A NOVEL MACHINE LEARNING CLASSIFIER BASED ON A QUALIA MODELING AGENT (QMA)

DISSERTATION

Sandra L. Vaughan

AFIT-ENG-DS-16-S-016

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-DS-16-S-016

A NOVEL MACHINE LEARNING CLASSIFIER BASED ON A QUALIA MODELING AGENT (QMA)

DISSERTATION

Presented to the Faculty

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

Sandra L. Vaughan, BS, MS

September 2016

AFIT-ENG-DS-16-S-016

A NOVEL MACHINE LEARNING CLASSIFIER BASED ON A QUALIA MODELING AGENT (QMA)

DISSERTATION

Sandra L. Vaughan, BS, MS

AFIT-ENG-DS-16-S-016

Abstract

This dissertation addresses a problem found in supervised machine learning (ML) classification, that the target variable, i.e., the variable a classifier predicts, has to be identified before training begins and cannot change during training and testing. This research develops a computational agent, which overcomes this problem. The Qualia Modeling Agent (QMA) is modeled after two cognitive theories: Stanovich's tripartite framework, which proposes learning results from interactions between conscious and unconscious processes; and, the Integrated Information Theory (IIT) of Consciousness, which proposes that the fundamental structural elements of consciousness are qualia.

By modeling the informational relationships of qualia, the QMA allows for retaining and reasoning-over data sets in a non-ontological, non-hierarchical qualia space (QS). This novel computational approach supports concept drift, by allowing the target variable to change *ad infinitum* without re-training while achieving classification accuracy comparable to or greater than benchmark classifiers. Additionally, the research produced a functioning model of Stanovich's framework, and a computationally tractable working solution for a representation of qualia, which when exposed to new examples, is able to match the causal structure and generate new inferences.

AFIT-ENG-DS-16-S-016

This dissertation is dedicated to my husband and my daughter, who have been invaluable sources of encouragement and support through the challenges of graduate school and in every other area of life.

To the memory of my parents, who are greatly missed.

Acknowledgements

Thank you to the mathematicians, statisticians, cognitive researchers, computer scientists and electrical engineers that contributed their knowledge and expertise to make this research complete. Thank you to my fellow students, friends and my wingman, who provided moral support and encouragement. A special appreciation to my advisor and research committee for their time, guidance and wisdom, which provided me the opportunity to earn a PhD.

Finally, I would like to thank my husband and daughter, who have been incredibly patient and supportive, sharing the sacrifices necessary to achieve our goals together.

Sandra L. Vaughan

Table of Contents

Page

Page

Page

Page

List of Figures

List of Tables

A NOVEL MACHINE LEARNING CLASSIFIER BASED ON A QUALIA MODELING AGENT (QMA)

I. Introduction

1.1 Motivation

Learning-by-experience, i.e., example, is the fundamental method by which humans obtain knowledge (Newell, 1990). In the field of Artificial Intelligence (AI), one category of tools that emulates this learning method are supervised learning classifiers. A supervised classifier is trained on a set of example input-output pairs (training samples) and learns a function to map input to output *labels.* After training, the classifier will be tasked to predict the output *label* in a new set of example inputs (test samples) based on the mapping function of the samples on which it was trained (Russell and Norvig, 2009).

A medical diagnosis classifier provides an intuitive example of a typical supervised classifier. In the medical diagnosis classifier each unique diagnosis is a *label.* Once adequately trained with diagnoses and symptoms, the classifier, given a set of symptoms, is able to predict a diagnosis with some statistical probability. Over time, as additional training samples are introduced, the classifier gains more knowledge and classifies with greater accuracy. One can imagine that this classifier, in order to be accurate enough to handle such a critical task, must retain a large set of training samples.

Now envision a different task, which can be solved using the same training samples. In this task, the diagnosis and some symptoms are observed, however, one critical symptom, vision loss, which is slow to manifest, has not been observed. This particular symptom is not always observed with this diagnosis, but if it is, it would indicate a different medical treatment. The physician will want to know the statistical probability of vision loss in a particular patient in order to properly treat him.

A standard learner, which has been trained to predict diagnoses, would have to re-train on all cumulative training samples in order to predict vision loss. The target variable, i.e., the variable a classifier predicts (diagnoses or vision loss), has to be identified before training begins. Re-training is a time consuming task typically required each time the target variable changes, and while the system is being re-trained and used to predict vision loss, it is not available for diagnoses.

A second limitation of these learners is that the test sample predictive variables are not retained. When the target variable remains constant, test sample predictive variables do not contribute to the prediction of subsequent test samples. However, should the target variable change, for example, from diagnoses to vision loss, previous test sample predictive variables may contribute to the correct prediction of subsequent test samples, e.g. vision loss.

A third limitation of these learners is that the response variables (predicted values) are also not retained. They cannot be retained as training samples, because they may distort the probability distributions applied to subsequent test samples. However, since they are not retained, identical test samples require redundant classification, a drain on limited resources and a possible waste of time.

The fourth limitation is that *some* learners cannot incorporate additional training samples once testing has begun, these are called batch classifiers. Incremental classifiers accept additional training samples after testing has begun (Almaksour, 2011).

The fifth limitation is the limited ability of these learners to properly classify test samples when it has been trained with incomplete or perturbated information.

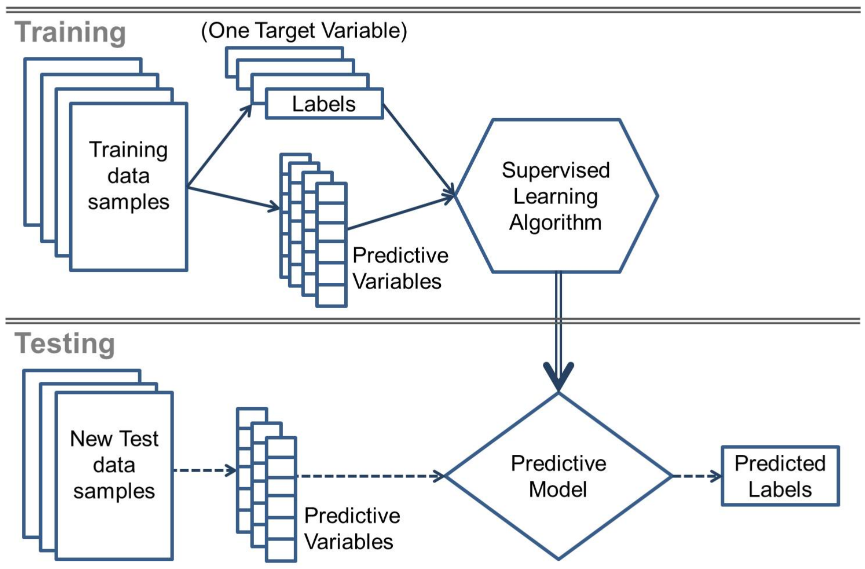
Tasks where the data are often incomplete or intentionally perturbated are malicious software (malware) identification, target recognition in contested environments, and Intelligence, Surveillance, and Reconnaissance (ISR).

Supervised learning classifiers aggregate large volumes of data to infer the general from the particular, i.e., inductive reasoning[1]. Whereas humans, rarely able to reason over large volumes of data, infer knowledge from partial or perturbated information, i.e., abductive reasoning. Abductive reasoning is inferring knowledge not available in the environment, generating a series of competing plausible explanatory hypothesis through cognitive simulation, and choosing the best hypothesis based on some set of criteria (Henson et al., 2012; Shanahan, 1996).

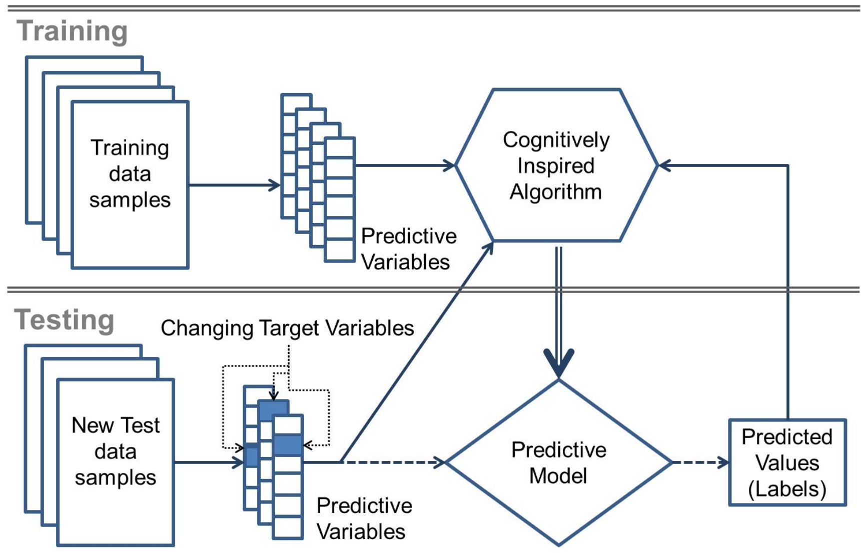
Humans apply previously learned knowledge to new tasks in the same domain, new tasks in similar domains, and even new tasks in foreign domains, with relative ease, using our full repertoire of previous experiences to make robust decisions. This capability is known as *transfer of skill* in human cognitive research (Anderson, 2005), and *transfer learning* in machine learning (ML) research (Pan and Yang, 2010).

The motivation for this dissertation is the opportunity to develop a new set of ML processes implemented in a computational agent that overcomes limitations of standard supervised classifiers by emulating the way humans learn by experience, in particular the way humans employ *transfer of skill* and abductive reasoning to transfer learning from one task to make improved decisions in other tasks. A standard supervised classifier and the proposed computational agent are illustrated in Figure 1.

[1]See Appendix A for logic/reasoning definitions.



(a) Standard Supervised Learning Classifier.



(b) Proposed Computational Agent.

Figure 1: Standard Supervised Learning Classifier Model Compared to Proposed Computational Agent. Solid lines indicate variables incorporated in the predictive model. Dashed lines indicate variables not incorporated in the predictive model.

The standard supervised learning classifier, illustrated in Figure 1a, has to be re-trained on all cumulative training samples each time the target variable changes, and test sample predictive variables and response variables (predicted labels) are not retained for their predictive values. The proposed computational agent, illustrated in Figure 1b, is a classifier that does not have to be re-trained each time the target variable changes, the target variable can change *ad infinitum* during testing, and test sample predictive variables and response variables are retained for their predictive values.

1.2 Hypothesis

This research proposes a computational agent, inspired by a theory of human learning-by-experience, can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes.

The theory of human learning and decision making, selected as the theoretical basis of this research, is Stanovich's tripartite framework (Stanovich and Evans, 2013), illustrated in Figure 2. Stanovich's framework is a specific, extended variation of the traditional Dual–Process Theory of Higher Cognition (DPT). The DPT has featured prominently and consistently in the cognitive literature since the 1960's (Patterson, 2016; Wason, 1966). Stanovich's framework was selected because it provides a detailed explanation of the roles and interactions between the *minds* and working memory (WM) and emphasizes the significance of consciousness in learning and decision-making.

Stanovich's framework proposes that learning and decision making rely on the interactions between three minds, which are situated in two levels. The unconscious level consists of the autonomous mind, without the agent's conscious awareness. The

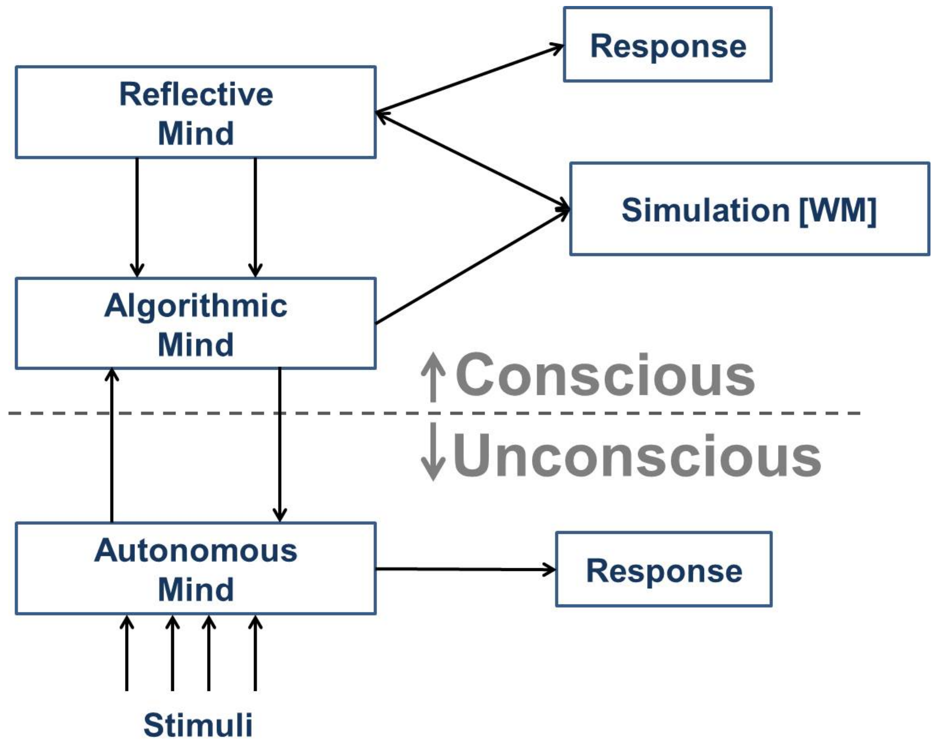


Figure 2: Stanovich's Tripartite Framework, a predominant theory of human cognition which proposes that learning and decision making result from blending conscious/reflective and unconscious/autonomous processes (Stanovich, 2009).

conscious level consists of the reflective mind and the algorithmic mind, of which the agent is consciously aware. In addition to the three interactive minds, Stanovich proposes that consciousness requires WM. WM is transitory and only exists during conscious mental simulation, i.e., hypothetical thinking. Put another way, consciousness is present only during mental simulation in WM.

This dissertation also proposes that by modeling the conscious processes of WM, the unconscious processes of the autonomous mind and their interactions the agent will be able to retain and utilize the predictive variables in test samples, retain and utilize the data available in response variables without distorting the probability distributions applied to subsequent test samples, and incorporate new training samples after the testing phase has begun efficiently and without re-training.

1.3 Research Questions

By conducting the research these questions will be answered:

RQ-1 What are the functions of the autonomous mind, algorithmic mind, reflective mind and WM in Stanovich's framework?

RQ-2 How can each of these components, in Stanovich's framework, be algorithmically implemented and coded in software?

RQ-3 Are there existing tools that can be adapted to support the implementation of each of these components?

RQ-4 What are the interactions between the minds and WM and how can they be modeled to produce a complete computational model of the framework?

RQ-5 What specific functions of Stanovich's framework can overcome the limitations of benchmark supervised classifiers by emulating the way humans employ transfer of skill from one task to make improved decisions in other tasks?

RQ-6 What are the appropriate metrics to assess the performance of existing supervised classifiers and the proposed cognitively inspired agent?

RQ-7 Under what conditions does the developed cognitively inspired agent perform better or worse than benchmark supervised classifiers?

RQ-8 Can the developed computational agent provide an effective decision aid in complex environments where data are too broad or diverse for a human to evaluate without computational assistance?

1.4 Assumptions and Scope

Stanovich's framework is a specific DPT (Kahneman, 2011). Addressing inconsistencies in the cognitive modeling literature including inconsistent terminology for the DPT *minds* and their functionality is well beyond the scope of this dissertation. In this dissertation the terms primarily used by Stanovich will be adopted, which are: *Type I* and *Type II*; unconscious/autonomous and conscious/reflective minds; cognitive simulation and working memory (WM).

Modeling the cognitive frameworks and theories on which this dissertation is based, Stanovich's framework, DPT and Integrated Information Theory (IIT) of Consciousness, will be limited to functionality that supports the research hypothesis and answers the research questions.

The mathematical formalism presented in this dissertation will be designed to accept categorical (nominal) attributes. Accepting ordinal, interval or ratio scales is beyond the scope. A proposed area of future research is to extend the formalism to support additional data types.

1.5 Outline of the Dissertation

This chapter introduced the motivation for this research, research hypothesis, research questions based on the hypothesis and stated assumptions and scope. Chapter 2 provides a literature review of the cognitive theories of learning and decision-making (Stanovich's tripartite framework) and consciousness (Integrated Information Theory (IIT) of Consciousness) on which this dissertation is primarily based. Chapter 3 describes the tools and methodologies used to develop the cognitively inspired computational agent, including Adaptive Control of Thought–Rational (ACT–R), hypernetwork theory and extensions to hypernetwork theory developed in this dissertation. Chapter 4 presents the results of the research, demonstrates support for the research hypothesis and compares the computational agent to benchmark supervised learning classifiers. Chapter 5 summarizes the findings and conclusions, the contributions of this dissertation and areas for future research are explored.

II. Literature Review

Chapter 1 introduced the motivation for this research, the hypothesis and research questions addressing the hypothesis. The research hypothesis proposes that a computational agent, inspired by a theory of human learning-by-experience (Stanovich's tripartite framework), can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and allow it to change after testing begins. In standard supervised classifiers, the target variable is identified *a priori* and represented differently than the other (predictive) variables.

This chapter begins with a review of Stanovich's framework (Section 2.1), an extended theory of the traditional Dual–Process Theory of Higher Cognition (DPT). Stanovich's framework emphasizes the significance of consciousness in learning and decision-making (Stanovich and Evans, 2013). After the discussion of Stanovich's framework, theories of consciousness (Section 2.2), contributing to a computational model, are reviewed. The research of consciousness results in a framework for *how* knowledge is represented, retrieved and reasoned over in the conscious experience base. The last section of this chapter (Section 2.3) reviews literature on conceptual knowledge, which provides a framework for *what* knowledge is represented, retrieved and reasoned over in the conscious experience base.

This dissertation proposes that a machine learning (ML) classifier is analogous to an intelligent agent, and all variables (target and predictive) are analogous to the important details of experiences that have been stored in the agent's memory. The intelligent agent does not know *a priori* what future experiences will require cognitive inference (pattern-completion), therefore it is reasonable to expect that all of the important details of experiences are represented and recalled using the same mechanism.

This literature review seeks a representation of memory storage and recall that represents all details of experiences with the same mechanism, with no ontology or hierarchy, and a formalism with which to reason-over the representation and generate and inference, e.g., predict a label. A computational agent, leveraging this mechanism, would allow the target variable to be identified *a posteriori*, and change after testing begins, in a manner that does not require re-training.

2.1 Stanovich's Tripartite Framework

The computational agent is based on Stanovich's framework (Stanovich, 2009; Stanovich and Evans, 2013), a cognitive theory of learning and decision making which extends the traditional DPT (Kahneman, 2011). Stanovich and Evans propose that learning and decision making rely on two interactive levels: the unconscious level, without the agent's awareness, and the conscious level, in which the agent is aware. The unconscious level consists of the autonomous mind. The conscious level consists of the algorithmic mind, reflective mind and working memory (WM). Figure 3 illustrates the internal processes of the conscious and unconscious components. Figure 4 illustrates the interactions between the conscious and unconscious components.

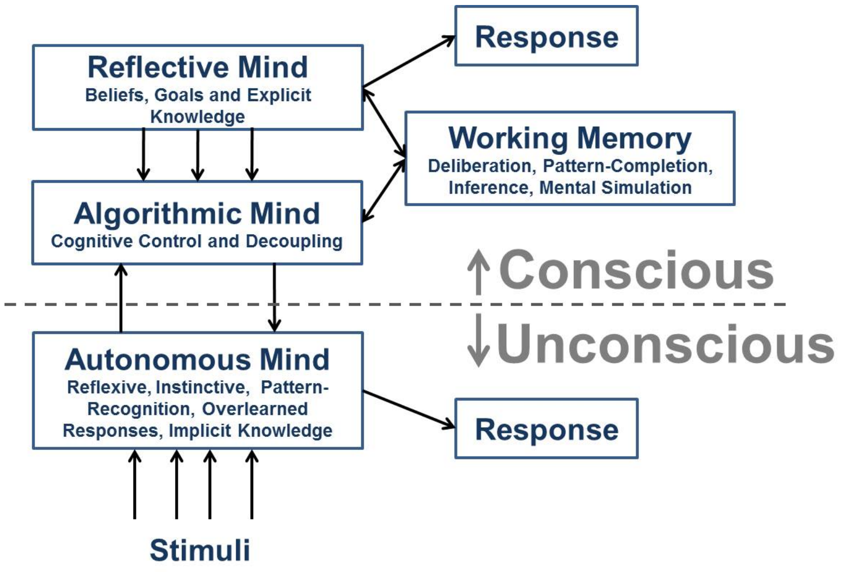

Response
Reflective Mind
Beliefs, Goals and Explicit Knowledge
Working Memory
Deliberation, Pattern-Completion, Inference, Mental Simulation
Algorithmic Mind
Cognitive Control and Decoupling
Conscious
Unconscious
Autonomous Mind
Reflexive, Instinctive, Pattern-Recognition, Overlearned Responses, Implicit Knowledge
Response
Stimuli

Figure 3: Stanovich's Tripartite Framework, Components of the Unconscious and Conscious Levels and Their Primary Roles (Stanovich, 2009).

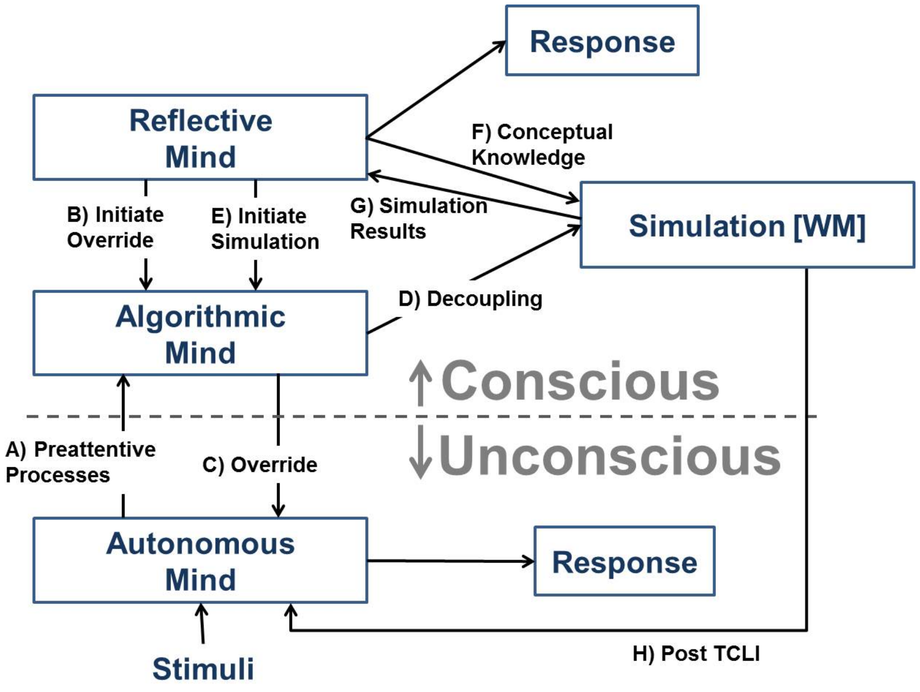


Figure 4: Stanovich's Tripartite Framework, Interactions Between the Three Minds and WM (Stanovich, 2009).

2.1.1 The Unconscious Level.

The unconscious level (lower portion of Figures 3 and 4) consists of the *autonomous mind* and experiences the real-world through stimuli. The unconscious level retains a primary representation of real-world experiences (Stanovich, 2009; Stanovich and Evans, 2013).

In the *autonomous mind* decision making employs pattern-recognition which supports fast, effortless reactive responses. One type of knowledge (memory) retained in the autonomous mind is Tightly Compiled Learned Information (TCLI). TCLI is knowledge generated in (conscious) WM that has become tightly compiled and posted to the autonomous mind due to overlearning and practice (step H in Figure 4). Preattentive processes (step A in Figure 4), perceptions of real-world experiences and the results of autonomous pattern-recognition, supply computations to the conscious level (Stanovich, 2009; Stanovich and Evans, 2013).

2.1.2 The Conscious Level.

The conscious level (upper portion of Figures 3 and 4) consists of the *algorithmic mind*, the *reflective mind*, (transitory) *WM*, does not interact directly with the real-world experiences and generates *consciousness*. (A definition of *consciousness* will be explored in Section 2.2.) The conscious level retains a secondary representation of the real-world experiences, available for manipulation without effecting the primary representation, as illustrated in Figure 5 (Stanovich, 2009; Stanovich and Evans, 2013).

The conscious *algorithmic* mind is responsible for sequencing behavior and controlling and maintaining cognitive decoupling (step D in Figure 4 and Figure 5). Cognitive decoupling allows for mental simulation in WM by decoupling the conscious level from the primary representation of the unconscious level. WM processes

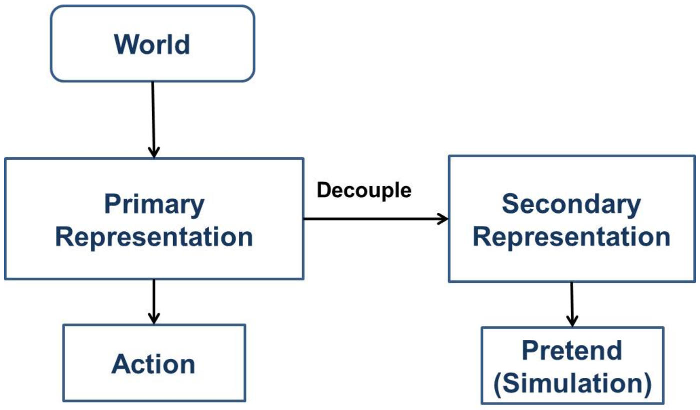


Figure 5: Cognitive Decoupling, adapted by Stanovich (2009) from Leslie (1987).

over the secondary representation that can be manipulated, providing a mechanism for simulation. This process leaves the primary representation intact (Leslie, 1987). During cognitive decoupling, the agent has limited awareness of the present real-world environment, subsequently the response generated by WM overrides the autonomous mind's response (step C in Figure 4).

The ability to maintain decoupling is difficult and costly in terms of cognitive capacity. Stanovich proposes an evolutionary purpose behind this difficulty. He proposes that it is critical for survival not to be unaware of ones present real-world environment for too long (Stanovich, 2009).

