. At each time and for each agent, an action is selected. Based on the action chosen, the corresponding model for that action, developed herein, is executed (Figures 4.7 and 4.9 for "form word" and "request letter," respectively, and Figure 16 of [45] for "reply to request"; the thinking action is a no-op). We run $n_{runs} = 100$ runs or simulation instances and average the results. We use the 5000-word corpus C^W. These are purely simulation studies and are not tied to the experiments. The goal is to demonstrate that the models alone provide insights into human behavior.

Study 1: Effects of model aptitude properties. We use a game configuration $G(V, E)$ consisting of six players that form a circle, with each player having two neighbors. The initial letter assignments are given in Table 4.1. We systematically vary the aptitudes of players in forming words b_i^{wf}, in requesting letters b_i^{req}, and in replying to letter requests b_i^{rpl}. See Table 4.2. Recall that these aptitudes correspond to the skill levels of players.

Table 4.1: Study 1 initial letter assignments to players in simulations for six players arranged as 2-regular graph.

Player #:	1	2	3	4	5	6
Init. Ltrs:	b, a, t	m, e, n	l, u, t	s, o, p	h, u, g	r, i, e

Table 4.2: Parameters that are systematically varied in the simulations of Study 1. These aptitude (b_i^{wf}, b_i^{req}, b_i^{rpl}) settings are the same for all agents in a simulation.

Sim. No.	b_i^{wf}	b_i^{req}	b_i^{rpl}	**Sim. No.**	b_i^{wf}	b_i^{req}	b_i^{rpl}
1	P_1	Q_1	FB	5	P_5	Q_5	FB
2	P_2	Q_2	FB	6	P_5	Q_5	LTFB
3	P_3	Q_3	FB	7	P_5	Q_5	NR
4	P_4	Q_4	FB	–	–	–	–

Figure 4.10 (left) shows the average number of interactions (requests sent, replies received, requests received, replies sent) and the average number of words formed per player for the first five simulation numbers (sim. no.) of Table 4.2. There is a drop-off in performance in going from $b_i^{wf} = P_1$ to P_5, $b_i^{req} = Q_1$ to Q_5, for fixed b_i^{rpl} =FB. We observe that decreasing the letter request aptitude b_i^{req} and the word formation aptitude b_i^{wf} decreases the quality of letters requested and hence the number of words that can be formed.

To determine how b_i^{rpl} affects performance, we plot in Figure 4.10 (right) results from simulation numbers 5, 6, and 7 of Table 4.2. Using $b_i^{wf} = P_5$ and $b_i^{req} = Q_5$ as a reference, there

is a large decrease in numbers of reply interactions in going from b_i^{rpl} =LTFB to b_i^{rpl} =NR, as expected, since NR means that agents do not reply to letter requests. There is a small decrease in numbers of replies in reducing b_i^{rpl} from FB to LTFB.

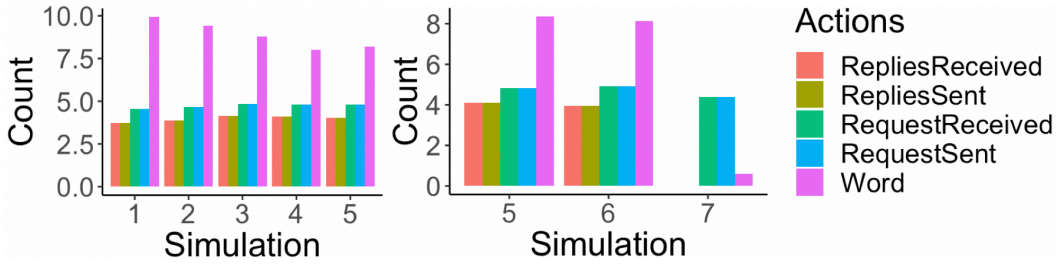


Figure 4.10: (Left) Simulation results for Sim. nos. 1 through 5 of Table 4.2. The average number of words formed per player drops in going from $b_i^{wf} = P_1$ to P_5, $b_i^{req} = Q_1$ to Q_5, for fixed b_i^{rpl} =FB. (Right) Simulation results for Sim. nos. 5, 6, and 7 of Table 4.2. Using $b_i^{wf} = P_5$ and $b_i^{req} = Q_5$ as a baseline, these results show a precipitous drop-off in replies to letter requests, and to words formed, in going from b_i^{rpl} =LTFB to b_i^{rpl} =NR. Results in counts for b_i^{rpl} =LTFB are slightly less than those for b_i^{rpl} =FB.

Study 2: Effects of heterogeneity: network connectivity and quality of letter assignments to players. We use a game configuration $G(V, E)$ consisting of four players v_i $(1 \leq i \leq 4)$ that form a star. The initial letter assignments are given in Figure 4.11. All players have the following conditions $b_i^{wf} = P_1$, $b_i^{req} = Q_1$, and b_i^{rpl} =FB. Players are assigned heterogeneous numbers and qualities of letters; see the figure caption. The numbers of requests received and replies sent are greatest for player v_1 owing to its centrality; this affects the number of words player v_1 forms, which is less than those for v_2 and v_3. Players v_2 and v_3 have more requests received from v_1 (compared to v_4) because their letters (i.e., popular consonants) create larger sets of possible words to form. The number of words formed is least for player v_4 because of the poorer quality of assigned letters.

4.7 Summary and Future Work

We have developed mechanistic and data-driven models for representing the decision-making and actions of humans in online networked GrAGs. Our contributions are in Section 4.2.4. We would like to conduct more experiments with more network structures. This would also (ideally) produce sufficient data to more finely partition aptitudes—player behavior—into ten 10% bins (currently, we have five 20% bins). These experiments would be used to further evaluate the models and improve them.

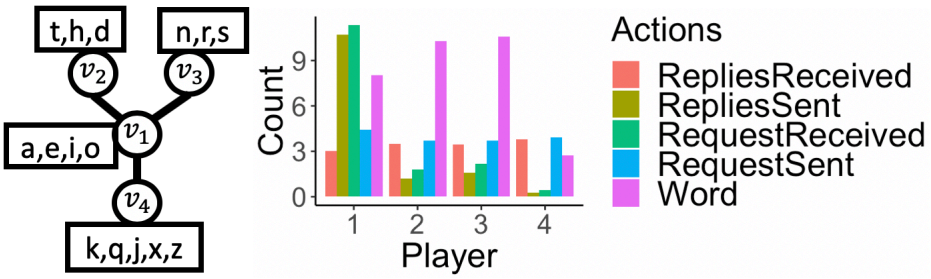


Figure 4.11: Simulation for players v_i $(1 \leq i \leq 4)$, arranged in a star. All players have the following conditions $b_i^{wf} = P_1$, $b_i^{req} = Q_1$, and b_i^{rpl} =FB. Player v_1 is at the center with three neighbors. v_1 is assigned the four most popular vowels in the alphabet; v_2, v_3 are assigned the six most popular consonants, and v_4 is assgned the five least popular consonants. See text for discussion of results.

Chapter 5

Conclusions

Online social science experiments are being used to understand behavior at-scale. Due to the considerable work required to perform data analytics for custom experiments and to build models from experimental data, the work in this thesis presents an automated and extensible system for evaluating social phenomena through iterative experiments and modeling. A set of five composable and extensible pipelines for studying networked social science phenomena has been presented, along with data and computational models for formal specification of experiments and modeling and simulation (MAS), for a particular class of networked social science experiments. We provide case studies on collective identity, complex contagion, and structure of communication networks, respectively, to illustrate the successful use of the system.

Also, we use these pipelines to study collective identity (CI) in a group activity. We formalize an abductive loop, implement it computationally, and exercise it in an experimental setting (the group anagram game) designed to induce CI, as operationalized by Swann's DIFI score. However, our abductive looping process is not tied to CI. As part of the abductive iterations, we provide novel experimental insights into CI and build and evaluate three ABMs. This work establishes the potential of iterative abductive looping for the (computational) social sciences.

We have presented a process for combining mechanistic and data-driven approaches. We provide new findings on how players behave and on explanations of behavior based on Levenshtein distance and utility maximization, motivated by cognitive and economic theories. Finally, mechanistic and data-driven models for representing the reasoning and actions of humans in networked group anagram games are developed.

Bibliography

[1] Word frequency data: Corpus of contemporary american english. `https://www.wordfrequency.info/free.asp`. Accessed: 2018-11-16.

[2] K. A. and R. A. H. Factors influencing the bias towards one's own group. 2:33–50, 01 1972.

[3] A. Abbott. Sequence analysis: New methods for old ideas. *Annual Review of Sociology*, 21:93–113, 1995.

[4] D. E. Abrams and M. A. Hogg. *Social identity theory: Constructive and critical advances.* Springer-Verlag Publishing, 1990.

[5] R. Ackland and M. O'Neil. Online collective identity: The case of the environmental movement. *Social Networks*, 33:177–190, 2011.

[6] A. Adiga, C. J. Kuhlman, M. V. Marathe, H. S. Mortveit, S. S. Ravi, and A. Vullikanti. Graphical dynamical systems and their applications to bio-social systems. *International Journal of Advances in Engineering Sciences and Applied Mathematics*, Dec 2018.

[7] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[8] A. E. Ahmed, J. Heldenbrand, Y. Asmann, F. M. Fadlelmola, D. S. Katz, K. K. M. C. Kendzior, et al. Managing genomic variant calling workflows with swift/t. *PLoS Computational Biology*, pages e1006843–1–e1006843–14, 2019.

[9] R. Aipperspach, E. Cohen, and J. Canny. Modeling human behavior from simple sensors in the home. In *Proceedings Of The IEEE Conference On Pervasive Computing*, 05 2006.

[10] J. C. Alexander, R. Eyerman, B. Giesen, N. J. Smelser, and P. Sztompka. *Cultural Trauma and Collective Identity.* University of California Press, 1 edition, 2004.

[11] P. Amstutz, M. R. Crusoe, N. Tijanic, B. Chapman, J. Chilton, M. Heuer, A. Kartashov, D. Leehr, H. Mnager, M. Nedeljkovich, M. Scales, S. Soiland-Reyes, and L. Stojanovic. Common workflow language, v1.0., 2016. [Online; accessed May-2019].

[12] E. A. M. Andrews and A. J. Bonner. Explaining genetic knock-out effects using cost-based abduction. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1635–1640, 2011.

[13] Anonymous. Workflow description language. `https://software.broadinstitute.org/wdl/documentation/spec`, 2019. [Online; accessed May-2019].

[14] J. Arlow and I. Neustadt. *UML 2.0 and the Unified Process: Practical Object-Oriented Analysis and Design (2Nd Edition)*. Addison-Wesley Professional, 2005.

[15] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor. Scientific workflows: Past, present and future. *Future Gener. Comput. Syst.*, 75:216–227, 2017.

[16] S. Atran, R. Axelrod, and R. Davis. Sacred barriers to conflict resolution. *Science*, 317:1039–1040, 2007.

[17] S. Atran, H. Sheikh, and A. Gomez. Devoted actors sacrifice for close comrades and sacred cause. *Proceedings of the National Academy of Sciences*, 111(50):17702–17703, 2014.

[18] S. Atran, H. Sheikh, and A. Gomez. For cause and comrade: Devoted actors and willingness to fight. *Cyclodynamics*, 5:41–57, 2014.

[19] T. Attema, P. P. van Maanen, and E. Meeuwissen. Development and evaluation of multi-agent models predicting twitter trends in multiple domains. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 1133–1140, Aug 2015.

[20] A. Bach. Boltzmann's probability distribution of 1877. *Archive for History of Exact Sciences*, 41:1–40, 1990.

[21] C. W. Bachman. Data structure diagrams. *SIGMIS Database*, 1(2):4–10, July 1969.

[22] M. Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, 2016.

[23] C. Barrett, H. B. Hunt, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, R. E. Stearns, and M. Thakur. Predecessor existence problems for finite discrete dynamical systems. *Theoretical Computer Science*, pages 3–37, 2007.

[24] C. L. Barrett, H. B. Hunt, M. V. Marathe, S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. Complexity of reachability problems for finite discrete dynamical systems. *J. Comp. Syst. Sci.*, 72(8):1317–1345, 2006.

[25] D. Barseghian, I. Altintas, M. B. Jones, D. Crawl, N. Potter, J. Gallagher, et al. Workflows and extensions to the kepler scientific workflow system to support environmental sensor data access and analysis. *Ecological Informatics*, 5(1):42–50, 2010.

[26] G. S. Becker. A theory of social interaction. *Journal of Political Economy*, 82:1063–1093, 1974.

[27] G. S. Becker. *The economic approach to human behavior*. University of Chicago Press Chicago, 1976.

[28] K. Belhajjame, J. Zhao, D. Garijo, M. Gamble, K. Hettne, R. Palma, E. Mina, O. Corcho, J. M. Gomez-Perez, S. Bechhofer, G. Klyne, and C. Goble. Using a suite of ontologies for preserving workflow-centric research objects. *Web Semantics: Science, Services and Agents on the World Wide Web*, 32(0), 2015.

[29] D. J. Benjamin, J. J. Choi, and G. Fisher. Religious identity and economic behavior. *The Review of Economics and Statistics*, 98(4):617–637, 2016.

[30] J. H. M. Bergmann, P. M. Langdon, R. E. Mayagoitia, and N. Howard. Exploring the use of sensors to measure behavioral interactions: An experimental evaluation of using hand trajectories. *PLOS ONE*, 9(2):1–10, 02 2014.

[31] A. Bokulich. How scientific models can explain. *Synthese*, 180:33–45, 2011.

[32] G. Bornstein and I. Yaniv. Individual and group behavior in the ultimatum game: Are groups more rational players? *Experimental Economics*, 1:101–108, 1998.

[33] M. B. Brewer. The social self: On being the same and different at the same time. 17:475–482, 10 1991.

[34] M. B. Brewer and W. Gardner. Who is this "we"? levels of collective identity and self representations. 71:83–93, 07 1996.

[35] M. B. Brewer and M. Silver. Ingroup bias as a function of task characteristics. *European Journal of Social Psychology*, 8(3):393–400, 1978.

[36] H. Brody. *The Other Side of Eden: Hunters, Farmers, and the Shaping of the World*. Farrar, Straus, and Giroux, New York, NY, 2000.

[37] A. R. Brunsdon. #misconstruedidentitiesmustfall collective: Identity formation in the current south african context: A practical theological perspective. *HTS Theological Studies*, 73:1–7, 2017.

[38] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal. *Pattern-Oriented Software Architecture - Volume 1: A System of Patterns*. Wiley Publishing, 1996.

[39] J. Butler. Performative acts and gender constitution: An essay in phenomenology and feminist theory. *Theatre Journal*, 40(4):519–531, 1988.

[40] C. B. Cadsby et al. Sorting and incentive effects of pay for performance: An experimental investigation. *Acad. of Mgmt. Jour.*, 2007.

[41] C. B. Cadsby, F. Song, and F. Tapon. Are you paying your employees to cheat? an experimental investigation. *The B.E. Journal of Economic Analysis & Policy*, 10:1–30, 2010.

[42] S. Callaghan, E. Deelman, D. Gunter, G. Juve, P. Maechling, C. Brooks, K. Vahi, K. Milner, R. Graves, E. Field, D. Okaya, and T. Jordan. Scaling up workflow-based applications. *J. Comput. Syst. Sci.*, 76(6):428–446, Sept. 2010.

[43] D. Cameron. Performing gender identity: Young men's talk and the construction of heterosexual masculinity. In S. Johnson and U. Hanna, editors, *Language and Masculinity*, pages 47–64. Oxford: Blackwell, 1997.

[44] V. Capraro. A model of human cooperation in social dilemmas. *PLoS One*, 8:e72427–1–e72427–6, 2013.

[45] V. Cedeno-Mieles et al. Modeling anagram games. Technical report, 2019. `https://github.com/vcedeno/asonam19/blob/master/tr.pdf`.

[46] V. Cedeno-Mieles, Y. Ren, S. Ekanayake, B. J. Goode, C. J. Kuhlman, D. Machi, M. V. Marathe, H. H. Mortveit, Z. Hu, X. Deng, N. Ramakrishnan, P. Saraf, N. Self, N. Contractor, J. M. Epstein, and M. W. Macy. Pipelines and their compositions for modeling and analysis of controlled online networked social science experiments. In *2018 Winter Simulation Conference (WSC)*, pages 774–785, Dec 2018.

[47] D. Centola. The spread of behavior in an online social network experiment. *Science*, 329:1194–1197, 2010.

[48] D. Centola. An experimental study of homophily in the adoption of health behavior. *Science*, 334:1269–1272, 2011.

[49] T. Cerny, M. J. Donahoo, and M. Trnka. Contextual understanding of microservice architecture: Current and future directions. *Applied Computing Review*, 17(4):29–45, 2017.

[50] E. Chamiak and E. S. Jr. Dynamic map calculations for abduction. In *Proceedings AAAI Conference on Artificial Intelligence*, pages 552–557, 1992.

[51] G. Charness, R. Cobo-Reyes, et al. Identities, selection, and contributions in a public-goods game. *Games and Economic Behavior*, 2014.

[52] G. Charness, P. Kuhn, and M. C. Villeval. Competition and the ratchet effect. *Journal of Labor Economics*, 29:513–547, 2011.

[53] G. Charness, L. Rigotti, and A. Rustichini. Individual behavior and group membership. *American Economic Review*, 97:1340–1352, 2007.

[54] D. L. Chen et al. otree–an open-source platform for laboratory, online and field experiments. *J. Beh. & Exp. Fin.*, 9:88–97, 2016.

[55] D. L. Chen and S. Yeh. The construction of morals. *Journal of Economic Behavior & Organization*, 104, 2014.

[56] P. P.-S. Chen. The entity-relationship model-toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, Mar. 1976.

[57] R. Chen and Y. Chen. The potential of social identity for equilibrium selection. *American Economic Review*, 101(6):2562–89, October 2011.

[58] Y. Chen and S. Li. Group identity and social preferences. *American Economic Review*, 99:431–457, 2009.

[59] Y. Chen, S. X. Li, T. X. Liu, and M. Shih. Which hat to wear? impact of natural identities on coordination and cooperation. *Games and Economic Behavior*, 84:58–86, 2014.

[60] F. Chierichetti, J. Kleinberg, R. Kumar, M. Mahdian, and S. Pandey. Event detection via communication pattern analysis. *Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014*, pages 51–60, 01 2014.

[61] A. M. Choup. The formation and manipulation of collective identity: A framework for analysis. *Social Movement Studies*, 7(2):191–207, 2008.

[62] J. Clarke, V. Srikumar, M. Sammons, and D. Roth. In *LREC*, pages 3276–3283. European Language Resources Association (ELRA), 2012.

[63] S. Cohen-Boulakia, K. Belhajjame, O. Collin, J. Chopard, C. Froidevaux, A. Gaignard, K. Hinsen, P. Larmande, Y. L. Bras, F. Lemoine, F. Mareuil, H. Menager, C. Pradal, and C. Blanchet. Scientific workflows for computational reproducibility in the life sciences: Status, challenges and opportunities. *Future Gener. Comput. Syst.*, 75:284–298, 2017.

[64] A. Cohn, E. Fehr, and M. A. Maréchal. Business culture and dishonesty in the banking industry. *Nature*, 516:86–89, 2014.

[65] A. Cohn, M. A. Marechal, and T. Noll. Bad boys: How criminal identity salience affects rule violation. *The Review of Economic Studies*, 82(4):1289–1308, 06 2015.

[66] T. M. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[67] H. Cunningham, D. Maynard, and K. Bontcheva. *Text Processing with GATE*. Gateway Press CA, 2011.

[68] S. Currarini, M. Jackson, and P. Pin. An economic model of friendship: homophily, minorities and segregation. *Econometrica*, 77:1003–1045, 2009.

[69] R. F. da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman. A characterization of workflow management systems for extreme-scale applications. *Future Generation Computer Systems*, 2017.

[70] W. L. Davis and D. E. Davis. Internal-external control and attribution of responsibility for success and failure. *J. of Personality*, 1972.

[71] M. de Oliveira. Isotropic majority-vote model on a square lattice. *Journal of Statistical Physics*, 66:273–281, 01 1992.

[72] L. A. DeChurch and J. R. Mesmer-Magnus. The cognitive underpinnings of effective teamwork: A meta-analysis. *Journal of Applied Psychology*, 95:32–53, 2010.

[73] E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Gener. Comput. Syst.*, 25(5):528–540, 2009.

[74] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. Ferreira da Silva, M. Livny, and K. Wenger. Pegasus, a workflow management system for science automation. *Future Gener. Comput. Syst.*, 46:17–35, 2015.

[75] R. L. Dominowski. The effect of pronunciation practice on anagram difficulty. *Psychonomic Science*, 16(2):99–100, Feb 1969.

[76] M. Drouvelis, R. Metcalfe, and N. Powdthavee. Priming Cooperation in Social Dilemma Games. IZA Discussion Papers 4963, Institute for the Study of Labor (IZA), May 2010.

[77] E. Durkheim. *Suicide*. Free Press, 1951.

[78] M. Echenim, N. Peltier, and S. Tourret. An approach to abductive reasoning in equational logic. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 531–537, 2013.

[79] C. Eckel and P. Grossman. Managing diversity by creating team identity. *J. Econ. Behav. Organ.*, 58:371–392, 2005.

[80] E. Elmroth, F. Hernández, and J. Tordsson. Three fundamental dimensions of scientific workflow interoperability: Model of computation, language, and execution environment. *Future Gener. Comput. Syst.*, 26(2):245–256, Feb. 2010.

[81] J. M. Epstein. *Generative Social Science: Studies in Agent-Based Computational Modeling*. Princeton University Press, 2007.

[82] T. H. Eriksen. *Ethnicity and Nationalism: Anthropological Perspectives*. Pluto Press, 3 edition, 2010.

[83] E. H. Erikson. *Identity and the Life Cycle*. W. W. Norton & Company, 1980.

[84] N. T. Feather. Attribution of responsibility and valence of success and failure in relation to initial confidence and task performance. *Journal of Personality and Social Psychology*, 13:129–144, 1969.

[85] N. T. Feather and J. G. Simon. Attribution of responsibility and valence of outcome in relation to initial confidence and success and failure of self and other. *Journal of Personality and Social Psychology*, 18:173–188, 1971.

[86] N. T. Feather and J. G. Simon. Causal attributions for success and failure in relation to expectation of success based upon selective or manipulative control. *Journal of Personality and Social Psychology*, 39:527–541, 1971.

[87] O. Feher, E. Wonnacott, and K. Smith. Structural priming in artificial languages and the regularisation of unpredictable variation. *Journal of Memory and Language)*, 91:158–180, 2016.

[88] S. T. Fiske, D. T. Gilbert, and G. Lindzey. *Handbook of Social Psychology*. Wiley, 5 edition, 2010.

[89] P. A. Flach and A. C. Kakas. *Abduction and Induction: Essays on their Relation and Integration*. Springer, New York, NY, 2010.

[90] C. F. Fominaya. Collective identity in social movements: Central concepts and debates. *Sociology Compass*, 4:393–404, 2010.

[91] M. Fowler. *Patterns of Enterprise Application Architecture*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[92] R. Fujimoto, C. Bock, W. Chen, E. Page, and J. H. Panchal. *Research Challenges in Modeling and Simulation for Engineering Complex Systems*. Springer, 2017b.

[93] R. M. Fujimoto, C. Carothers, A. Ferscha, D. Jefferson, M. Loper, M. Marathe, and S. J. E. Taylor. Computational challenges in modeling simulation of complex systems. In *2017 WSC*, pages 431–445, 2017.

[94] D. Garijo, P. Alper, K. Belhajjame, O. Corcho, Y. Gil, and C. Goble. Common motifs in scientific workflows: An empirical analysis. In *2012 IEEE 8th International Conference on E-Science*, pages 1–8, Oct 2012.

[95] M. A. Gates, J. W. Suchow, and T. L. Griffiths. Empirical tests of large-scale collaborative recall. In *CogSci*, 2017.

[96] A. L. Gibbs and F. E. Su. On choosing and bounding probability metrics. *INTERNAT. STATIST. REV.*, pages 419–435, 2002.

[97] Y. Gil, E. Deelman, M. Ellsman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, and J. Myers. Examining the challenges of scientific workflows. *IEEE*, pages 24–32, 2007.

[98] Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim. Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In *Proceedings of the 19th National Conference on Innovative Applications of Artificial Intelligence - Volume 2*, IAAI'07, pages 1767–1774. AAAI Press, 2007.

[99] K. J. Gilhooly and C. E. Johnson. Effects of solution word attributes on anagram difficulty: A regression analysis. *Quarterly Journal of Experimental Psychology*, 30(1):57–70, 1978.

[100] F. J. GilWhite. Are ethnic groups biological species to the human brain?: Essentialism in our cognition of some social categories. *Current Anthropology*, 42(4):515–553, 2001.

[101] J. Ginges and S. Atran. Sacred values and cultural conflict. In *Advances in Culture and Psychology*, pages 273–305. 2013.

[102] J. Goecks, A. Nekrutenko, and J. Taylor. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8):R86, Aug 2010.

[103] C. A. Goldberg. Haunted by the specter of communism: Collective identity and resource mobilization in the demise of the workers alliance of america. *Theory and Society*, 32(5/6):725–773, 2003.

[104] M. Goldman, J. W. Stockbauer, et al. Intergroup and intragroup competition and cooperation. *J. of Exper. Soc. Psych,*, 1977.

[105] A. Gomez, M. L. Brooks, M. D. Buhrmester, A. Vazquez, J. Jetten, and W. B. Swann. On the nature of identity fusion: Insights into the construct and a new measure. *Journal of Personality and Social Psychology*, pages 918–933, 2011.

[106] A. Gomez, J. Morales, S. Hart, A. Vazquez, J. Jetten, and W. B. Swann. Rejected and excluded forevermore, but even more devoted: Irrevocable ostracism intensifies loyalty to the group among identity-fused persons. *Personality and Social Psychology Bulletin*, pages 1574–1586, 2011.

[107] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[108] B. Greene. *Letters to a Young Poet*. New World Library, 2000.

[109] B. Greenhill. Recognition and collective identity formation in international politics. *European Journal of International Relations*, 14(2):343–368, 2008.

[110] V. Guralnik and K. Z. Haigh. Learning models of human behaviour with sequential patterns. In *Proceedings of the AAAI-02 workshop "Automation as Caregiver"*, pages 24–30, 2002. AAAI Technical Report WS-02-02.

[111] H. H. Kendler and T. S. Kendler. Vertical and horizontal processes in problem solving. *Psychological review*, 69:1–16, 02 1962.

[112] B. D. Haig. An abductive theory of scientific method. *Psychological Methods*, 10:371–388, 2005.

[113] D. J. Hand, P. Smyth, and H. Mannila. *Principles of Data Mining*. MIT Press, Cambridge, MA, USA, 2001.

[114] A. G. Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[115] N. Heller. The philosopher redefining equality: Elizabeth anderson thinks we've misunderstood the basis of a free and fair society. *The New Yorker*, 07 January, 2019. accessed online at `https://www.newyorker.com/magazine/2019/01/07/the-philosopher-redefining-equality`.

[116] K. Hoff and P. Pandey. Discrimination, social identity, and durable inequalities. *American Economic Review*, 96(2):206–211, May 2006.

[117] K. Hoff and P. Pandey. Making up people-the effect of identity on performance in a modernizing society. *Journal of Development Economics*, 106(C):118–131, 2014.

[118] J. M. Hofman, A. Sharma, and D. J. Watts. Prediction and explanation in social systems. *Science*, 355:486–488, 2017.

[119] M. A. Hogg and D. Abrams. Intergroup behavior and social identity. In M. A. Hogg and J. Cooper, editors, *The SAGE Handbook of Social Psychology*, pages 335–360. Sage Publishing, 2007.

[120] G. Homans. *Social Behavior: Its Elementary Forms*. Harcourt Brace, 1961.

[121] J. Hu, M. Liu, and J. Zhang. A semantic model for academic social network analysis. In *2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014)*, pages 310–313, Aug 2014.

[122] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153. IEEE, 2009.

[123] J. Jebeile. Explaining with simulations: Why visual representations matter. *Perspectives on Science*, 26:213–238, 2018.

[124] J. Jebeile and A. G. Kennedy. Explaining with models: The role of idealization. *Int. Studies in the Philos. of Science*, pages 383–392, 2015.

[125] J. Jiménez, A. Gomez, M. D. Buhrmester, et al. The dynamic identity fusion index: A new continuous measure of identity fusion for web-based questionnaires. *Soc. Sci. Comp. Rev.*, pages 215–228, 2016.

[126] Y. Jo, G. Tomar, O. Ferschke, C. P. Rosé, and D. Gašević. Pipeline for expediting learning analytics and student support from data in social learning. In *Proceedings of the Sixth International Conference on Learning Analytics & Knowledge*, LAK '16, pages 542–543, New York, NY, USA, 2016. ACM.

[127] B. Juba. Learning abductive reasoning using random examples. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 999–1007, 2016.

[128] S. Judd, M. Kearns, and Y. Vorobeychik. Behavioral dynamics and influence in networked coloring and consensus. *Proceedings of the National Academy of Sciences*, 107(34):14978–14982, 2010.

[129] M. Juergensmeyer. *Terror in the Mind of God: The Global Rise of Religious Violence*. University of California Press, 3 edition, 2003.

[130] G. Kaushik, S. Ivkovic, J. Simonovic, N. Tijanic, B. Davis-Dusenbery, and D. Kural. Rabix: An open-source workflow executor supporting recomputability and interoperability of workflow descriptions. In *Biocomputing*, pages 154–165, 2013.

[131] M. Kearns, S. Judd, et al. Behavioral experiments on a network formation game. In *EC*, pages 690–704, 2012.

[132] M. Kearns, S. Judd, J. Tan, and J. Wortman. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences*, 106(5):1347–1352, 2009.

[133] K. Kennedy and R. Allen. *Optimizing Compilers for Modern Architectures: A Dependence-based Approach*. Morgan Kaufmann, 2001.

[134] J. S. Kinnebrew, K. M. Loretz, and G. Biswas. A contextualized, differential sequence mining method to derive students' learning behavior patterns. *Journal of Educational Data Mining*, 5(1), 2013.

[135] R. Korolov, D. Lu, J. Wang, G. Zhou, C. Bonial, C. Voss, L. Kaplan, W. Wallace, J. Han, and H. Ji. On predicting social unrest using social media. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2016*, pages 89–95. Institute of Electrical and Electronics Engineers Inc., 11 2016.

[136] R. Korolov, J. Peabody, A. Lavoie, S. Das, M. Magdon-Ismail, and W. Wallace. Actions are louder than words in social media. In *2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 292–297, Aug 2015.

[137] B. Kozegi. Ego utility, overconfidence, and task choice. *Journal European Economic Association*, 4:673–707, 2006.

[138] S. W. Kozlowski and D. R. Ilgen. Enhancing the effectiveness of work groups and teams. *Psychological Science in the Public Interest*, 7(3):77–124, 2006.

[139] S. Kramer. The biggest thing amazon got right: The platform. `https://gigaom.com/2011/10/12/419-the-biggest-thing-amazon-got-right-the-platform/`, October 2006. [Online; accessed May-2019].

[140] S. Kullback. *Information Theory and Statistics*. Wiley, New York, 1959.

[141] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951.

[142] T. Kurashima, T. Althoff, and J. Leskovec. Modeling interdependent and periodic real-world action sequences. In *World Wide Web Conference WWW*, pages 803–812. ACM, 2018.

[143] T. Kyle, P. DeScioli, O. Haque, and S. Pinker. The psychology of coordination and common knowledge. *Journal of Personality and Social Psychology*, 107:657–676, 2014.

[144] G. P. Latham and E. A. Locke. Self-regulation through goal setting. *Organizational Behavior and Human Decision Processes*, 50:212–247, 1991.

[145] B. Laurenczy, A. Kashev, H. Stockinger, P. Escobar Lpez, and S. Maffioletti. Guidelines for pipeline interoperability using containers, July 2017. [Online; accessed May-2019].

[146] D. Lazer and A. Friedman. The network structure of exploration and exploitation. *Administrative Science Quarterly*, 52(4):667–694, 2007.

[147] D. Lazer, A. Pentland, L. Adamic, S. Aral, A.-L. Barabási, D. Brewer, N. Christakis, N. Contractor, J. Fowler, M. Gutmann, T. Jebara, G. King, M. Macy, D. Roy, and M. Van Alstyne. Computational social science. *Science*, 323(5915):721–723, 2009.

[148] J. O. Ledyard. Public Goods: A Survey of Experimental Research. Public economics, University Library of Munich, Germany, May 1994.

[149] J. Y. Lee and J. C. Oh. A model for recursive propagations of reputations in social networks. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 666–670, New York, NY, USA, 2013. ACM.

[150] J. Leipzig. A review of bioinformatics pipeline frameworks. *Briefings in Bioinformatics*, 18(3):530–536, 2017.

[151] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 1966.

[152] J. Lewis. Microservices - java, the unix way. *Proceedings of the 33rd Degree Conference for Java Masters*, March 2012.

[153] J. Lewis and M. Fowler. Microservices. `https://martinfowler.com/articles/microservices.html`, March 2014. [Online; accessed May-2019].

[154] B. Li, D. Sun, Z. Lin, and C. Ou. Agent-based simulation research on group emotion evolution of public emergency. In *ASONAM*, 2014.

[155] C. Li, A. Kowdle, A. Saxena, and T. Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. *CoRR*, abs/1110.5102, 2011.

[156] E. A. Locke and G. P. Latham. *A theory of goal setting and task performance*. Prentice-Hall, Englewood Cliffs, NJ, 1990.

[157] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. B. Jones, E. A. Lee, J. Tao, and Y. Zhao. Scientific workflow management and the kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[158] C. C. Luhmann and S. Rajaram. Memory transmission in small groups and large networks: An agent-based model. *Psychological science*, 26 12:1909–17, 2015.

[159] I. Lustick. Agent-based modelling of collective identity: Testing constructivist theory. *Journal of Artificial Societies and Social Simulation*, 3(1), 2000.

[160] C. M. Macal and M. J. North. Agent-based modeling and simulation. In *Winter Simulation Conference*, WSC '09, pages 86–98, 2009.

[161] N. MacGregor. *Living with the Gods: On Beliefs and Peoples*. Knopf, 1 edition, 2018.

[162] M. W. Macy, , and R. Willer. From factors to factors: Computational sociology and agent-based modeling. *Annual Review of Sociology*, 28(1):143–166, 2002.

[163] W. Manchester. *A World Lit Only By Fire: The Medieval Mind and the Renaissance: Portrait of an Age*. Little, Brown and Company, 1993.

[164] W. Mason and D. J. Watts. Collaborative learning in networks. *Proceedings of the National Academy of Sciences*, 109(3):764–769, 2012.

[165] T. Mauro. Adopting microservices at netflix: Lessons for architectural design. https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/, February 2015. [Online; accessed May-2019].

[166] M. S. Mayzner and M. E. Tresselt. Anagram solution times: A function of letter order and word frequency. *Journal of Experimental Psychology*, 56(4):376, 1958.

[167] K. McAuliffe and Y. Dunham. Group bias in cooperative norm enforcement. *Phil. Trans. R. Soc. B*, 371:20150073–1–20150073–9, 2015.

[168] D. A. McFarland, J. Moody, D. Diehl, J. A. Smith, and R. J. Thomas. Network ecology and adolescent social structure. *American Sociological Review*, pages 1–34, 2014.

[169] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, et al. Active learning machine learns to create new quantum experiments. *Proceedings of the National Academy of Sciences*, 2018.

[170] A. Melucci. *Nomads of the Present: Social Movement and Identity Needs in Contemporary Society*. Temple University Press, 1989.

[171] A. Melucci. The process of collective identity. In H. J. H. and B. Klandermans, editors, *Social Movements and Culture*, pages 104–130. University of Minnesota Press, 1995.

[172] D. T. Miller and M. Ross. Self-serving biases in the attribution of causality: Fact or fiction? *Psychological Bulletin*, 82:213–225, 1975.

[173] S. Mitra, A. Bagchi, and A. Bandyopadhyay. Design of a data model for social network applications. *J. Database Manag.*, 18:51–79, 10 2007.

[174] H. Mortveit and C. Reidys. *An Introduction to Sequential Dynamical Systems*. Springer, 2007.

[175] W. Muller. *How Then, Shall We Live? Four Simple Questions That Reveal the Beaty and Meaning of Our Lives*. Bantam Books, 1996.

[176] M. Munafo and G. Davey Smith. Repeating experiments is not enough. *Nature*, 553(7689):399–401, 1 2018.

[177] J. Nagel. *American Indian Ethnic Renewal: Red Power and the Resurgence of Identity and Culture*. Oxford University Press, New York, 1996.

[178] S. Newman. *Building Microservices*. O'Reilly, 2015.

[179] C. Nguyen, K. J. Schlesinger, and J. M. Carlson. Data-driven models for individual and group decision making. In *ASONAM*, pages 852–859, 2017.

[180] D. A. Nguyen, S. Tan, R. Ramanathan, and X. Yan. Analyzing information sharing strategies of users in online social networks. In *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 247–254, Aug 2016.

[181] N. P. Nguyen, G. Yan, et al. Containment of misinformation spread in online social networks. In *Web Science Conference*, pages 213–222, 2012.

[182] D. O'Callaghan et al. Uncovering the wider structure of extreme right communities spanning popular online networks. In *Web Sci*, 2013.

[183] J. O'Doherty and P. Bossaerts. Toward a mechanistic understanding of human decision making; contributions of functional neuroimaging. *Current Directions in Psychological Science*, 17:119–123, 2008.

[184] A. Oldenquist. Loyalties. *Journal of Philosophy*, 79:173–193, 1982.

[185] M. Olson. *The Logic of Collective Action: Public Goods and the Theory of Groups.* Harvard Univ. Press, Cambridge, Mass., 1965.

[186] Open Science Collaboration. Estimating the reproducibility of psychological science. *Science*, 349:943–943, 2015.

[187] S. Overbeek, B. Klievink, D. Hesketh, F. Heijmann, and Y.-H. Tan. A web-based data pipeline for compliance in international trade. In *Proceedings of the 1st Workshop on IT Innovations Enabling Seamless and Secure Supply Chains (WITNESS 2011) held in conjunction with the EGOV 2011 Conference, Delft, The Netherlands, August 29, 2011*, volume 769 of *CEUR Workshop Proceedings*, pages 32–48. Sun SITE Central Europe, Aachen, Germany, EU, 2011.

[188] T. J. Owens. Self and identity. In *Handbook of Social Psychology*, pages 205–232. 2006.

[189] C. Pahl and P. Jamshidi. Microservices: A systematic mapping study. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science - Volume 1 and 2 (CLOSER)*, pages 137–146, 2016.

[190] W. D. Paris, G. C. Budapest, H. D. D. Konstanz, C. G. Bristol, K. L. Oregon, and A. O. Stirling. An experimental investigation into the formation of intergroup representations. *European Journal of Social Psychology*, 2(2):202–204, 1972.

[191] K.-C. Pau, Y.-W. Si, and D. Marlon. Data warehouse model for audit trail analysis in workflows. In *Proceedings of the Student Workshop of IEEE International Conference on e-Business ENgineering*, ICEBE'07, 2007.

[192] A. Paxton, T. J. H. Morgan, J. W. Suchow, and T. Griffiths. Interpersonal coordination of perception and memoryin real-time online social experiments. In *In Proceedings of the 40th Annual Meeting of the Cognitive Science Society*, Austin, TX, USA, 2018.

[193] L. Peek. Becoming muslim: The development of a religious identity. *Sociology of Religion*, 66(3):215–242, 2005.

[194] C. Perdue, J. F. Dovidio, M. B. Gurtman, and R. B. Tyler. Us and them: Social categorization and the process of intergroup bias. 59:475–486, 09 1990.