The conscious *reflective mind* is effortful, responsible for the call to initiate override (step B in Figure 4), instantiates WM (step E in Figure 4) and provides the secondary representation, in the form of *conceptual knowledge*, to WM for mental simulation (step F in Figure 4). Conceptual knowledge is aggregate declarative memory; memories are abstracted from the perceptual details and aggregated into a meaning of an experience and unimportant details are forgotten. The term *declarative* corresponds to knowledge we are aware we know and can usually describe to others

(Anderson, 2007; Lynott and Connell, 2009; Stanovich, 2009). Conceptual knowledge will be explored in greater detail in Section 2.3.

Working memory (WM) performs mental simulation, i.e., hypothetical thinking, and employs pattern-completion for decision making which requires slow, reflective deliberation. The results of simulation are sent to the reflective mind (step G in Figure 4) for posting the response. WM is active only during cognitive decoupling. Cognitively demanding decision making, using WM, has a higher threshold for activation than reactive autonomous decision making, therefore the conscious level is said to *override* unconscious responses that the reflective mind determines are suboptimal (Stanovich, 2009; Stanovich and Evans, 2013).

2.1.3 Responses.

As illustrated in Figures 3 and 4, Stanovich's framework produces two types of *responses*: unconscious autonomous, based on fast (primary representation) pattern-recognition; and conscious deliberative, based on slow (secondary representation) pattern-completion in WM. An optimal response requires *fully disjunctive reasoning (FDR)* in WM, which is evaluating all available knowledge when selecting options or choosing a response in a reasoning task. Human participant research has shown that humans often do not have the cognitive capacity or inclination to employ *FDR*, resulting in suboptimal decision making (Stanovich, 2009; Wason, 1966). The computational agent developed in this dissertation will employ FDR. Modeling more human-like behavior, using the formalisms developed here, are left as an area for future research.

2.1.4 Discussion.

The literature review of Stanovich's framework defined the functions of the components — the autonomous mind, algorithmic mind, reflective mind, and WM — and their interactions. However, gaps in the literature were identified, leading to additional questions.

2.1.4.1 Gaps in Stanovich's Tripartite Framework.

The review of Stanovich's framework revealed two gaps in the literature (as illustrated in Figure 6) which need to be addressed in order to develop a computational agent. First, a definition of *consciousness.* Second, an adequate description of *conceptual knowledge.*

Throughout the literature *consciousness* is discussed as necessary for reflective deliberation, and even necessary for developing the autonomous memory base (TCLI), yet a definition of consciousness was not found in the review of DPT. Stanovich and Evans acknowledge this gap in the literature, based on "both vague and disputable definitions of consciousness (Stanovich and Evans, 2013)" referencing definitions of consciousness by DPT researchers Churchland (2002) and Dennett (1991). The literature also did not address the detailed processes by which conscious reasoning performs mental simulation, or define the information structures that generate consciousness.

The type of knowledge retained in the reflective mind, and provided to WM for deliberation, is *conceptual knowledge.* The general description of conceptual knowledge in the DPT literature is overly broad for the purposes of developing a computational model. The literature also does not provide a detailed explanation for the process of encoding knowledge captured from stimuli, the recall mechanism or inference generation.

These gaps in the literature raise additional questions:

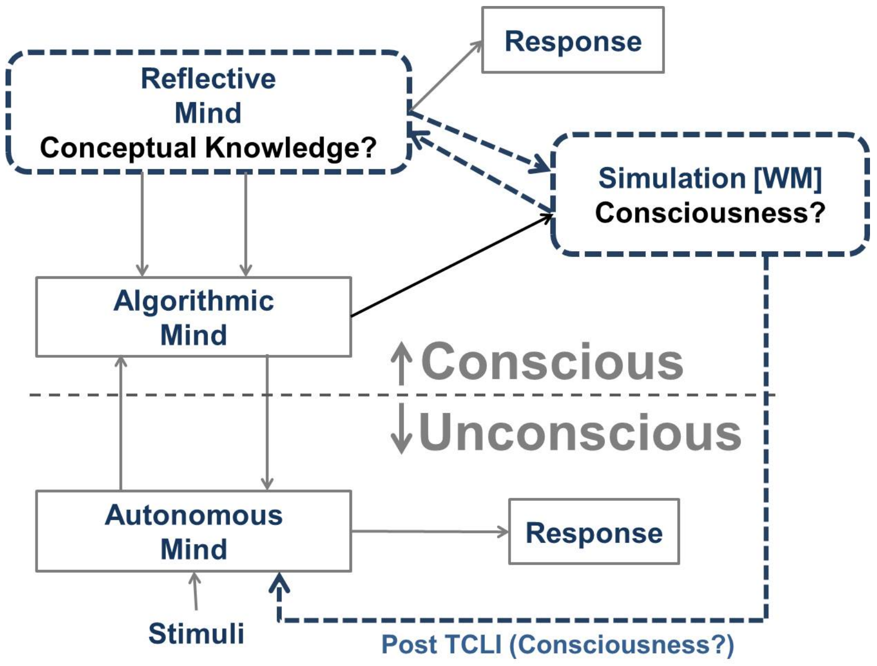


Figure 6: Gaps in Stanovich's Tripartite Framework, illustrated by dashed lines: no definition of consciousness; inadequate details of *conceptual knowledge* and *consciousness* for a computational model. Conceptual knowledge is retained in the reflective mind and provided to WM for mental simulation. Consciousness is necessary for deliberation in WM, and for generating TCLI (Anderson, 2007; Lynott and Connell, 2009; Stanovich, 2009).

- Is there another research area that can fill the *definition of consciousness* gap, and provide specificity for a computational model that can be integrated into the computational agent proposed?
- What are the theoretical structural elements of consciousness, how can they be implemented in software, and are there existing tools that can be adapted to support the implementation?
- What is the structure of conceptual knowledge, how can it be algorithmically implemented and coded in software and are there existing tools that can be adapted to support the implementation?

- How does conceptual knowledge relate to consciousness, and can they be modeled as the same structures for the purposes of this research?
- How can the structural elements of consciousness and conceptual knowledge be processed-over to produce an inference, pattern-completion?

These additional questions will be answered in the remaining literature review and following chapters.

2.1.4.2 Functions of Stanovich's Tripartite Framework that Address the Research Hypothesis.

The research suggests functions of Stanovich's framework that may overcome some limitations of supervised classifiers: all experiences (analogous to both test and training samples) are captured in the reflective mind through preattentive processes, analogous to retaining test sample predictive variables, as well as training samples, contributing to predictions of subsequent test samples; once an appropriate response (analogous to a test sample/response variable pair) is learned in WM it is posted to the autonomous mind as TCLI, therefore providing a fast response to the same stimulus in the future, mitigating the requirement of redundant classification. However, in an incremental learning environment a response may become obsolete or incorrect over time as new samples are incorporated. The proposed capability of reusing responses raises an additional concern:

- Is there a mechanism in the autonomous mind which identifies obsolete responses, or allows for responses to be removed from the autonomous mind over time, and how can it be implemented?

A partial answer to this question was found in the cognitive modeling literature. Memories fade over time when they are not recently or frequently recalled. The cognitive mechanism that controls this phenomenon is called an activation level (Anderson, 2007). By modeling memory activation levels based on recency, memories (TCLI) posted to the autonomous mind can fade with time and be *forgotten*.

2.2 Consciousness

Literature on DPT, and specifically Stanovich's framework, propose that consciousness is necessary for deliberative decision making, and even for developing the primary knowledge base of the unconscious, reactive processes (TCLI). However, as acknowledged by Stanovich and Evans, there is no established definition of consciousness in the DPT literature due to vague and disputable definitions. This literature review will address that gap by identifying a definition of consciousness which is well defined and based on contemporary research.

2.2.1 A Philosophical Foundation of Consciousness.

The study of consciousness has it's foundation in philosophy. The French philosopher René Descartes (1596–1650) defined consciousness as reflexive thought and self-awareness (Van Gulick, 2014). The English philosopher and physician John Locke (1623–1704) asserted that humans were born with a *blank slate* and consciousness gradually unfolds as sensations and perceptions are experienced (Uzgalis, 2015). Immanuel Kant (1724–1804) argued that phenomenal consciousness is an integrated group of experiences, unique to the subject, and it requires a rich structure of mental organization (Brook, 2013). American biologist and Nobel laureate Gerald Edelman (1929–2014) posited that past and present conscious experiences, as well as a hypothetical future, are, in fact, *imagined* past experiences, *imagined* present experiences, as well as, an *imagined* future. It is Edelman's position that conscious experiences are all imagined and are generated by a repertoire of qualia[1] states corresponding to a set of experiences, in which new experiences are integrated into preexisting circuitry to generate *consciousness* (Edelman, 1989).

Despite the lack of consensus of a precise definition of consciousness, there is some agreement on the principle features of consciousness, which include: (1) some form of phenomenal structure (i.e., the order and organization of our internal representation of experiences); (2) unity (i.e., the integration of diverse elements of conscious content); (3) it is a dynamic process; and (4) throughout the literature (and in this dissertation) consciousness is discussed synonymously with *experiences* (Van Gulick, 2014).

2.2.2 A Definition of Consciousness.

The definition of consciousness selected as a basis for this research supports Edelman's position that "Qualia are the *phenomenal contents* of experiences", and is also based on the teaching of the American philosopher, John R. Searle (Searle et al., 1997). Christopher Williams Cowell, a student of Searle's, proposes "Consciousness is the experiencing of qualia. A system must continue to experience qualia if it is to remain conscious; any periods during which no qualia are experienced are periods in which the system has lost consciousness (Cowell, 2001)".

This definition of consciousness is further supported by more recent theories of consciousness. Arrabales et al. (2009), Samsonovich and Nadel (2005) and Tononi (2012) also propose that the fundamental structural elements of individual conscious experiences are considered to be qualia.

2.2.3 Qualia.

Qualia are not physical, objective properties, such as hues, angles, or speed of movement that can be measured, but rather qualia are the subjective (*agent-centric*

[1]Qualia are defined in Section 2.2.3.

when discussing computational models), nonphysical, phenomenal qualities that are experienced consciously from the interaction with stimuli. Although qualia are often discussed as being synonymous with emotion, they are not limited to emotions. All conscious experiences are represented by qualia: sensory experiences from stimuli, perceptions, bodily sensations, moods, and emotions (Tye, 2015). It is proposed by Rogers et al. (2003), that the purpose of qualia is to make it unnecessary for an agent to remember the high-bandwidth details of specific memories, enabling an agent to react to a new stimulus, quickly and intelligently, with bit-reduced qualia representations.

These definitions of qualia, and consciousness, are too abstract to support a computational formalism, therefore literature of contemporary theories of consciousness, based on the definition accepted for this dissertation, were reviewed.

2.2.4 Integrated Information Theory (IIT) of Consciousness.

Systematic reviews (Gamez, 2008; Reggia, 2013) reveal three contemporary theories of consciousness based on qualia as the fundamental structural elements of consciousness: Internal self-model (Samsonovich and Nadel, 2005), Higher-level representations (Arrabales et al., 2009), and Integrated Information Theory (IIT) of Consciousness (Balduzzi and Tononi, 2009; Tononi, 2012). Most applicable to the approach in this research is IIT, because it provides a description of qualia in a non-ontological, non-hierarchical qualia space (QS). Balduzzi and Tononi provide adequate detail in their theory for a computational model of *integrated information generated by a complex of elements*. Therefore, the theoretical model of *consciousness*, selected for modeling consciousness in WM is primarily based on IIT, as illustrated in Figure 7.

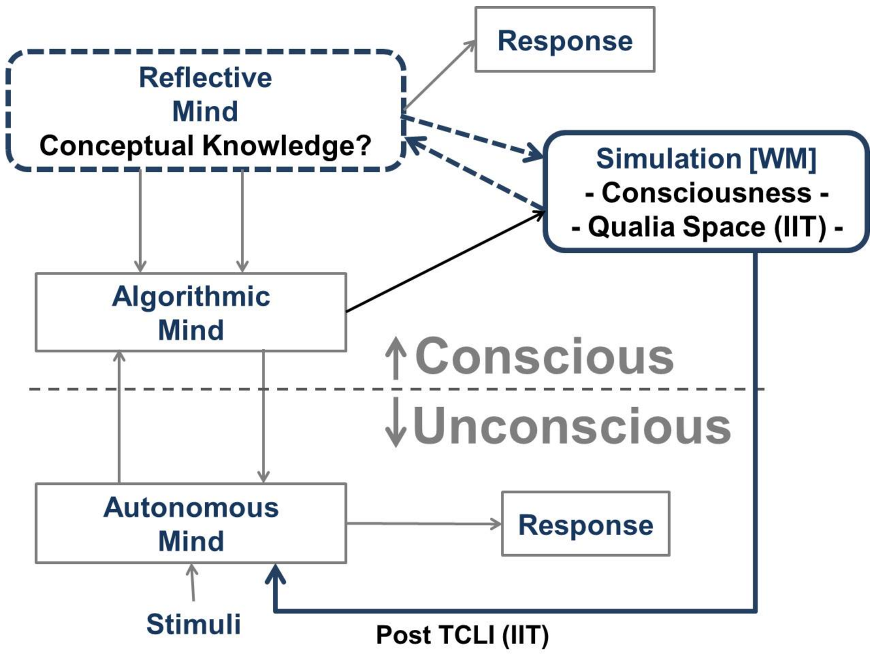


Figure 7: Qualia and the Integrated Information Theory (IIT) of Consciousness Fill Gaps in Stanovich's framework. Consciousness is the experiencing of qualia, reasoned-over in a non-ontological, non-hierarchical QS (Cowell, 2001; Balduzzi and Tononi, 2009).

2.2.5 Discussion.

There are two research areas that, together, provide a definition and description of *consciousness*, with specificity for a computational model: philosophy and cognitive theories of consciousness. The *philosophical* definition of consciousness states that consciousness is the *experiencing of qualia.* A model of qualia, with specificity for a computational model, was found in *cognitive theories of consciousness*, specifically, the Integrated Information Theory (IIT) of Consciousness. How IIT can be computationally modeled and integrated into the computational agent, as proposed by this research, is not answered in the literature review.

Aleksander and Gamez (2011) developed a performance metric by implementing the *maximally integrated information structures* of IIT as defined by Balduzzi and Tononi (2009). The results of their research demonstrated that the integrated information structures, modeled precisely as defined by Balduzzi and Tononi, are not computationally tractable with current technology. A feature space of 30 elements (analogous to 30 binary variables in a data set) would require 10^{10} years to fully analyze, i.e., perform *FDR*. No computationally tractable solutions were found that can be adapted to implement QS in software.

The literature review further revealed that *conceptual knowledge* is retained in the reflective mind and provided to QS, that then generates consciousness. A literature review of *conceptual knowledge* was pursued in hopes of finding a computationally tractable formalism with which to model it, and to better establish how the structural elements of *conceptual knowledge* are related to consciousness.

2.3 Conceptual Knowledge

The literature reveals that the type of knowledge retained in the reflective mind, and provided to WM for mental simulation in QS, is *conceptual knowledge* (Anderson, 2007; Lynott and Connell, 2009; Stanovich, 2009). Some features of conceptual knowledge are: (1) memories are abstracts from the perceptual details and encoded into a meaning of an experience; (2) memory for detail is available initially but is forgotten rapidly, whereas memory for meaning is retained; (3) unimportant details of previous experiences are forgotten, and we retain abstractions of experiences; and (4) we abstract from specific experiences to general categories of the properties of that class of experiences (Anderson, 2005; Mandler and Ritchey, 1977; Wanner, 1974).

Additional theories of conceptual knowledge (Bartlett, 1932; Rumelhart et al., 1976) propose that it is retained in *schemata*. In schemata knowledge is represented in a structure of related concepts, and is a network of interrelationships of knowledge, as opposed to specific instances of experiences. Furthermore, schemata can be used to make inferences about specific instances of the abstract concepts they represent (Anderson and Pearson, 1988).

One of the three theories of consciousness discussed in section 2.2.4, the Internal self-model (Samsonovich and Nadel, 2005), also proposed that qualia are retained in *schemata*, specifically as defined by Bartlett (1932). (No computational models based on the Internal self-model were found in the literature.) This finding brings the research full-circle answering questions raised in the literature review, revealing that the theoretical structure of *conceptual knowledge*, as well as the structural elements of *QS*, have been based on schemata, as illustrated in Figure 8.

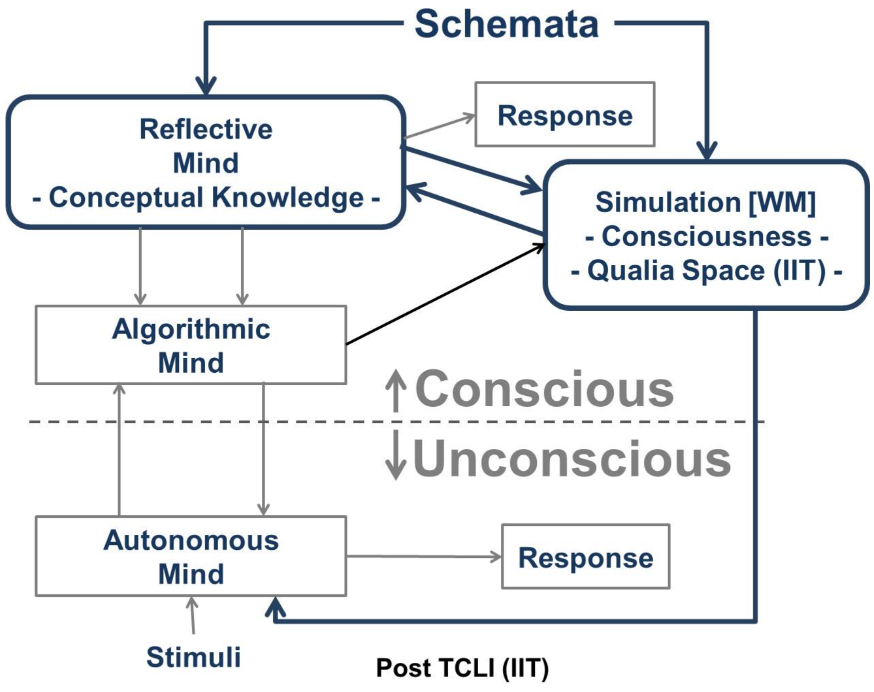


Figure 8: Schemata Fill Gaps in Stanovich's framework. The structural elements of *conceptual knowledge*, as well as the structural elements of *QS*, have been based on schemata. Furthermore, schemata can be used to make inferences about specific instances of the abstract concepts they represent (Anderson and Pearson, 1988; Samsonovich and Nadel, 2005).

2.4 Summary

This chapter presented a review of the relevant literature, seeking both a theoretical framework and tools for implementing the proposed computational agent. A solid foundation of integrated theoretical concepts was established, however, the literature did not reveal any tools for implementing the integrated theoretical framework in a computational agent.

The literature review of DPT and Stanovich's framework identified the functions of the autonomous mind, algorithmic mind, reflective mind and WM, as detailed in Section 2.1.4. However, two gaps in the literature on DPT and Stanovich's framework were identified. First, an accepted definition of consciousness was not found. Stanovich and Evans acknowledge this gap in the literature, due to vague and disputable definitions of consciousness. Second, a description of conceptual knowledge, with adequate specificity for a computational model was not found. As a result, the literature also did not address the detailed processes by which conscious reasoning performs mental simulation, or define the information structures that generate consciousness.

These gaps in the literature raised additional research questions, that were subsequently addressed in a literature reviews of consciousness and conceptual knowledge:

- There are two research areas that, together, provide a definition of consciousness, with specificity for a computational model: philosophy and cognitive theories of consciousness. The definition of consciousness from philosophy, states that consciousness is the *experiencing of qualia.* A model of qualia, with specificity for a computational model, comes from the research area of cognitive theories of consciousness, specifically, the Integrated Information Theory (IIT) of Consciousness. How IIT can be computationally modeled and integrated into the computational agent, as proposed by this research, is not answered in the literature review.
- The theoretical structural elements of consciousness were identified as qualia, reasoned-over in QS. No computationally tractable solutions or tools were found that can be adapted to implement of QS in software.
- The structure for conceptual knowledge, as well as the structure for the *maximally integrated information structures* of QS, are based on schemata — a network of interrelationships of knowledge.

The literature did not reveal how the structural elements of conceptual knowledge or QS, can be processed-over to produce an inference, pattern-completion, ideally performing *FDR*. The literature review also does not address how can each of Stanovich's framework components be algorithmically implemented and coded in software, and did not identify any tools that can be adapted to support the functions of the autonomous mind, algorithmic mind, reflective mind or WM, which are necessary for a complete computational model of the framework.

The research suggests functions of Stanovich's framework that can overcome some limitations of existing supervised classifiers, when modeled in a computational agent, such as: all experiences are captured in the reflective mind through preattentive processes, effectively retaining test sample predictive variables, as well as training samples, contributing to predictions of subsequent test samples; once a response is learned in WM it can be posted to the autonomous mind as TCLI, therefore providing a fast response to the same stimulus in the future, mitigating the requirement of redundant classification. However, in an incremental learning environment a response may become obsolete or incorrect as new training samples are incorporated.

The capability of retaining responses, which may become obsolete or incorrect as new training samples are incorporated, raises an additional concern, which was answered with a review of the cognitive modeling literature:

- Memories have activation levels based on recency and frequency (Anderson, 2007). By modeling the activation levels, based on recency, memories (TCLI) posted to the autonomous mind can fade with time and be *forgotten*. However, no solution was found in the literature review for implementing this mechanism.

The following Methodology Chapter will provide a description of the steps taken to implement a computational agent based on the theoretical framework presented in the present chapter. Specifically, a functioning model of Stanovich's framework with a computationally tractable working solution for a representation of qualia, inspired by

IIT, as the conscious component. Chapter 4 will present the results of this research, as compared against analogous supervised learning classifiers. The Chapter 5 will present the findings, conclusions, the contributions of this dissertation and areas for future research are explored.

III. Methodology

Chapter 1 introduced the motivation for this research, the hypothesis and research questions addressing the hypothesis. Chapter 2 reviewed the relevant literature and revealed functions of Stanovich's tripartite framework, which may lead to a computational agent that supports the research hypothesis. The hypothesis proposes that a computational agent, inspired by a theory of human learning-by-experience (Stanovich's framework), can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes. Previous machine learning (ML) approaches supporting pattern-completion (i.e., predicting the class labels for new samples), require re-training when the target variable changes. Chapter 2 also revealed a computational theory of consciousness, Integrated Information Theory (IIT) of Consciousness, providing a framework for *how* knowledge is represented, retrieved and reasoned over in the conscious experience base, and a theory of conceptual knowledge, which provides a framework for *what* knowledge is represented, retrieved and reasoned over in the conscious experience base. The IIT is based on *qualia* as the fundamental structural elements of consciousness, therefore, the cognitively inspired computational agent developed in this chapter will be the Qualia Modeling Agent (QMA).

This chapter details the steps taken, mathematical formalism and tools used to develop the QMA, address the hypothesis and remaining research questions. The first section (Section 3.1) reviews the theoretical basis identified in the literature review, proposes an abstract representation of consciousness for modeling the conscious level of the framework, and proposes a Cognitive Modeling Architecture (CMA) for modeling the unconscious level of the framework. The next section (Section 3.2) presents an overview of the methodology used to implement the QMA. The remaining sections provide the details of how the functional modules were developed and integrated into a complete computational algorithm: the human-computer interface module (Section 3.3), the autonomous module (Section 3.4), the algorithmic module (Section 3.5) and the QS Computational Module (Section 3.6). The QS Computational Module incorporates functionality of both the reflective mind and working memory (WM).

3.1 Extending the Theoretical Basis

The QMA is primarily based on Stanovich's tripartite framework, a cognitive theory of learning and decision-making. The goal of the literature review was to reveal a more complete mechanism by which conscious reasoning performs mental simulation, and reasons over the important details of experiences in a non-ontological, non-hierarchical structure. Stanovich's framework did not completely satisfy this goal. Gaps in Stanovich's framework were addressed by other lines of research, resulting in a theoretical framework which includes: qualia and IIT, conceptual knowledge and schemata, illustrated in Figure 9.

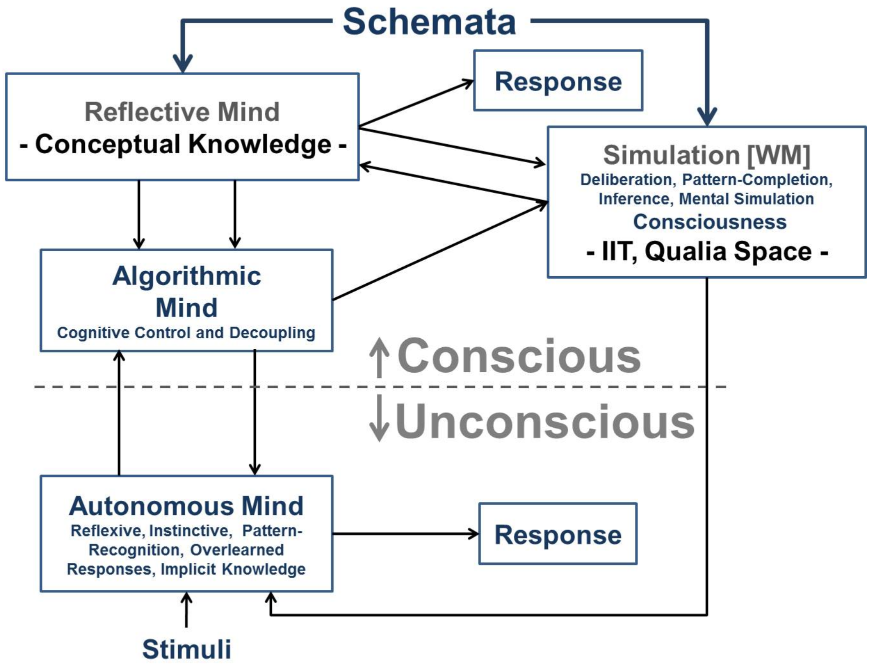

Schemata
Response
Reflective Mind
- Conceptual Knowledge -
Simulation [WM]
Deliberation, Pattern-Completion, Inference, Mental Simulation
Consciousness
- IIT, Qualia Space -
Algorithmic Mind
Cognitive Control and Decoupling
↑Conscious
↓Unconscious
Autonomous Mind
Reflexive, Instinctive, Pattern-Recognition, Overlearned Responses, Implicit Knowledge
Response
Stimuli

Figure 9: The QMA Theoretical Basis. Stanovich's Tripartite Framework: three *minds*, WM and their interactions. Schemata as a formalism to represent both conceptual knowledge and an abstract interpretation of qualia space (QS) as proposed by IIT.

The literature review identified the functions of Stanovich's framework *minds* and WM, as well as, the interactions between the components but did not reveal how these functions can be implemented in software. The review identified some functions and interactions, that when computationally modeled, may overcome the specific limitations of supervised classifiers identified in the hypothesis. They are presented here, with the proposed implementation approach.