[195] L. Pereira and F. Moreira. Majority-vote model on random graphs. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 71:016123, 02 2005.

[196] A. Pfandler, S. Rümmele, and S. Szeider. Backdoors to abduction. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1046–1052, 2013.

[197] C. S. Pierce. Elements of logic. In C. Hartshorn et al., editors, *Collected Papers of Charles Sanders Pierce*. 1931.

[198] A. Pilny, M. S. Poole, A. Reichelmann, and B. Klein. A structurational group decision-making perspective on the commons dilemma: results from an online public goods game. *Journal of Applied Communication Research*, 45(4):413–428, 2017.

[199] E. Plutzer and J. Zipp. Identity politics, partisanship, and voting for women candidates. *Public Opinion Quarterly*, 60(1):30–57, 3 1996.

[200] F. Polletta and J. M. Jasper. Collective identity and social movements. *Annual Review of Sociology*, 27:283–305, 2001.

[201] M. Qin, D. Jin, D. He, B. Gabrys, and K. Musial. Adaptive community detection incorporating topology and content in social networks. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2017*, ASONAM '17, pages 675–682, New York, NY, USA, 2017. ACM.

[202] Y. Ren, V. Cedeno-Mieles, Z. Hu, X. Deng, A. Adiga, C. Barrett, S. Ekanayake, B. J. Goode, G. Korkmaz, C. J. Kuhlman, D. Machi, M. V. Marathe, N. Ramakrishnan, S. S. Ravi, P. Sarat, N. Selt, N. Contractor, J. Epstein, and M. W. Macy. Generative modeling of human behavior and social interactions using abductive analysis. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 413–420, Aug 2018.

[203] F. Rioux, F. Bernier, and D. Laurendeau. Design and implementation of an xml-based, technology-unified data pipeline for interactive simulation. In *Winter Simulation Conference*, pages 1130–1138, 2008.

[204] D. Rousseau and A. M. van der Veen. The emergence of a shared identity: An agent-based computer simulation of idea diffusion. *Journal of Conflict Resolution*, 49(5):686–712, 2005.

[205] D. G. Russell and I. G. Sarason. Test anxiety, sex, and experimental conditions in relation to anagram solution. *Journal of Personality and Social Psychology*, 1:493–496, 1965.

[206] T. Salah, M. J. Zemerly, C. Y. Yeun, M. AI-Qutayri, and Y. AI-Hammadi. The evolution of distributed systems towards microservices architecture. In *The 11th International Conference for Internet Technology and Secured Transactions (ICITST)*, pages 318–325, 2016.

[207] M. J. Salganik and D. J. Watts. Web-based experiments for the study of collective social dynamics in cultural markets. *topiCS*, 1(3):439–468, 2009.

[208] I. G. Sarason. Test anxiety and cognitive modeling. *Journal of Personality and Social Psychology*, 28:58–61, 1973.

[209] I. G. Sarason. Test anxiety and social influence. *Journal of Personality*, 41:261–271, 1973.

[210] T. C. Schelling. *Micromotives and Macrobehavior*. Norton, 2006.

[211] J. W. Schooler. Metascience could rescue the 'replication crisis'. *Nature*, 515:9–9, 2014.

[212] T. W. Schubert and S. Otten. Overlap of self, ingroup, and outgroup: Pictorial measures of self-categorization. *Self and Identity*, 1(4):353–376, 2002.

[213] M. Schweitzer, L. Ordonez, and B. Dumaz. Goal-setting as a motivator of unethical behavior. *Academy of Management Journal*, 47:422–433, 2004.

[214] C. Sell and I. Braun. Using a workflow management system to manage emergency plans. 04 2009.

[215] R. Sethi and E. Somanathan. A simple model of collective action. *Economic Development and Cultural Change*, 54(3):725–747, 2006.

[216] M. Shanahan. Perception as abduction: Turning sensor data into meaningful representation. *Cognitive Science*, 29:103–134, 2005.

[217] D. B. Shank, Y. Kashima, S. Saber, et al. Dilemma of dilemmas: How collective and individual perspectives can clarify the size dilemma in voluntary linear public goods dilemmas. *PLOS ONE*, 10:1–19, 03 2015.

[218] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[219] A. Shepherd, S. Rauch, C. Schloer, D. Kinkade, H. Ake, M. Biddle, N. Copley, M. Saito, P. Wiebe, and A. York. Towards Capturing Data Curation Provenance using Frictionless Data Package Pipelines, Oct. 2018.

[220] A. Silke. Holy warriors: Exploring the psychological processes of jihadi radicalization. *European Journal of Criminology*, 5(1):99–123, 2008.

[221] P. Singla and R. J. Mooney. Abductive markov logic for plan recognition. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[222] R. O. Sinnott and S. Hussain. Security-oriented workflows for the social sciences. In *International Conference on Network and System Security*, pages 152–159, 2010.

[223] K. Smith, O. Feher, and J. Culbertson. The influence of word-order harmony on structural priming in artificial languages. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society, CogSci 2017, London, UK, 16-29 July 2017*, 2017.

[224] D. Snow. Collective identity and expressive forms. In N. J. Smelser and P. B. Baltes, editors, *International Encyclopedia of the Social & Behavioral Sciences*, pages 2212–2219. Elsevier, 2001.

[225] D. Snow and D. McAdams. Identity work processes in the context of social movements: Clarifying the identity/movement nexus. In S. Stryker, T. Owens, and R. W. White, editors, *Self, Identity and Social Movements*, pages 2212–2219. University of Minneapolis Press, 2000.

[226] S. Spaulding et al. A social robot system for modeling children's word pronunciation: Socially interactive agents track. In *AAMAS*, 2018.

[227] C. R. Stones. Self-determination and attribution of responsibility: Another look. *Psychological Reports*, 53:391–394, 1983.

[228] C. T. Stout, K. Kretschmer, and L. Ruppanner. Gender linked fate, race/ethnicity, and the marriage gap in american politics. *Political Research Quarterly*, 70(3):509–522, 2017.

[229] S. Stryker. *Symbolic interactionism: A social structural version*. Benjamin/Cummings, 1980.

[230] J. Stubbs, W. Moreira, and R. Dooley. Distributed systems of microservices using docker and serfnode. In *7th International Workshop on Science Gateways*, pages 34–39, 2015.

[231] S. X. Sun and J. L. Zhao. Formal workflow design analytics using data flow modeling. *Decision Support Systems*, 55(1):270 – 283, 2013.

[232] S. Suri and D. J. Watts. Cooperation and contagion in web-based, networked public goods experiments. *PLOS ONE*, 6:1–18, 03 2011.

[233] C. Sutton and A. McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2011.

[234] W. B. Swann, A. Gomez, J. F. Dovidio, S. Hart, and J. Jetten. Dying and killing for one's group: Identity fusion moderates responses to intergroup versions of the trolley problem. *Psychological Science*, pages 1176–1183, 2010.

[235] M. S. Swanson. *Composing Democracy: Collective Identity Formation in Small Group Composition*. PhD thesis, University of Washington, 2015.

[236] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:257–285, 1988.

[237] D. Taibi, V. Lenarduzzi, C. Pahl, and A. Janes. Microservices in agile software development: a workshop-based study into issues, advantages, and disadvantages. pages 1–5, 05 2017.

[238] H. Tajfel. Social identity and intergroup behavior. *Social Science Information*, 13:65–93, 1974.

[239] Y. Tanaka, T. Iwata, T. Kurashima, H. Toda, and N. Ueda. Estimating latent people flow without tracking individuals. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 3556–3563. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[240] S. Tarrow. Mentalities, political cultures, and collective action frames: constructing meanings through action. *Sociology Compass*, 4:174–202, 2010.

[241] B. D. Tatum. *Why Are All the Black Kids Sitting Together in the Cafeteria?: And Other Conversations About Race*. Basic Books, 2 edition, 2003.

[242] V. Taylor and N. E. Whittier. Collective identity in social movement communities: lesbian feminist mobilization. In A. D. Morris and C. M. Mueller, editors, *Frontiers of Social Movement Theory*, pages 104–130. Yale University Press, 1992.

[243] P. Thagard. Explanatory coherence. *Behavioral and Brain Sciences*, 12:435–502, 1989.

[244] R. H. Thaler. *Misbehaving: The Making of Behavioral Economics*. W. W. Norton & Company, 2016.

[245] S. Timmermans. Social death as a self-fulfilling prophecy: David Sudnow's 'passing on' revisited. *Sociological Quarterly*, 39:453–472, 1999.

[246] S. Timmermans and I. Tavory. Theory construction in qualitative research: From grounded theory to abductive analysis. *Sociological Theory*, 30:167–186, 2012.

[247] R. Toivonen, L. Kovanen, M. Kivelä, J.-P. Onnela, J. Saramäki, and K. K. Kaski. A comparative study of social network models: Network evolution models and nodal attribute models. *Social Networks*, 31:240–254, 2009.

[248] P. D. Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame. Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35:316–319, 2017.

[249] T. Tran and K. Lee. Understanding citizen reactions and ebola-related information propagation on social media. In *Proceedings of the 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '16, pages 106–111, Piscataway, NJ, USA, 2016. IEEE Press.

[250] M. E. Tresselt. Reexamination of anagram problem solving. *Transactions of the New York Academy of Sciences*, 30:1112–1119, 1968.

[251] K. J. Turner and P. S. Lambert. Workflows for quantitative data analysis in the social sciences. *Int. J. Softw. Tools Technol. Transf.*, 17:321–338, 2015.

[252] J. D. Ullman and J. Widom. *A First Course in Database Systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[253] D.-J. van der Zee and B. Holkenborg. Conceptual modelling for simulation-based serious gaming. In *Winter Simulation Conference*, pages 522–534, 2010.

[254] T. A. van Dijk. *Ideology: a multidisciplinary approach*. SAGE Publications, Ltd, 2000.

[255] P.-P. van Maanen and B. van der Vecht. An agent-based approach to modeling online social influence. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, ASONAM '13, pages 600–607, New York, NY, USA, 2013. ACM.

[256] M. van Zomeren, T. Postmes, and R. Spears. Toward an integrative social identity model of collective action: A quantitative research synthesis of three socio-psychological perspectives, 2008.

[257] R. J. Vance and A. Colella. Effects of two types of feedback on goal acceptance. *Journal of Applied Psychology*, 75:68–77, 1990.

[258] F. Vanderhaegen and P. Caulier. A multi-viewpoint system to support abductive reasoning. *Information Sciences*, 181:5349–5363, 2011.

[259] M. Verkuyten and A. A. Yildiz. National (dis)identification and ethnic and religious identity: A study among turkish-dutch muslims. *Personality and Social Psychology Bulletin*, 33(10):1448–1462, 2007.

[260] J. Vivian, A. A. Rao, F. A. Nothaft, C. Ketchum, J. Armstrong, et al. Toil enables reproducible, open source, big biomedical data analyses. *Nature Biotechnology*, 35:314–316, 2017.

[261] K. D. Vryan, E. A. Adler, and P. Adler. Identity. In L. T. Reynolds and N. J. Herman-Kinney, editors, *Handbook of Symbolic Interactionism*, pages 367–390. Altamira Press, 2003.

[262] G. Wang and B. Peng. Script of scripts: A pragmatic workflow system for daily computational research. *PLoS Computational Biology*, pages e1006843–1–e1006843–14, 2019.

[263] M. W. Warren and W. J. Thomson. Anagram solution as a function of transition probabilities and solution word frequency. *Psychonomic Science*, 17(6):333–334, Dec 1969.

[264] D. J. Watts. Should social science be more solution-oriented? *Nat. Hum. Behav.*, pages 1–5, 2017.

[265] F. Wei-Kleiner, Z. Dragisic, and P. Lambrix. Abduction framework for repairing incomplete el ontologies: Complexity results and algorithms (extended version). In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[266] A. Wendt. Collective identity formation and the international state. *American Political Science Review*, 88(2):384–396, 1994.

[267] M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. T. Foster. Swift: A language for distributed parallel scripting. *Parallel Computing*, 37:633–652, 2011.

[268] J. William B. Swann, A. Gomez, M. D. Buhrmester, L. Lopez-Rodriguez, J. Jimenez, and A. Vazquez. Contemplating the ultimate sacrifice: identity fusion channels pro-group affect, cognition, and moral decision making. *Journal of Personality and Social Psychology*, pages 713–727, 2014.

[269] J. William B. Swann, A. Gomez, et al. Identity fusion and self-sacrifice: Arousal as a catalyst of pro-group fighting, dying, and helping behavior. *J. Pers. and Soc. Psych.*, 2010.

[270] J. William B. Swann, A. Gomez, D. C. Seyle, J. F. Morales, and C. Huici. Identity fusion: The interplay of personal and social identities in extreme group behavior. *Journal of Personality and Social Psychology*, pages 955–1011, 2009.

[271] K. Wolstencroft, R. Haines, D. Fellows, A. Williams, et al. The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research*, page gkt328, 2013.

[272] S. Worchel, V. A. Andreoli, and R. Folger. Intergroup cooperation and intergroup attraction: The effect of previous interaction and outcome of combined effort. *Journal of Experimental Social Psychology*, 13(2):131 – 140, 1977.

[273] J. M. Wozniak, T. G. Armstrong, M. Wilde, D. S. Katz, E. Lusk, and I. T. Foster. Swift/t: Scalable data flow programming for many-task applications. In *ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 309–310, 2013.

[274] J. Y. Lee and G. J. Collins. On using ilities of non-functional properties for subsystems and components. *Systems*, 5:47, 07 2017.

[275] Z. Yao, J. Barrick, Z. Weinberg, et al. A computational pipeline for high- throughput discovery of cis-regulatory noncoding rna in prokaryotes. *PLOS Computational Biology*, pages 1–12, 2007.

[276] H. Zhang and Y. Vorobeychik. Empirically grounded agent-based models of innovation diffusion: a critical review. *Artificial Intelligence Review*, pages 1–35, 2017.

[277] H. Zhang, Y. Vorobeychik, J. Letchford, and K. Lakkaraju. Data-driven agent-based modeling, with application to rooftop solar adoption. *Autonomous Agents and Multi-Agent Systems*, 30:1023–1049, 2016.

[278] Z. Zhao, M. Madaio, et al. Socially-conditioned task reasoning for a virtual tutoring agent. In *AAMAS*, pages 2265–2267, 2018.

Appendix A

Appendix: Pipelines

A.1 Data Common Specification

This appendix provides a concrete view into the system. The definition of a data common specification in Figure 2.1 provides the bridge between the abstract data model and the implementation of the pipelines; see Figure 2.3. JSON schemas provide a detailed specific view of the implementation aspect of our pipelines. Because we go into detail, this is an exemplar for other types of problems. These are the types of files we use in the case studies in Section 2.8.

Figure A.1 shows the "Experiment" definition. Figure A.2 shows the "Phase" definition. Figure A.3 shows the "Phase Description" definition. Figure A.4 shows the "Player" definition. Figure A.5 shows the "Action" definition.

Table A.1: Data Common Specification.

#	**Component Name**	**Parameter from Data Model (Table 2.1)**	**Table in Data Model UML (Figure 2.4)**	**Description**
1	Experiment	Experiment Schema	Experiment Schema	Experiment description and definition of initial parameters (i.e., experiment id, number of phasers, number of players, begin time, duration and list of players).
2	Phase	Phase Schema: Phase schema id, Sequence, Phase Begin, Phase duration, Unit of time, Network definition, Meaning of an edge.	Phase Schema	An experiment can have many phases. This is the Phase description and definition of initial parameters (i.e., phase id, order in experiment, begin time, duration and list of players, connections between players, number of players).
3	Phase Description	Phase Schema: Node attributes for a phase, Edge attributes for a phase, Initial conditions for nodes, Initial conditions for edges.	Edge, Initial Conditions Edge, Edge Attributes, Initial Conditions Node.	A phase has a description (i.e., phase id, beginning parameters, end parameters, actions, relations between actions).
4	Player	Experiment Schema: Player id.	Player.	Player description (i.e., player id, experiment id, phase id).
5	Action	Phase Schema: Action set, Action sequence.	Action, Action Tuple	A experiment, a phase and players have actions associated with them (i.e., action id, phase id, action tuple id, player id, timestamp, payload).

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Experiments",
    "description": "Experiment initial parameters",
    "type": "array",
    "items" : {
    "type": "object",
    "properties": {
        "experimentid": {
            "description": "The unique identifier for an experiment",
            "type": "integer"
        },
        "p": {
            "description": "Number of phases in experiment",
            "type": "integer"
        },
        "n": {
            "description": "Number of players in experiments",
            "type": "integer"
        },
        "startime": {
            "description": "Date and time of experiment",
            "type": "string"
        },
        "duration": {
            "description": "Experiment duration in minutes",
            "type": "number"
        },
        "activeplayers" : {
          "type" : "array",
          "items" : {
              "type" : "object",
              "properties" : {
                   "playerid": {
                   "description": "Active player in experiment",
                   "type": "string"
                   }
               }
          }
       }
    },
    "required": ["experimentid","p","n"]
    }
}
```

Figure A.1: JSON schema for the "Experiment" of the Data common specification.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Phases",
    "description": "Phases in an experiment",
    "type": "array",
    "items" : {
    "type": "object",
    "properties": {
        "phaseid": {
            "description": "The unique identifier for a phase","type": "string"
        },
        "phasedescriptionid": {
            "description": "The phase description id","type": "string"
        },
        "experimentid": {
            "description": "Phase experiment id","type": "integer"
        },
        "phaseorder": {
            "description": "Order of phase in experiment","type": "integer"
        },
        "begin": {
            "description": "Start time of a phase in experiment","type": "number"
        },
        "duration": {
            "description": "Duration of a phase in seconds","type": "integer"
        },
        "d": {
            "description": "Connections between players","type": "integer"
        },
        "n": {
            "description": "Number of players","type": "integer"
        }
    },
    "required": ["phaseid","phasedescriptionid","experimentid",
                "phaseorder","begin","duration"]
    }
}
```


Figure A.2: JSON schema for the “Phase” of the Data common specification.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Phases",
    "description": "Phases in an experiment",
    "type": "array",
    "items" : {
    "type": "object",
    "properties": {
        "phaseid": {"description": "The unique identifier for a phase","type": "string"},
        "beginparameters" : {
          "type" : "array","items" : {"type" : "object",
               "properties" : {
                    "parameter": {
                    "description": "Beginning parameters in phase","type": "string"}}},
        "endparameters" : {
          "type" : "array","items" : {"type" : "object",
               "properties" : {
                    "parameter": {
                    "description": "End parameters in phase","type": "string"}}},
        "actions" : {
          "type" : "array","items" : {"type" : "object",
               "properties" : {
                    "action": {
                    "description": "Actions in phase","type": "string"}}},
         "synchronousactions" : {"type" : "array","items" : {"type" : "object",
               "properties" : {
                    "actionrequest": {
                    "description": "Action request id","type": "string"},
                    "actionreply": {"description": "Action reply id",
                    "type": "string"}}}},
        "features" : {"type" : "array",
           "items" : {"type" : "object",
               "properties" : {"feature": {
                    "description": "feature vector for models","type": "string"}}}},
    "required": ["phaseid"]
    }
}
```

Figure A.3: JSON schema for the "Phase Description" of the Data common specification.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Players",
    "description": "Player phases parameters",
    "type": "array",
    "items" : {
    "type": "object",
    "properties": {
        "phaseid": {
            "description": "The unique identifier for a phase",
            "type": "string"
        },
        "playerid": {
            "description": "Player id",
            "type": "string"
        },
        "neighbors" : {
          "type" : "array",
          "items" : {
              "type" : "object",
              "properties" : {
                   "neighbor": {
                   "description": "Neighbor of a player",
                   "type": "string"
                   }
               }
           }
         },
         "beginparameters" : {
          "type" : "array",
          "items" : {
              "type" : "object",
              "properties" : {
                   "parameter": {
                   "description": "Beginning parameter in phase",
                   "type": "string"
                   },
                   "value": {
                   "description": "Beginning parameter value",
                   "type": "string"
                   }
               }
           }
         },
        "endparameters" : {
          "type" : "array",
          "items" : {
              "type" : "object",
              "properties" : {
                   "parameter": {
                   "description": "End parameter in phase",
                   "type": "string"
                   },
                   "value": {
                   "description": "End parameter value",
                   "type": "string"
                   }
               }
           }
         }
    },
    "required": ["playerid","phaseid"]
    }
}
```


Figure A.4: JSON schema for the “Player” of the Data common specification.

```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "title": "Actions",
    "description": "Actions during an experiment phase",
    "type": "array",
    "items" : {
    "type": "object",
    "properties": {
        "phaseid": {
            "description": "The unique identifier for a phase",
            "type": "string"
        },
        "n": {
            "description": "Number of players in phase",
            "type": "integer"
        },
        "d": {
            "description": "Number of connections between players",
            "type": "integer"
        },
         "actionlist" : {
           "type" : "array",
           "items" : {
               "type" : "object",
               "properties" : {
                    "player1": {
                    "description": "Player that initiates the action",
                    "type": "string"
                    },
                    "player2": {
                    "description": "Player that receives the action",
                    "type": "string"
                    },
                    "actionid": {
                    "description": "action id",
                    "type": "string"
                    },
                    "playerActionSeqid": {
                    "description": "Player unique action sequence",
                    "type": "integer"
                    },
                    "timestamp": {
                     "description": "Timestamp of action",
                     "type": "number"
                    },
                    "payload": {
                    "description": "payload",
                    "type": "string"
                     }
                },
                "required": ["player1","actionid","playerActionSeqid","timestamp","payload"]
            }
         }
    },
    "required": ["phaseid","n","d"]
    }
}
```


Figure A.5: JSON schema for the “Action” of the Data common specification.