3.1.1 Hypernetwork Theory Model of QS.

One line of research, *Models of Consciousness*, revealed a theory that the fundamental structural elements of individual conscious experiences are considered to be qualia, which are reasoned-over in a non-ontological, non-hierarchical QS. A separate line of research, *Cognitive Modeling*, addressing the *type* of knowledge maintained in the conscious experience base, revealed a theory that conceptual knowledge is maintained in the conscious level. QS is *how* knowledge is represented, retrieved and reasoned-over. Conceptual knowledge is *what* knowledge is represented, retrieved and reasoned-over. These separate lines of research are both based on an earlier theory of *schemata*, as defined by Bartlett (1932). This link in the research suggests a mathematical formalism applied to computationally modeling schemata may prove successful as an approach for modeling QS and conceptual knowledge. Further research revealed Young (1998) modeled elementary elements of schematic memory using the mathematical formalism of hypernetwork theory[1] Young's formalism did not *maximally integrate* the structural elements, perform pattern-completion or make inferences about specific instances of abstract concepts. Hypernetwork theory will be extended in this dissertation to meet those requirements.

Rogers et al. (2008) propose that the conscious representation is constrained to maintain relationships, as opposed to sensory values. The hypernetwork approach achieves that constraint — resulting in a stable, consistent and useful representation. Therefore, hypernetwork theory will be used as a formalism for representing, retrieving and reasoning-over a representation of conceptual knowledge in QS. Hypernetwork theory is a theory which extends network theory to multidimensional hypernetworks for modeling relationships between qualitative data, psychological and social relations, in particular systems in nature, society and cognition. Also referred to as Polyhedral dynamics in earlier literature (Casti, 1977; Johnson, 2013; Wang et al., 2010)

Retaining the *Important details of experiences* in a knowledge base is analogous to retaining both training and test samples with no particular variable defined as

[1]At the time of Young's research, hypernetwork theory was referred to as polyhedral dynamics.

the target variable. By modeling this structure using hypernetwork theory the agent should be able to classify new test samples with any variable defined as the *target variable* for each test sample, therefore addressing the research hypothesis.

3.1.2 Activation levels Proposed to Allow Responses to Obsolesce.

The literature also revealed that once a response (analogous to a test sample/response variable pair) is learned in WM it may be posted to the autonomous mind as Tightly Compiled Learned Information (TCLI), therefore providing a fast response to the same stimulus in the future, mitigating the requirement of redundant classification. However, in an incremental learning environment, a response may become obsolete or incorrect over time as new training samples are incorporated. Another feature of the autonomous mind addresses this concern. Memories fade over time when they are not recently or frequently recalled. The cognitive mechanism that controls this phenomenon is called an activation level (Anderson, 2007). By modeling memory activation levels based on recency, memories (TCLI) posted to the autonomous mind can fade with time and be *forgotten*. Activation levels, as well as the primary autonomous function of pattern-recognition, can be implemented with a CMA, specifically, Adaptive Control of Thought–Rational (ACT–R). CMAs and ACT–R are discussed in more detail in Appendix C.

3.2 Overview of the Implementation

This present chapter transitions from the theoretical basis to implementation of the cognitively inspired agent, a ML computational algorithm, as illustrated in Figure 10.

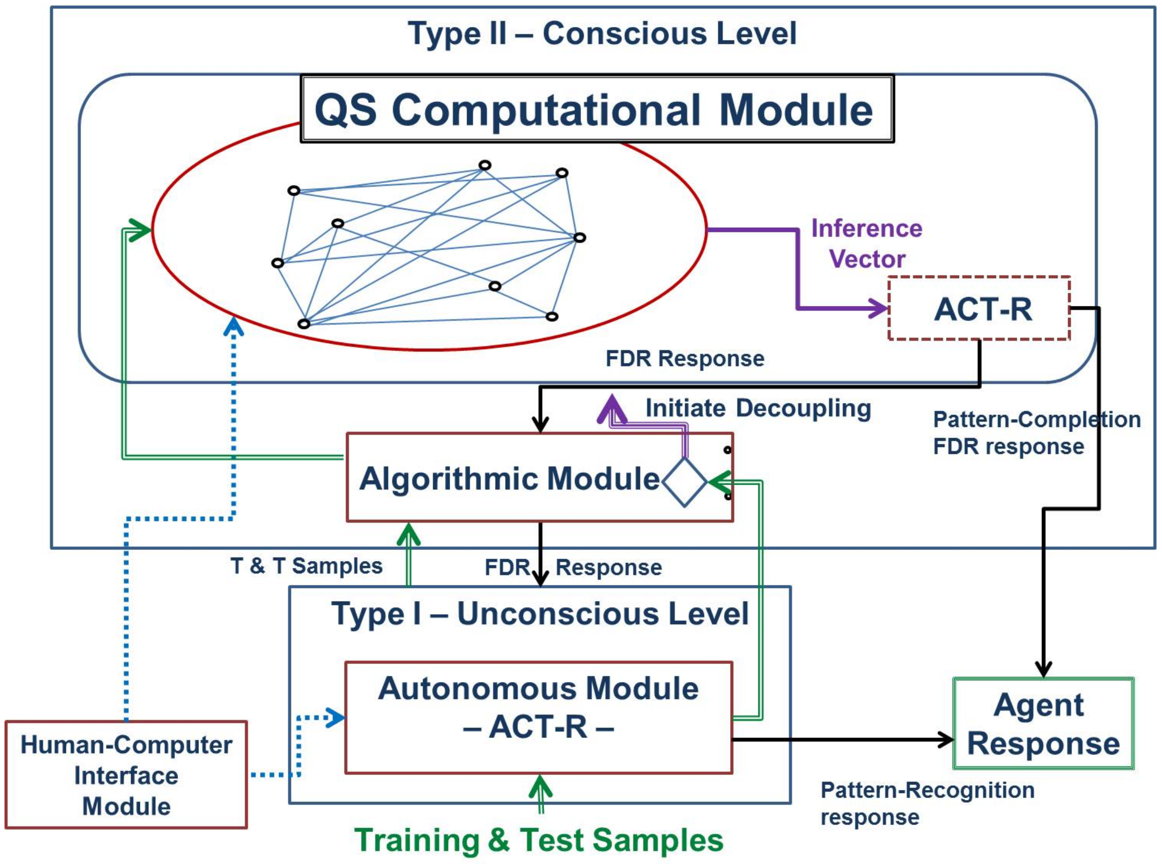

Type II – Conscious Level
QS Computational Module
Inference Vector
ACT-R
FDR Response
Initiate Decoupling
Pattern-Completion FDR response
Algorithmic Module
T & T Samples
FDR Response
Type I – Unconscious Level
Autonomous Module – ACT-R –
Agent Response
Human-Computer Interface Module
Pattern-Recognition response
Training & Test Samples

Figure 10: Overview of the Cognitively Inspired Computational Agent. The *Autonomous Module* is an extended ACT–R model. The *QS Computational Module* incorporates the functionality of both the reflective mind and WM and is implemented with hypernetwork theory. A second ACT–R model is integrated to convert the hypernetwork theory vector representation into a *chunk* for posting responses to the Autonomous Module. The *Algorithmic Module* and all other interactions are custom software written in Common Lisp (CL).

Terminology used in this chapter will adapt to the change in focus. The *minds* and WM will be modeled with software *modules*. Stimuli and the *important details of experiences* are modeled by the predictive variables in training and test samples from categorical data sets. TCLI is modeled by *responses* (test sample/response variable pairs) produced by fully disjunctive reasoning (FDR) in the QS Computational Module.

The Autonomous Module consists of an ACT–R model extended to interact with the Algorithmic Module and Human-Computer Interface Module. The *QS Computational Module* models the combined functionality of the reflective mind and WM, implemented in hypernetwork theory. The *QS Computational Module* also contains an atypical ACT–R model which converts the hypernetwork theory vector representation of an inference into a *chunk* for posting responses to the Autonomous Module. The *Algorithmic Module*, Human-Computer Interface Module and interactions are custom software written in CL. The primary modules will be discussed in more detail below.

3.3 The Human-Computer Interface Module

The Human-Computer Interface Module allows a domain expert to change attributes in training and test samples, which have already been incorporated in both the autonomous knowledge base and the QS Computational Module, without retraining. This feature allows a domain expert to correct an error, or accept a response variable as truth, therefore allowing the predictive values in the response variable to contribute to subsequent inferences.

3.4 The Autonomous Module

ACT–R is the CMA selected with which to implement the Autonomous Module, as illustrated in Figure 11. ACT–R models sensory perception, pattern-recognition and configurable activation levels based on human participant research. Activation levels allow memories to fade with infrequency or lack of recency, eventually becoming forgotten. ACT–R is open-source and can be extended, through custom software, to model the interactions between the Autonomous Module and other components. Technical details, and terminology, for ACT–R are available in Appendix C.

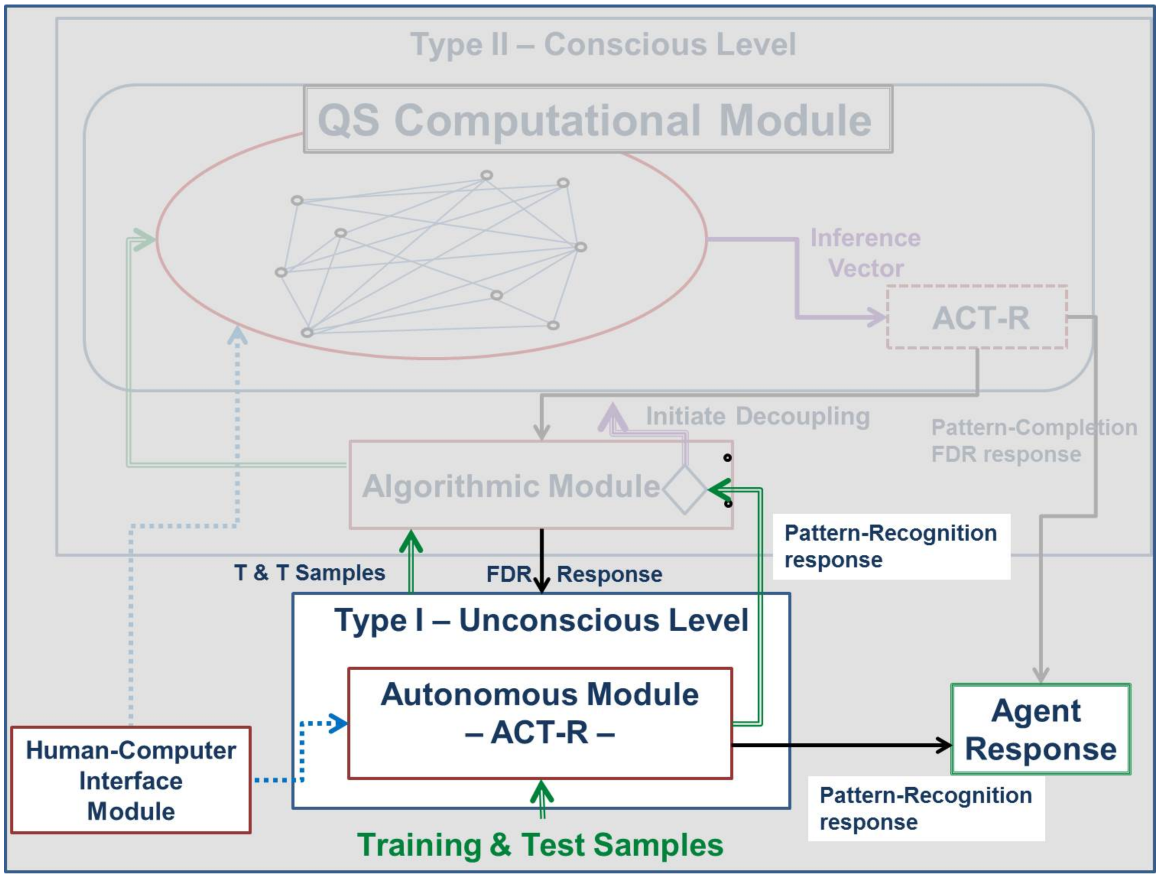
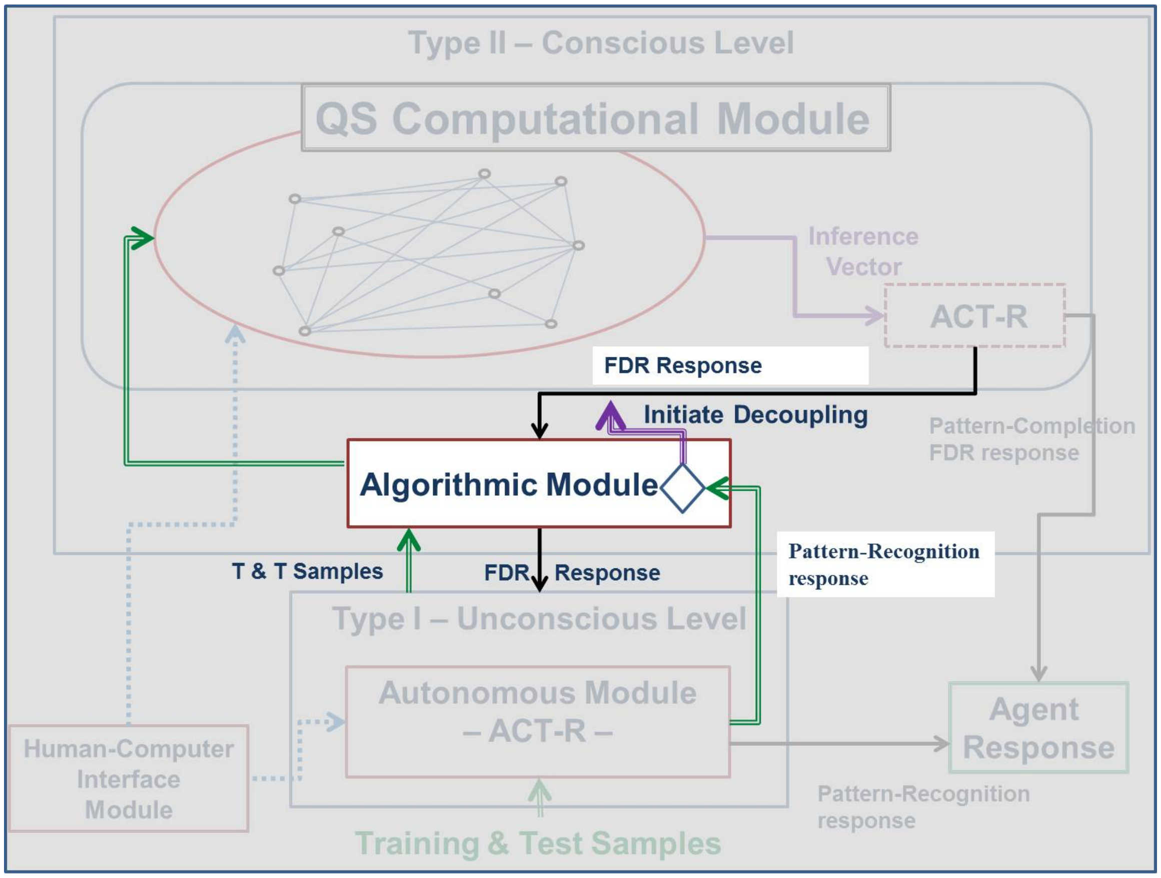

Type II – Conscious Level
QS Computational Module
Inference Vector
ACT-R
Initiate Decoupling
Pattern-Completion FDR response
Algorithmic Module
Pattern-Recognition response
T & T Samples
FDR Response
Type I – Unconscious Level
Autonomous Module – ACT-R –
Agent Response
Human-Computer Interface Module
Pattern-Recognition response
Training & Test Samples

Figure 11: Type I — Unconscious level, Autonomous Module. The Autonomous Module is the QMA interface to the training and test samples, and generates an immediate agent response if there is pattern-recognition in the autonomous knowledge base. This module forwards the preattentive processes, represented by green double arrows — training and test samples and pattern-recognition response — to the Algorithmic Module, and receives responses, represented by solid black arrows, from the Algorithmic Module.

The Autonomous Module will: perform pattern-recognition over previously stored (FDR) responses to produce a fast response; perform preattentive processes, i.e., send training and test samples to the Algorithmic Module; accept reprogramming (FDR responses) from inferences generated in the QS Computational Module; and, allow FDR responses to obsolesce over time.

3.5 The Algorithmic Module

The processes and interactions for the Algorithmic Module are illustrated in Figure 12. This module is the gate-keeper of QMA, responsible for sequencing processes.



Type II – Conscious Level
QS Computational Module
Inference Vector
ACT-R
FDR Response
Initiate Decoupling
Pattern-Completion FDR response
Algorithmic Module
Pattern-Recognition response
T & T Samples
FDR Response
Type I – Unconscious Level
Autonomous Module – ACT-R –
Agent Response
Human-Computer Interface Module
Pattern-Recognition response
Training & Test Samples

Figure 12: Type II — Conscious level, Algorithmic Module. The Algorithmic Module is the gate-keeper of Stanovich's framework, responsible for sequencing behavior. It forwards the preattentive process, represented by a green double arrow, — training and test samples and pattern-recognition response — from the Autonomous Module to the QS Computational Module, and forwards conscious responses, represented by solid black arrows, from the QS Computational Module to the Autonomous Module. This module also initiates cognitive decoupling, which allows the QS Computational Module to override the Autonomous Module response.

This module forwards training and test samples to the QS Computational Module for incorporation in the maximally integrated information structures in QS Computational Module. When there is no pattern-recognition response from the Autonomous Module, the Algorithmic Module initiates decoupling, allowing the QS Computational Module to generate an inference. This module also forwards responses, generated by QS Computational Module, to the Autonomous Module for incorporation in autonomous knowledge base for subsequent pattern-recognition.

3.6 QS Computational Module

The QS Computational Module, illustrated in Figure 13, is the most complex module of the QMA and a significant contribution of this dissertation, requiring detailed discussion.

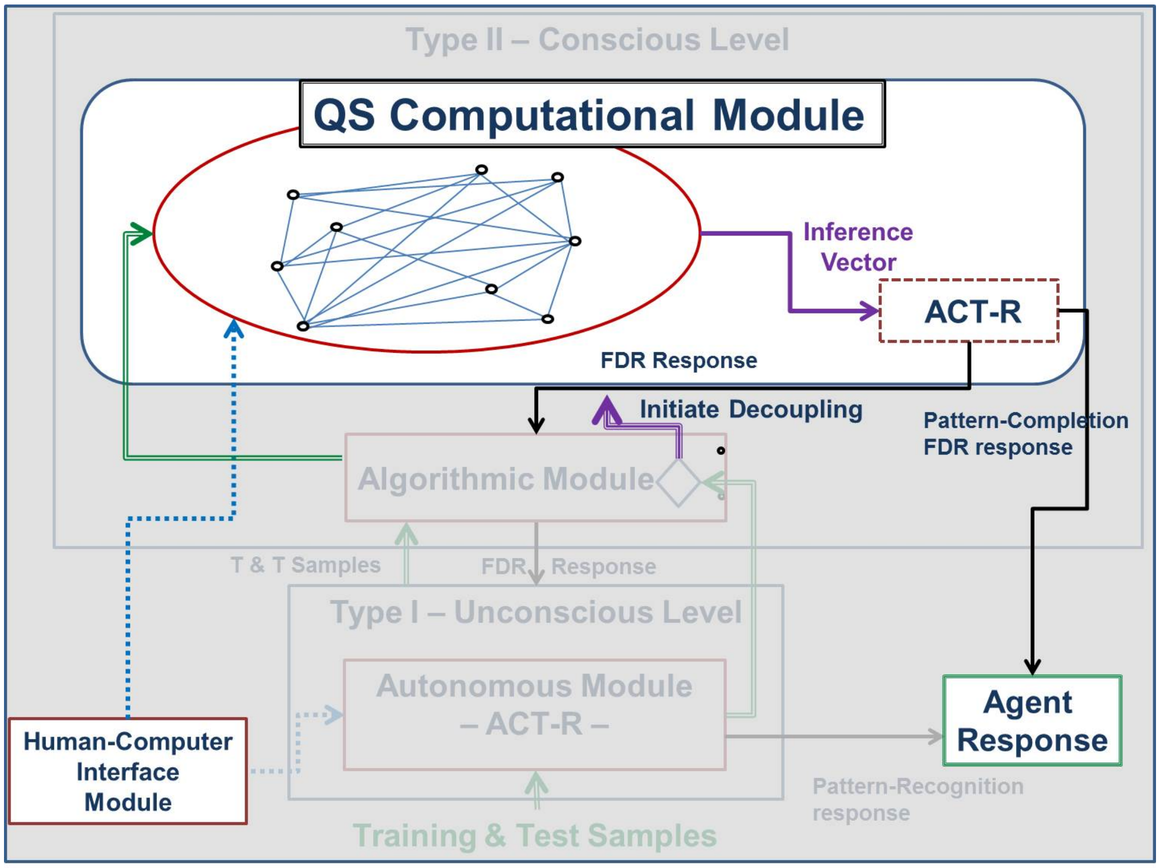

Type II – Conscious Level
QS Computational Module
Inference Vector
ACT-R
FDR Response
Initiate Decoupling
Pattern-Completion FDR response
Algorithmic Module
T & T Samples
FDR Response
Type I – Unconscious Level
Autonomous Module – ACT-R –
Agent Response
Human-Computer Interface Module
Pattern-Recognition response
Training & Test Samples

Figure 13: Type II — Conscious Level, QS Computational Module. The QS Computational Module incorporates functionality of the reflective mind and WM. The triple purple arrow represents the initiation of decoupling, which triggers the QS Computational Module inference generating process. The green double arrow represent the preattentive process — training and test samples and pattern-recognition response — forwarded from the Algorithmic Module to QS Computational Module. The dotted blue arrow represents input from the human-computer interface allowing for updates or corrections of data entries. The black solid arrows leaving the QS Computational Module represent the resulting inference vector converted into an ACT–R chunk for saving to the autonomous knowledge base and the agent response.

The QS Computational Module is a mathematical formalism which incorporates the functionality of two conscious level components, the reflective mind and WM. The QS Computational Module is implemented with hypernetwork theory, and extensions of hypernetwork theory[2].

[2]A more detailed discussion of *hypernetwork theory* is provided in Appendix B.

3.6.1 A Computationally Tractable QS Representation.

As discussed in Section 2.2.5, modeling the maximally integrated information structures of QS, as proposed by Balduzzi and Tononi (2009) was proven to be computationally intractable given the current state of technology, as illustrated in Figure 14.

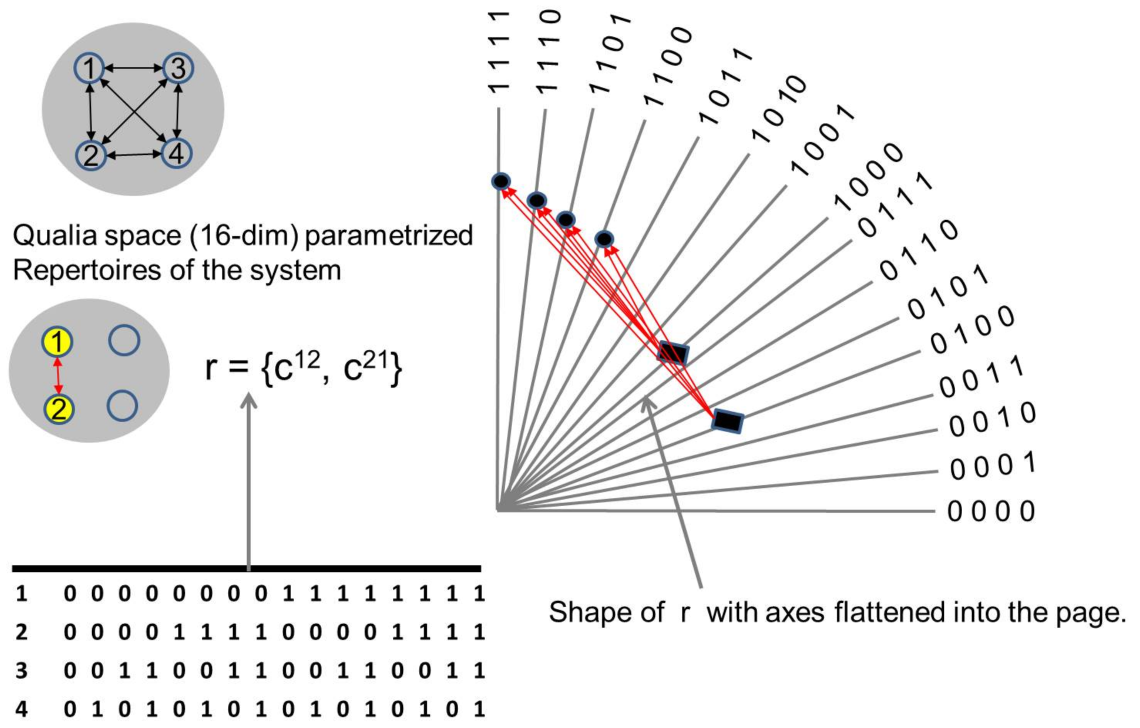
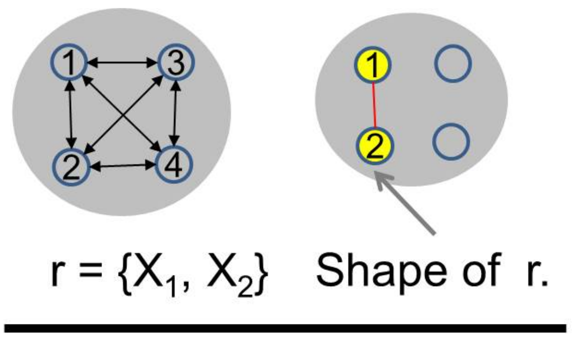


Figure 14: Dimensionality of QS as Defined by Balduzzi and Tononi (2009). QS for a system of 4 elements is 16-dimensional, *with an axis for each of the* 2^4 *possible states of the complex.* Notation c^{ij} refers to a connection from element n^i to n^j, r is a subset of possible connections (Balduzzi and Tononi, 2009). Dimensionality of QS in this formalism grows exponentially as elements are added to the system, i.e, for a system of 5 elements QS is 2^5, 32-dimensional, etc.

Therefore, a computationally tractable mathematical formalism, which could model an abstract, i.e., less precise, representation of QS is required, as illustrated in

Figure 15.