A.2 Appendix: Mapping of Model onto the Software System

In this appendix, we describe the characteristics of the the implementation of an individual pipeline. Figure A.11 shows an example of a Configuration Input file JSON schema describing how to execute up to five functions in a pipeline.

```
"experiment":
    {
    "type": "string",
    "description":"Experiments description file"
    },
```

Figure A.6: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This figure shows a portion of the schema for a configuration file that specifies the experiment JSON schema file location.

```
"phasedesc":
    {
    "type": "string",
    "description":"Phases description file"
    },
```

Figure A.7: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the phase description JSON schema file location.

```
"phase":
    {
    "type": "string",
    "description":"Phases registration file"
    },
```

Figure A.8: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the phase JSON schema file location.

```
"action":
    {
    "type": "string",
    "description":"Actions registration file"
    },
```

Figure A.9: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the action description JSON schema file location.

```
"player":
    {
    "type": "string",
    "description":"Players parameters registration file"
    },
```

Figure A.10: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. This Figure shows a portion of the schema for a configuration file that specifies the player description JSON schema file location.

```
"functions": {
    "type": "array",
    "description": "Functions to be run for this pipeline",
    "minItems": 1,
    "items": {
        "type": "object",
        "properties": {
            "function": {
                "description": "Pipeline functions to execute",
                "type": "string",
                "enum": ["h1","h2","h3","h4","h5"]},
            "actionId": {
                "type": "string",
                "description": "Required for h2, h3, h4, h5"},
            "windowSize": {
                "type": "integer",
                "description": "Required for h3, h4"},
            "bin": {
                "type": "integer",
                "description": "Required for h5"}
        },
        "required": ["function"]
    }
}
```

Figure A.11: To run a pipeline (called a job), a configuration input file specifies functions and their order of execution. In this configuration file there are five possible functions that can be executed in any order. This Figure shows a portion of the schema for a configuration file that specifies how to compose and execute one or more functions of a simple pipeline. For example, here it defines that a parameter called "actionId" is only necessary for functions h_2 through h_5.

A.3 Appendix: Examples of the Software System

This Appendix shows examples of input files for the Experimental Data Transformation Pipeline (Figure A.12), and the Data Analytics Pipeline (Figure A.13). Here we show how a function is executed in a generic pipeline. Input files are validated against their corresponding JSON schema. If necessary, file contents are transformed (possibly outputs from upstream functions) to obtain the direct input for a function in the correct format. After verification of formats by the corresponding JSON schemas, the function is executed and output files are generated (these digital object outputs may be, e.g., plot files, ASCII data files, and binary data files).

Figure A.12 shows an example of the (1) Experimental Data Transformation Pipeline input files and the transformations they go through. Here, the function h_1 takes experimental raw data and transforms it to our Data Common Specification. CSV files are transformed into JSON files, then verified for input before executing function h_1. After execution, function h_1 outputs JSON schemas that become inputs for the Data Analytics Pipeline.

Figure A.13 shows an example of the (2) Data Analytics Pipeline execution of function h_7 with configuration input files examples. Here, the input JSON files are verified, then transformed into function h_7 direct input. After verifying the input for the function, h_7 is executed and the output files returned. In this example, the output file is an input for the (3) Property Inference pipeline.

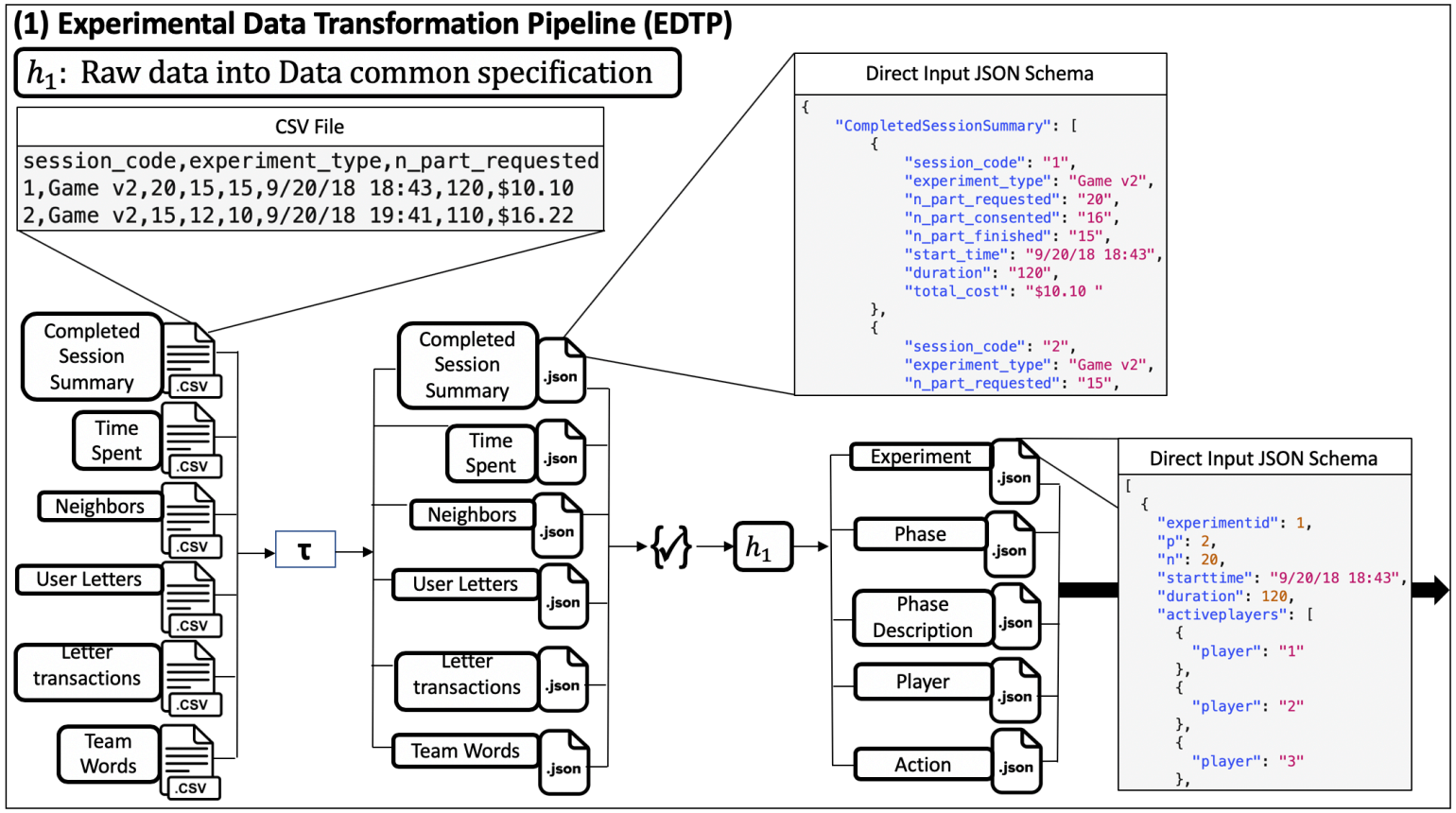


Figure A.12: This is an example of the (1) Experimental Data Transformation pipeline execution to transform raw experimental data into the data common specification. Here we show how function h_1 is executed. Here we show an input CSV file as an example for the "Completed Session Summary" input file. If necessary, file contents are transformed to obtain the direct input for a function in the correct format. Here we show how the "Completed Session Summary" CSV input file is transformed into a "Completed Session Summary" json file that becomes the input for the function. After verification of formats by the corresponding JSON schemas, the function is executed and output files are generated. Here we show the output json file for the "Experiment" data common specification.

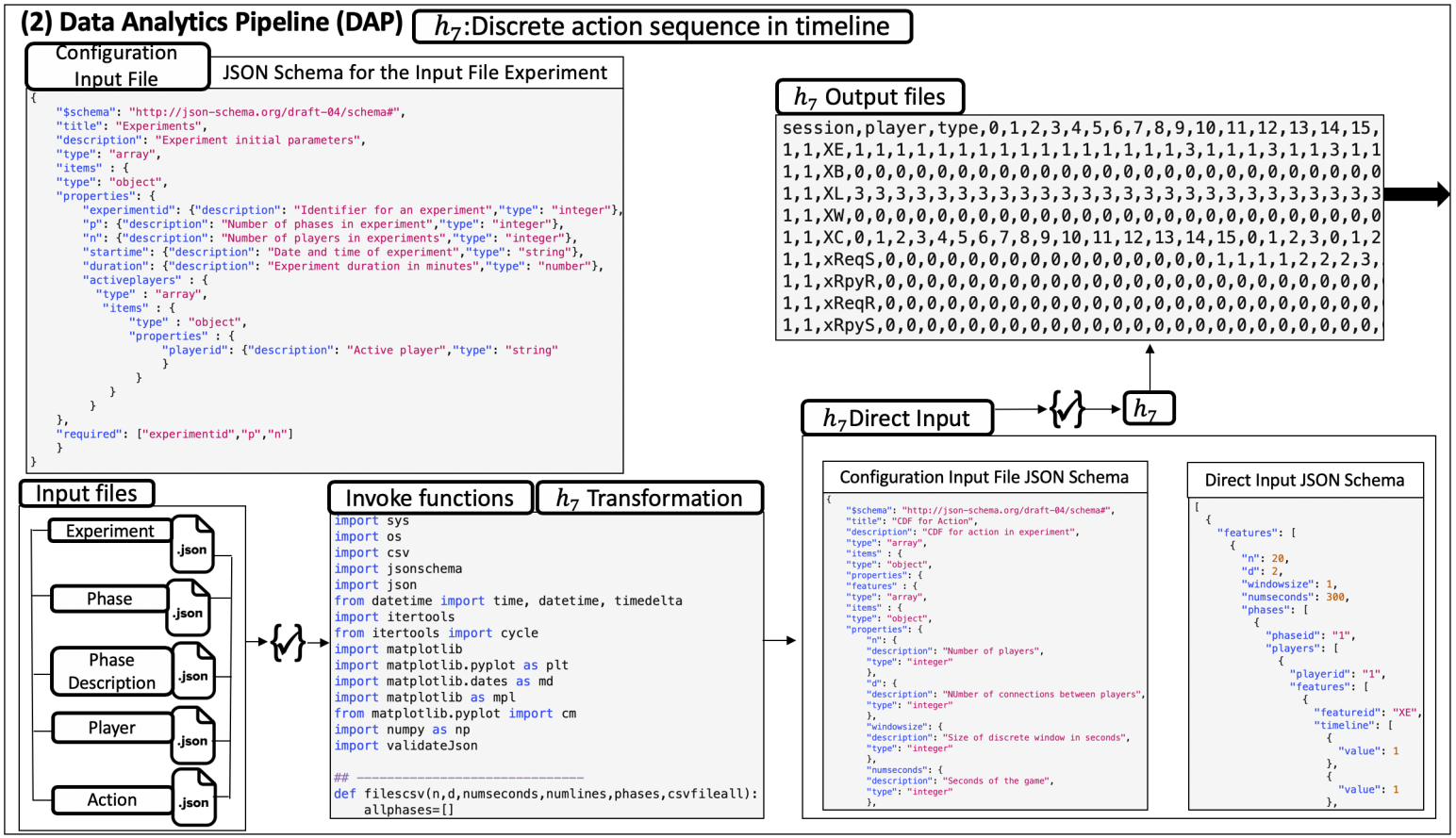


Figure A.13: This is an example of the (2) Data Analytics pipeline execution to analyze files of data in the common specification. Here we show how function h_7 is executed. Input files are validated against their corresponding JSON schema. Here we show an example of a json schema file for the "Experiment" description input file. Figure A.1 contains the whole file. After verification of formats by the corresponding JSON schemas, if necessary, file contents are transformed to obtain the direct input for a function in the correct format. After verification of formats by the corresponding JSON schemas, function h_7 is executed and output files are generated. In this example the output file is an input for the (3) Property Inference pipeline.

A.4 Appendix: Pipeline Functions

In this Appendix, we describe the characteristics of the the atomic element of a pipeline: the function. If a new component is added to the pipeline, it is introduced by a new function. We provide a listing of types of functions as microservices within each of the five pipelines. We show five tables, one for each pipeline, with a list of available functions. Table A.2 shows one function for the (1) Experimental Data Transformation Pipeline (EDTP). Table A.3 shows fourteen functions for the (2) Data Analytics Pipeline (DAP) Table A.4 shows four functions for the (3) Property Inference Pipeline (PIP). Table A.5 shows five functions for the (4) Modeling and Simulation Pipeline (MASP). Table A.6 shows five functions for the (5) Model Evaluation and Prediction pipeline (MEAPP).

The functions provide a range of capabilities from simple plotting routines to cleaning and organizing, storing and accessing data sets, and inferring properties and running simulations. Users may add other functions and continue community-based development, as these functions are not exhaustive. Each function completes one well-defined task. Many of these functions can be used in multiple contexts; functions use the pipeline as a universal interface. For example, the action progression function h_3 of the Data Analytics Pipeline generates a plot of the number of actions a_i per player in time $\forall a_i \in A$. Also, often a function represents a category of operation; e.g., there are six different agent-based models (ABMs) under h_1 of the Modeling and Simulation Pipeline. Currently, functions are written in the following Programming Languages (PLs) C++, Python, and R.

Table A.2: Listing of types of functions as microservices for the (1) Experimental Data Transformation Pipeline (EDTP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model.

Pipeline: Experimental Data Transformation (EDTP)				
#	**Name**	**Description**	**Significance**	**Output type**
h_1	Raw data into Data common specification	Transform experimental raw data into our data common specification.	This is the only way an experiment data can go through our pipelines.	Data files

Table A.3: Listing of types of functions as microservices for the (2) Data Analytics Pipeline (DAP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model.

Pipeline: Data Analytics (DAP)				
#	**Name**	**Description**	**Significance**	**Output type**
h_1	Player interactions	Generate a timeline of individual and between-players actions. Each player represents a lane. Each action has a unique color.	Detect common patterns between players and actions.	Visualization
h_2	Timestamp Delta between related actions	Construct a visualization of the timestamp delta between related actions. A request action has a correspondent receive action. Each request action represents a lane, a horizontal line represents the length of time it takes to receive a requested action.	Detect bursts in types of actions. Detect time patterns in types of actions.	Visualization
h_3	Action progression	Generate a cumulative distribution plot for an action, by player.	Show how an action progresses in time during an experiment phase.	Data files and plot
h_4	Average action	Generate plot of the average number of actions between players in a window size s.	Show how an average action progresses in time between experiments phases.	Data files and plot
h_5	Action histogram	Generate a histogram of timestamps of an action.	Compare histograms among all experiment phases.	Data files and plot
h_6	Histogram of related actions	Generate a histogram of timestamp delta between related actions.	Compare histograms between all experiments phases.	Data files and plot
h_7	Discrete action sequence in timeline	Generate a discrete-time action sequence by phase. Each action, from the action set A has a unique id definition.	Generate input for the Property Inference pipeline.	Time series data files
h_8	Summary of actions.	Generate for each unique action the number of occurrences at the end of a phase, and the number of occurrences at the end of all experiments in the pipeline run.	Compare action occurrences among all experiments.	Data files
h_9	Player categories.	Categorize players by performance in each action.	Analyze player performance by clustering them in categories.	Data files and plot
h_{10}	Actions heat-map.	Generate heat-map by player for actions in a phase.	Analyze player performance by a heat-map visualization.	Data files and plot
h_{11}	Summary of related actions.	Generate a summary at the end of a phase with the possible actions between neighbors and the occurred actions.	Compare related action occurrences among all experiments.	Data files
h_{12}	Distance between actions.	Generate a file with distance between two actions. The distance has to be provided by the analyst (e.g, for the action of forming a word, the Levenshtein distance between two words formed).	Compare action characteristics in an experiment.	Data files
h_{13}	Rank of actions.	Generate a file with rank of an action. The rank has to be provided by the analyst (e.g, for the action of requesting a letter, the letter rank comes from a specified list).	Compare action characteristics in an experiment.	Data files
h_{14}	Score of actions.	Generate a file with a score of an action. The method to calculate the score has to be provided by the analyst (e.g, for the action of forming a word, the scrabble score for a word formed).	Compare action characteristics in an experiment.	Data files

Table A.4: Listing of types of functions as microservices for the (3) Property Inference Pipeline (PIP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model.

Pipeline: Property Inference (PIP)				
#	**Name**	**Description**	**Significance**	**Output type**
h_1	Properties for Markovian transition matrix.	Use of the sequences of discrete actions to generate the probability of transition from an action a_i to an action a_j as measured in the experiment data.	Generates the properties for a Markovian transition matrix.	Data files
h_2	Properties for an adapted CRF model	Use of the sequences of discrete actions to generate a derived feature vector accounting for history effects, where the vector corresponds to the discrete-time sequences from the Data Analytics h_7 output.	Generates properties for an adapted conditional random fields (CRF) model.	Data files
h_3	Coefficients in a hierarchical model	Generalize the model to take the number of neighbors into consideration, and also digest the additional experiment data where player degree increases or decreases.	Generate coefficients in a hierarchical model to augment the CRF model.	Data files
h_4	Multilinear regression model	Construct multilinear regression model on action set A.	Generate structure of the model and parameter values	Data files

Table A.5: Listing of types of functions as microservices for the (4) Modeling and Simulation Pipeline (MASP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model.

Pipeline: Modeling and Simulation (MASP)				
#	**Name**	**Description**	**Significance**	**Output type**
h_1	Agent based model (ABM)	Execute agent based simulation models. Currently, six different models (stationary, dynamic conditional random fields (CRF).	Generate Agent Based Model Simulations outputs for self-consistency checks and predictions.	Data files
h_2	Statistical regression	Compute a relation between selected and observed values.	Predict most probable value of the observed values for any selected values.	Data files
h_3	Statistical regression	Regression equation that uses results from Phase 1 ABM to predict the Phase 2 DIFI.	Predict the Phase 2 DIFI (i.e., DIFI2) score per player.	Data files
h_4	Statistical regression	Regression equation that uses results from ABM Phases to predict the Publics Good Game Contributions in the corresponding Phase.	Predict the Publics Good Game Contributions per player.	Data files
h_5	Component model prediction	Execute agent based simulation component models to compare outputs with real actions from a real experiment.	Compare outputs between models.	Data files

Table A.6: Listing of types of functions as microservices for the (5) Model Evaluation and Prediction pipeline (MEAPP). Many functions may be considered as collections of functions because they can handle multiple types of data through the data model.