X_1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X_3	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X_4	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Figure 15: Dimensionality of QS as implemented in this dissertation. QS for a system of 4 elements is at most 3-dimensional, *with an axis for each of the 4 elements*, and 2^4 possible states of the complex, r is a subset of possible connections. Dimensionality is equal to the number of axes minus one. By means of comparison to Figure 14, dimensionality of QS in this formalism grows linearly as elements are added to the system.

The computationally tractable representation of QS, illustrated in Figure 15, is modeled using hypernetwork theory. Hypernetwork theory provides a hypergraph representation for high-order relationships between elements, whether they be the important details of individual experiences or the values of predictive variables in ML training and test samples. Furthermore, hypernetwork theory is extended, by this dissertation, with a weighted distance measure to generate pattern-completion and a hypothetical inference.

3.6.2 Key Terms and Concepts.

The mathematical formalism of the QS Computational Module will be explained by means of examples, beginning in Section 3.6.4. The following are key terms and concepts used in the examples.

3.6.2.1 Variables and their Attributes.

An *attribute* is a specific value of a predictive or response *variable*. Put another way, a variable is a logical set of attributes. For example, the variable *gender* is a logical set of two attributes, male and female (Babbie, 1998). In the formalism presented in this dissertation all variables can be represented as missing, or unobserved. Therefore, for gender there are three possible attributes: male, female and unobserved.

3.6.2.2 Samples.

There is no differentiation between training and test sample vector representations once they have been incorporated in the QS Computational Module hypernetwork theory formalism. Test samples will have at least one unobserved element, but training samples may also have unobserved elements. As a convention in this dissertation, the term *sample*, is taken to include both training and test samples.

3.6.2.3 Ordered Binary Representation.

Before training or test samples can be processed by QS Computational Module they have to be converted to ordered binary representations, also known as binarizing (Lourenco et al., 2004), discussed in detail in Appendix E.1. Converting the data set to an ordered binary representations is effectively the training portion of the QS Computational Module algorithm. An explanation of this process is included in the example in Section 3.6.4.

3.6.2.4 Query Element.

The Query Element is the specific element that is to be inferred. A test sample can have multiple unobserved elements, but only one (at a time) can be a Query Element. Alternatively, the domain expert may choose to make an *observed element* the Query Element, therefore generating inferences from an imagined test sample. Imagined test samples are discussed in more detail in Section 3.6.6.

3.6.2.5 Target Variable.

Once the Query Element is selected, the *target variable* is identified as the target of the inference/pattern-completion formalism, whose value is to be inferred. The term *target variable* is also referred to as *class* or *category* in ML terminology (Pang–Ning et al., 2006).

3.6.2.6 Concept Drift and Transfer Learning (TL).

A phenomena, particular to incremental learning, is *concept drift*, which refers to a learning problem that changes over time. In particular, the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways (Žliobaitė, 2010). Concept drift is discussed in more detail in Appendix D.4.2.

Transfer Learning (TL) is the ability of a system to apply knowledge or skills learned in previous tasks to subsequent tasks or new domains, which are similar in some way. Concept drift is one method in which TL can be achieved (Pan and Yang, 2010). TL is discussed in more detail in Appendix D.8.

3.6.2.7 Attributes Conceptualized as Simplices and Polyhedra.

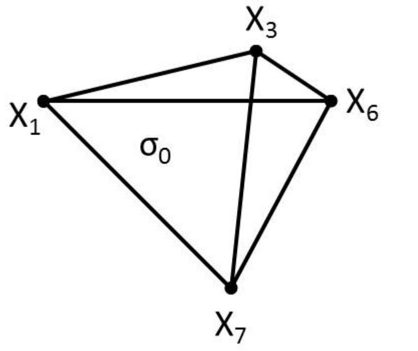
Once the *target variable* is selected, each ordered binary representation (sample) can be conceptualized as a simplex. A simplex (for example, σ_0 in Table 1) and its polyhedron geometric representation, a polyhedron (for example, σ_0 in Figure 16a) represent the attributes of a unique sample.

Table 1: Soybean Sample in Simplicial Complex Form, Given *Disease* as Target Variable.

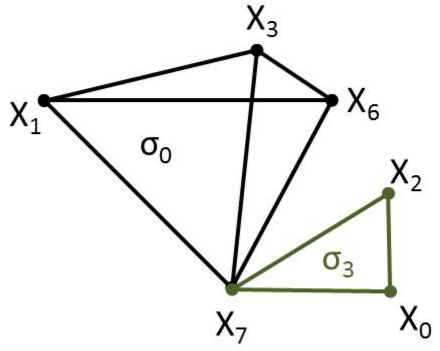
	(notional) Attribute Set											
	disease	hail		seed -discolor		precipitation			leaf -mildew			
	Simplex	present	absent	present	absent	lt normal	normal	gt normal	absent	upper-surface	lower-surface	
ID	Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	dim
σ_0	bact.-blight	0	1	0	1	0	0	1	1	0	0	(3)
σ_1	bact.-blight	0	1	1	0	0	0	1	1	0	0	(3)
σ_2	cyst-nem.	1	0	1	0	1	0	0	0	0	1	(3)
σ_3	cyst-nem.	1	0	1	0	0	0	0	1	0	0	(2)
σ_4	herb.-injury	1	0	0	1	0	0	1	0	1	0	(3)
σ_q	$\langle query \rangle$	1	0	0	1	0	0	1	1	0	0	(3)

Unique samples, conceptualized as simplices (rows in tables, shapes in geometric representation), can share attributes, conceptualized as axes (columns in tables, axes in geometric representation), with other samples. Similarities in samples are represented by shared simplex axes (e.g., $X_0, X_1, \ldots, X_9$ in Table 1), visualized as polyhedral components/shapes: points, edges, faces, tetrahedrons (e.g., Figures 16b and 16c) and greater dimensionality beyond the ability to model on paper.

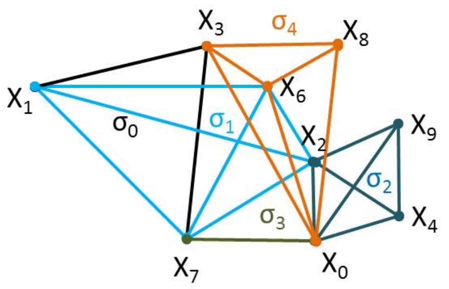
Take, for example, two simplices, σ_0 and σ_3, from Table 1 (illustrated in Fig-



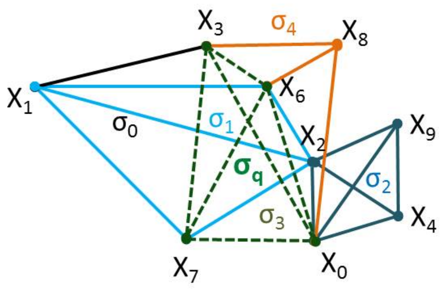
(a) Simplex σ_0 and its geometric representation (a tetrahedron) which contains axes $\langle X_1 X_3 X_6 X_7 \rangle$.



(b) σ_0 (a tetrahedron) and σ_3 (a plane) share one attribute: axis X_7 (leaf-mildew, absent).



(c) Geometric representation of a simplicial complex. All training simplices, σ_0 through σ_4, and their relationships.



(d) Geometric representation of a simplicial complex. All training simplices and the Query Simplex, σ_q, in green dashed lines, illustrating it's relationship to the training simplices.

Figure 16: Geometric Actualization of Simplices and Their Relationships from Table 1.

ure 16b). The 3-simplex (i.e., 3 dimensional), σ_0, can be written as $\langle X_1, X_3, X_6, X_7 \rangle$. The 2-simplex (i.e., 2 dimensional), σ_3, can be written as $\langle X_0, X_2, X_7 \rangle$. One attribute, axis X_7, representing *leaf-mildew is absent*, is shared by σ_0 and σ_3. It is the combination of shared axes that the formalism uses to infer Query Elements response variable.

3.6.2.8 QS Represented by a Simplicial Complex.

In the formalism a simplicial complex represents all samples as simplices and their relationships, as illustrated in Table 1 and Figure 16d. In Table 1 each row, σ_0 through σ_q, is a simplex. In Figure 16d all simplices from Table 1 are integrated into a geometric representation of stable and cinsistent maximally integrated information structures.

3.6.2.9 Simplicial Family.

Any set of simplices can be defined to be a *Simplicial Family* (Johnson, 2013). To support generating an inference, a *Simplicial Family* is defined in this dissertation to be all simplices with the same Simplex Label. In the soybean disease example (Table 1) there are three *Simplicial Families*: bacterial-blight, cyst-nematode and herbicide-injury.

3.6.2.10 A Measure of Connectivity between Simplices.

(Johnson, 2013) Eccentricity is an asymmetric measure of connectivity between two simplices. Let σ and σ' be two simplices. The eccentricity of a simplex with respect to another is:

$$ecc(\sigma|\sigma') \overset{def}{=} \frac{|\sigma \setminus \sigma'|}{|\sigma|} = \frac{\text{number of } \sigma \text{ vertices not shared with } \sigma'}{\text{number of vertices of } \sigma} \tag{1}$$

Table 2 illustrates how eccentricity is calculated and the asymmetric nature of the measure, with a simple example.

Table 2: Example of Asymmetric Eccentricity: $ecc(\sigma|\sigma')$ and $ecc(\sigma'|\sigma)$.

$ecc(\sigma|\sigma') = 1/5$

	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
σ	0	1	0	$\boxed{1}$	0	0	1	1	1	0
σ'	1	1	1	$\boxed{0}$	1	1	1	1	1	0

$ecc(\sigma'|\sigma) = 4/8$

	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
σ	$\boxed{0}$	1	$\boxed{0}$	1	$\boxed{0}$	$\boxed{0}$	1	1	1	0
σ'	$\boxed{1}$	1	$\boxed{1}$	0	$\boxed{1}$	$\boxed{1}$	1	1	1	0

3.6.3 Extensions to Hypernetwork Theory.

These extensions to hypernetwork theory (Vaughan et al., 2015) were introduced by this research to create a distance measure and an inference (response variable) from the QS Computational Module knowledge base.

3.6.3.1 Applying a Hausdorff Distance.

Eccentricity cannot be used as a distance measure because it is not symmetric. The eccentricity of σ with respect to σ', $ecc(\sigma|\sigma')$, is not guaranteed to be equal to the eccentricity of σ' with respect to σ, $ecc(\sigma'|\sigma)$. In order to satisfy the requirement of symmetry, a Hausdorff distance[3] was applied, and the distance (d_Q) between two simplices defined to be the maximum of $ecc(\sigma|\sigma')$ and $ecc(\sigma'|\sigma)$ [4].

$$d_Q(\sigma, \sigma') \overset{def}{=} \max\{ecc(\sigma|\sigma'), ecc(\sigma'|\sigma)\} \tag{2}$$

It is beyond the scope of this research to prove, or disprove, this distance measure

[3]See Hausdorff Distance in Appendix B

[4]An alternate, standardized format for d_Q is $\max\left(\left(\frac{c}{a+c}\right), \left(\frac{b}{a+b}\right)\right)$, Range: $[0, 1]$, see Appendix E.

satisfies the four properties of a metric, as discussed in Appendix B.2. A proposed area of future research is to conduct the proof.

3.6.3.2 Distance Between a Simplex and a Family.

In order to achieve the objective of inferring the Query Element, the distance between the Query Simplex and each Simplicial Family needs to be determined. The Query Element is inferred to be the Simplex Label of the *closest* Simplicial Family. Therefore, hypernetwork theory is further extended by defining the distance, *dist*, between a simplex (σ_q) and a Simplicial Family (F) and a weight, w, associated with *dist*.

Let σ_q be a simplex, and F be a family, that σ_q is not a member of. The distance, $dist_Q$, between σ_q and F is defined to be the minimum of the distances (d_Q) between σ_q and each member of the family:

$$dist_Q(\sigma_q, F) \overset{def}{=} \min\{d_Q(\sigma_q, \sigma) : \sigma \text{ belongs to } F\} \tag{3}$$

3.6.3.3 A Weighting Function.

It is foreseeable that two or more families can share the closest distance, $dist_Q$, from a simplex, therefore a weighting function is added to the formalism to determine the closest family in this situation. The weight, w_Q, illustrated in Figure 17 is defined to be the proportion of simplices, σ, in family, F, where $d_Q(\sigma_q, \sigma)$ is equal to $dist_Q(\sigma_q, F)$:

$$w_Q(\sigma_q, F) \overset{def}{=} \frac{card\{\sigma \in F : d_Q(\sigma, \sigma_q) = dist_Q(\sigma_q, F)\}}{card(F)} \tag{4}$$

The value of the Query Element is inferred by finding the Simplicial Family to

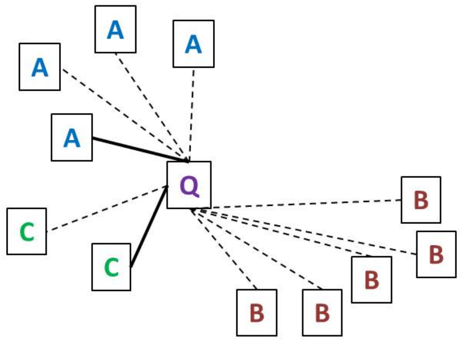


Figure 17: Illustration of Weight Function. Given three families, A, B, C, and a Query Simplex, Q. The minimum distance, $dist_Q$, is represented with a solid line. Since $dist_Q(Q, A) = dist_Q(Q, C)$, the closest *family* is determined by the *largest* weight function. The weight for family $A = 1/4$, and the weight for family $C = 1/2$, therefore C is the closest family.

which the Query Simplex is closest, i.e., the family with the smallest distance, $dist_Q$. Should more than one family share the closest distance, $dist_Q$, the closest family is determined to be the one with the greatest weight, w_Q.

3.6.4 QS Computational Module Examples.

The QS Computational Module is better understood by means of two simple examples. The first example illustrates details of the algorithm, using the large soybean disease diagnosis data set (Michalski and Chilausky, 1980) from the University of California, Irvine (UCI) ML Repository (Lichman, 2013). The first example also illustrates that the target variable is not identified before training or testing begin, supporting the hypothesis if this dissertation.

The second example illustrates more completely the hypothesis of this research, demonstrating that the QMA can function as a supervised classifier and overcome the necessity to identify the target variable before training (or testing) begins and overcomes the necessity to re-train the cumulative training samples when the target variable changes. The second example also demonstrates concept drift. In the second example, the algorithm is applied to a subjective problem space, one created entirely by human cognition, which is dynamic, contested and current — malicious software (malware) classification based on malware behavior and metadata.

3.6.4.1 Example 1, Illustrating Detailed Inference Generation Process.

This example will demonstrate that the QMA can function as a supervised classifier and overcome the necessity to identify the target variable before training (or testing) begins. To provide an intuitive illustration, a clearly objective and frequently referenced problem space is employed: the large soybean disease diagnosis data set. A notional subset of data are illustrated in Table 3.

The UCI Large Soybean Disease Categorical Data Set

Table 3: Illustrative Subset of Raw Data from UCI Large Soybean Data Set.

obs.	disease	hail	seed-discolor	precip.	leaf-mildew
(1)	bacterial-blight	absent	absent	gt normal	absent
(2)	bacterial-blight	absent	present	gt normal	absent
(3)	bacterial-blight	absent	present	gt normal	absent
(4)	cyst-nematode	present	present	lt normal	lower-surface
(5)	cyst-nematode	present	present	$\langle$*unobserved*$\rangle$	absent
(6)	cyst-nematode	present	present	$\langle$*unobserved*$\rangle$	absent
(7)	cyst-nematode	present	present	$\langle$*unobserved*$\rangle$	absent
(8)	herbicide-injury	present	absent	gt normal	upper-surface
(9)	$\langle$*unobserved*$\rangle$	present	absent	gt normal	absent
(10)	$\langle$*unobserved*$\rangle$	$\langle$*unobserved*$\rangle$	absent	normal	absent

The UCI large soybean data set has 683 samples and 36 categorical variables. Presented in Table 3 is a notional subset of 10 samples (rows) and 5 variables (columns). Note that some samples are indistinguishable (i.e., identical), for example rows 2 and 3 are identical, and some elements are *unobserved*. All raw samples, from Table 3, are converted into an ordered binary representation (binarized), as illustrated in Table 4. Converting the categorical data into the ordered binary representation is the training step in this formalism.

Training Step, Samples Convert to Ordered Binary Representation

In Table 4 each row of number represents a unique sample identified by a unique set of attributes. The *frequency* column is a count of indistinguishable samples. A *1* in an attribute cell indicates the presence of the attribute, a *0* indicates the absence of the attribute. The attributes for each variable (disease, hail, etc.) are mutually exclusive, but not required. Notice, there is no *precipitation* in row (4), no *disease* in row (6), and no *disease* or *hail* in row (7).

Table 4: Training Step, Soybean Sample Converted Ordered Binary Representation.

		(notional) Attribute Set													
		disease			hail		seed-discolor		precipitation			leaf-mildew			
row	frequency	bacterial-blight	cyst-nematode	herbicide-injury	present	absent	present	absent	lt normal	normal	gt normal	absent	upper-surface	lower-surface
(1)	1	1	0	0	0	1	0	1	0	0	1	1	0	0
(2)	2	1	0	0	0	1	1	0	0	0	1	1	0	0
(3)	1	0	1	0	1	0	1	0	1	0	0	0	0	1
(4)	3	0	1	0	1	0	1	0	***0***	***0***	***0***	1	0	0
(5)	1	0	0	1	1	0	0	1	0	0	1	0	1	0
(6)	1	***0***	***0***	***0***	1	0	0	1	0	0	1	1	0	0
(7)	1	***0***	***0***	***0***	***0***	***0***	0	1	0	1	0	1	0	0

Training and test samples are represented identically. Test samples will have at least one unobserved element. Training samples may also have unobserved elements. In order to initiate an inference one variable (usually an *unobserved* element) is selected as the Query Element from a sample, which will be inferred.

Selecting Target Variable after Training

For this example, the *disease* variable in Row (6) of Table 4 is selected as the Query Element. Therefore, the target variable is identified as *disease*. Once the target variable of *disease* has been identified, the samples from Table 4 can be conceptualized as a simplicial complex, as illustrated in Table 5. The target variable becomes the Simplex Label, and the remaining variables become axes. It is important to note, that the algorithm does not convert the data to the simplicial complex, but calculates over the data on the ordered binary representation.

Table 5: Soybean Sample in Simplicial Complex Form, Given *Disease* as Target Variable.

		(notional) Attribute Set											
		disease	hail		seed -discolor		precipitation			leaf -mildew			
			present	absent	present	absent	lt normal	normal	gt normal	absent	upper-surface	lower-surface	
ID	fr	Simplex Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	dim
σ_0	1	bact.-blight	0	1	0	1	0	0	1	1	0	0	(3)
σ_1	2	bact.-blight	0	1	1	0	0	0	1	1	0	0	(3)
σ_2	1	cyst-nem.	1	0	1	0	1	0	0	0	0	1	(3)
σ_3	3	cyst-nem.	1	0	1	0	0	0	0	1	0	0	(2)
σ_4	1	herb.-injury	1	0	0	1	0	0	1	0	1	0	(3)
σ_q	1	$\langle query \rangle$	1	0	0	1	0	0	1	1	0	0	(3)

Extending the Hypernetwork Theory Simplicial Complex Formalism

The research presented in this dissertation extends the simplicial complex formalism with four features, represented in blue and green text, and underlined, in Table 5. First, multiple simplices with the same *Simplex Label* are supported (i.e., σ_0 and σ_1 are both *bacterial-blight*, σ_2 and σ_3 are both *cyst-nematode*), representing different samples with the same target variable. Second, a *frequency* column (fr) was added, which captures the count of samples with the same attribute set. Third, the option of a Query Element was added; $\langle query \rangle$ in Table 5, indicating a Query Element which is the goal of the inference generation. Fourth, a *Query Simplex* (σ_q as defined in Table 5), which is a simplex with one Query Element. These extensions to hypernetwork theory are necessary for the inference generating algorithm.

This formalism is a ML algorithm, with the data set σ_0 through σ_4 as training samples, and σ_q as a test sample. Note, row (7) of Table 4 is not carried over to Table 5. Because the target variable is *disease*, and row (7) has no *disease* value, row (7) is not used in this inference generating formalism. The Query Element is the disease intended for inference, conceptualized as $\langle query \rangle$ in Table 5.

Alternatively, the unobserved *precipitation* from row (4), in Table 4, could have been selected for this inference example. In that case, the target variable would be *precipitation* and the samples from Table 4 would be presented into a hypernetwork theory formalism with *precipitation* as the Simplex Label column. In this alternate simplicial complex, the number and labels of the simplices (i.e., rows) and axes (i.e., columns) would differ from Table 5, representing a different set of shapes. This dynamic, an example of concept drift[5], is illustrated in the malware example (see Section 3.6.4.2) which follows this detailed example.

[5]Concept Drift is defined in Appendix D.4.2.

Apply the Novel Distance and Weight Measures

Now the novel distance and weight measures will be applied to the samples represented in Table 5. The objective is to infer the label of the Query Element, $\langle query \rangle$ in σ_q. The following calculations determine which disease is inferred. Note, the *frequency* column is used in the calculation of the weight (w).

For family bacterial-blight, $F_{BB} = \{\sigma_0, \sigma_1\}$:

$$ecc(\sigma_0|\sigma_q) = 1/4, \quad ecc(\sigma_q|\sigma_0) = 1/4 \quad \therefore d_Q(\sigma_0|\sigma_q) = 1/4 \tag{5}$$

$$ecc(\sigma_1|\sigma_q) = 2/4, \quad ecc(\sigma_q|\sigma_1) = 2/4 \quad \therefore d_Q(\sigma_1|\sigma_q) = 2/4 \tag{6}$$

$$\therefore\ dist_Q(\sigma_q, F_{BB}) = 1/4, \quad w = 1/3 = .33 \tag{7}$$

For family cyst-nematode, $F_{CN} = \{\sigma_2, \sigma_3\}$:

$$ecc(\sigma_2|\sigma_q) = 3/4, \quad ecc(\sigma_q|\sigma_2) = 3/4 \quad \therefore d_Q(\sigma_2|\sigma_q) = 3/4 \tag{8}$$

$$ecc(\sigma_3|\sigma_q) = 1/3, \quad ecc(\sigma_q|\sigma_3) = 2/4 \quad \therefore d_Q(\sigma_3|\sigma_q) = 2/4 \tag{9}$$

$$\therefore\ dist_Q(\sigma_q, F_{CN}) = 2/4, \quad w = 3/4 = .75 \tag{10}$$

For family herbicide-injury, $F_{HI} = \{\sigma_4\}$:

$$ecc(\sigma_4|\sigma_q) = 1/4, \quad ecc(\sigma_q|\sigma_4) = 1/4 \quad \therefore d_Q(\sigma_4|\sigma_q) = 1/4 \tag{11}$$

$$\therefore\ dist_Q(\sigma_q, F_{HI}) = 1/4, \quad w = 1/1 = 1.00 \tag{12}$$

The label is inferred to be the family closest to the Query Simplex, σ_q, with the greatest weight. In this example, there are two families equally close with a distance of 1/4: bacterial-blight and herbicide-injury. Herbicide-injury has the greater weight, 1.00, therefore herbicide-injury is inferred as the Query Element, $\langle query \rangle$.

Report the Response and Update the Autonomous Module

Once these calculations are complete a vector representation of the hypothetical sample, an Inference Vector, is sent to QS Computational Module ACT–R (Figure 13) to be converted in to a chunk slot/value representation (i.e., disease: herbicide-injury, hail: present, seed-discolor: absent, precipitation: gt normal, leaf-mildew: absent) and forwarded to autonomous ACT–R. A copy of the Inference Vector is not stored in QS Computational Module, as that may introduce miscalculations in subsequent inferences. However, the Human-computer pairing interface (discussed in Section 3.3) can be used to confirm the inference and update the Query Element in QS Computational Module.

3.6.4.2 Example 2, Supporting the Research Hypothesis.

Like the first example, this example demonstrates that the QMA can function as a supervised classifier and overcome the necessity to identify the target variable before training (or testing) begins. This example also demonstrates how the formalism overcomes the necessity to re-train the cumulative training samples when the target variable changes, supporting the research hypothesis.

In this example the algorithm is applied to a subjective, contemporary social and economic problem space: malware classification-types (i.e., malware types), behavior and metadata. Classifying malware, and identifying its behavior, is a necessary step in malware mitigation and helps to identify new and emerging threats (Szor, 2005). Metadata, among other useful features, potentially identifies the malware source, which can help analysts recognize persistent threats and take appropriate actions to protect the network, as well as support forensic investigations (Sikorski and Honig, 2012). A data set of 2088 samples was created, consisting of 9 malware types, 46 behavioral and 3 metadata variables, from the VirusTotal (2016) malware repository.

A more detailed analysis of the malware data collection process is in Appendix G.

A notional subset of ten samples is illustrated in Table 6, consisting of 3 malware types, 2 behavioral and 2 metadata variables. The data have been converted to ordered binary representation.

Training Step, Samples Convert to Ordered Binary Representation

Table 6: Training Step, Malware Samples Converted to Ordered Binary Representation.

		(notional) Attribute Set														
		Malware Type			AutoExec		MntPnt		Charset			Lang Code				
row	frequency	NeshtaVirus	WormViking	WormBrontok	present	absent	present	absent	Win Cyrillic	Win Korean	Unicode	English	Chinese	Korean	Xhosa
(1)	1	1	0	0	0	1	0	1	0	0	1	1	0	0	0
(2)	1	1	0	0	0	1	1	0	0	1	0	0	0	1	0
(3)	1	1	0	0	0	1	1	0	0	0	1	0	1	0	0
(4)	2	0	1	0	1	0	0	1	1	0	0	0	0	0	1
(5)	1	0	1	0	1	0	0	1	0	0	1	0	1	0	0
(6)	1	0	1	0	1	0	1	0	0	0	1	***0***	***0***	***0***	***0***
(7)	1	***0***	***0***	***0***	1	0	0	1	1	0	0	***0***	***0***	***0***	***0***
(8)	1	0	0	1	0	1	0	1	0	1	0	0	0	1	0

Selecting the Target Variable after Training

In this test sample, the unobserved *Language Code* from Table 6, row (6), is selected as the *Query Element*. Therefore, the target variable is *Language Code*, and row (6) is conceptualized as the *Query Simplex*, σ_q, in the simplicial complex in Table 7 .

Table 7: A Malware Sample Simplicial Complex with *Language Code* as Target Variable.