Pipeline: Model Evaluation and Prediction (MEAPP)				
#	**Name**	**Description**	**Significance**	**Output type**
h_1	Model Validation	Compares experiment outputs with simulation outputs.	Demonstrate that the model is a reasonable representation of the actual system.	Data files and plot
h_2	Model Prediction	Generates statistical models to predict outcomes.	Forecast outcomes in an experiment.	Data files and plot
h_3	Model Fusion	Generates model to predict outcomes by combining outputs from different models.	Predict the Phase 2 DIFI (i.e., DIFI2) score per player.	Data files
h_4	Model Evaluation	Generates R-squared values by comparing experiment outputs with simulation outputs.	R-squared is a statistical measure of how close the data are to the fitted regression line.	Data files
h_5	Cross-Validation	The original experiment sample is randomly partitioned into k equal size subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining k-1 subsamples are used as training data. The cross-validation process is then repeated k times, with each of the k subsamples used exactly once as the validation data.	Demonstrate that the model is a reasonable representation of the actual system. All observations are used for both training and validation, and each observation is used for validation exactly once.	Data files

A.5 Appendix: Microservices

A.5.1 Characteristics

We provide a compact description of microservices [49, 152, 153, 178, 206, 237]. While there is no universally accepted of what a microservice is, we take the term to have the following features;

1. Autonomous (isolated, simple entity): a microservice is a separate entity. Although isolated services can add overhead, the resulting simplicity is worth it. This is analogous to the trade-offs between a distributed system and a shared memory system.

2. Smallness: the code for a microservice can be rewritten (constructed, tested, verified, documented) in two weeks. Often, they are less than 100 lines of code.

3. Smallness: people tend to have good intuition when a code base is too large; so sufficiently small is when this intuition does not hint at being too large.

4. Smallness: if the code base is too large to be managed by a small team, then it is not small enough.

5. Interdependence: there should be interdependence among a collection of services. As services get smaller, the benefits of interdependence increase. But smaller services create complexity (the "edges" between services). But teams should learn to handle this complexity.

6. Communication among services: all cooperation among services is through network calls (versus direct invocation) to avoid tight coupling.

7. Change/upgrade: all microservices should be capable of changing independent of other microservices. In practice, this can be hard to do it, for example, a collection of services depend on lower level infrasctructure.

8. Independent deployment: each microservice should be deployable, independent of all others.

9. Weary of Sharing Capability Between Services: the more multiple microservices share, the more services become coupled to internal representations and decreases autonomy.

10. APIs (Application Programming Interfaces): specify/select/prefer technology-agnostic APIs so that the services are not constrained by technology. Achieve decoupling: the success of the "Change/upgrade" feature is an evaluation of decoupling success. Decoupling also requires good models.

A.5.2 Benefits

Many of the benefits of microservices stem from their isolated, independent scope [49, 152, 153, 178, 206, 237].

1. Technology heterogeneity, including technology stacks, across microservices.
2. Technology changeout.
3. Technology evaluation in a controlled, limited way.
4. Easier to isolate problems and failures.
5. Scale-up can be focused to particular services. So, too, with on-demand provisioning.
6. Deployments/redeployments can be isolated to particular microservices. Smaller increments of (re)deployment means reducing the possibility of adverse ripple effects.
7. Improvements/new versions are eased in and old versions are eased out.
8. Smaller services translates to smaller teams.
9. Composability, reuse.
10. Choices to throw away code are made more easily (less ownership, less cost of construction).
11. Easier unit testing (generating, executing, and interpreting tests). For example, there are fewer paths through the code.

A.5.3 Microservices as a Type of Service Oriented Architecture

Pipelines are intimately tied to microservices. While microservices may be used individually, typically, the small scope and limited features (or one feature) per service implies that they must be composed to accomplish many tasks. This composition can be accomplished with pipelines. This is not necessarily true with larger, more monolithic service oriented architectures (SOAs): these may provide broader-scope services within one module.

Microservices are one type of service oriented architecture (SOA). One example of the difference between the two is that microservices generally tend to avoid shared libraries that are used across microservices. This is because use of shared libraries means increased coupling of services. Based on the authors' experiences, this difference between microservices and SOAs in general is analogous to the difference between shared memory multi-process systems versus distributed systems, as described next.

By multi-process *shared memory* systems, we mean a software system that is composed of multiple processes that run asynchronously and use shared memory to exchange information (e.g., no message passing). In this environment, the processes are tightly coupled because if one process requires changes in shared storage structures, these will affect all other processes that use those storage structures. That is, the software for these other processes needs to be changed, too, leading to increased maintenance. Hence, there are a lot of interdependencies. However, in an asynchronous *distributed system*, each process has its own storage structures and memory, so that changes in storage structures for one process has no effect on other processes. While it is the case that additional infrastructure is required for distributed systems (e.g., for message passing), this additional requirement is offset by the autonomy realized for each process. The analogy here is that a multi-process shared memory system is a classic SOA, while microservices are the distributed system.

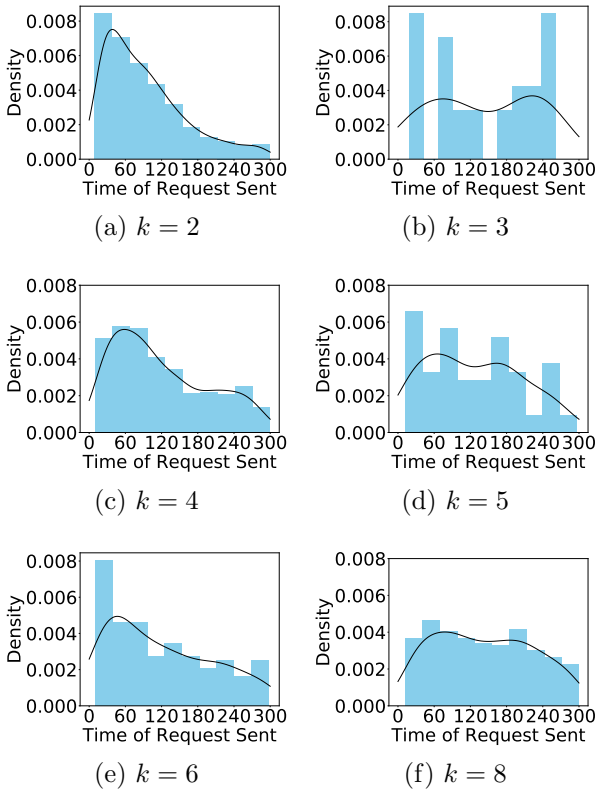
Appendix B

Appendix: Iterative Abduction Framework

B.1 Experimental Data

B.1.1 Timestamp for Letter Request

Figure B.1 shows histograms with 10 bins of 30-seconds each for the timestamps of **request sent** for experiments with k= 2, 3, 4, 5, 6, 8. The same trends exist for each value of k. However, if there are few neighbors (k=2) and consequently fewer available letters (3 letters per neighbor), there are fewer letter requests and letter replies near the end of the game.



(a) $k = 2$ (b) $k = 3$

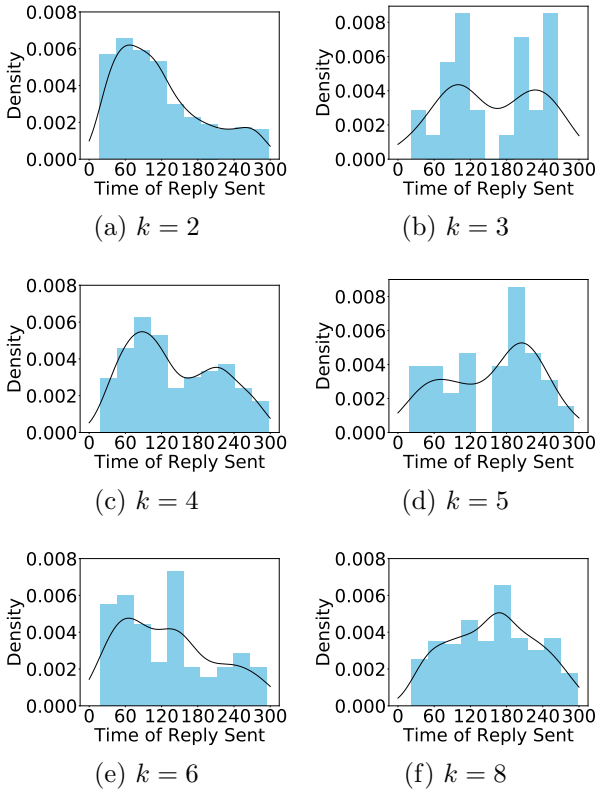
(c) $k = 4$ (d) $k = 5$

(e) $k = 6$ (f) $k = 8$

Figure B.1: Probability density distribution for requests sent over the 300 second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. (a) shows 1 experiment, for 28 experiments with k=2; (b) shows 1 experiment with k=3; (d) shows 9 experiments with k=4; (a) shows 2 experiments with k=5; (e) shows 3 experiments with k=6; (f) shows 4 experiments with k=8. A kernel-density estimation with Gaussian kernels is used. Letter requests are made throughout the game, rather than solely at the outset. However, if there are few neighbors (k=2) and consequently fewer available letters (3 letters per neighbor), there are fewer letter requests and letter replies near the end of the game.

B.1.2 Timestamp for Letter Reply

Figure B.2 shows histograms with 10 bins of 30-seconds each for the timestamps of **Reply Sent** for experiments with k= 2, 3, 4, 5, 6, 8. Similar trends are obtained when data are broken down by k. We find that letter reply are made throughout the game, rather than solely at the outset.



(a) $k = 2$ (b) $k = 3$

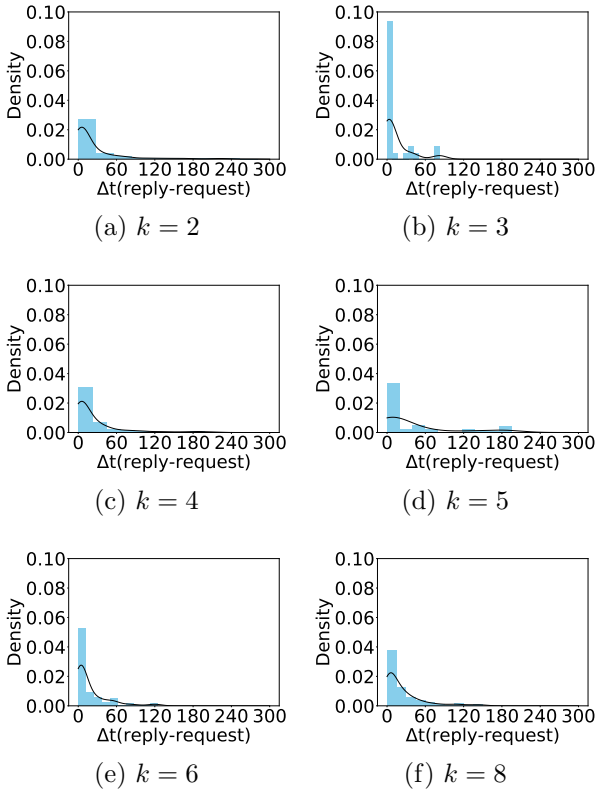
(c) $k = 4$ (d) $k = 5$

(e) $k = 6$ (f) $k = 8$

Figure B.2: Probability density distribution for reply sent over the 300 second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. (a) shows 28 experiments with k=2; (b) 1 experiment with k=3; (c) 9 experiments with k=4; (d) 2 experiments with k=5; (e) 3 experiments with k=6; (f) 4 experiments with k=8. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Letter replies are made throughout the game, rather than solely at the outset.

B.1.3 Timestamp delta between Reply Received and Request Sent

Figure B.3 shows histograms with 10 bins of 30-seconds each for the timestamps of duration between Reply Received and Request Sent for experiments with k= 2, 3, 4, 5, 6, 8. The number of neighbors doesn't affect this type of action, players generally respond relatively quickly to their neighbors letter requests with replies typically made within 30 seconds of the request.



(a) $k = 2$ (b) $k = 3$

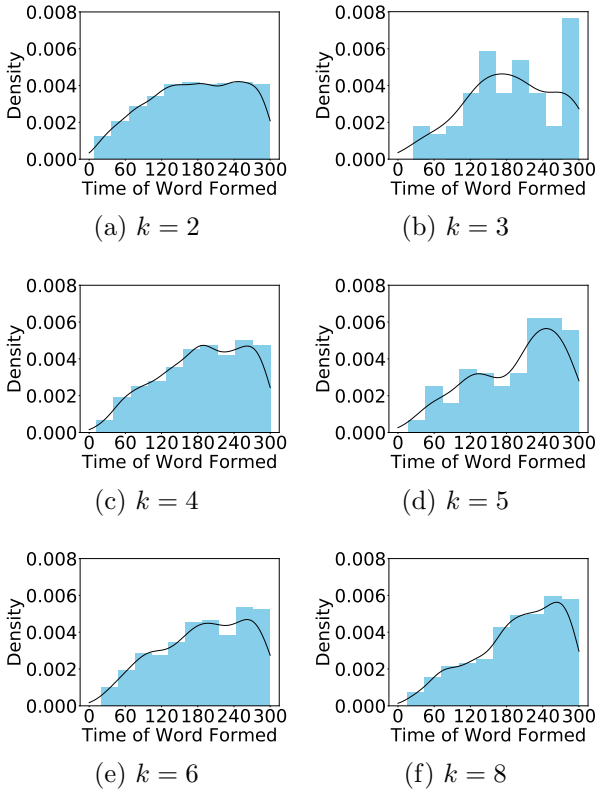
(c) $k = 4$ (d) $k = 5$

(e) $k = 6$ (f) $k = 8$

Figure B.3: Probability density distribution for the time duration between Reply Received and Request Sent over the 300 second anagram game. (a) shows 28 experiments with k=2; (b) shows 1 experiment with k=3; (c) shows 9 experiments with k=4; (d) shows 2 experiments with k=5; (e) shows 3 experiments with k=6; (f) shows 4 experiments with k=8. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Players generally respond relatively quickly to their neighbors letter requests with replies typically made within 30 seconds of the request, the number of neighbors doesn't affect this type of action.

B.1.4 Timestamp for Word Formed

Figure B.4 show histograms with 10 bins of 30-seconds each for **Word Formed** timestamps for experiments with k= 2, 3, 4, 5, 6, 8. Word submissions are made throughout the game, and the number of neighbors and available letters, doesn't affect this type of action.



(a) $k = 2$ (b) $k = 3$

(c) $k = 4$ (d) $k = 5$

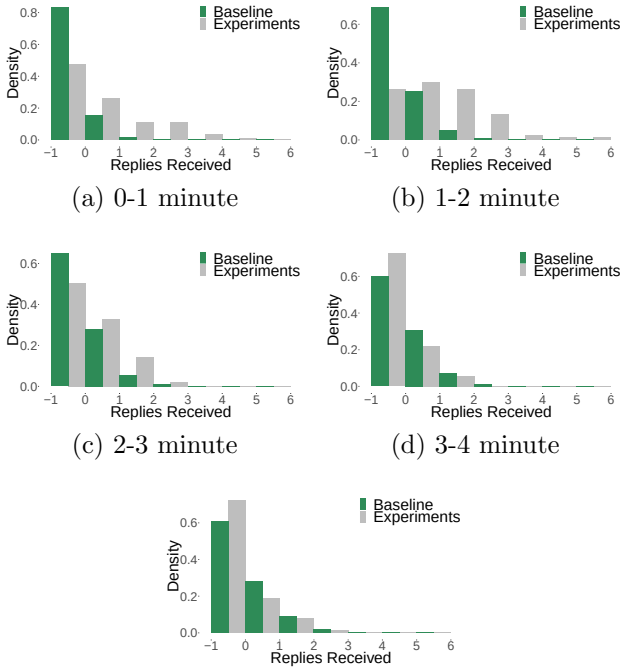
(e) $k = 6$ (f) $k = 8$

Figure B.4: Probability density distribution for the time of words formed over the 300 second anagram game. Each of the bins on the x-axis correspond to 30-second intervals. (a) shows 28 experiments with $k = 2$; (b) shows 1 experiment with $k = 3$; (c) shows 9 experiments with $k = 4$; (d) shows 2 experiments with $k = 5$; (e) shows 3 experiments with $k = 6$; (f) shows 4 experiments with $k = 8$. A kernel-density estimation with Gaussian kernels is used to estimate the probability density function. Word submissions are made throughout the game, and the number of neighbors and available letters, doesn't affect this type of action.

B.1.5 Temporal Comparisons of Distributions Between Model M0 and Experiments for Individual Player Actions

This section shows the figures resulting from the temporal analysis by minute of distributions between Model M0 and Experiments for $k = 2$. Each plot contains data over a time window. Figure B.5 shows temporal analysis for the number of replies received at the end of each minute. Figure B.6 shows temporal analysis for the number of replies sent at the end of each minute. Figure B.7 shows temporal analysis for the number of requests received at the end of each minute. Figure B.8 shows temporal analysis for the number of requests sent at the end of each minute. Figure B.9 shows temporal analysis for the number of words formed at

the end of each minute. For all these plot the SubFigures show the following intervals (a) the 0-1 minute, (b) the 1-2 minute, (c) the 2-3 minute, (d) the 3-4 minute, and (e) the 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. Often, but not always, the largest discrepancies between the model predictions and experiments occur in the first minute of the game.



(a) 0-1 minute

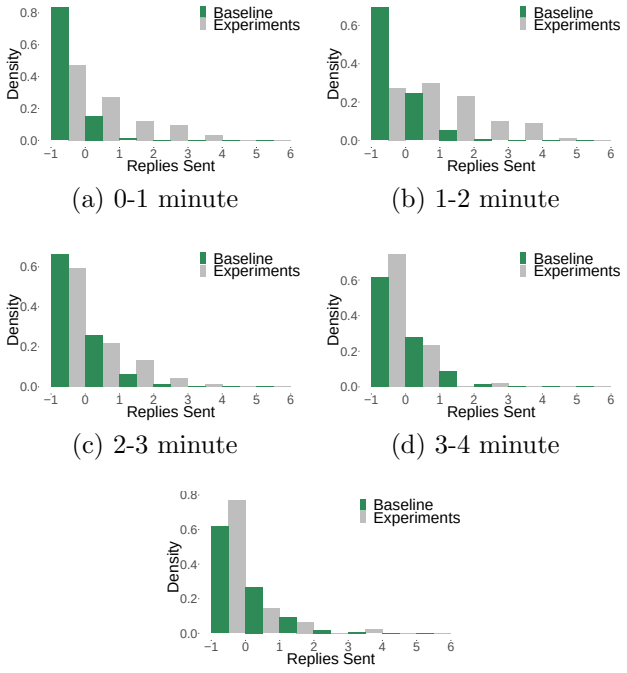
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.5: ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Replies Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, replies received predictions are better in the second half of the five-minute anagram games.

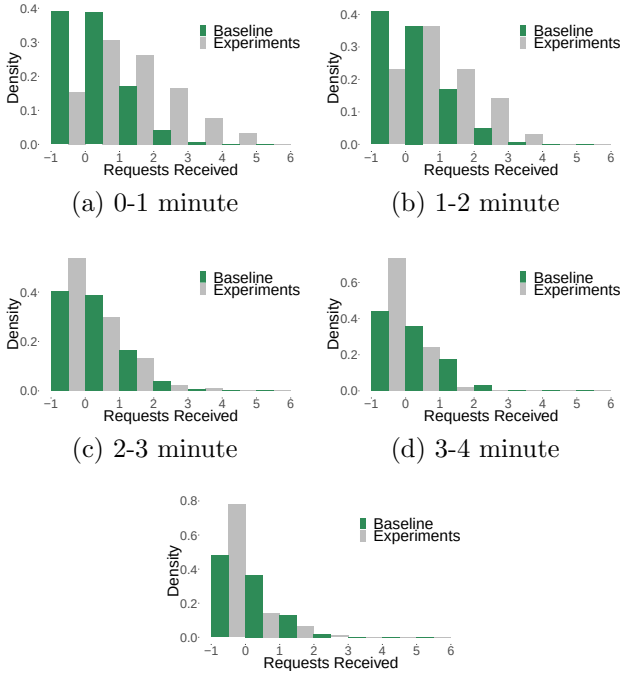


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.6: ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Replies Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, replies sent predictions are better in the second half of the five-minute anagram games.