		(notional) Attribute Set											
		Lang Code	Malware Type			AutoExec		MntPnt		Charset			
		Simplex	NeshtaV.	WormVik.	WormBr.	present	absent	present	absent	Win Cyrillic	Win Korean	Unicode	
ID	fr	Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	dim
σ_0	1	English	1	0	0	0	1	0	1	0	0	1	(3)
σ_1	1	Korean	1	0	0	0	1	1	0	0	1	0	(3)
σ_2	1	Chinese	1	0	0	0	1	1	0	0	0	1	(3)
σ_3	2	Xhose	0	1	0	1	0	0	1	1	0	0	(3)
σ_4	1	Chinese	0	1	0	1	0	0	1	0	0	1	(3)
σ_5	1	Korean	0	0	1	0	1	0	1	0	1	0	(3)
σ_q	1	$\langle query \rangle$	0	1	0	1	0	1	0	0	0	1	(3)

By applying the novel distance and weight measures (as detailed in the first example, Section 3.6.4.1), the Query Element, $\langle query \rangle$, is inferred to be Chinese, with distance of 1/4, weight 0.5. Xhosa has a distance of 2/4, and a weight of 1.00. English has a distance of 3/4, and a weight of 1.00. Korean has a distance of 3/4, and a weight of 0.5.

Changing the Target Variable with no Re-training

In this next test sample, the unobserved *Malware Type* from Table 6, row (7), is selected as the *Query Element.* Therefore, the target variable is *Malware Type*, and row (7) is conceptualized as the *Query Simplex*, σ_q, in the simplicial complex in Table 8. The target variable has changed, and no re-training has occurred.

Table 8: A Malware Sample Simplicial Complex with *Malware Type* as Target Variable.

		(notional) Attribute Set												
		Malware Type	AutoExec		MntPnt		Charset			Lang Code				
		Simplex	present	absent	present	absent	Win Cyrillic	Win Korean	Unicode	English	Chinese	Korean	Xhosa	
ID	fr	Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	dim
σ_0	1	NeshtaV.	0	1	0	1	0	0	1	1	0	0	0	(3)
σ_1	1	NeshtaV.	0	1	1	0	0	1	0	0	0	1	0	(3)
σ_2	1	NeshtaV.	0	1	1	0	0	0	1	0	1	0	0	(3)
σ_3	2	WormVik.	1	0	0	1	1	0	0	0	0	0	1	(3)
σ_4	1	WormVik.	1	0	0	1	0	0	1	0	1	0	0	(3)
σ_5	1	WormVik.	1	0	1	0	0	0	1	***0***	***0***	***0***	***0***	(2)
σ_6	1	WormBr.	0	1	0	1	0	1	0	***0***	***0***	***0***	***0***	(3)
σ_q	1	$\langle query \rangle$	1	0	0	1	1	0	0	0	1	0	0	(3)

By applying the novel distance and weight measures (as detailed in the first example, Section 3.6.4.1), the Query Element, $\langle query \rangle$, is inferred to be WormViking, with distance of 1/4, weight 0.5. NeshtaVirus has a distance of 3/4, and a weight of 0.33. WormBrontok has a distance of 3/4, and a weight of 1.00.

This example illustrates a primary advantage of the QS algorithm, compared to conventional ML algorithms — the target variable can change from test sample to test sample, without re-training or repopulating the data structure. The algorithm does not convert the data to the simplicial complex representation, but calculates over the data on the ordered binary representation. The maximally integrated information structure remains stable.

3.6.5 Concept Drift and Transfer Learning (TL).

In ML research, a change in the source of the target is referred to as *concept drift* and is a feature of some incremental ML algorithms. *Concept drift* is one method in which TL can be achieved. TL is the ability of a system to apply knowledge or skills learned in previous tasks to subsequent tasks or new domains, which are similar in some way (Pan and Yang, 2010).

3.6.6 Imagined Experiences.

Inherent in the QMA is the ability to *imagine* a different set of simulated sensory experiences. Generating inferences from an *imagined* set of experiences supports the *imagined* past, present or future as posited by Edelman (1989). This is accomplished by applying the inference generation algorithm to hypothetical test samples, thereby generating a series of competing hypothesis, a form of abductive reasoning.

This approach will be useful in domains where the accuracy of the training or test samples are suspect, for example, in malware identification and classification. In order to avoid detection, malicious actors employ various forms of obfuscation. For example, malware may import libraries, or call functions, that are not actually used in the execution of the code. On the other hand, they may rename a library on the target computer and call that unexpected, renamed library. They may use a foreign language code, instead of their native language. These actions are intended to evade detection and make analysis and mitigation more difficult. A domain expert could use the QMA formalism to generate a series of competing hypothesis to help identify, classify and mitigate threats.

3.7 Summary

This chapter detailed the steps taken and tools used to develop the cognitively inspired computational agent, thereby answering the research questions of how the components can be modeled to produce a complete computational model of the framework and the data sets can be processed-over to produce a pattern-completion inference by using hypernetwork theory and custom software.

The research hypothesis was supported with two examples using two different data sets, demonstrating that a computational agent, inspired by a theory of human learning-by-experience (Stanovich's framework), can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes. The research questions addressing metrics and performance, as compares to benchmark classifiers, will be addressed in Chapter 4.

IV. Results and Analysis

Chapter 1 introduced the motivation for this research, the hypothesis and research questions addressing the hypothesis. Chapter 2 reviewed the relevant literature and revealed functions of Stanovich's tripartite framework which lead to the development of the Qualia Modeling Agent (QMA). Chapter 3 detailed the steps taken, mathematical formalism and tools used to develop the QMA, and demonstrated support for the research hypothesis.

This chapter presents the results of data analysis and findings that further support the research hypothesis. The research hypothesis proposes that a computational agent, inspired by a theory of human learning-by-experience, the QMA, can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes. Previous machine learning (ML) approaches supporting pattern-completion (i.e., predicting the class labels for new samples), require re-training when the target variable changes.

The first section of this chapter (Section 4.1) presents the appropriate metrics to assess the performance of benchmark nonparametric classifiers and the QMA. The second section (Section 4.2) presents further support for the research hypothesis, by reporting the results of classifying various malicious software (malware) data set Query Elements as the target variable changes from test sample to test sample. The next section (Section 4.3) demonstrates concept drift. The last section (Section 4.4) presents a comparative analysis of classification accuracy between the QMA and two benchmark nonparametric classifiers, using 4 factual data sets. In this section the QMA demonstrates improved classification accuracy in data sets with greater dimensionality.

QMA is a novel algorithm that can be used as either an incremental, or a batch learning classifier. As an *incremental* classifier the QMA supports the research hypothesis and demonstrates support for concept drift when the target variable changes. When used as a batch classifier QMA demonstrates increased classification accuracy with multivalued/multiclass data sets.

4.1 Metrics

Based on the motivation for his research and the hypothesis, the appropriate metrics, with which to assess the performance of benchmark nonparametric classifiers and the QMA, are: how efficiently the learner adjusts to changes in the target variable; if the learner retains test samples in a manner that allows their predictive values to contribute to subsequent test samples; if the learner retains response variables in a manner that prevents unnecessary redundant classification, and does not distort the probability distributions applied to subsequent test samples; and, classification accuracy, i.e., the proportion of correctly classified test samples. These metrics will be applied the the benchmark learners and to the QMA.

4.2 Supporting the Research Hypothesis

The research hypothesis proposes that a computational agent, inspired by a theory of human learning-by-experience, can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes.

Details of the formalism, allowing the target variable to change, were illustrated with notional examples in Section 3.6.4.2. Table 9 presents results when the formalism was applied to the factual malware data set, detailed in Appendix G.

Table 9: Support for the Research Hypothesis, Results of QMA with a Factual Malware Data Set. The QMA was trained with 1250 Malware data set samples (3 randomized folds).

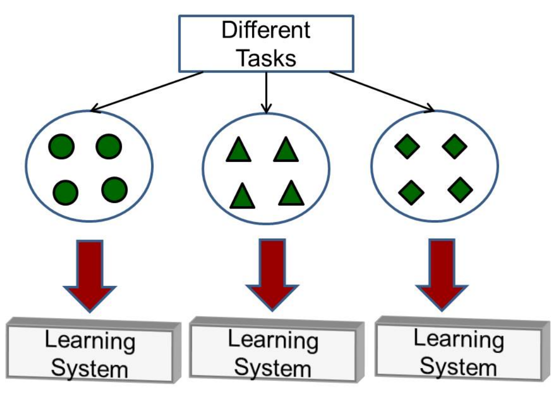
Test	Hash (ID)	Target Variable	Classes	Truth	Response Variable
1	72445...	Char. Set (Metadata)	10	Windows Latin1	Windows Latin1
2	0b6ca...	Malware Type	9	WormRamnit	WormRamnit
3	2d012...	Subsystem (Metadata)	5	Windows GUI	Windows GUI
4	386f5...	Lang. Code (Metadata)	14	Chinese Simpl.	Chinese Simpl.
5	386f5...	Char. Set (Metadata)	10	UNICODE	UNICODE

Six test samples were run, and the target variable was changed between each test sample. The last two test runs use the same test sample with different target variables selected. With such a large number of training samples, it is not unexpected that all 6 test samples classified accurately, based on classification accuracy of this data set and the QMA algorithm demonstrated later in this chapter. The results reported here support the research hypothesis and demonstrate the target variable changing with no re-training required. Also, the test sample's predictive variables are retained in the hypernetwork theory representation for their predictive value in subsequent test samples.

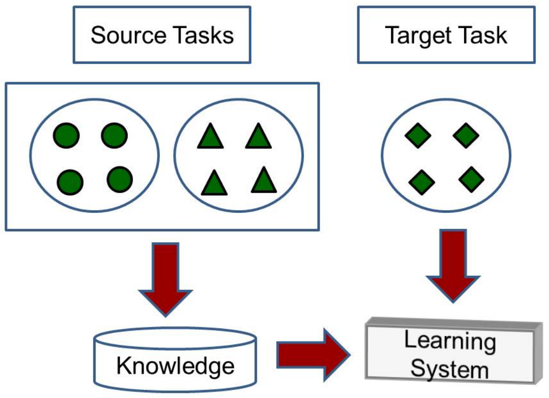
4.3 Concept Drift and Transfer Learning (TL)

Concept drift refers to a learning problem that changes over time. In particular, the statistical properties of the Target Variable, which the model is trying to predict, change over time in unforeseen ways (Žliobaitė, 2010). In QMA *concept drift* occurs when the target variable changes from test sample to test sample. Concept drift is discussed in more detail in Appendix D.4.2.

Concept drift is one method in which Transfer Learning (TL) can be achieved (Geng and Smith-Miles, 2009). TL is the ability of a system to apply knowledge or skills learned in previous tasks to subsequent tasks or new domains, which are similar in some way (Pan and Yang, 2010). Figure 18 illustrates the conceptual difference between the learning processes of traditional and TL techniques presented in a TL survey by Pan and Yang.



(a) Learning Process of Traditional ML.



(b) Learning Process of Transfer Learning.

Figure 18: Different Learning Processes between Traditional Machine Learning and Transfer Learning (Pan and Yang, 2010).

The learning process presented in this dissertation is novel, and is conceptually illustrated in Figure 19. When the target variable changes, the new target variable becomes the Label Space, and the previous target variable joins the other variables in the Feature space. Subsequently, the domain marginal probability distribution (mpd) changes and the predictive function of the task changes. Also, when the target variable changes each test sample adds knowledge to all other all other domains. For example, in the malware classification example (Section 3.6.4.2), when the target variable is changed from *language code* to *malware type*, the domain (variables and predictive function of the simplicial complex) changed, and the task (Query Element)

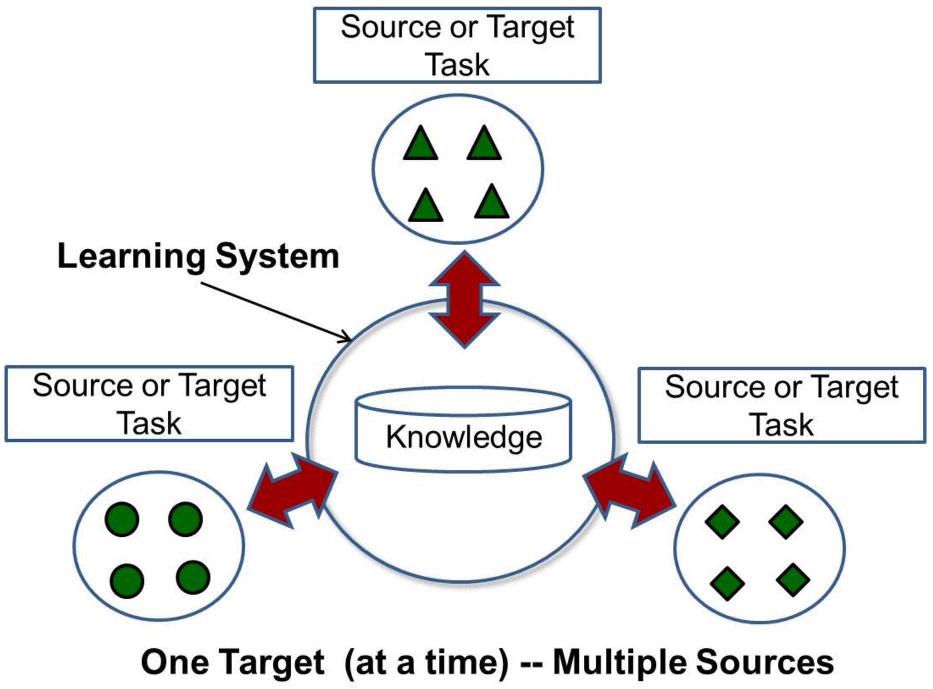


Figure 19: Learning Process of QS. When the target variable changes, the new target variable becomes the label space, and the previous target variable joins the other variables in the feature space (knowledge).

changed, yet, the declarative knowledge captured in qualia space (QS) remains the same.

In order for TL to be achieved improved performance, along some axis, must be demonstrated, as well as, knowledge from one domain and task must be transferred to another domain and task which share a common feature space (Pan and Yang, 2010). Applying the definitions of (Pan and Yang, 2010) to the QS formalism:

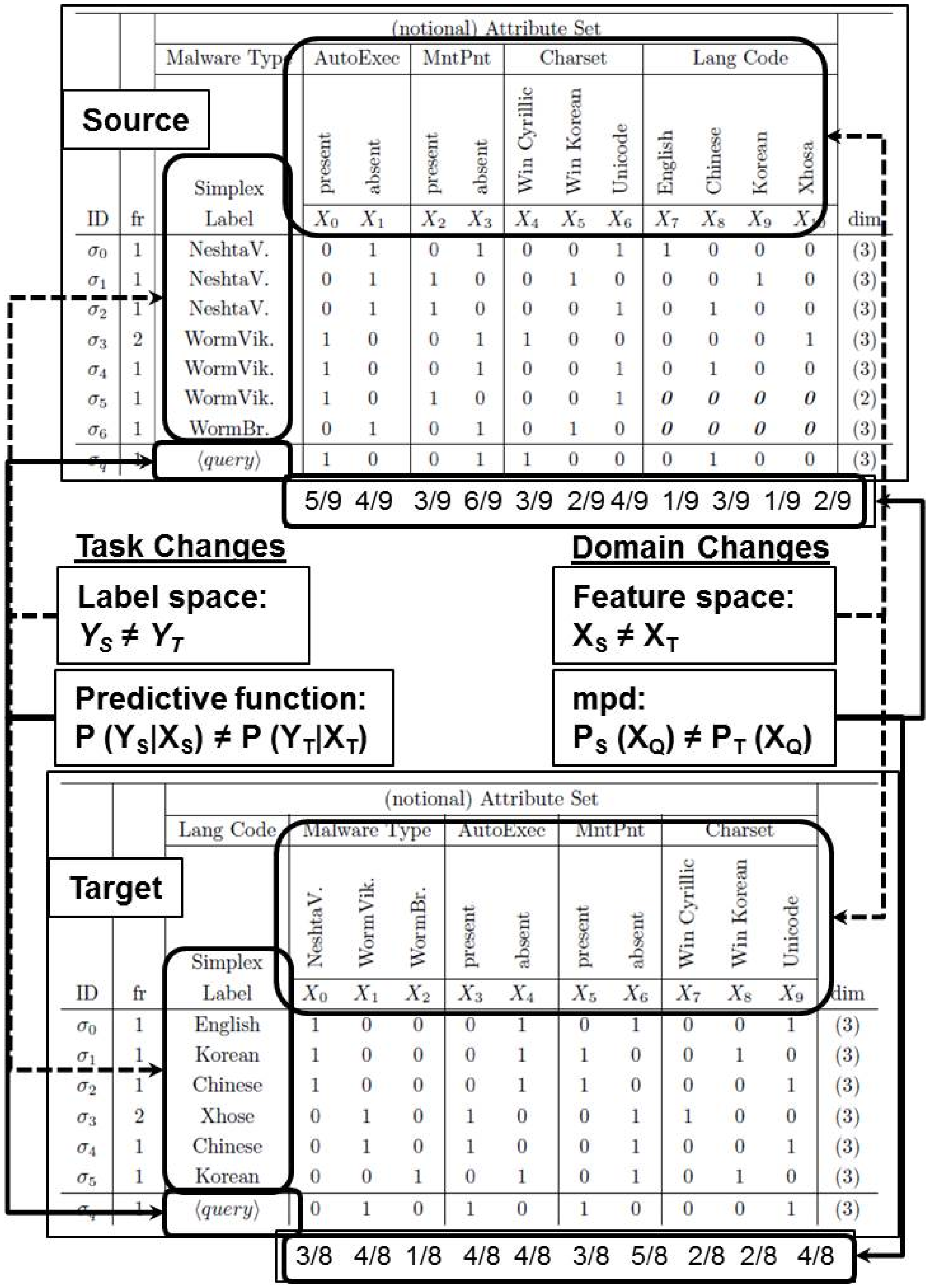
Definition 4.3.1 (QS Domain)**.** The *Domain*, $\mathcal{D}_Q$, consists of two components: a feature space, $\mathcal{X}_Q$, **i.e, axes** $(X_1, \ldots, X_n)$ **in the simplicial complex**, and a mpd, $P(X_Q)$, where $X_Q = \{x_1, \ldots, x_n\} \in \mathcal{X}_Q$, **i.e., a unique mpd created when a target variable is identified and the Simplicial Complex is created.**

Definition 4.3.2 (QS Task)**.** Given a specific domain, $\mathcal{D}_Q = \{\mathcal{X}_Q, P(X_Q)\}$, a *task* consists of two components: a label space $\mathcal{Y}_Q$, **i.e., Simplex Labels in the Simplicial Complex**, and an objective predictive function $f(\cdot)$ (denoted by $\mathcal{T}_Q = \{\mathcal{Y}_Q, f(\cdot)\}$), which is not observed but can be learned from the training data, which consists of pairs $\{x_i, y_i\}$, where $x_i \in X_Q$ and $y_i \in \mathcal{Y}_Q$. The function $f(\cdot)$ can be used to predict the corresponding label, $f(x)$, of a new instance x. From a probabilistic viewpoint, $f(x)$ can be written as $P_Q(y|x)$.

Given these definitions, Figure 20 uses Table 7 and Table 8 from the second example in Chapter 3, to illustrate concept drift and change of domain and task components presented in this research.

- A domain is a pair $\mathcal{D}_Q = \{\mathcal{X}_Q, P(X_Q)\}$. Thus, the condition $\mathcal{D}_S \neq \mathcal{D}_T$ implies that either $\mathcal{X}_S \neq \mathcal{X}_T$ or $P_S(X_Q) \neq P_T(X_Q)$. **In the QMA formalism both the feature space and the mpd change:** $\mathcal{X}_S \neq \mathcal{X}_T$ **and** $P_S(X_Q) \neq P_T(X_Q)$.
- A task is defined as a pair $\mathcal{T}_Q = \{\mathcal{Y}_Q, P(Y_Q|X_Q)\}$. Thus, the condition $\mathcal{T}_S \neq \mathcal{T}_T$ implies that either $\mathcal{Y}_S \neq \mathcal{Y}_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$. **In the QMA formalism both the label space and the objective predictive function have changed:** $\mathcal{Y}_S \neq \mathcal{Y}_T$ **and** $P(Y_S|X_S) \neq P(Y_T|X_T)$.

A proposed area of future research is to demonstrate improved performance, along some axis, when algorithm accommodates target variable changes, therefore demonstrating TL has been achieved.


Source
Task Changes
Label space: $Y_S \neq Y_T$
Predictive function: $P(Y_S|X_S) \neq P(Y_T|X_T)$
Domain Changes
Feature space: $X_S \neq X_T$
mpd: $P_S(X_Q) \neq P_T(X_Q)$
Target

Source:

		(notional) Attribute Set												
		Malware Type	AutoExec		MntPnt		Charset			Lang Code				
		Simplex	present	absent	present	absent	Win Cyrillic	Win Korean	Unicode	English	Chinese	Korean	Xhosa	
ID	fr	Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	dim
σ_0	1	NeshtaV.	0	1	0	1	0	0	1	1	0	0	0	(3)
σ_1	1	NeshtaV.	0	1	1	0	0	1	0	0	0	1	0	(3)
σ_2	1	NeshtaV.	0	1	1	0	0	0	1	0	1	0	0	(3)
σ_3	2	WormVik.	1	0	0	1	1	0	0	0	0	0	1	(3)
σ_4	1	WormVik.	1	0	0	1	0	0	1	0	1	0	0	(3)
σ_5	1	WormVik.	1	0	1	0	0	0	1	*0*	*0*	*0*	*0*	(2)
σ_6	1	WormBr.	0	1	0	1	0	1	0	*0*	*0*	*0*	*0*	(3)
σ_q	1	⟨*query*⟩	1	0	0	1	1	0	0	0	1	0	0	(3)
			5/9	4/9	3/9	6/9	3/9	2/9	4/9	1/9	3/9	1/9	2/9	

Target:

		(notional) Attribute Set											
		Lang Code	Malware Type			AutoExec		MntPnt		Charset			
		Simplex	NeshtaV.	WormVik.	WormBr.	present	absent	present	absent	Win Cyrillic	Win Korean	Unicode	
ID	fr	Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	dim
σ_0	1	English	1	0	0	0	1	0	1	0	0	1	(3)
σ_1	1	Korean	1	0	0	0	1	1	0	0	1	0	(3)
σ_2	1	Chinese	1	0	0	0	1	1	0	0	0	1	(3)
σ_3	2	Xhose	0	1	0	1	0	0	1	1	0	0	(3)
σ_4	1	Chinese	0	1	0	1	0	0	1	0	0	1	(3)
σ_5	1	Korean	0	0	1	0	1	0	1	0	1	0	(3)
σ_q	1	⟨*query*⟩	0	1	0	1	0	1	0	0	0	1	(3)
			3/8	4/8	1/8	4/8	4/8	3/8	5/8	2/8	2/8	4/8	

Figure 20: Illustration of Domain and Task Change in QMA. Tables 7 and 8 are used as examples.

4.4 Data Analysis and Findings as a Batch Supervised Learning Classifier

Now the methods used to test the validity of the QMA as a batch supervised ML classifier are presented. The comparative analysis required factual data sets (Table 10) that could determine the degree to which the model could correctly classify patterns and infer unobserved elements (classes). In addition to determining the validity of the model, how well the results generalize to other data sets needed to be assessed. Therefore, the following steps were taken.

4.4.1 Experimental Design.

Two standard nonparametric supervised classifiers were selected as benchmarks for this analysis, k-nearest neighbors (kNN) and decision tree (DT). Nonparametric classifiers make no assumptions about the training data distribution, all training samples are maintained for the test phase, and classification decisions are based on the entire training set (see Appendix D.1) (Thirumuruganathan, 2010). QMA is also a nonparametric supervised classifier.

In addition to the soybean disease and malware data sets, two additional University of California, Irvine (UCI) ML Repository data sets were obtained: the 1984 Congressional Voting (Almanac, 1984) and (standardized) Audiology[1] diagnosis data sets. The three classification algorithms (kNN, DT and the QMA) were combined with the four data sets (Soybean, Voting, Audiology and Malware) to create a 3x4 factorial between-participant design.

4.4.2 Procedure.

In order to compare the results of the three algorithms, all three were applied to the four data sets in *batch* mode, which assumes all training (learning) is complete

[1]Original Owner: Professor Jergen at Baylor College of Medicine.

[2]*Classes* is the number of unique values (Simplex Labels) for the target variable.

Table 10: Non-trivial, Factual, Data Sets used to Test the Validity of the Model.

Data Set (*target variable*)	Samples	Classes[2]	Binary variables	Multivalued variables	Missing values
Soybean (*diagnosis*)	683	19	15	20	∼10%
Cong.Voting (*Dem/Rep*)	435	2	16	0	∼5%
Audiology (*diagnosis*)	226	23	61	8	∼20%
Malware (*type*)	2088	9	46	3	∼2%

before the testing begins. The goal was to create a process that would produce a sufficient number of independent test runs from each data set. Each test run would subsequently be subjected to each of the three algorithms, kNN, DT and the QMA, to produce three *scores*, with each score being a measure of classification accuracy.

For each data set all of the samples were randomly structured into separate partitions of equal size. Different subsets of the partitions were identified as training, validation, and testing partitions to create one test run. To create subsequent test runs the partitions were rotated through each of 20 possible permutations.

This process is otherwise known as k-fold cross-validation (Russell and Norvig, 2009), k being the total number of partitions. k = 5 was chosen[3], which produced a total of 20 test runs for each possible combination of 3 training, 1 validation and 1 test partition. The entire process was repeated a second time to produce a total of 40 independent test runs per data set. Each test run was applied to each of three algorithms: the QMA model, and two analogous ML supervised learning classifiers, kNN and DT[4], producing 40 scores for each data set.

[3]see Appendix D.2 for k-fold cross-validation discussion and rationale for the choice of k.

[4]Both of the ML classifiers, kNN and DT, have a validation procedure requiring an additional validation partition (Russell and Norvig, 2009). The QMA does not have a validation procedure, therefore the validation partition is simply not used in the QMA test runs.

4.4.2.1 Results as a Batch ML Classifier.

The 40 scores per data set, collected under each condition of the 3x4 factorial design, were analyzed in a two-way between-participant analysis of variance (ANOVA), as illustrated in Table 11 and Figure 21. Figure 21 is computed as the proportion of accurately classified test samples (ordinate), for three classification algorithms (abscissa). Each data point represents the mean (M) of 120 scores. *Algorithm* (kNN, DT and the QMA) and type of *data set* (Soybean, Voting, Audiology and Malware) were both between-participant factors. As seen in Table 11 the 2-way ANOVA also revealed a significant *interaction* effect. To probe this interaction, tests of simple main effects were conducted. To do this a one-way ANOVA was computed on the *algorithm* factor at each level of the *data set* factor, as seen in Table 12 and Figure 22. Figure 22 is computed as the proportion of accurately classified test samples (ordinate), for three classification algorithms (abscissa), for each of four data sets (shown as parameter). Each data point represents the mean of 40 scores. (Standard error bars are smaller than symbol used to plot points on graph.)

Table 11: Two-way ANOVAs Revealed a Significant Main Effect for *Algorithm* and *Data Set*, as well as the Interaction ($\alpha = 0.050$).

Algorithm	$F(2, 468) = 7.54, p = 0.0006$
Data Set	$F(3, 468) = 1217.91, p < 0.0001$
Interaction	$F(6, 468) = 5.14, p < 0.0001$

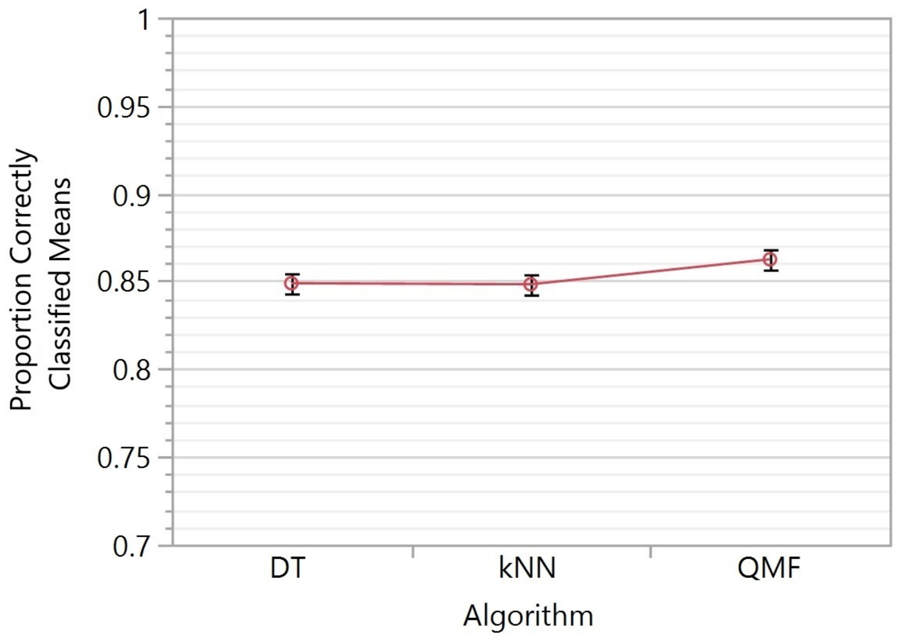


Figure 21: Means of Proportion of Correctly Classified Samples, All Data Sets Combined.

Table 12: One-way ANOVA Computed on the *Algorithm* Factor at each Level of the *Data set* Factor ($\alpha = 0.050$).

	DT (1)		kNN (2)		QMA (3)				Tukey's
Data Set	M	SE	M	SE	M	SE	$F(2,117)$	p	HSD
Soybean	0.887	0.0038	0.896	0.0038	0.914	0.0038	12.95	<0.0001	$1,2<3$
Voting	0.952	0.0026	0.928	0.0026	0.926	0.0026	30.55	<0.0001	$2,3<1$
Audiology	—	—	—	—	—	—	2.55	0.0824	—
Malware	0.888	0.0048	0.899	0.0048	0.914	0.0048	7.04	0.0013	$1<3$

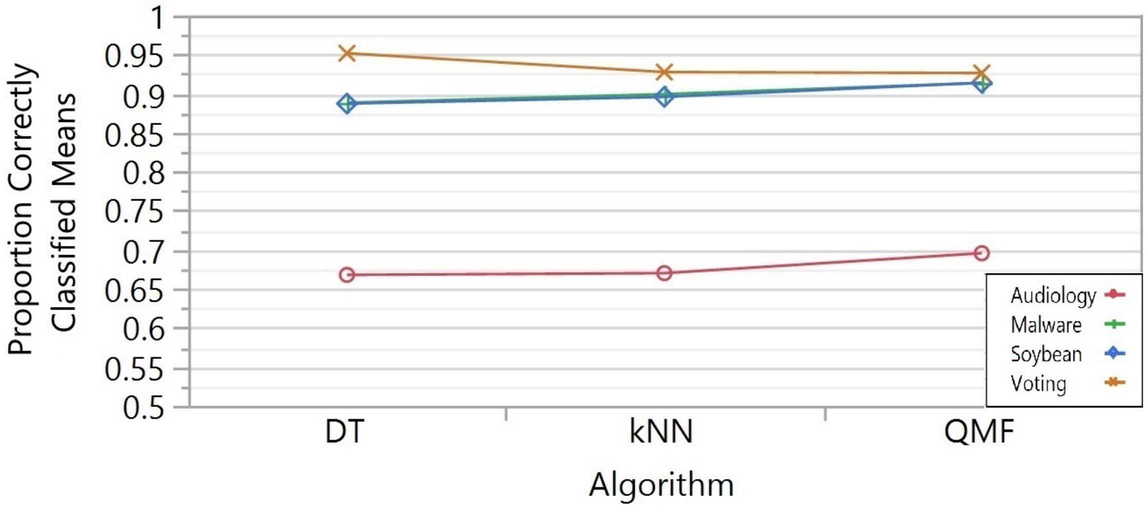


Figure 22: Means of Proportion of Correctly Classified Samples, each Data Set Presented Individually.

4.4.2.2 Exploring Reduced Response Variable Dimensionality.

Empirical observations of the results indicate that greater dimensionality of the response variable may improve relative classification accuracy of the QMA algorithm over the benchmark ML algorithms. In the original experiment, the QMA algorithm demonstrated the highest relative classification accuracy with the soybean data set, therefore this is the data set selected to explore the effect of dimensionality.

In the original experiment the soybean data set had 19 classes and the QMA classified with more accuracy than both kNN and DT, as seen in Figure 23. To explore this observation further data are removed from the soybean data set. First six randomly selected classes, and then an additional four, were removed from the soybean data set, leaving 13 and 9 classes respectively. The three algorithms were applied again using the same technique. Under these new conditions, the classification accuracy of the QMA and kNN were statistically equal and greater than the classification accuracy of DT, as seen in Figure 24 and Figure 25, suggesting that greater response variable dimensionality increases QMA classification accuracy.

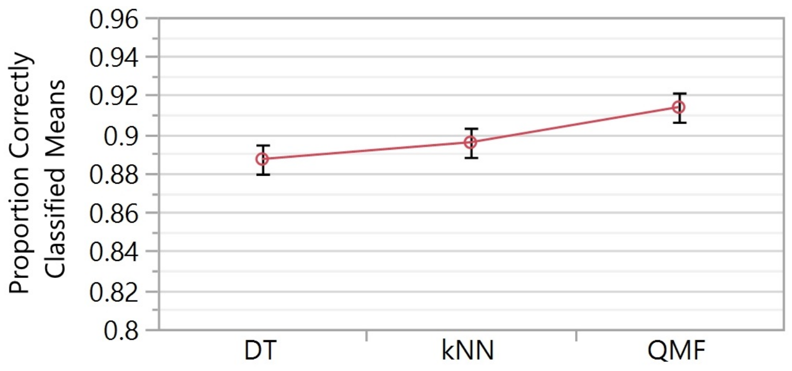


Figure 23: Means of Proportion of Correctly Classified Samples, Soybean Data Set with Original 19 Classes, 683 Samples, 20 Independent Test Runs.
DT - M: 0.887; SE: 0.0038, kNN - M: 0.896, SE: 0.0038; QMA - M:0.914, SE: 0.0038.

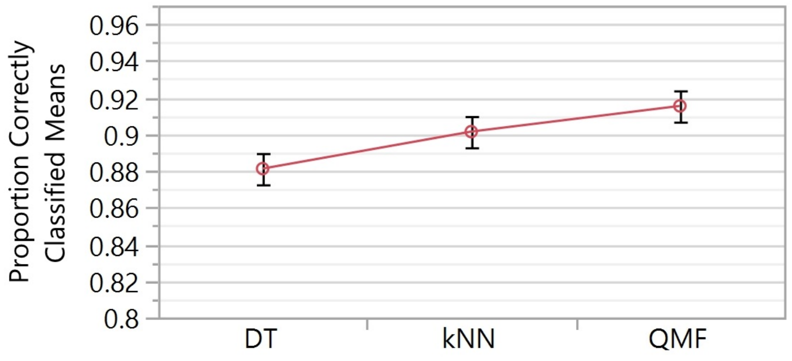


Figure 24: Means of Proportion of Correctly Classified Samples, Soybean Data Set with 13 Classes, 545 Samples, 20 Independent Test Runs.
DT - M: 0.881; SE: 0.0043, kNN - M: 0.901, SE: 0.0043; QMA - M:0.915, SE: 0.0043.

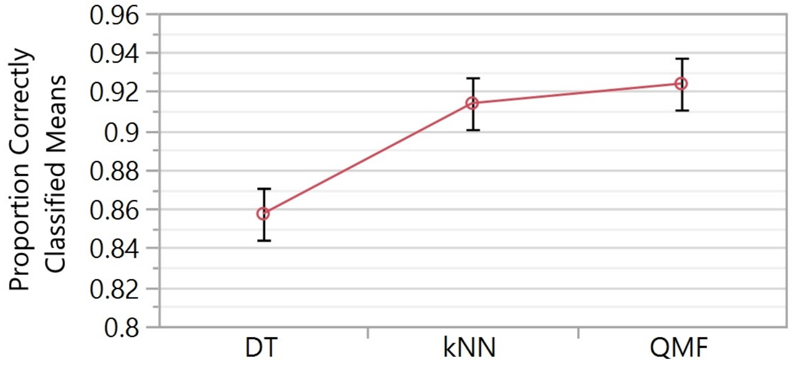


Figure 25: Means of Proportion of Correctly Classified Samples, Soybean Data Set with 9 Classes, 501 Samples, 20 Independent Test Runs.
DT - M: 0.858; SE: 0.0066, kNN - M: 0.914, SE: 0.0066; QMA - M:0.924, SE: 0.0066.

4.4.2.3 Summary of Results as a Batch Classifier.

In summary, the results reveal that the QMA algorithm yielded more accurate classification than kNN and DT algorithms for the Soybean data set, and more accurate classification than DT for the malware data set. In contrast, the QMA yielded fewer accurate classifications than DT for the Voting data set, and provided equivalent classification accuracy for the Audiology data set.

While the QMA model classified queries with increased accuracy relative to the benchmark ML algorithms, the processing overhead of situating the model in a cognitive architecture, and concurrently training the autonomous module with inferred responses, results in a slower processing speed, compared to the benchmark algorithms.

4.4.3 Computational Complexity Order Of Magnitude (OOM).

Table 13 presents a summary of the computational complexity Order Of Magnitude (OOM) for kNN, DT and QS given a 5-fold cross-validation technique.

Table 13: Computational Complexity OOM for kNN, DT and QS given 5-fold Cross-Validation Technique. m = sample size, a = number of features (variables).

	kNN	DT	QMA
Training:	$O(a(3/5)m)+$ $O(a(3/25)m^2)$	$O(a(3/5)mLog(3/5m))$	$O(a(3/5)m)$
Test one sample:	$O(a(3/5)m)$	$O(Log(3/5m))$	$O(a(6/5)m)$
Test all samples:	$O(a(3/25)m^2)$	$O(Log(3/5m)(1/5m))$	$O(a(6/25)m^2)$
Total cost:	$O(a(3/5)m)+$ $O(a(6/25)m^2)$	$O([a(3/5)+$ $(1/5)]mLog(3/5m))$	$O(a(3/5)m)+$ $O(a(6/25)m^2)$

The total cost for training and testing using the 5-fold cross-validation technique for kNN and QS are the same. The QS algorithm is affectively a 1NN algorithm with no validation step and a novel distance measure, therefore the training for QS is just the cost of converting the categorical data from a positional parameter list into an ordered binary representation, $O(a(1/5)m)$. The kNN training includes the conversion of data to an ordered binary representation, $O(a(1/5)m)$, and the validation step, $O(a(3/25)m^2)$. However, in the testing phase kNN has only one (Euclidean) distance to evaluate. In QS the distance measure requires evaluating two eccentricity measures, therefore the test OOM is twice as large and the total costs are the same. The complete discussion of OOM for all three algorithms is lengthy, and can be found in Appendix D.3.

4.4.4 QMA Ordering Effect and Convergence.

In QMA the ordering effect (defined in Appendix D.5) depends on whether or not the framework is used as a batch or incremental learning algorithm, and the variability of the *target variable*. When the QMA is used as a batch algorithm and the *target variable* remains constant in the test samples, there is no ordering-effect. This is due to the fact that the test samples all have the same target variable, therefore they do not contribute to the inference (predictive function) of subsequent test samples. Furthermore, batch QMA does not *forget* any previously learned samples. Therefore, when used in this way, a classifier is order-independent (Cornuéjols, 1993).

QMA is *order-dependent* when it is used as an incremental learning algorithm or when the *target variable* does not remain constant in the test samples. Each incoming *training* sample potentially contributes to the inference (predictive function) for each subsequent *test* sample. Each incoming *test* sample potentially contributes to the inference (predictive function) for any subsequent *test* sample in which the *target variable* differs.

For both batch and incremental learning, the QMA is effectively a weighted 1-nearest neighbor (kNN, k = 1) nonparametric classifier. It does not maintain any information, or make any calculations, based on the underlying probability function. Therefore, QMA becomes a *Bayes classifier* if the probability estimates converge to the true probability when an infinite number of samples are used. The resulting error is the *Bayes error* (Keinosuke, 1990). Bayes probability of error for a multiclass classifier with discrete features is defined in Appendix D.6.

Theoretical results, and experimental studies, demonstrate that the Generalization Error (GE) of the 1NN algorithm converges to Bayes error, asymptotically, as the number of samples increases (Dhurandhar and Dobra, 2013; Imandoust and Bolandraftar, 2013). Keinosuke (1990, 307) and (Duda et al., 2001, 26) further demonstrate that the error for 1NN is less than twice the Bayes error in the case of infinite number of samples.

4.5 Summary

This chapter presented the results of data analysis and findings. This chapter supports the research hypothesis, demonstrating that the computational agent can function as a supervised classifier and overcome the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes. As a supervised learning *incremental* classifier the QMA demonstrates support for concept drift when target variable occurs. In addition to increased classification accuracy with the multivalued/multiclass data sets, and support for changing target variables, the QMA model has additional advantages over the benchmark ML algorithms: one test sample can have multiple *target variables*, which can be inferred in series; at any time a domain expert can correct a sample, again, without re-training the entire model; and, over time, as the autonomous module is trained with responses, subsequent queries are more likely to have a pattern-matching response available, therefore, the slower QS Computational Module inference will not be evoked, resulting in a faster response.

As a supervised learning *batch* classifier the research reveals that a cognitive architecture, modeled after a theory of human consciousness, can yield more accurate classification results than modern ML algorithms kNN and DT. Further research suggests that the QMA demonstrates improved classification accuracy in data sets with greater dimensionality.

V. Findings and Conclusions

In this chapter the key findings are summarized, details of the contributions of this dissertation are provided and areas for future research are explored.

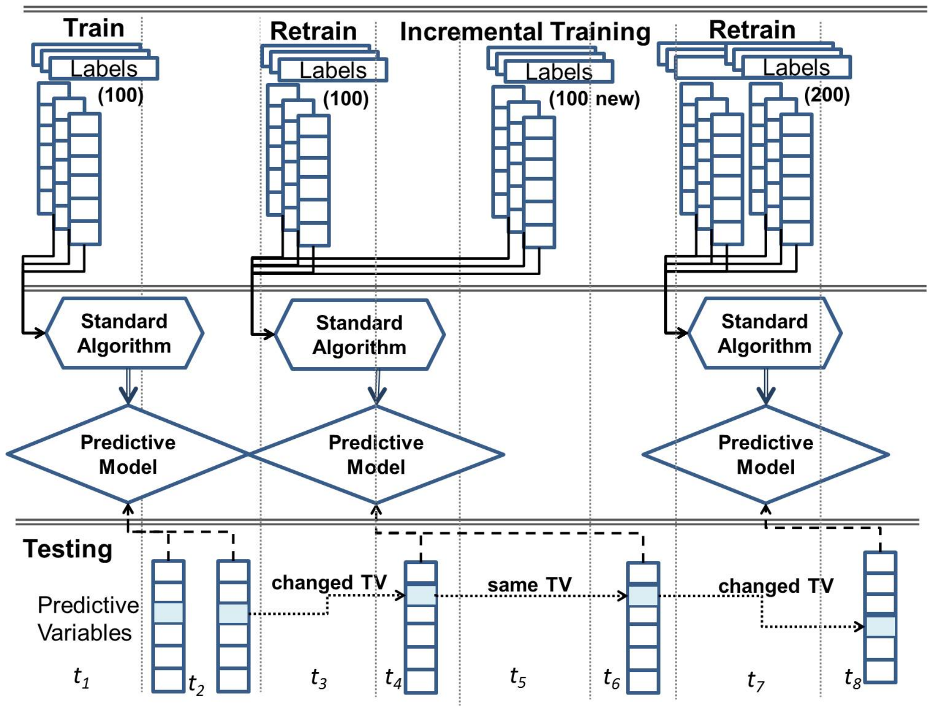
5.1 Key Findings

5.1.1 Support for the Research Hypothesis.

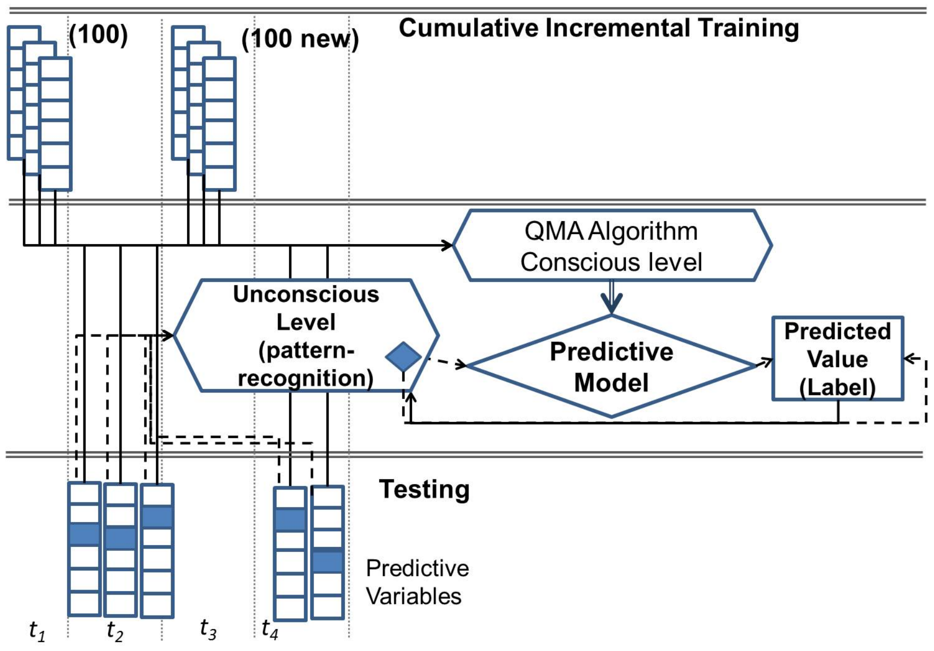
The key findings support the hypothesis of this dissertation, revealing that a computational agent, modeled after a theory of human learning-by-experience, can function as an incremental supervised learning classifier, overcome the necessity to identify the target variable before training begins, and the necessity to re-train the cumulative training samples when the target variable changes. The Qualia Modeling Agent (QMA) overcame additional limitations of standard learners: the failure to retain test samples and their predictive values; the failure to retain response variables and their predictive values; and, with *some* learners, the inability to incorporate additional training samples after testing begins. This last limitation applies only to learners that work exclusively as batch classifiers.

5.1.2 Comparing Training Requirements between QMA and Benchmark Learners.

Figure 26 illustrates a comparison of computational complexity Order Of Magnitude (OOM) for training requirements between a standard incremental learner and the QMA, when the target variable changes, for 200 unique training samples and 5 test samples.



(a) Standard Supervised Learning Incremental Classifier.



(b) Qualia Modeling Agent (QMA) Cumulative Incremental Classifier.

Figure 26: Example of Training OOM Comparison Between (a) Standard Learner and (b) QMA.

The standard learner (Figure 26a) trains a total of 500 samples: time t_1, the original 100 samples; t_3, the original 100 samples again, when the target variable changes; t_5, the next 100 incremental samples; and, t_7, the total 200 cumulative samples, when the target variable changes again. The QMA (Figure 26b) trains once for each unique training sample: 100 at time t_1; and another 100 at t_3. Additionally, it incorporates the predictive variables from the test samples which potentially contribute to improved classification accuracy of subsequent test samples, for a total of 205 training samples. The QMA also retains responses (test sample/response variable pairs) for subsequent pattern-recognition of duplicate test samples. In Figures 26a and 26b solid lines indicate variables incorporated in the predictive model. Dashed lines indicate variables not incorporated in the predictive model. Filled cells in predictive variables represent changing target variables.

5.1.3 Improved Classification Accuracy.

The QMA overcame the necessity to identify the target variable before training begins and the necessity to re-train when the target variable changes, while achieving classification accuracy comparable to, or greater than, benchmark classifiers. The QMA was compared against two nonparametric benchmark supervised learning classifiers: decision tree (DT) and k-nearest neighbors (kNN). DT does not support incremental learning, therefore DT must start training from scratch in order to incorporate additional training samples or accommodate a change in target variable. The QMA classified 2 out of 4 data sets with greater classification accuracy than DT, 1 of the 4 data sets with statistically equal accuracy, and the remaining data set DT classified with greater accuracy. kNN does support incremental learning, however, if the target variable changes kNN must re-train all cumulative test samples. The QMA classified 1 out of 4 data sets with greater classification accuracy than kNN, and the remaining 3 of the 4 data sets with statistically equal accuracy.

5.1.4 Practical Application.

Supervised classifiers are used to support decision making in diverse, often crucial, industries, for example: military, health care, farming, marketing, financial, manufacturing. Organizations frequently maintain large volumes of data (terabytes) for their own analysis, or to satisfy regulatory requirements (Maimon and Rokach, 2005). Standard supervised learning classifiers are typically trained on these data to answer a specific question or predict a specific outcome, for example: a diagnosis, approve a mortgage, a crop yield, the demand for a certain product, etc. If a different question is asked from the same data, the standard learner has to be re-trained on the cumulative training samples, which is not only time consuming, but the learner is out of service for it's primary purpose while answering alternate questions, for example: predicting specific symptoms given a diagnosis; predict profit/loss if the mortgage is approved; given a desired crop yield, estimate water or fertilizer requirements; and, given a demand for a certain product, estimate factory hiring requirements.

The QMA, efficiently adjusts to changing target variables and would provide increased efficiency and availability over standard classifiers. The QMA can function as a decision aid in complex environments where data are too broad or diverse for a human to evaluate without computational assistance, with increased flexibility and efficiency over these standard learners.

5.1.5 How QMA Contributes to Machine Learning (ML) Research.

The findings of this dissertation fit into concept drift and Transfer Learning (TL) research, areas of machine learning (ML) which address the challenge of leveraging previously acquired knowledge in order to improve efficiency or accuracy in a new domain that is in some way related to the original domain (Mooney, 2016). The QMA efficiently adapts to concept drift. Concept drift occurs when the statistical properties of the target variable change (Pan and Yang, 2010). The QMA is efficient, because no re-training is required when the target variable changes. The QMA potentially improves accuracy of classification, because the predictive variables from test samples are retained.

5.1.6 How QMA Contributes to Cognitive Modeling Research.

This research developed an abstract implementation of the Integrated Information Theory (IIT) of Consciousness (Balduzzi and Tononi, 2009) that is computationally tractable. More specifically, an implementation of the *maximally integrated information structures* of qualia space (QS) as proposed by IIT, as well as, a formalism to perform *fully disjunctive reasoning (FDR)* over QS. Previous research (Aleksander and Gamez, 2011) modeled QS, as specified by Balduzzi and Tononi, and demonstrated the implementation to be computationally intractable with as few as 30 elements.[1] The implementation presented here was applied to over 180[2] elements, and is computationally tractable on a typical desktop computer. The QMA implementation of IIT is a unique contribution to computational models of consciousness.

This research also developed an implementation of a theoretical model of learning and decision making, specifically, Stanovich's tripartite framework. The primary functions of the *minds* and working memory (WM) modeled both unconscious decision making by pattern-recognition, and conscious decision making by pattern-completion, as well as re-training the unconscious level with knowledge learned from *FDR* in the conscious level. The QMA implementation of Stanovich's framework is a unique contribution to the cognitive modeling literature.

[1]Each unique value from all predictive variables is an *element*.

[2]The University of California, Irvine (UCI) audiology diagnosis data set has the largest dimensionality of the data sets tested, containing 181 *elements*.

5.1.7 How QMA Contributes to Computational Models of Narrative (CMN).

An area of research dedicated to understanding the structure of narratives is Computational Models of Narrative (CMN). Cognitive research has demonstrated that almost every aspect of our lives are comprehended through narrative. We understand events, make decisions and structure our thinking based on verbal narratives. Narratives allow us to generate a stable, consistent and useful representation of reality, and allow us to select between competing plausible narratives simulated in WM. Computer systems attempting to model narrative need to address the underlying cognitive structure of both the conscious and unconscious elements of narrative (Lakoff and Narayanan, 2010; Vaughan et al., 2014). The QMA implementation of Stanovich's framework and IIT is a novel contribution to CMN, by presenting an architecture which models the underlying cognitive structure of both the conscious and unconscious decision making, allowing for competing plausible narratives simulated in WM.

5.1.8 How QMA was Built.

The QMA was built by modeling a widely accepted theory of human learning and decision making, Stanovich's framework, consisting of a conscious and an unconscious level. The unconscious level supports fast decision making using pattern-recognition of previous responses (analogous to test sample/response variable pairs) in an extended implementation of an Adaptive Control of Thought–Rational (ACT–R) cognitive modeling toolset. As the agent learns over time, previously predicted responses may no longer be accurate, therefore they are allowed to obsolesce using the activation levels integral to the ACT–R cognitive model, which represents memories fading from lack of recency. ACT–R functionality was extended to provide preattentive processes to the algorithmic mind and to accept reprogramming from inferences generated in WM.