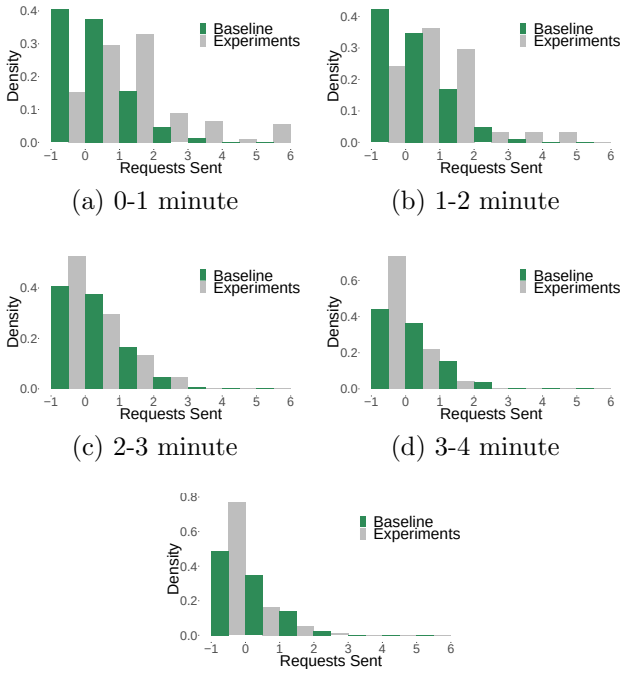


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.7: ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Requests Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, requests received predictions, compared to the other variables, are good through the five minutes of the game.

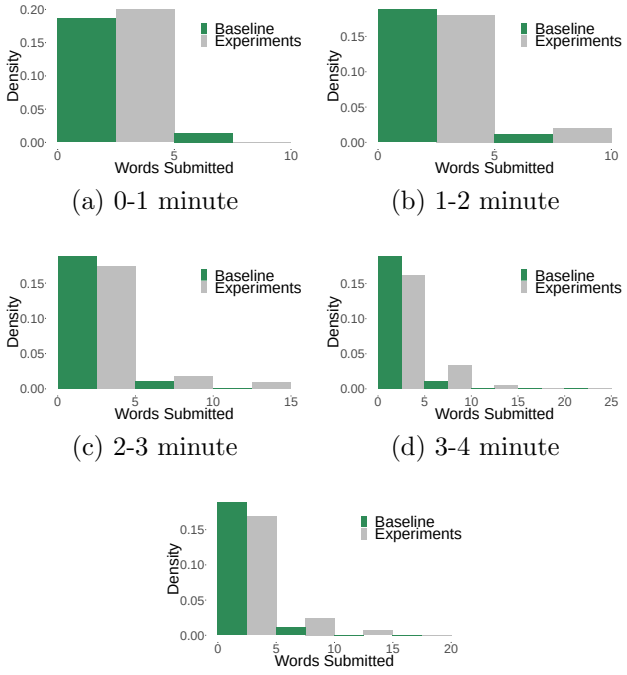


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.8: ABM M0 predictions of the $k = 2$ experiments for the distributions of letters Requests Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, requests sent predictions, compared to the other variables, are good through the five minutes of the game.



(a) 0-1 minute

(b) 1-2 minute

(c) 2-3 minute

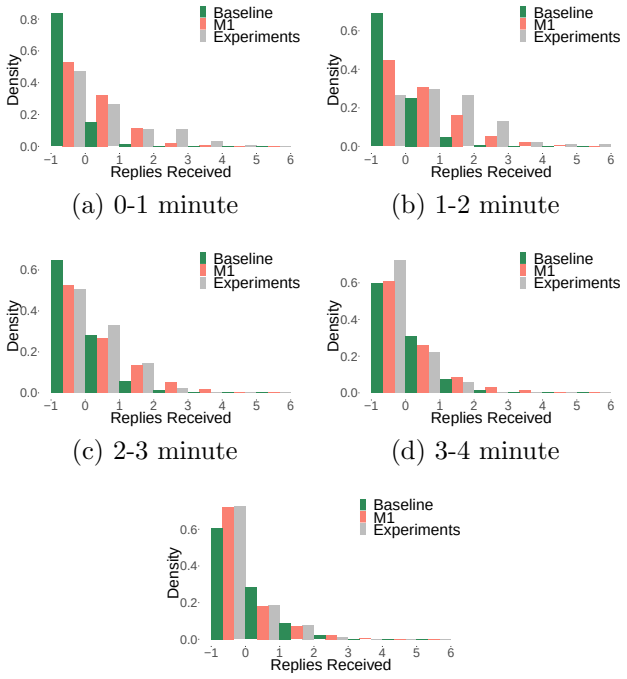
(d) 3-4 minute

(e) 4-5 minute

Figure B.9: ABM M0 predictions of the $k = 2$ experiments for the distributions of Words Formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to Baseline M0 predictions (green) for 100 simulations of an $n = 10$ player game. These plots show that for Model M0, words formed predictions are better in the first two minutes of the five-minute anagram games.

B.1.6 Temporal Comparisons of Distributions Between Models M0, M1 and Experiments for Individual Variables

This section shows the figures resulting from the temporal analysis by minute of distributions between Models M0, M1 and Experiments for $k = 2$. Each plot contains data over a time window. Figure B.10 shows temporal analysis for the number of replies received at the end of each minute. Figure B.11 shows temporal analysis for the number of replies sent at the end of each minute. Figure B.12 shows temporal analysis for the number of requests received at the end of each minute. Figure B.13 shows temporal analysis for the number of requests sent at the end of each minute. Figure B.14 shows temporal analysis for the number of words formed at the end of each minute. For all these plot the SubFigures show the following intervals (a) the 0-1 minute, (b) the 1-2 minute, (c) the 2-3 minute, (d) the 3-4 minute, and (e) the 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. It is clear from visual inspection that model M1 predictions are in better agreement with the experiment data than are M0 predictions. Often, but not always, for Model M1 the largest discrepancies between the model predictions and experiments occur after the first minute and in larger magnitude for the words formed variable.

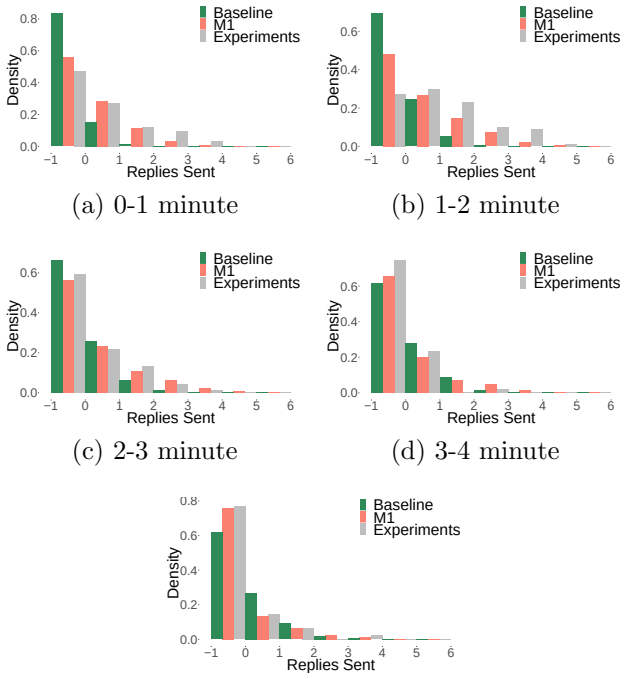


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.10: ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Replies Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, replies received predictions are better in minute 3, and minute 5 of the five minute anagram games.

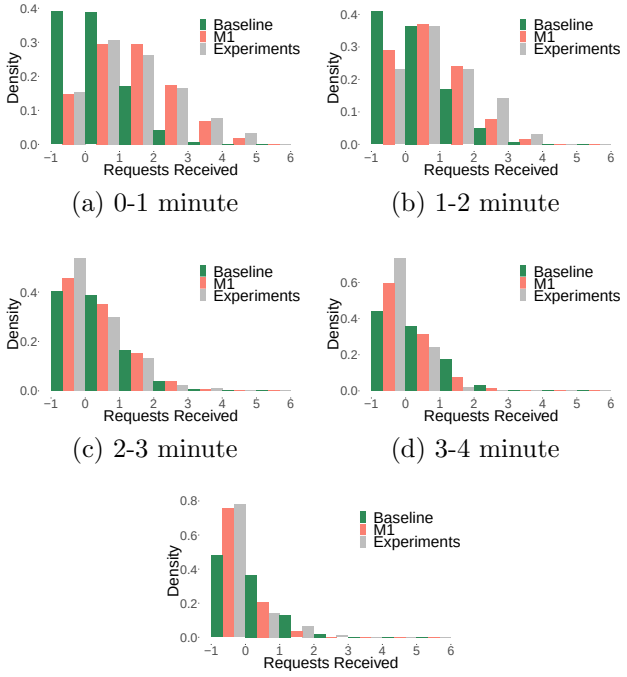


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.11: ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Replies Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, replies sent predictions are better in minute 1, minute 3, and minute 5 of the five minute anagram games.

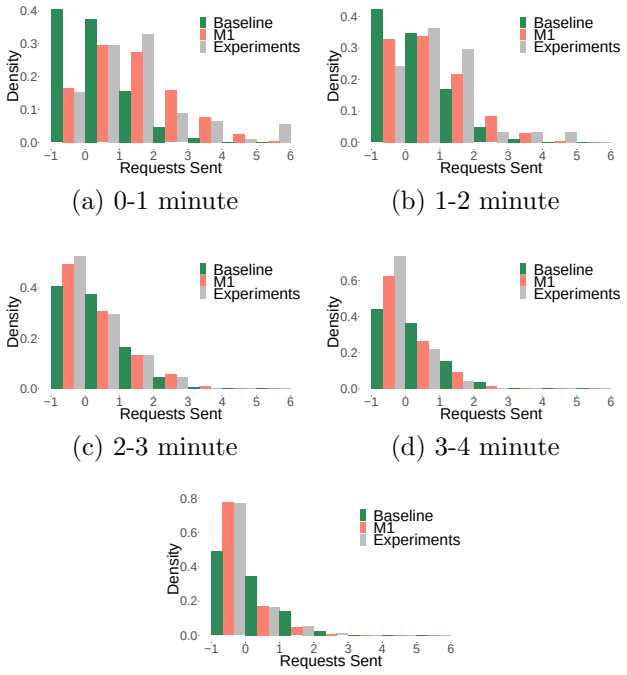


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.12: ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Requests Received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, requests received predictions are good throughout the five minute anagram games.

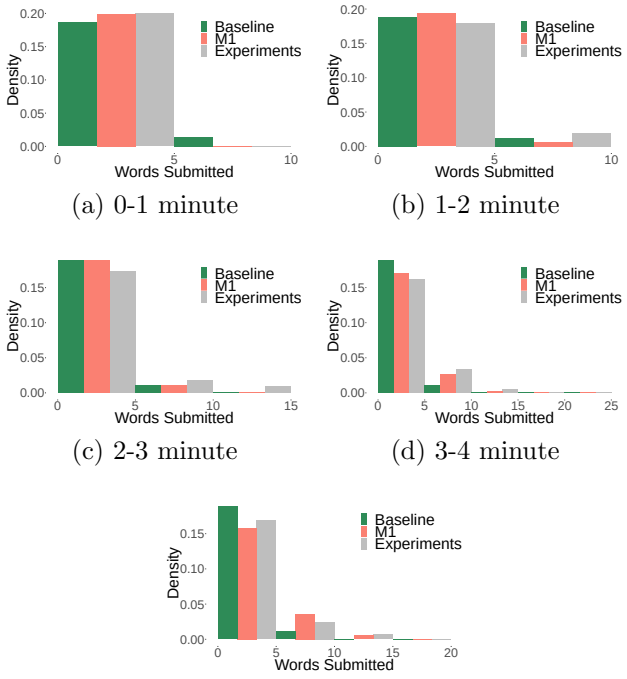


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.13: ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of letters Requests Sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, requests sent predictions are good throughout the five minute anagram games.



(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.14: ABM M0 and M1 predictions of the $k = 2$ experiments for the distributions of Words Formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M1 predictions (red) for 100 simulations of an $n = 10$ player game. The baseline model M0 is shown in green for comparison. These plots show that for Model M1, words formed predictions are good only in the first minute of the five minute anagram games.

B.1.7 Comparisons of Distributions Between Models M1, M2 and Experiments for Individual Variables at the End of the Anagram Game.

This section shows the comparisons of distributions between models M1, M2 and experiments for individual variables at the end of the 5-minute anagram game. Figure B.15 shows data distributions (gray bars) for all $k = 6$ compared to M2 predictions (blue bars). Figure B.16 shows data distributions (gray bars) for all $k = 8$ compared to M2 predictions (blue bars). The model M1 is shown in red for comparison. Figure (a) shows the distributions of replies received, Figure (b) shows the distributions of replies sent, Figure (c) shows the distributions of requests received, Figure (d) shows the distributions of requests sent, and Figure (e) shows the distributions of Words Formed. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. As expected, M1 and M2 perform equally well for $k = 2$ as M1 is learned from $k = 2$ experimental data. For $k > 2$, M2 performs better. Figure 3.21 shows data distributions for all $k = 2$ and Figure 3.22 shows data distributions for all $k = 4$.

B.1.8 Temporal Comparisons of Distributions Between Models M1, M2 and Experiments for Individual Variables.

This section shows the temporal analysis by minute of distributions between Models M1, M2 and Experiments for $k = 2, 4, 6, 8$. Each plot contains data over a time window, SubFigures (a) the 0-1 minute, (b) the 1-2 minute, (c) the 2-3 minute, (d) the 3-4 minute, and (e) the 4-5 minute, of the 5-minute anagram game experiments (gray bars), compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. Figure B.17, B.22, B.27, and B.32 shows temporal analysis for the number of replies received at the end of each minute for $k = 2, 4, 6, 8$ accordingly. Figure B.18, B.23, B.28, and B.33 shows temporal analysis for the number of replies sent at the end of each minute for $k = 2, 4, 6, 8$ accordingly. Figure B.19, B.24, B.29, and B.34 shows temporal analysis for the number of requests received at the end of each minute for $k = 2, 4, 6, 8$ accordingly. Figure B.20, B.25, B.30, and B.35 shows temporal analysis for the number of requests sent at the end of each minute for $k = 2, 4, 6, 8$ accordingly. Figure B.21, B.26, B.31, and B.36 shows temporal analysis for the number of words formed at the end of each minute for $k = 2, 4, 6, 8$ accordingly. M2 gives much better performance, as expected, as it explicitly accounts for agent degree. As expected, M1 and M2 perform equally well for $k = 2$ as M1 is learned from $k = 2$ experimental data. For $k > 2$, M2 performs better.

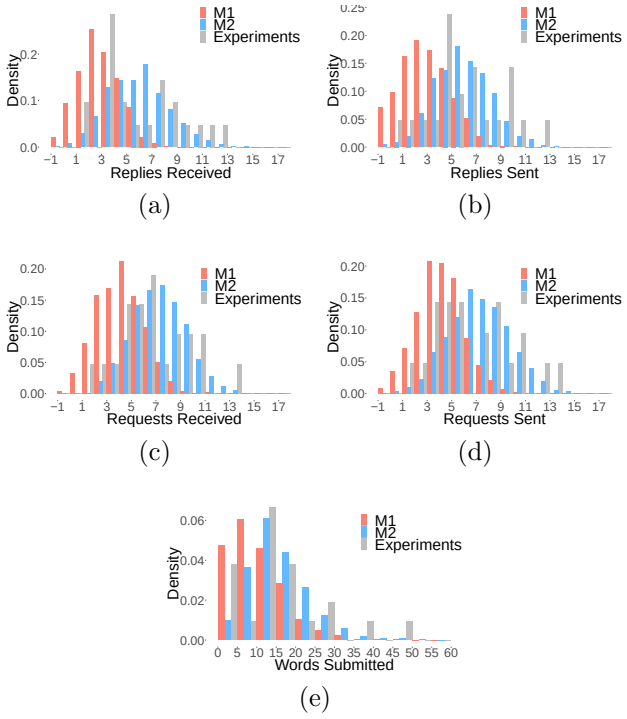


Figure B.15: ABM M1 and M2 predictions of the $k = 6$ experiments and experimental data. (a) Distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that model M2 predictions are in better agreement with the experiment data than are M1 predictions (with the exception of the words formed variable). We make this comparison more precise using KL-divergence in Figure B.37.

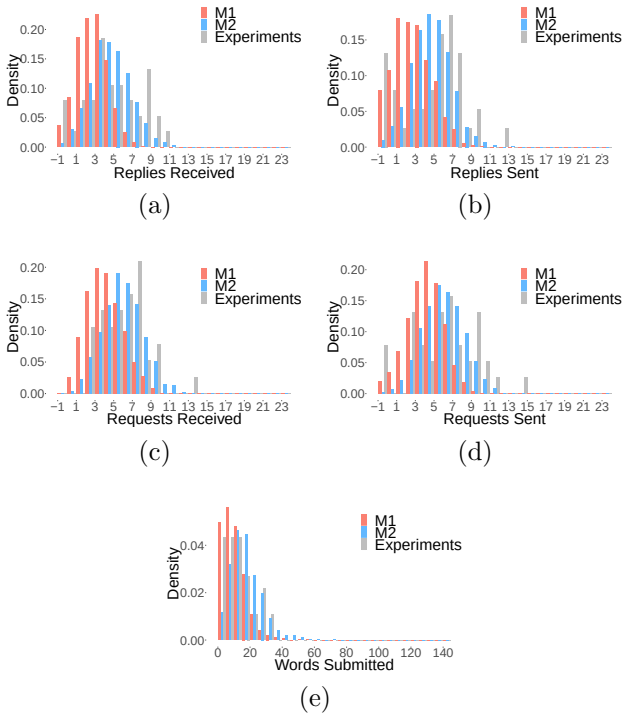
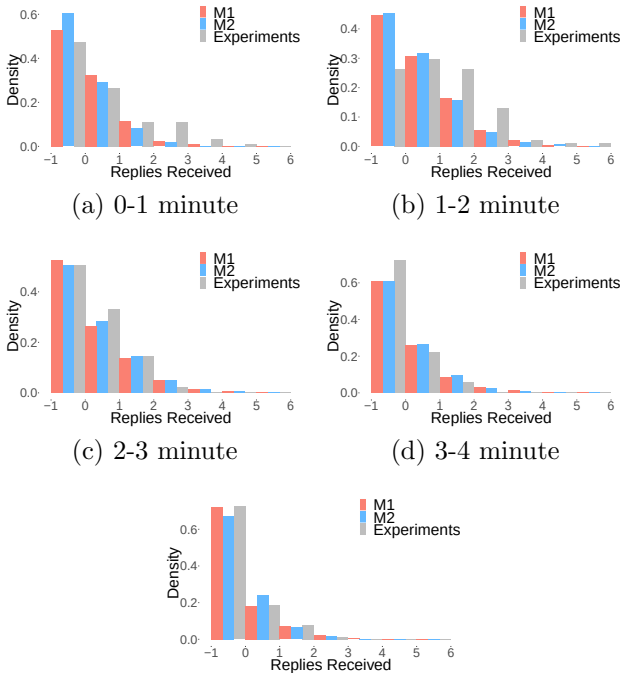


Figure B.16: ABM M1 and M2 predictions of the $k = 8$ experiments and experimental data. (a) Distribution of replies received, (b) distribution of replies sent, (c) distribution of requests received, (d) distribution of requests sent, and (e) distribution of words formed, each at the end of the 5-minute anagram game (gray bars are experimental data) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 predictions are shown in red for comparison. It is clear from visual inspection that model M2 predictions are in better agreement with the experiment data than are M1 predictions (with the exception of the words formed variable). We make this comparison more precise using KL-divergence in Figure B.38.