The conscious level performs *FDR* over a multidimensional array representation of all cumulative training samples and test samples. The mathematical formalism, which performs FDR and generates an inferred response variable, is based on Hypernetwork theory and extensions to hypernetwork theory which are a principal contribution of this dissertation. All of the custom software for this implementation was written in Common Lisp (CL), including the hypernetwork theory formalism and interfaces with ACT–R. A second instantiation of ACT–R is used to convert the inference vector to ACT–R format for reprogramming the unconscious level.

The QMA mathematical formalism, representing the conscious level, is analogous to a standard first nearest neighbor (1NN) supervised classifier. However, the formalism uses a novel, weighted, distance measure for inference. Standard learners, such as kNN, use the Euclidean distance measure, or other standard distance measure, discussed in Appendix E. The QMA distance measure is inspired by a computationally tractable model of the maximally integrated information structures of consciousness as proposed by IIT.

5.1.9 Limitations.

The research presented in this dissertation was not able to demonstrate improved classification accuracy with partial or corrupt training samples, when compared to the benchmark classifiers: DT and kNN. When data elements were randomly removed or corrupted, the three algorithms continued to classify test samples with the same relative accuracy. Another limitation revealed in this research, is that the QMA classified with less accuracy, than DT, when there was less dimensionality in the data set. For example, the 1984 Congressional voting data set is entirely binary, for all variables there are only two values: Democratic or Republican, or, yes or no. DT classified the voting data set with greater accuracy than both QMA and kNN. The audiology diagnosis data set variables were predominantly binary (61 out of 70), and all three algorithms classified with statistically comparable accuracy.

5.2 Research Contributions

This dissertation's novel research contributions are:

1. A computational agent, the QMA which functions as an incremental supervised learning classifier, overcoming specific limitations of standard learners.
2. Extensions to hypernetwork theory resulting in an inference generating formalism.
3. A novel binary distance measure based on extensions to hypernetwork theory.
4. A computationally tractable implementation of the Integrated Information Theory (IIT) of Consciousness.
5. A computational model of Stanovich's Tripartite Framework.
6. A novel contribution to CMN, by presenting an architecture which allows for competing plausible narratives.

The primary contribution of this dissertation is a computational agent, the QMA, that functions as a batch or an incremental supervised learning classifier and overcomes the necessity to identify the target variable before training begins and the necessity to re-train the cumulative training samples when the target variable changes. The QMA achieves classification accuracy comparable to or greater than benchmark classifiers, and performs at the same computational complexity OOM as the kNN benchmark classifier. A computational agent that overcomes these limitations has not appeared before in the research literature.

The second contribution are extensions to hypernetwork theory to create a distance measure and an inference (response variable). The *simplicial complex* concept is extended with four features. First, multiple simplices with the same *Simplex Label* are supported, Second, a *frequency* variable was added to each simplex, which captures the count of experiences with the same variable set (i.e., important details of experiences). Third, the option of a *Query Element* was added, which indicates a Simplex Label element which will result in an inference generation. Fourth, a *Query Simplex* is defined, which is a simplex with one *Query Element*. The concept of *eccentricity*, an asymmetric measure of connectivity between simplices, is extended to complete the inference generating formalism. Eccentricity was extended with a Hausdorff metric to produce a distance measure. The novel distance measure was further extended with a weight and a distance measure between an individual simplex and a family of simplices, to produce an inference. Leveraging the multidimensional relational structures inherently available in hypernetwork theory, to infer Query Elements, simulate the maximally integrated information structures, i.e., shapes, in QS, produce a binary distance measure, and generate an inference is not found elsewhere in the research literature.

The third contribution is a novel binary distance measure. The extensions to hypernetwork theory resulted in a binary distance measure which calculates a distance between point sets in multidimensional space: $d_Q = \max\left(\left(\frac{c}{a+c}\right), \left(\frac{b}{a+b}\right)\right)$, Range: $[0, 1]$ (see Appendix E). This distance measure, along with the weighting function, were incorporated in the formalism presented in this dissertation which demonstrated increased classification accuracy with some multivalued/multiclass data sets. This distance measure is not found elsewhere in the research literature. A proposed area of future research is comparative analysis of d_Q with other binary distance measures, given select performance criteria, such as improved classification accuracy, consistency or outlier identification.

The fourth contribution is a computationally tractable model of consciousness, i.e., an implementation of the *maximally integrated information structures* of QS as proposed by IIT, as well as, a formalism to perform *fully disjunctive reasoning (FDR)* over QS. No previous research presenting a computationally tractable implementation of IIT was found in the literature, and none that implemented FDR or generated an inference were found in the literature.

The fifth contribution is a computational model of Stanovich's Tripartite Framework. This contribution provides a decision-making framework in which to place the computational model of consciousness. The model simulates cognitive decoupling, which allows the reflective mind to override the autonomous mind and engage QS for inference generation, and reprogramming of the unconscious level by the conscious level based on overlearning and practice. A computational model of Stanovich's framework is not found elsewhere in the research literature.

The sixth contribution is a novel contribution to CMN, by presenting an architecture which models the underlying cognitive structure of both the conscious and unconscious decision making, allowing for competing plausible narratives simulated in WM.

This research presents contributions to four bodies of knowledge: *machine learning (ML)* with a novel classification algorithm, support for concept drift and a novel distance measure; extensions to *hypernetwork theory*; *cognitive modeling* with operational, computationally tractable, models of Stanovich's framework and IIT; and, *Computational Models of Narrative (CMN)*, by presenting an architecture which models the underlying cognitive structure of conscious and unconscious decision making, allowing for competing plausible narratives.

5.3 Areas for Future Research

Develop a Threshold for Tightly Compiled Learned Information (TCLI). In the QS Computational Module all inferences generated in QS are sent to the Autonomous Module as TCLI. This assumes that all inferences meet the requirement that these inferences have been "...overlearned and practiced (Stanovich, 2009)" enough to become TCLI. Future work could integrate a threshold, based on participant experimentation, below which inferences would not be sent to the Autonomous Module for reprogramming. ACT–R activation levels, already present in the conscious ACT–R model could be modulated to implement this additional detail.

Modulate agent response to model the cognitive miser. The agent response, as configured in this dissertation, models *FDR* (Stanovich, 2009), and is always either an inference directly from QS, or TCLI, which was previously generated from an inference. However, humans do not always use *FDR*, which is evaluating all alternative hypothesis. Stanovich (2009) present several scenarios where humans tend to be *cognitive misers*, and do not use *FDR*. Advancing the QMA model, representing these *cognitive miser* processes and more accurately modeling human decision making, could be implemented.

Ordinal, Interval or Ratio Scales. Future research could extend this formalism to incorporate real-world data sets with ordinal, interval or ratio scaled attributes which will allow for generating a more complete set of inferences, including spatial and temporal relationships. In addition, this model could be compared to observed behavior from a participant–based experiment, similar to Faghihi et al. (2015) experiments in which associative memory responses from animal subjects were modeled, and the visual qualia modeled by Arrabales et al. (2010).

Standard Web Formats. Henson et al. (2012) propose that "...the perception process must generate abstractions [inferences] from observations [experiences] encoded in Web languages. Much sensor data is encoded in standard Web formats, is made accessible on the Web, and is increasingly annotated with a sensor ontology." Integrating the QMA with web-available sensor technology would demonstrate additional utility, and provide current real-world data analytics.

Distance Measure Research. Binary similarity and dissimilarity (i.e., distance) measures perform critical roles in research, such as, pattern analysis, classification, statistics, clustering, and pattern recognition. Numerous distance measures (over 100 unique binary distance measures in Appendix F) have been developed in various research fields. The same measure is often developed independently from different research areas, as is evident in the first column of Appendix F. Applying an appropriate distance measure is important to improved data analysis, yet, despite voluminous literature on the subject, no guidelines exist for selecting optimal measures for given data set features or research goals (Morris, 2012). Distance measures are generally selected based on proven superior performance, given select criteria, on a certain data sets (McCune et al., 2002). In other words, the recommended approach is empirical evidence based on trial and error.

This dissertation proposes a novel distance measure developed from hypernetwork theory measures of eccentricity, $d_Q = \max\left(\left(\frac{c}{a+c}\right), \left(\frac{b}{a+b}\right)\right)$, Range: $[0,1]$. A proposed area of future research is comparative analysis of d_Q with other binary distance measures, given select performance criteria, such as improved classification accuracy, consistency or outlier identification.

This dissertation has compared the d_Q, using a weighting measure, against the Euclidean distance in kNN formalism, with four specific data sets. Future research might demonstrate utility by:

- comparing d_Q against the Euclidean distance for classification accuracy with no (d_Q) weighting factor and no kNN validation (optimal k-selection) step, i.e., 1-Nearest Neighbor, or,
- comparing d_Q against other distance measures using the same, or perhaps additional, data sets, given select performance criteria. Some candidate distance measures, for comparison, are those that also seek maximal distance measures, such as Goodman and Kruskal (1954) Probability, Goodman and Kruskal (1954) Lambda, and Relative Decrease of Error Probability, see Appendix F.

Prove, or Disprove, the d_Q Distance Measure is a Metric. A recommended area for future research is to to prove, or disprove, this distance measure, d_Q, as defined in Section 3.6.3.1 satisfies the four properties of a metric.

Demonstrate the Algorithm's Transfer Learning (TL) Capabilities. In order for TL to be achieved three criteria are necessary: first, one or more, of the domain or task components must have changed; second, the new domain must in some way be related to the previous domain; and, third, improved performance, along some axis, must be demonstrated (Pan and Yang, 2010). In the QMA, when the target variable changes, concept drift occurs and all four of the task and domain components change and, with the exception of the source and target *target variables*, the domain feature space remains the same. However, further research is necessary to demonstrate that improved performance, along some axis, has been achieved. Some axis of performance are: improved classification accuracy and improved efficiency.

In order to test for and potentially demonstrate improved classification accuracy or efficiency, the QMA algorithm can be run in two separate modes, comparing performance against itself when concept drift is not supported, and when concept drift is supported. First, with no target variable change without retraining supported, and subsequently in the formalism for which it was designed, testing efficiency and classification accuracy against itself.

In the first case, multiple batch tests would be run, restarting the algorithm and changing the target variable between batches. Noting the computational complexity OOM of retraining and testing and the classification accuracy of the combined multiple batches. Subsequently, the same training and test runs would be run in one continuous test, changing the target variable to match the original test, without restarting the algorithm or retraining the cumulative samples. The efficiency of the second test should demonstrate improvement over the first. Because the *test sample* predictive variables are retained within the training samples, given a large enough test sample set, the classification accuracy of the second set is expected to be improved over the first, as well.

Appendix A. Types of Logical Reasoning

Abductive reasoning. "Abductive reasoning is inferring a case (particular abstract relationship) from a rule (abstract, general claim) and a result (empirical observation) (Shanahan, 1996)."

The following is the theory of abductive reasoning originally written by Peirce (1974), as summarized by Svennevig (2001):

> Abduction is not just choosing any hypothesis, but selecting one as more plausible than the others. Peirce presents a set of criteria for choosing the best hypothesis. And here his theory is more explicitly concerned with the methodology of scientific inquiry. He mentions three criteria for favoring one hypothesis over others, namely:
>
> 1. The hypothesis should explain the facts
> 2. It should be economical
> 3. It should be capable of being subjected to experimental testing. (Svennevig, 2001)

Furthermore, humans use abductive reasoning to infer knowledge not available in the environment. Perception, i.e., inferring knowledge, isn't a deductive or inductive process, but rather an abductive process, meaning an inference to the best explanation (Henson et al., 2012; Shanahan, 1996). Generating a series of competing plausible explanatory hypothesis, and choosing the best based on some set of criteria (Henson et al., 2012).

Henson et al. (2012) propose an abductive reasoning framework, Parsimonious Covering Theory (PCT), for evaluating *abstractions* generated by sensory observations and context. PCT has primarily been used in medical disease diagnosis. PCT provides a model of abductive reasoning, i.e., computing the best explanation given a set of observations. PCT uses an hypothesize-and-test abductive inference process.

Deductive reasoning. A general rule is applied to a specific case, i.e., from the general to the specific. Deductive reasoning is the only way to achieve a provable, logical, solution (Svennevig, 2001).

Inductive reasoning. A generalization is inferred given a specific case, i.e., from the specific to the general. In inductive reasoning the conclusion is not assured to be correct (Svennevig, 2001).

Transductive reasoning. Transductive reasoning is inferring from one specific experience to another specific case (Vapnik, 2006).

Appendix B. Hypernetwork Theory: A Mathematical Formalism

This appendix is a summary of (1) the basic components of set theory on which hypernetwork theory is based, (2) the definition of metric space, which is used in the extensions to hypernetwork theory developed in this dissertation, (3) the definition of a Hausdorff distance (aka Hausdorff metric), which is also used in the extensions to hypernetwork theory developed in this dissertation, (4) the elements of hypernetwork theory used in this dissertation, and in the interest of integrity, (5) repeated in this appendix are the extensions to hypernetwork theory developed as part of this dissertation. A comprehensive, contemporary, discussion of hypernetwork theory can be found in Johnson (2013).

B.1 The Mathematical Basis of Hypernetwork Theory: Set Theory

Set theory notation and definitions, necessary for this discussion of hypernetwork theory, are summarized here, excerpts from Apostol (1974).

B.1.1 Notation.

The following notation is customarily used in set theory and will be used in this dissertation.

Sets will be denoted by capital letters: $A, B, C, \ldots, Z$; and sometimes in mathematical formulae as: $\mathbb{A}, \mathbb{B}, \mathbb{C}, \ldots, \mathbb{Z}$, or, $\mathcal{A}, \mathcal{B}, \mathcal{C}, \ldots, \mathcal{Z}$.

Elements will be denoted by lower-case letters: $a, b, c, \ldots, z$. Elements in an unordered set will be surrounded by curly braces, e.g., $A = \{a, b, c, \ldots, z\}$. Elements in an ordered set, i.e., a set where the order is relevant, will be surrounded by parenthesis, e.g., $A = (a, b, c, \ldots, z)$.

Element belongs to a set will be denoted by $x \in S$, which means x belongs to S, or x is an element of S.

Element does not belong to a set will be denoted by $x \notin S$, which means x does not belong to S, or x is not an element of S.

B.1.2 Definitions.

Definition B.1 (Set)**.** A set S is a collection, infinite or finite, of distinct elements.

Definition B.2 (Ordered pair)**.** A set of two entities in which the order is important. Let A and B be two sets. Given $a \in A$ and $b \in B$, define the ordered pair with the first component a and the second component b, (a, b). Where $(a, b) \neq (b, a)$.

Definition B.3 (Cartesian product of two sets)**.** Cartesian product of two sets A and B is a new set denoted by $A \times B$ which consists of all ordered pairs of the form (a, b), where $a \in A$, and $b \in B$. That is, $A \times B = \{(a, b) : a \in A \text{ and } b \in B\}$.

Definition B.4 (Relation)**.** A relation is any set of ordered pairs.

Definition B.5 (Function)**.** A function F is a set of ordered pairs, $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, such that no two have the same first element. That is, if $x_1 = x_2$, then $y_1 = y_2$.

B.2 Metric Space

Definition B.6 (a metric space)**.** A metric space is a nonempty set $\mathbb{M}$ of objects (called points) together with a function d from $\mathbb{M} \times \mathbb{M} \Rightarrow \mathbb{R}$ (called the metric of the space) satisfying the following four properties for all points x, y, z in $\mathbb{M}$:

1. $d(x, x) = 0$

2. $d(x, y) > 0$ if $x \neq y$

3. $d(x, y) = d(y, x)$

4. $d(x, y) \leq d(x, z) + d(z, y)$. [triangle inequality]

The nonnegative number $d(x, y)$ is to be thought of as the distance from x to y.

A narrative explanation of metric space, provided by Peeler (2011), is presented here, referencing the definition provided by Apostol (1974):

> A metric space is a precise formal measure of distance given any set. Then we call d a metric on $\mathbb{M}$, and $\mathbb{M}$ together with d is called a metric space $(\mathbb{M}; d)$. A metric as defined gives us a formal way to view the notion of distance between points in a set. The [second] property is simply justified -distance is never taken to be negative by convention and is only zero for two nondistinct points. The [third] property makes good sense as well-distance ought not to depend on which point is considered first. The [fourth] property comes as a necessity. Consider the set of points comprising a Euclidean plane: any three non-collinear points form a triangle, and any one side length of such a triangle must be less than the sum of the other two side lengths. Therefore, the distance between two points must be less than the sum of the distances between each of those points and a third point. (In the case of points along the same line we have equality). This property is often referred to as the triangle inequality. (Peeler, 2011)

The reason for ensuring the distance measure used is a metric space is to guarantee that the distances between all members of the set are defined. There is no set of members, however obscure, where the distance is undefined. Additionally, a metric space creates well-understood topological properties such as open and closed sets, which contribute to a better understood behavior of the model (Apostol, 1974).

However, there are many similarity and distance measures, which are, in fact, not metric spaces. They are often referred to as *robust non-metric distances* (Jacobs et al., 2000), and successfully used in classification algorithms. For instance, the Euclidean distance is a metric space, but the Squared Euclidean (aka Hamming, Manhatten, City Block, when data are binarized), is not a metric space (see Appendix F). The

Squared Euclidean fails the triangle inequality test.

If $x = (0,0), y = (0,2)$, and $z = (0,1)$ as elements of $\mathbb{R}^2$ for which the pair-wise distances are: $d_{SE}(x,y) = 4, d_{SE}(y,z) = 1$, and $d_{SE}(x,z) = 1$, fails the triangle inequality, $4 \leq 1 + 1$ (Gardner et al., 2014).

B.3 Hausdorff Distance

A Hausdorff distance, aka Hausdorff metric, is a metric, because it satisfies the four properties of a metric, defined in Metric Space (Sternberg, 2010).

Definition B.7 (Hausdorff distance)**.** The Hausdorff distance between two sets of numbers $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_m\}$ is defined as:

$$d(A,B) = \max\left\{\max_{a\in A}\min_{b\in B}|b-a|, \max_{b\in B}\min_{a\in A}|a-b|\right\}. \tag{13}$$

In other words, the Hausdorff distance between A and B is the smallest value d such that every point of A has a point of B within distance d and every point of B has a point of A within distance d.

The Hausdorff distance can also be defined for point sets in two or more dimensions, where $|a - b|$ must be replaced by the Euclidean distance between a and b (or any other appropriate distance measure.) (Rote, 1991)

The last paragraph of Rote's definition is utilized in the Qualia Modeling Agent (QMA) formalism, in which multiple dimensions are defined, and $|a - b|$ is replaced by the appropriate distance measure, d_Q, defined in Definition B.15.

B.4 Hypernetwork Theory Definitions

Definition B.8 (Incidence Matrix)**.** "A matrix is called the **incidence matrix** (denoted by Λ) of the relation λ is an array of numbers λ_{ij}, with each λ_{ij} being either

0 or 1. The number λ_{ij} equals 1 if a_i is λ-related to b_j and is 0 otherwise (Johnson, 2013)."

Example B.1. Incidence Matrix. If $A = \{1, 2, 3\}$, $B = \{0, 4, 8, 10\}$ and λ is the relation *less than*, then λ is the subset of $A \times B$ defined by the ordered pairs:
$\lambda = \{(1, 4), (1, 8), (1, 10), (2, 4), (2, 8), (2, 10), (3, 4), (3, 8), (3, 10)\}$. For this example, the relation is a 2-dimensional *incidence matrix*, illustrated in Table 14:

Table 14: Example Incidence Matrix.

	(B)				
	λ	0	4	8	10
$\Lambda = (A)$	1	0	1	1	1
	2	0	1	1	1
	3	0	1	1	1

Definition B.9 (Polyhedron)**.** In algebraic topology [polyhedron] is defined as a space that can be built from such *building blocks* as line segments, triangles, tetrahedra, and their higher dimensional analogs by connecting them at points, or along their edges, faces, or greater dimensional structures. Original source (Munkres, 1993), as found at (Wolfram, 2015).

> Polyhedra are the geometric realisation of more abstract objects called simplices. Let V be a set of vertices. An abstract p-simplex is determined by a set of $p + 1$ vertices, written as $\langle v_0, v_1, ..., v_p \rangle$. Simplices are often represented by the symbol σ. (Johnson, 2013)

Definition B.10 (Simplex)**.** "A simplex is the generalization of a tetrahedral [or greater] region of space to n dimensions. The boundary of a k-simplex has $k + 1$

0-faces (vertices), $k(k+1)/2$ 1-faces (edges) and $\binom{k+1}{i+1}$ i-faces, where $\binom{n}{k}$ is a binomial coefficient (Johnson, 1995)."

More specifically, a simplex can be defined as the underlying space of a *simplicial complex*, with the additional constraint sometimes imposed that the complex be finite (Monk, 1974; Munkres, 1993).

> The simplex $\sigma = \langle v_0, v_1, ..., v_p \rangle$ is a p-dimensional face, or p-face, of the simplex $\sigma' = \langle v_0, v_1, ..., v_p \rangle$ if every vertex of σ is also a vertex of σ'. For example, the 3-dimensional tetrahedron $\langle v_0, v_1, v_2, v_3 \rangle$ has four 2-dimensional triangular faces $\langle v_1, v_2, v_3 \rangle$, $\langle v_0, v_2, v_3 \rangle$, $\langle v_0, v_1, v_3 \rangle$, $\langle v_0, v_1, v_2, \rangle$. (Johnson, 2013)

Definition B.11 (Simplicial Complex)**.** "A set of simplices with all of their faces (Johnson, 2013)."

A *simplicial complex* can be represented by an incidence matrix:

- A p-dimensional polyhedron has $p + 1$ vertices.
- The vertices of networks, $\langle v \rangle$, have dimension zero.
- Edges, $\langle v_0, v_1 \rangle$, have dimension one but two vertices.
- For higher dimensional polyhedra, a triangle $\langle v_0, v_1, v_3 \rangle$ has dimension two but three vertices. A tetrahedron $\langle v_0, v_1, v_3, v_4 \rangle$ has dimension three but four vertices, and so on.
- By labelling the first vertex v_0, the last vertex of a p-dimensional polyhedron can be labelled v_p, and this convention will be used as appropriate throughout this appendix.
- Thus the generality is that a p-dimensional polyhedron will be written as $\langle v_0, v_1, ..., v_p \rangle$. (Johnson, 2013)

Definition B.12 (Eccentricity of a simplex with respect to another (Johnson, 2013))**.**

$$ecc(\sigma|\sigma') \overset{def}{=} \frac{|\sigma \smallsetminus \sigma'|}{|\sigma|} = \frac{\text{number of } \sigma \text{ vertices not shared with } \sigma'}{\text{number of vertices of } \sigma} \tag{14}$$

Definition B.13 (Eccentricity of a simplex with respect to a family of simplices F

(Johnson, 2013)).

$$ecc(\sigma|F) \overset{def}{=} \min\{ecc(\sigma|\sigma')|\sigma' \text{ belongs to } F\} \tag{15}$$

Definition B.14 (Eccentricity of a simplicial family F with respect to family F' (Johnson, 2013)).

$$ecc(F|F') \overset{def}{=} \min\{ecc(\sigma|\sigma')|\sigma' \text{ in } F'\} \tag{16}$$

Eccentricity cannot be used directly as a metric because it does not satisfy the requirement for symmetry, the third property in Definition B.6. The eccentricity of σ with respect to σ', $ecc(\sigma|\sigma')$, is not guaranteed to be equal to the eccentricity of σ' with respect to σ, $ecc(\sigma'|\sigma)$.

$$(ecc(\sigma|\sigma')) \neq (ecc(\sigma'|\sigma)) \tag{17}$$

B.5 Extending Hypernetwork Theory

In order to satisfy the requirement of symmetry in a metric a Hausdorff distance (Rote, 1991) was applied, and the distance (d_Q) between two simplices defined to be the maximum of $ecc(\sigma|\sigma')$ and $ecc(\sigma'|\sigma)$.

Definition B.15 (Distance between two simplices). Let σ and σ' be two simplices. The distance (d_Q) between σ and σ' is defined to be:

$$d_Q(\sigma, \sigma') \overset{def}{=} \max\{ecc(\sigma|\sigma'), ecc(\sigma'|\sigma)\} \tag{18}$$

Definition B.16 (Distance between a simplex and a family of simplices F). Let σ be a simplex, and F be a family, i.e., a set of simplices. The distance ($dist_Q$) between

σ and F is defined to be:

$$dist_Q(\sigma, F) \overset{def}{=} \min\{d_Q(\sigma, \sigma') : \sigma' \text{ belongs to } F\} \tag{19}$$

The weight, w_Q, illustrated in Figure 27 is defined to be the proportion of simplices, σ, in family, F, where $d_Q(\sigma_q, \sigma)$ is equal to $dist_Q(\sigma_q, F)$:

$$w_Q(\sigma_q, F) \overset{def}{=} \frac{card\{\sigma \in F : d_Q(\sigma, \sigma_q) = dist_Q(\sigma_q, F)\}}{card(F)} \tag{20}$$

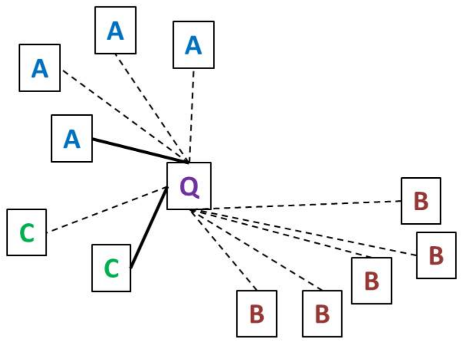


Figure 27: Illustration of Weight Function. Given three families, A, B, C, and a Query Simplex, Q. The minimum distance, $dist_Q$, is represented with a solid line. Since $dist_Q(Q, A) = dist_Q(Q, C)$, the closest *family* is determined by the *largest* weight function. The weight for family $A = 1/4$, and the weight for family $C = 1/2$, therefore C is the closest family.

The value of the Query Element is inferred by finding the Simplicial Family to which the Query Simplex is closest, i.e., the family with the smallest distance, $dist_Q$. Should more than one family share the closest distance, $dist_Q$, the closest family is determined to be the one with the greatest weight, w_Q.

Appendix C. ACT-R

Note: Adaptive Control of Thought–Rational (ACT–R) is open-source, modular, written in Common Lisp (CL) and extendable[1]. The source code, documentation, tutorials and scientific papers based on ACT–R models, can be found at (Anderson et al., 2015).