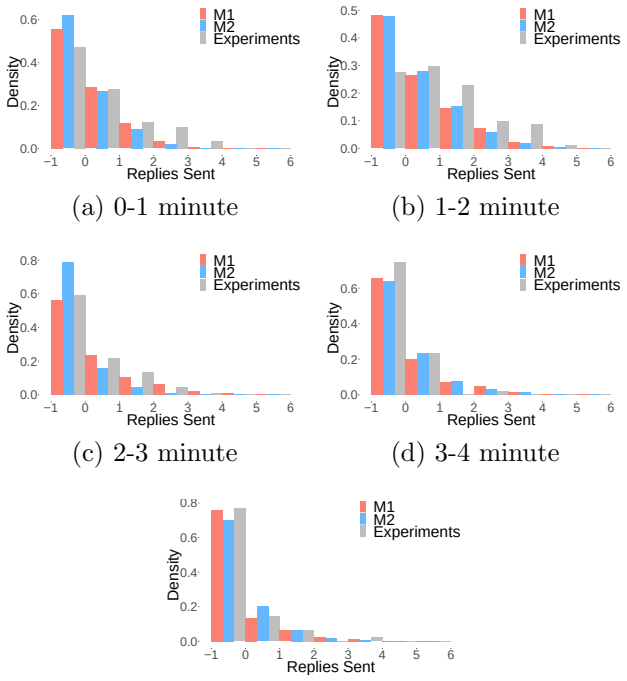


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.17: ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received, M1 and M2 perform equally well for k = 2 throughout the five minute anagram games.



(a) 0-1 minute

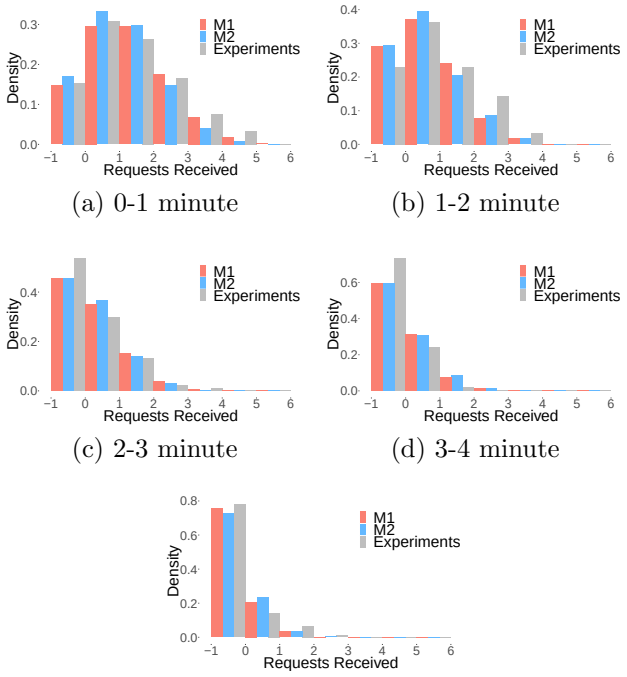
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.18: ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies sent Model M1 predictions are slightly better in minute 1, minute 2, minute 3, and minute 5 of the five minute anagram games.

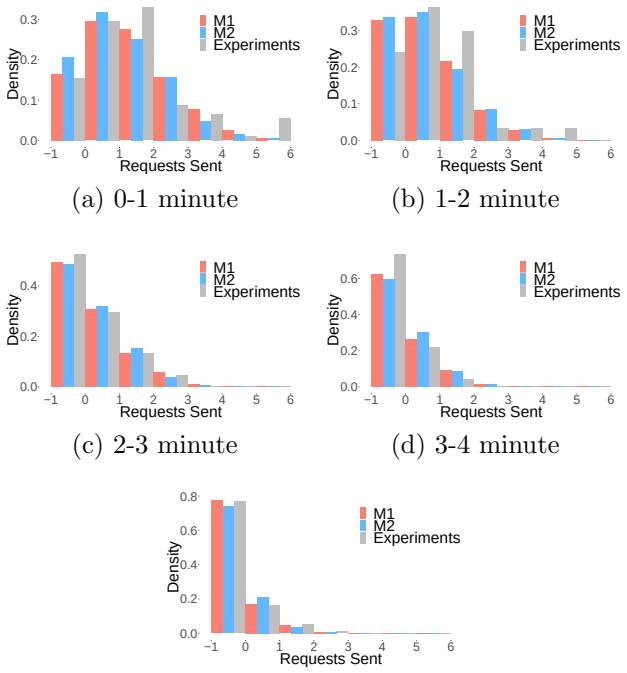


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.19: ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M2 predictions are slightly better than M1 throughout the five minute anagram games.



(a) 0-1 minute

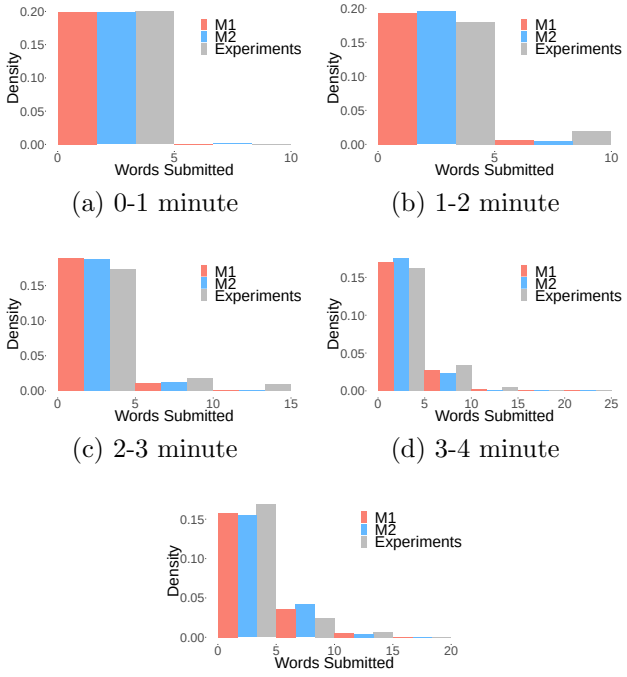
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.20: ABM M2 and M1 predictions of the $k = 2$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M1 predictions are slightly better than M2 throughout the five minute anagram games.

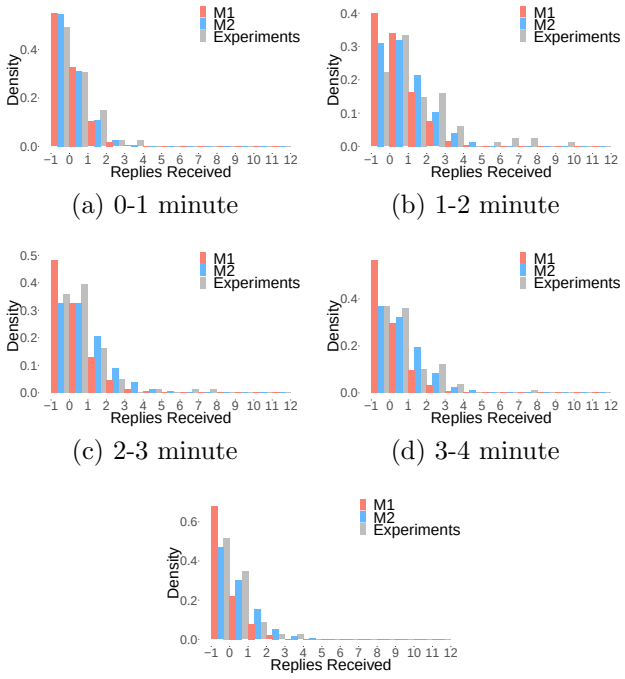


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.21: ABM M1 and M2 predictions of the $k = 2$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 2$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed Model M1 predictions are slightly better than M2 throughout the five minute anagram games.

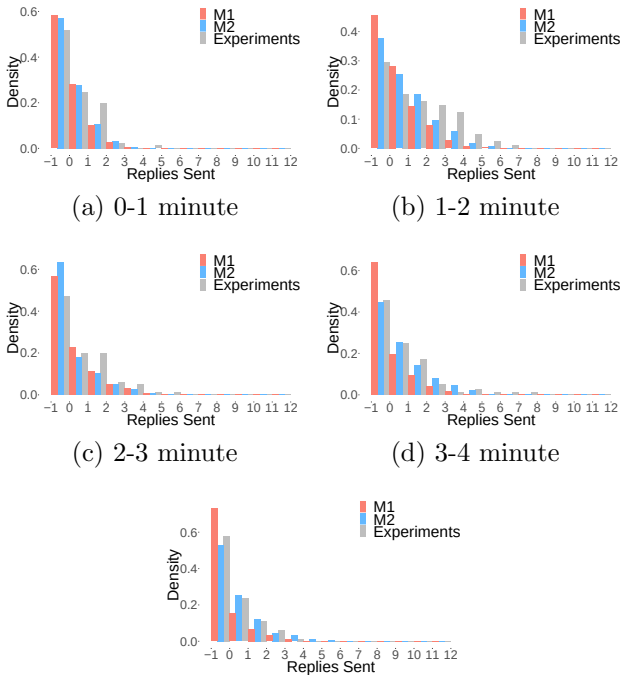


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.22: ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received Model M2 predictions are better than M1 throughout the five minute anagram games.

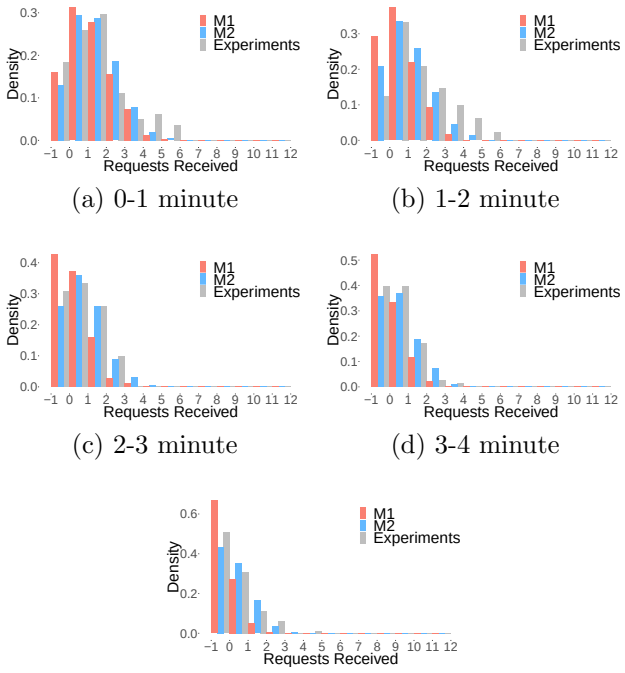


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.23: ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies sent Model M2 predictions are better than M1 (except for minute 3) throughout the five minute anagram games.



(a) 0-1 minute

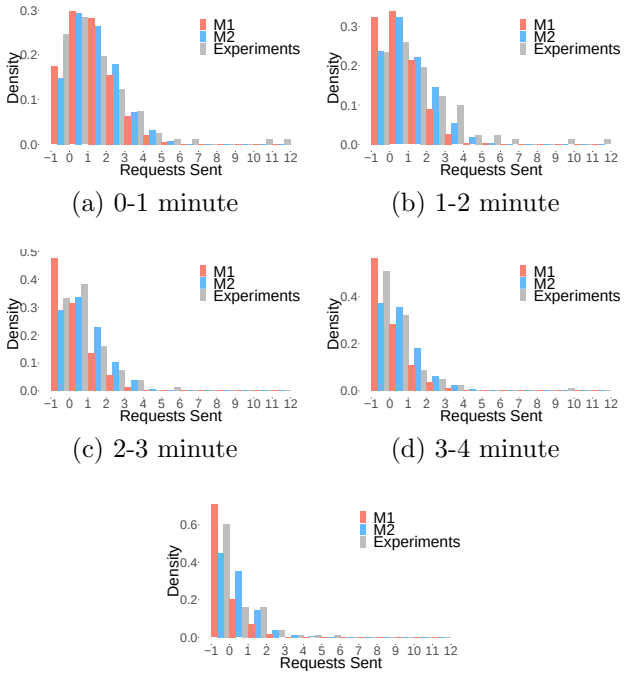
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.24: ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M2 predictions are better than M1 throughout the five minute anagram games.

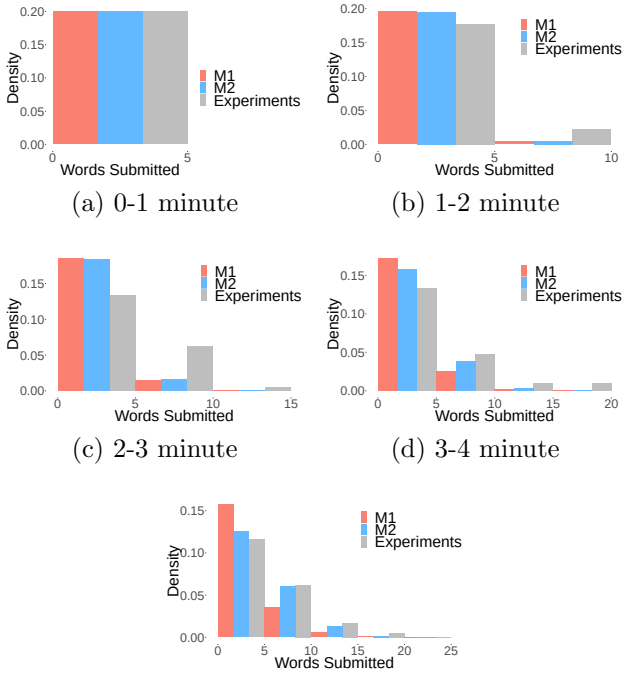


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.25: ABM M2 and M1 predictions of the $k = 4$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests sent Model M2 predictions are slighlty better than M1 throughout the five minute anagram games.

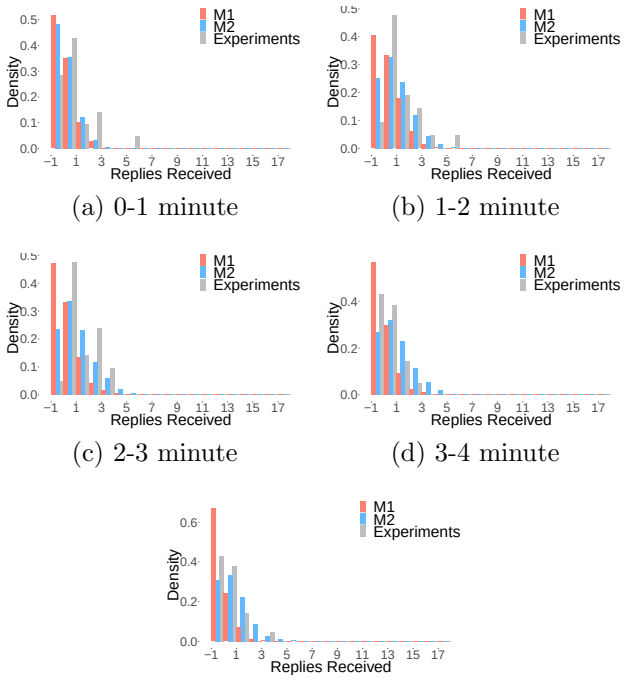


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.26: ABM M1 and M2 predictions of the $k = 4$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 4$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed Model M2 predictions are better than M1 for minute 2, minute 4 and minute 5 of the five minute anagram games.



(a) 0-1 minute

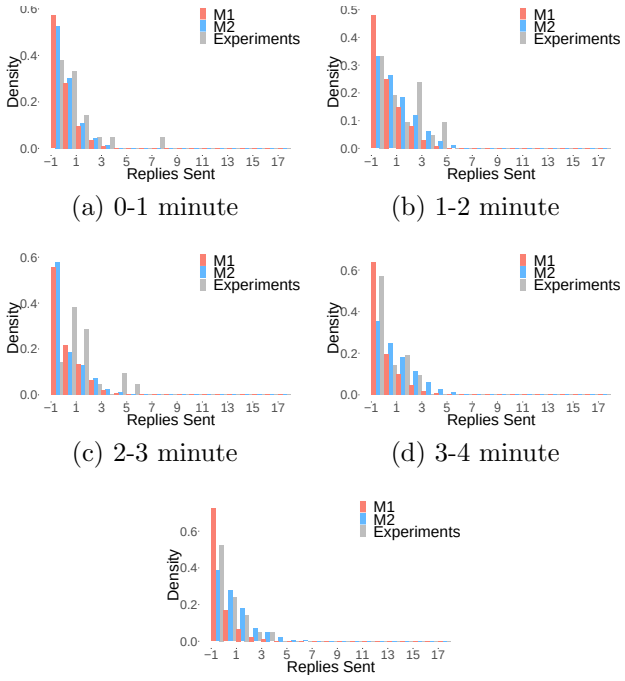
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.27: ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received most Model M2 predictions are better than M1 throughout the five minute anagram games.



(a) 0-1 minute

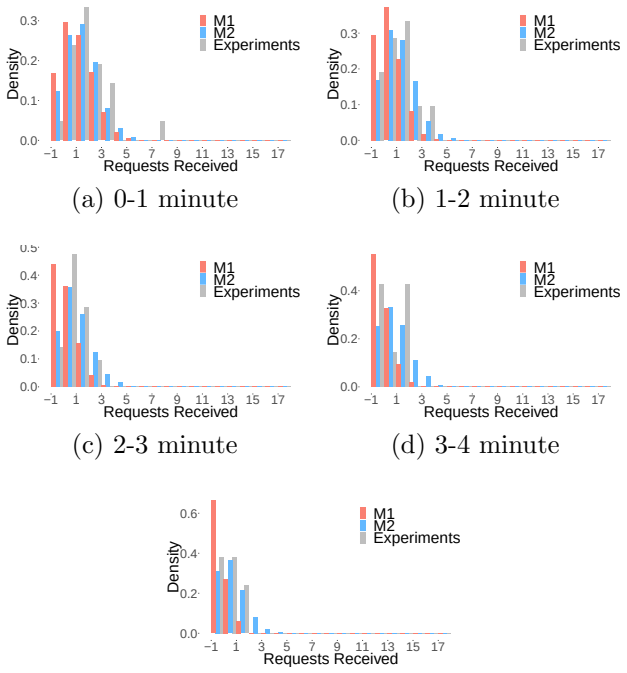
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.28: ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies sent Model M2 predictions are better than M1 throughout the five minute anagram games.



(a) 0-1 minute

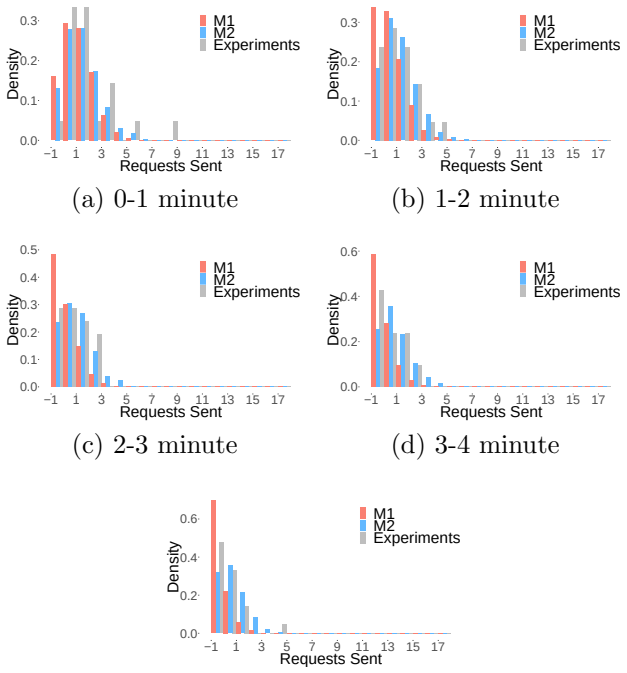
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.29: ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received Model M2 predictions are better than M1 throughout the five minute anagram games.



(a) 0-1 minute

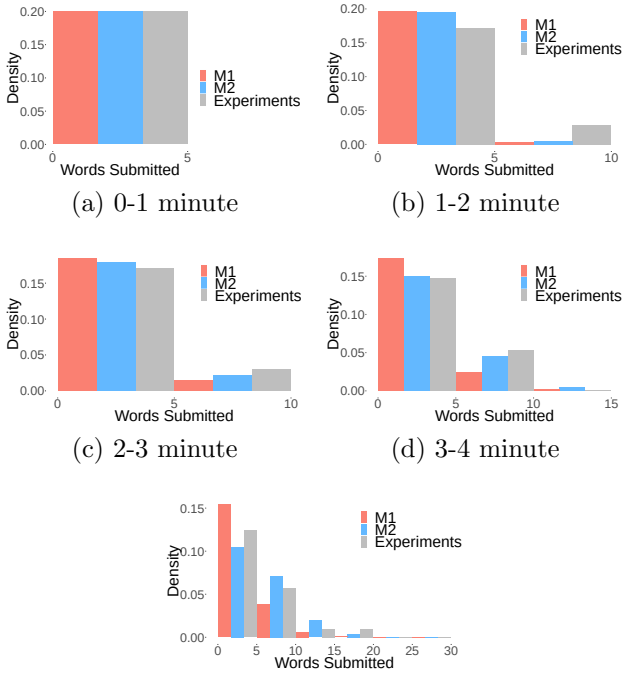
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.30: ABM M2 and M1 predictions of the $k = 6$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests sent Model M2 predictions are better than M1 throughout the five minute anagram games.



(a) 0-1 minute

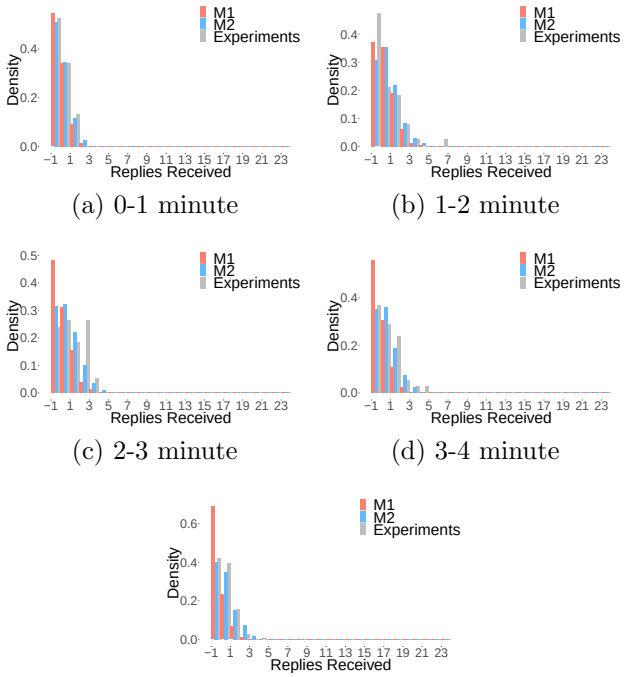
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.31: ABM M1 and M2 predictions of the $k = 6$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 6$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed, Model M2 predictions are better than M1 after the first two minutes of the five minute anagram game.



(a) 0-1 minute

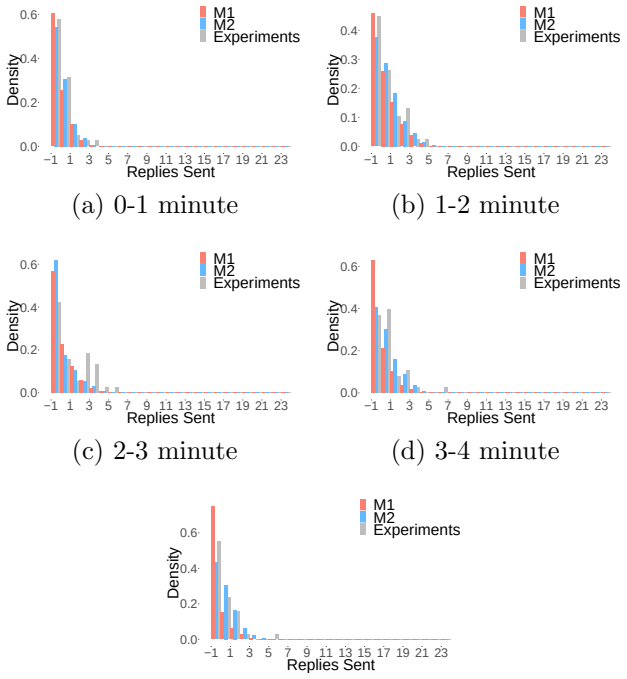
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.32: ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of letters replies received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The baseline model M1 is shown in red for comparison. These plots show that for replies received, Model M2 predictions are better than M1 throughout the five minute anagram games.

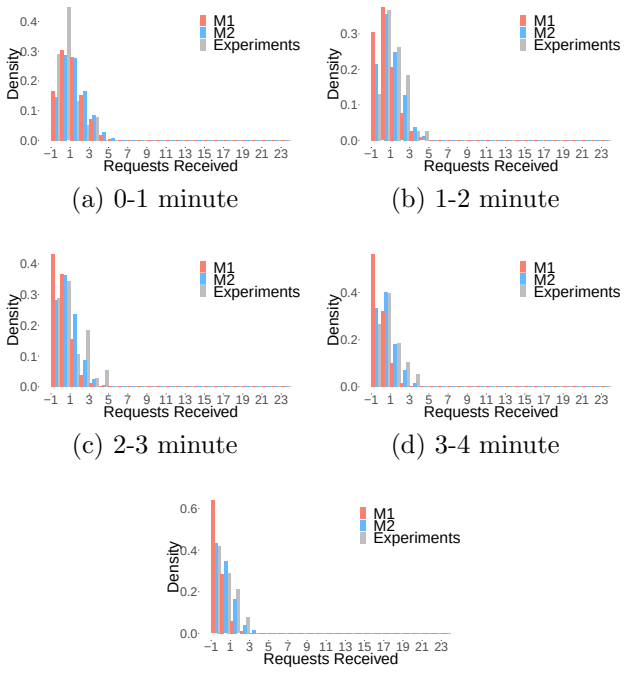


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.33: ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of letters replies sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for replies received Model M2 predictions are better than M1 throughout the five minute anagram games.



(a) 0-1 minute

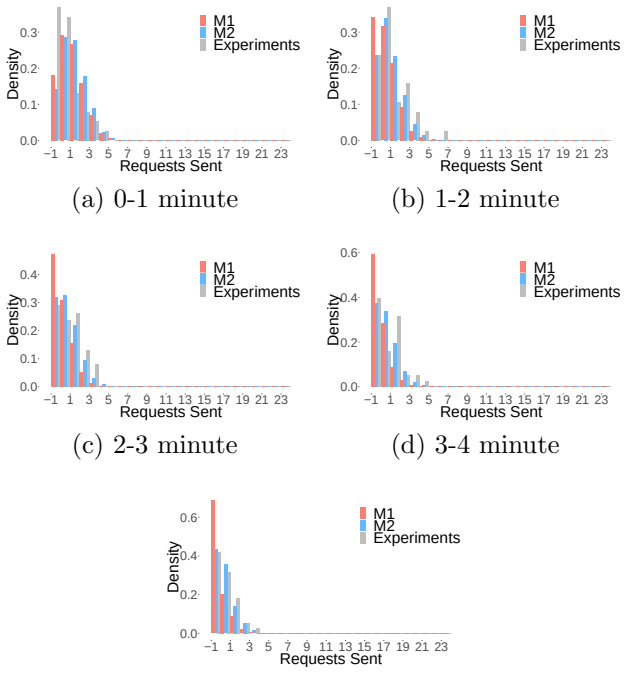
(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.34: ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of letters requests received. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests received, Model M2 predictions are better than M1 throughout the five minute anagram games.

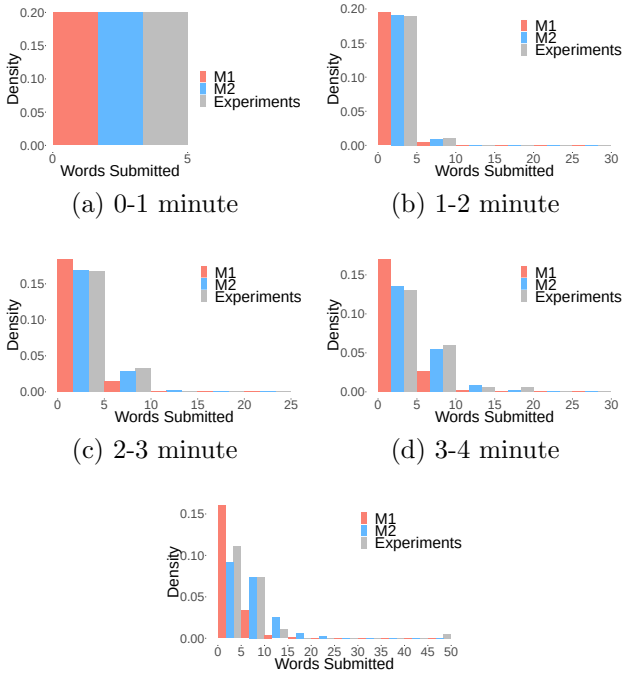


(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.35: ABM M2 and M1 predictions of the $k = 8$ experiments for the distributions of letters requests sent. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for requests sent Model M2 predictions are better than M1 throughout the five minute anagram games.



(a) 0-1 minute (b) 1-2 minute

(c) 2-3 minute (d) 3-4 minute

(e) 4-5 minute

Figure B.36: ABM M1 and M2 predictions of the $k = 8$ experiments for the distributions of words formed. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game (gray bars) for all $k = 8$ experiments, compared to M2 predictions (blue) for 100 simulations of an $n = 10$ player game. The model M1 is shown in red for comparison. These plots show that for words formed, Model M2 predictions are better than M1 throughout the five minute anagram games.

B.1.9 Comparisons of KL Divergence Distributions Between Models M1, M2 and Experiments for Individual Variables at the End of the Anagram Game.

This sections shows the KL divergence values for comparing distributions of models M1 and M2 outputs with corresponding distributions of experimental data, for the end of the five minute anagram game. The models are M1 (red bars) and M2 (blue bars) and the data sets used in comparison are for experiments with $n = 10$, with Figure B.37 experiments with $k = 6$, and Figure B.38 experiments with $k = 8$. Figure 3.23 shows experiments with $k = 2$ and Figure 3.24 shows experiments with $k = 4$. For each experiment/model combination, the variables (and hence distributions) compared are: number of replies received, number of replies sent, number of requests received, number of requests sent, and number of words

formed. Thus, there are 5 distributions and correspondingly 5 DKL values in each distribution. KL-divergence values for the baseline M1 and M2 models across the five parameters of x are shown, lower values are better. Although M1 performs well for $k = 2$, M2 betters M1 for $k > 2$, as M2 incorporates experimental data with $2 \leq k \leq 8$. This improvement is consistent among other x variables as shown in Figure 3.28.

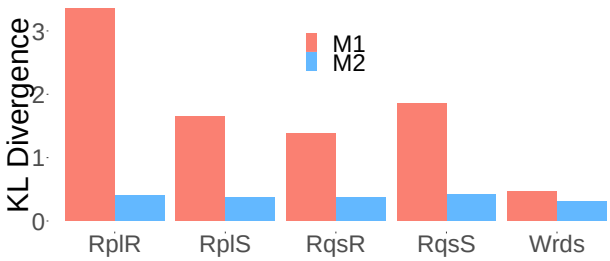


Figure B.37: The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 6$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M2 gives much better performance than M1 predicting the time to generate an action for an agent. M2 gives much better performance, as expected, as it explicitly accounts for agent degree.

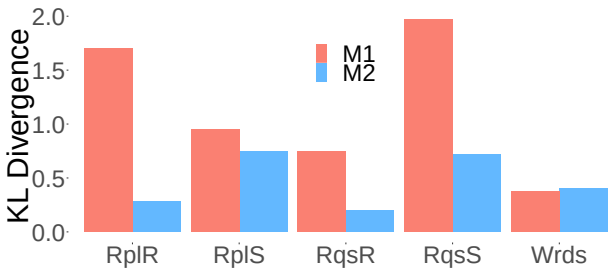
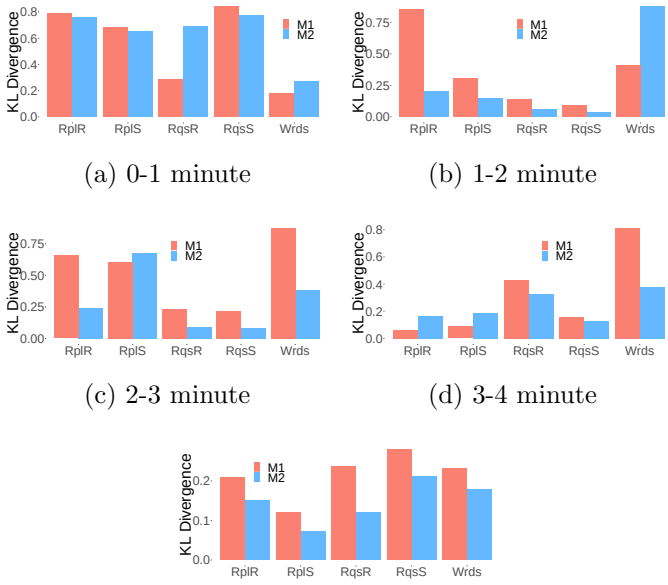


Figure B.38: The plot shows on the x axis KL-divergence values for the M1 and M2 models predictions at the end of the 5 minute anagram game. Here we compare $k = 8$ M1 and M2 models predictions to the experiments across the five parameters of x: lower values are better. This figure shows that M2 gives much better performance than M1 predicting the time to generate an action for an agent. M2 gives much better performance, as expected, as it explicitly accounts for agent degree.

B.1.10 Temporal Comparisons of KL Divergence Distributions Between Models M1 and M2, and Experiments for Individual Player Actions.

This section shows the temporal KL-divergence values for the model M1 and M2 predictions across the five parameters of x. Lower values are better. Figure B.39 shows $k = 6$ experiments, and Figure B.40 shows $k = 8$ experiments. Figure 3.25 shows $k = 2$ experiments, and Figure 3.26 shows $k = 4$ experiments. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. It is clear from visual inspection that model M2 predictions are in better agreement with the experiment data than are M1 predictions for $k > 2$.



(a) 0-1 minute

(b) 1-2 minute

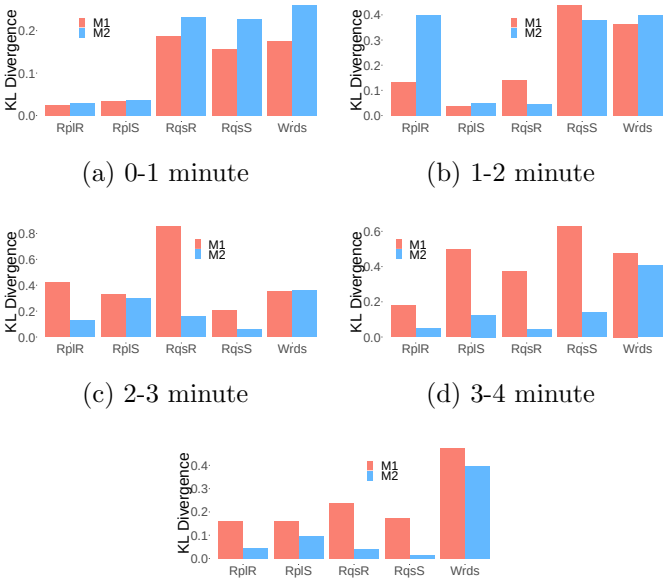
(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.39: KL-divergence values for the Models M1 and M2 predictions of the $k = 6$ experiments across the five parameters of x: lower values are better. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. M2 gives much better performance than M1 predicting the time to generate an action for an agent after the minute two. M2 gives much better performance, as expected, as it explicitly accounts for agent degree.

B.1.11 Temporal Comparisons of KL Divergence Values Between Models M0, M1 and M2, and Experiments by k.



(a) 0-1 minute

(b) 1-2 minute

(c) 2-3 minute

(d) 3-4 minute

(e) 4-5 minute

Figure B.40: KL-divergence values for the Models M1 and M2 predictions of the $k = 8$ experiments across the five parameters of x: lower values are better. Each plot contains data over a time window: (a) 0-1 minute, (b) 1-2 minute, (c) 2-3 minute, (d) 3-4 minute, and (e) 4-5 minute, of the 5-minute anagram game. M2 gives much better performance than M1 predicting the time to generate an action for an agent after the minute three. M2 gives much better performance, as expected, as it explicitly accounts for agent degree.

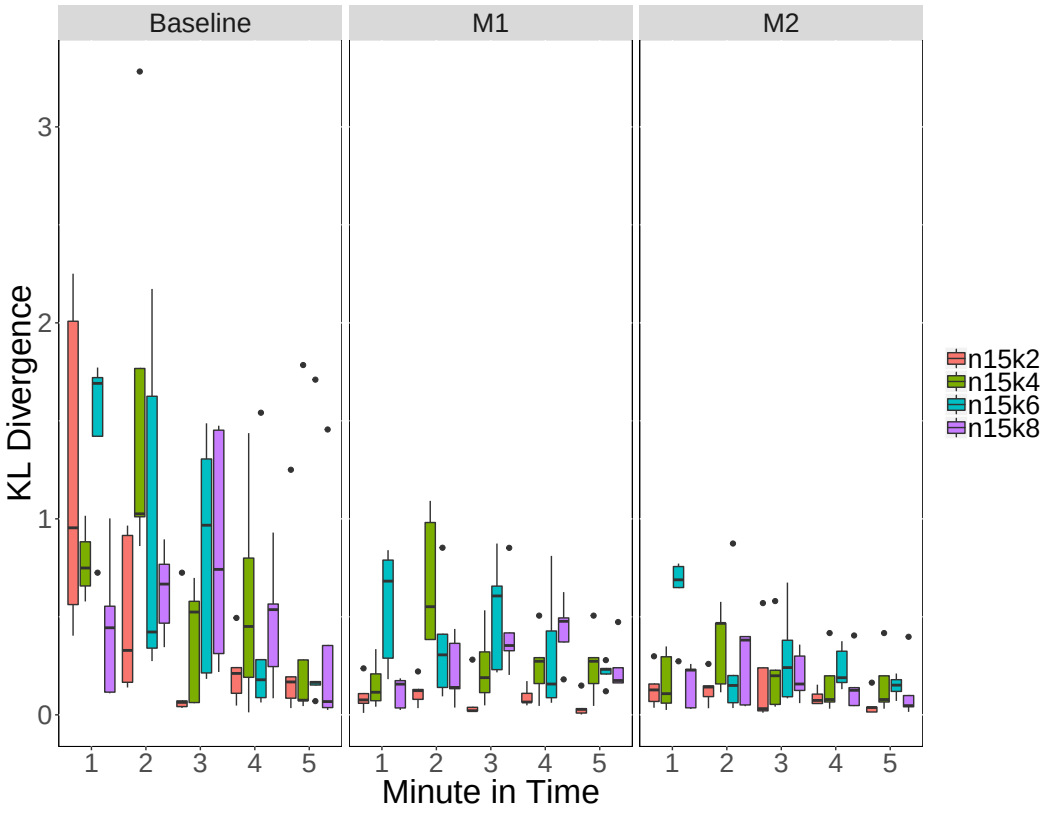


Figure B.41: KLD values from the comparison of models M0 (Baseline), M1, M2, versus the experimental data. On the x axis we show the anagram game by the minute of the five minute game (i.e. [0-1), [1-2), [2-3), [3-4), [4-5)). Each box, by type of k, contains five values of KLD corresponding to the five x variables at the end of each minute. Our models show highest median values on the first two minutes of the game.