C.1 A Cognitive Modeling Architectures (CMAs)

A *cognitive architecture* is a theory of the detailed processes of cognitive performance and learning. The primary purpose of a well-defined *cognitive architecture* is to provide a framework to facilitate research and understanding of components and processes of cognition. A secondary purpose is the advancement of the computational sciences by leveraging what is learned from research incorporating *cognitive architectures*. No *cognitive architecture* claims to provide a complete, or even nearly complete, model with the level of human intelligence, however, important design principles have emerged, pointing to promising models of generic and scalable architectures with close analogy to human cognitive processing. A cognitive architecture, that has been implemented in a computer-based modeling tool, is referred to as a Cognitive Modeling Architecture (CMA) (Chong et al., 2007; Hélie and Sun, 2014; Newell, 1990).

C.2 Overview of ACT-R

ACT–R is both a theory of human cognition, originally published by Anderson and Lebiere (1998), and a software toolset (a CMA) in which many aspects of the theory are modeled. The ACT–R toolset has been continuously updated and extended, as the field of cognitive modeling has matured, until present. The goal of the toolset is to provide a general theory of cognition, explain how information is encoded,

[1]Extendible ACT–R source code is only available for Windows operating systems.

retrieved, and processed and help researchers model, predict and explain human behavior (Banks, 2013). The title of the theory, *Adaptive Control of Thought–Rational (ACT–R)*, is interesting, as many would argue, and some cognitive researchers, in particular Stanovich (2009), have demonstrated that human behavior is often irrational, but that discussion is beyond the scope of this dissertation.

Banks, a long time user and contributor to ACT–R theory and toolset, provides a succinct description of ACT–R theory and features:

> According to ACT–R theory, cognition emerges through the interaction of several independent modules in the brain. These modules have specialized functions. The declarative memory module governs the storage and retrieval of declarative information. The problem state (or imaginal) module is responsible for the storage of intermediary mental representations that are used during thinking. The control (or goal) module contains information about the current goal when completing a task. The procedural module controls how each of these modules interacts.
>
> Procedural memory controls the interaction of modules through a series of if-then production rules that comprise procedural memory. The conditions of each rule are matched against the modules, and if one is met, then it fires and executes its actions. Typically, these actions involve the further encoding, retrieval, or processing of information and so the cognitive process is advanced. The cycle then continues and another production rule (or possibly the same one) will match and fire and so on, until the process completes.
>
> The conditions of each production rule are matched against various module buffers. Buffers provide an interface to each of the modules by holding chunks that are the output of the current processing of that module. A chunk is a symbolic representation of facts that have been encoded. Only one chunk can be held in a buffer at any one time. Depending on the function of the module, different types of chunk will be placed in these buffers. For example, the chunk in the declarative memory buffer is the last item retrieved from declarative memory; the chunk in the problem state buffer is the intermediary representation that is being used in the current task. Hence, production rules typically match a pattern of the buffer contents from several modules and determine the action that should be taken given that situation, moving it along a step. Different patterns will trigger different rules and therefore different actions. (Banks, 2013)

ACT–R does not provide an organic feature to represent, i.e., abstract, aggregate, blended knowledge, or the *maximally integrated information structures* of qualia space (QS) as discussed in Section 2.3.

C.3 Declarative Memory (DM)

In ACT–R instances of declarative memory (DM) are called **chunks**, which are programmatically defined as **chunk-types**, and **slots**. A **chunk-type** can be thought of as a category (e.g., Soybean-Sample) and **slots** as variables (e.g., disease, precipitation, temperature, hail, mold-growth, etc.).

A **chunk-type** is defined by the syntax: (chunk-type name slot-name-1 slot-name-2 . . . slot-name-n) Individual chunks are populated by declaring a chunk of a particular chunk-type with the ISA keyword, and filling the slots values by position. Empty slot values are supported with the *nil* Lisp symbol (Bothell, 2015; Anderson et al., 2012). The following example defines two memory chunks:
(chunk-type Soybean-Smpl disease precip temp hail mold-growth)
(sample001 ISA Soybean-Smpl brown-stem-rot nil gt-normal yes present)
(sample002 ISA Soybean-Smpl cyst-nema gt-normal lt-normal no absent)

C.4 Procedural Memory (PM)

Procedural Memory (PM) is modeled programmatically with condition-action pairs, called **productions**. Productions respond to the patterns in the DM chunks as contingency, i.e., if-then, rules which control behavior of the model. Pattern-matching in ACT–R can be either complete or partial (Bothell, 2015; Anderson et al., 2012). For the purposes of this dissertation, productions implement complete pattern-matching.

A limitation of the toolset is that productions are somewhat inflexible. Productions do not accept variable parameters, therefore chunk-types, the number of slots, and slot names (sample variables) have to be hard-coded into the model.

C.5 Overcoming PM Limitations

In order to make the Qualia Modeling Agent (QMA) generalizable, and to avoid hand-writing the productions for each new data set, code was written that dynamically generates the productions, based on the variables of the data set.

C.6 Activation Level

Cognitively, an activation level is "A state of memory traces that determines both the speed and the probability of access to a memory trace (Anderson, 2005, 183)". The following discussion describes how activation levels are implemented in ACT–R.

> Every chunk in ACT-R's declarative memory has associated with it a numerical value called its activation. The activation reflects the degree to which past experiences and current context indicate that chunk will be useful at any particular moment. When a retrieval request is made the chunk with the greatest activation among those that match the specification of the request will be the one placed into the retrieval buffer. There is one constraint on that however. There is a parameter called the retrieval threshold which sets the minimum activation a chunk can have and still be retrieved. It is set with the :rt parameter: (sgp :rt -0.5) [in the QMA :rt is set to the default, 0.0]
> If the chunk with the highest activation among those that match the request has an activation which is less than the retrieval threshold, then no chunk will be placed into the retrieval buffer and an error state will indicate the failure. The activation A_i of a chunk i is computed from three components the base-level, a context component and a noise component. We will discuss the context component in the next unit. So, for now the activation equation is:
>
> $$A_i = B_i + \epsilon \tag{21}$$
>
> B_i: The base-level activation. This reflects the recency and frequency of practice of the chunk i.
> ϵ: The noise value. The noise is composed of two components: a permanent noise associated with each chunk and an instantaneous noise computed at the time of a retrieval request.
>
> **Base-level Learning**
> The equation describing learning of base-level activation for a chunk i is:

$$B_i = \ln\left(\sum_{j=1}^{n} t_j^{-d}\right) \qquad (22)$$

n: The number of presentations for chunk i.
t_j: The time since the jth presentation.
d: The decay parameter which is set using the :bll (base-level learning) parameter. This parameter is almost always set to 0.5. [in the QMA :bll is set to 0.5]

This equation describes a process in which each time an item is presented there is an increase in its base-level activation, which decays away as a power function of the time since that presentation. These decay effects are summed and then passed through a logarithmic transformation. There are two types of events that are considered as presentations of a chunk. The first is its initial entry into declarative memory. The other is any merging of that chunk with a chunk that is already in declarative memory. (Anderson et al., 2012) (Unit #4).

Appendix D. Machine Learning (ML) Concepts

D.1 Nonparametric Classifiers

Like k-nearest neighbors (kNN) and decision tree (DT), the qualia space (QS) algorithm is nonparametric, meaning that it makes no assumptions about the training data distribution, all training samples are maintained for the test phase, and classification decisions are based on the entire training set. The training cost is very low for nonparametric classifiers, but the testing phase time and memory cost can be very high (Thirumuruganathan, 2010). For kNN and QS test cost includes comparison to all data points.

The QS hypernetwork algorithm can be considered most analogous to a 1-nearest neighbor classification algorithm with the differences:

1. a modified distance measure,
2. a weighting function,
3. test samples are added to the training data set,
4. the target variable (class) can change from test sample to test sample.

D.2 k-Fold Cross Validation

With k-fold cross validation each sample serves multiple duties, as training, validation, and test, therefore resulting in a more accurate result than training/validation/test methods that do not use all samples in all roles. Popular rule-of-thumb values for k in k-fold cross validation are 5 and 10 (Russell and Norvig, 2009).

The lower k-value (2-5) is less computationally expensive, as well as less time consuming for result evaluation and reporting, variance is increased, and bias decreased. Larger k-values (10-20) are more computationally expensive, as well as more time consuming for result evaluation and reporting, reduces the variance, while increasing the bias. Variance can, however, be reduced, without increasing bias, by re-randomizing the data and repeating cross-validation with the same k (Kohavi et al., 1995).

Due to the manually expensive process of running the various models, compiling and reporting the results, a moderate k value of 5 was selected for this research. In order to reduce variance, while not increasing bias, the entire process was repeated with the data set re-randomized.

D.3 Computation Complexity Order Of Magnitude (OOM)

The computational complexity Order Of Magnitude (OOM) of kNN, DT and Qualia Modeling Agent (QMA), summerized in Table 13, are evaluated with the same 5-fold cross-validation technique used to calculate the proportion of accurately classified test samples: three folds are used to train, one to validate and one to test. For the following discussion and equations:

$$\begin{aligned} a &= \text{number of features (variables)} \\ m &= \text{sample size} \\ Tr &= \text{training set}, (3/5)m \\ V &= \text{validation set}, (1/5)m \\ Te &= \text{test set}, (1/5)m \end{aligned} \tag{23}$$

D.3.1 K-Nearest Neighbors (kNN) Computation Complexity OOM.

In order to find the closest neighbor(s) to one test sample, the distances between the test sample and each of the samples in the training set, Tr, must be calculated. Therefore the OOM for one test sample is $O(aTr)$ (Thirumuruganathan, 2010). This calculation, however, does not provide any insight into the best choice of k for the particular data set. In order to select the best choice of k, 5-fold cross validation was applied, using the validation fold to select the best choice of k to apply to the test fold.

In the kNN algorithm, the training step includes converting the categorical data from a positional parameter list into a ordered binary representation, at a cost of $O(aTr)$. Additionally, in the 5-fold cross-validation technique the kNN algorithm training cost also includes the validation step.

In the validation step all validation samples, V, are compared to all training samples, Tr, to determines the choice for k, which will be used for the test queries.

The cost to train (i.e., determine the best choice for k) is evaluated as follows:

1. Converting the categorical data from a positional parameter list into a ordered binary representation, at a cost of $O(aTr)$.
2. The cost for one validation sample is $O(aTr)$ (Thirumuruganathan, 2010).
3. All validation samples are compared to all training samples, and a tally is kept of the distances, therefore kNN training cost is:

$$\begin{aligned} O(aTr) + O(aV(Tr)) &\Rightarrow O(a(3/5)m) + O(a(1/5)m(3/5)m) \\ &\Rightarrow O(a(3/5)m) + O(a(3/25)m^2). \end{aligned} \tag{24}$$

4. Each validation sample is evaluated at each level, $k = 1$ to x , looking at corresponding training labels and assigning the best guess of the k closest, based on majority vote.
5. At each k level the number of misclassified labels (training errors) are summed.

At the end of the training (validation) process the (smallest) k, with the least misclassification errors, is selected for the testing phase.

The cost to test is evaluated as follows:

1. Given one test sample, it is compared to all training samples, and a tally is kept of the distances, therefore the kNN query time for one test sample is:

$$O(a(Tr)) \Rightarrow O(a(3/5)m). \tag{25}$$

2. The k nearest neighbors are polled and the label selected by majority vote.
3. The kNN query time for all test samples is:

$$O(a(3/5)m(1/5)m) \Rightarrow O(a(3/25)m^2) \tag{26}$$

The total cost of kNN training and testing is:

$$O(a(3/25)m^2) + O(a(3/25)m^2) = O(a(3/5)m) + O(a(6/25)m^2) \tag{27}$$

D.3.2 Decision Tree (DT) Computation Complexity OOM.

(Pedregosa et al., 2011) The training cost of the DT algorithm is in building the tree, and the OOM is highly dependent on the specific balance of features of the data set, but in general, the cost in constructing a balanced tree is:[1]

$$O(n_{samples} n_{features} Log(n_{samples})) \tag{28}$$

and DT query time for one test sample is:

$$O(Log(n_{samples})). \tag{29}$$

Assuming the subtrees remain approximately balanced, the cost at each node consists of searching through $O(n_{features})$, to find the feature resulting in the largest reduction

[1]Log refers to natural, base-e, logarithm in these calculations.

in entropy. This has a cumulative cost, at each node, of:

$$O(n_{samples}n_{features}Log(n_{samples})) \tag{30}$$

resulting in a total cost (summing the cost at each node) of

$$O(n_{samples}^2 n_{features}Log(n_{samples})). \tag{31}$$

A more efficient method would be to pre-sort features over all samples, and retaining a running label count, reduces the complexity at each node to:

$$O(n_{features}Log(n_{samples})) \tag{32}$$

resulting in a total DT training cost of:

$$O(n_{samples}n_{features}Log(n_{samples})) \tag{33}$$

Given the more efficient method and 5-fold cross-validation, the total cost to train a balanced tree is:

$$O(a(Tr)Log(Tr)) \Rightarrow O(a(3/5)mLog(3/5m)), \tag{34}$$

the total cost to query all test samples is:

$$O(Log(Tr)Te) \Rightarrow O(Log(3/5m)(1/5m)). \tag{35}$$

The total cost of DT training and testing, is:

$$O(a(3/5)mLog(3/5m)) + (Log(3/5m)(1/5m)) = O([a(3/5) + (1/5)]mLog(3/5m)). \tag{36}$$

D.3.3 Qualia Space (QS) Computation Complexity OOM.

The OOM computed here is taking the QS hypernetwork algorithm in isolation, not considering the processing overhead of situating the algorithm in the QMA.

In the QS algorithm, the training step is converting the categorical data from a positional parameter list into a ordered binary representation. The cost to convert and store the training set in QS, i.e., the training cost is $O(aTr)$.

There is no validation step in QS, therefore the validation (V) samples are dismissed.

The cost to test is evaluated as follows:

1. Given one test sample, it is compared to all training samples, Tr, using the distance measure, d_Q, which requires two measures of eccentricity, $ecc(\sigma|\sigma')$ and $ecc(\sigma'|\sigma)$, therefore, the cost to test one sample in QS is:

$$O(2a(Tr)) \Rightarrow O(2a(3/5)m) \Rightarrow O(a(6/5)m). \tag{37}$$

2. The QS query time for all test samples is:

$$O(2a(Tr)(Te)) \Rightarrow O(2a(3/5)m(1/5)m) \Rightarrow O(a(6/25)m^2) \tag{38}$$

The total cost of QS training and testing is:

$$O(aTr) + O(2a(Tr)(Te)) \Rightarrow O((a(3/5)m + a(6/25)m^2) \tag{39}$$

D.4 Incremental vs Batch Machine Learning

The difference between incremental learning and batch machine learning (ML) algorithms is that incremental learning does not require the training to be complete before the testing (querying) process begins, but the test samples continue to be added to the training model after testing has begun. In batch learning all training samples are available and incorporated into the model before testing begins. A more rigorous definition of incremental learning is:

Definition D.1 (Incremental learning)**.** A sequence of instances [observations] is observed, one instance at a time, not necessarily in equally spaced time intervals.

Let $\boldsymbol{X}_t \in \mathcal{R}^p$ [be] a vector in $p-$dimensional feature space observed at time t and y_t is the corresponding label. We call $\boldsymbol{X}_t$ *an instance* and a pair $(\boldsymbol{X}_t, \boldsymbol{y}_t)$ a *labeled instance.* We refer to instances $(\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t)$ as *historical [or training] data* and instance $\boldsymbol{X}_{t+1}$ as *target* (or testing) instance.

At every time step t we have historical data (labeled) available $\boldsymbol{X}^{\boldsymbol{H}} = (\boldsymbol{X}_1, \ldots, \boldsymbol{X}_t)$. A target instance $\boldsymbol{X}_{t+1}$ arrives. The task is to predict a label $\boldsymbol{y}_{t+1}$. For that we build a learner $\mathcal{L}_t$, using all or a *selection* from the available historical data $\boldsymbol{X}^{\boldsymbol{H}}$. We apply the learner $\mathcal{L}_t$ to predict the label for $\boldsymbol{X}_{t+1}$).

At the next step after the classification or prediction decision is casted, the label $\boldsymbol{y}_{t+1}$ becomes available. How the instance $\boldsymbol{X}_{t+1}$ with a label is a part of historical data. The next testing instance $\boldsymbol{X}_{t+2}$ is observed. (Žliobaitė, 2010)

D.4.1 Instance Memory vs. Conceptual Memory.

(Maloof and Michalski, 2004) Incremental ML algorithms can have two types of memory, instance or concept, which are analogous to, and modeled after, human episodic and conceptual memory (knowledge), respectively:

Instance: a memory for representing each individual instance, or episode;

Concept: a memory for representing aggregate information compiled into a concept.

In the QMA all sensory data are stored in both QS and in autonomous Adaptive Control of Thought–Rational (ACT–R) declarative memory (DM), however, due to activation levels, older or unaccessed instances in autonomous DM fade over time.

The goal of the QS model is to represent conceptual knowledge, abstracted from aggregate sensory data to general categories of the properties of that class of experience.

D.4.2 Incremental Learning with Concept Drift.

A phenomena, particular to incremental learning, is *concept drift*, which refers to a learning problem that changes over time. In particular, the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways (Žliobaitė, 2010).

Definition D.2 (Concept Drift)**.** Every instance $\boldsymbol{X}_t$ is generated by a source S_t. If all the data is sampled from the same source, i.e., $S_1 = S_2 = \ldots = S_{t+1} = \boldsymbol{S}$ we say the *concept* is stable. If for any two points in time i and j $S_i \neq S_j$, we say that there is a *concept drift.*

The core assumption when dealing with the concept drift problem is uncertainty about the future. We assume that the source of the target instance $\boldsymbol{X}_{t+1}$ is not known with certainty. It can be assumed, estimated or predicted but there is no certainty. (Žliobaitė, 2010)

Classification problem. A classification problem, independent of the presence or absence of concept drift, may be described as follows.

Let $\boldsymbol{X}_t \in \mathcal{R}^p$ [be] an instance in $p-$dimensional feature space. $\boldsymbol{X} \in c_i$, where $c_1, c_2, \dots c_k$ is the set of class labels. The optimal classifier to classify $X \rightarrow c_i$ is completely determined by prior probabilities for the classes $P(c_i)$ and the class-conditional probability density function (pdf) $p(\boldsymbol{X}|c_i), i = 1, \dots, k$. (Narasimhamurthy and Kuncheva, 2007)

Data source. We define a set of prior probabilities of the classes and class-conditional pdfs as concept or *data source*:

$$\boldsymbol{S} = \{(P(c_1), p(\boldsymbol{X}|c_1)), (P(c_2), p(\boldsymbol{X}|c_2)), \dots, (P(c_k), p(\boldsymbol{X}|c_k))\} \tag{40}$$

When referring to a particular source at time t we will use the term *source*, while when referring to a fixed set of prior probability and the classes and class-conditional pdfs we will use the term *concept* and denote it with $\boldsymbol{S}$. (Narasimhamurthy and Kuncheva, 2007)

D.5 Ordering Effect and Convergence

An ordering effect, in incremental learning, occurs when the test samples, introduced to the algorithm in a different order, produce different results. To demonstrate no ordering effect, using n test sample, the algorithm must produce the same results at the end of the testing phase for the $n!$ possible orders (Almaksour, 2011; Cornuéjols, 1993).

Convergence, in ML is when the method, or algorithm, converges toward an optimal hypothesis, or estimation. Convergence is often achieved by increasing the number of training samples, if available (Mitchell, 1997).

D.6 Bayes Error

Definition D.6.1 (Bayes probability of error, B, for a multiclass classifier with discrete features)**.**

$$B = 1 - P(\omega_j|x), \tag{41}$$

where

$$P(\omega_j|x) = \frac{P(x|\omega_j)P(\omega_j)}{P(x)}, \tag{42}$$

and

$$P(x) = \sum_{j=1}^{c} P(x|\omega_j)P(\omega_j) \tag{43}$$

where $c =$ number of classes, and $P(x|\omega_j)$ is the conditional probability that x equal the state of nature ω_j (Duda et al., 2001, 36).

D.7 Supervised vs. Unsupervised Machine Learning

"Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class **labels** in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown (Kotsiantis et al., 2007)"

The defining difference between supervised and unsupervised learning is the existence of *labels* which presumably accurately classify the training set samples.

In unsupervised learning the agent learns patterns, even though no explicit labels are provided, detecting potentially useful clusters of samples (Russell and Norvig, 2009).

D.8 Transfer Learning (TL)

(Pan and Yang, 2010) In the research field of ML, *Transfer Learning (TL)* is the ability of a system to apply knowledge or skills learned in previous tasks to subsequent tasks or new domains, which are similar in some way. Pan and Yang's formal definitions of a domain, a task and TL, as applied to the domain and task, are:

Definition D.8.1 (Domain)**.** A domain, $\mathcal{D}$, consists of two components: a feature space, $\mathcal{X}$, and a marginal probability distribution (mpd), $P(X)$, where $X = \{x_1, \ldots, x_n\} \in \mathcal{X}$.

Definition D.8.2 (Task)**.** Given a specific domain, $\mathcal{D} = \{\mathcal{X}, P(X)\}$, a *task* consists of two components: a label space $\mathcal{Y}$ and an objective predictive function $f(\cdot)$ (denoted by $\mathcal{T} = \{\mathcal{Y}, f(\cdot)\}$), which is not observed but can be learned from the training data, which consists of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \mathcal{Y}$. The function $f(\cdot)$ can be used to predict the corresponding label, $f(x)$, of a new instance x. From a probabilistic viewpoint, $f(x)$ can be written as $P(y|x)$.

Definition D.8.3 (Transfer Learning (TL))**.** Given a source domain, $\mathcal{D}_S$, and learning task, $\mathcal{T}_S$, a target domain, $\mathcal{D}_T$ and learning task, $\mathcal{T}_T$, *Transfer Learning (TL)* aims to help improve the learning of the target predictive function $f_T(\cdot)$ in $\mathcal{D}_T$ using the knowledge in $\mathcal{D}_S$ and $\mathcal{T}_S$, where $\mathcal{D}_S \neq \mathcal{D}_T$, or $\mathcal{T}_S \neq \mathcal{T}_T$.

Given these definitions, TL is realized under the following conditions:

- A domain is a pair $\mathcal{D} = \{\mathcal{X}, P(X)\}$. Thus, the condition $\mathcal{D}_S \neq \mathcal{D}_T$ implies that either $\mathcal{X}_S \neq \mathcal{X}_T$ or $P_S(X) \neq P_T(X)$.
- A task is defined as a pair $\mathcal{T} = \{\mathcal{Y}, P(Y|X)\}$. Thus, the condition $\mathcal{T}_S \neq \mathcal{T}_T$ implies that either $\mathcal{Y}_S \neq \mathcal{Y}_T$ or $P(Y_S|X_S) \neq P(Y_T|X_T)$.
- When the domains are different, then either (1) the feature spaces between the domains are different, i.e., $\mathcal{X}_S \neq \mathcal{X}_T$, or (2) the feature spaces between the domains are the same but the marginal probability distributions between domain data are different; i.e., $P(X_S) \neq$

> $P(X_T)$, where $X_{S_i} \in \mathcal{X}_S$ and $X_{T_i} \in \mathcal{X}_T$.
>
> When the target and source domains are the same, i.e., $\mathcal{D}_S = \mathcal{D}_T$, and their learning tasks are the same, i.e., $\mathcal{T}_S = \mathcal{T}_T$, the learning problem becomes a traditional machine learning problem (Pan and Yang, 2010).

Pan and Yang categorize various ML methods based on the relationships between source and target domains and tasks, as well as, whether or not the data are labeled, as illustrated in Table 15.

Table 15: Relationship between Traditional Machine Learning (ML) and Various Machine Learning (ML) Methods (Pan and Yang, 2010).

Learning Method	**Source & Target Domains**	**Source & Target Tasks**	**Source Data Labeled**	**Target Data Labeled**
Traditional ML	the same	the same	both labeled and unlabeled	both labeled and unlabeled
Inductive[2] TL (case I)	the same	different but related	labeled	some
Inductive TL (case II)	the same	different but related	none	some
Unsupervised TL	different but related	different but related	none	none
Transductive[3] TL	different but related	the same	labeled	none

[2]see Inductive reasoning in Appendix A.
[3]see Transductive reasoning in Appendix A.

Appendix E. Distance Measures for Binary Data

Measuring distance, or the inverse, similarity, between two observations, captured as entities in a data set, is a core requirement for this research. Widely used distance measures for continuous and ordinal data are the *Manhattan Distance* and *Euclidean Distance* measures, which are specific cases of the more general *Minkowski Distance* (Gentle, 2009). However, the data sets to which the qualia space (QS) algorithm apply, are exclusively *categorical* data (also known as nominal or qualitative multi-state data). The notion of a distance measure is not as straightforward for categorical data as for continuous data values, because there is no natural ordering or unit of measurement (Boriah et al., 2008).

One contribution of this research is a novel distance measure for categorical data, d_Q, developed from hypernetwork theory, using measures of eccentricity. This section will compare the d_Q distance measure to some common methods for measuring similarity and distances between categorical data points, specifically the Simple Matching, Euclidean, and Hamming Distance measures.

E.1 Binarizing Categorical Data

The similarity and distance measures described in this research require that the categorical attributes be converted to binary values, also known as binarizing.

A commonly used method, explained by Lourenco et al. (2004), published originally by Bação (2002), is to create one binary variable for each categorical attribute. This is explained here with a simple example. Take the case of a categorical object with two variables $x = (x_1, x_2)$. The first variable having three possible values (attributes), e.g. $x_1 \in \{A, B, C\}$, and the second variable having two possible values (attributes), e.g., $x_2 \in \{D, E\}$. The resulting binary object will be:

$$x' = (x'_1, x'_2, x'_3, x'_4, x'_5) \tag{44}$$

where

$$x'_1 = \begin{cases} 1, \text{if } (x_1 = A) \\ 0, \text{if } (x_1 \neq A) \end{cases} \quad \text{and, } x'_2 = \begin{cases} 1, \text{if } (x_2 = B) \\ 0, \text{if } (x_2 \neq B) \end{cases} \quad \text{and, } x'_3 = \begin{cases} 1, \text{if } (x_3 = C) \\ 0, \text{if } (x_3 \neq C) \end{cases}$$

and,

$$x'_4 = \begin{cases} 1, \text{if } (x_4 = D) \\ 0, \text{if } (x_4 \neq D) \end{cases} \quad \text{and, } x'_5 = \begin{cases} 1, \text{if } (x_5 = E) \\ 0, \text{if } (x_5 \neq E) \end{cases}$$

Given this binary arrangement of the data the features of two objects with the same number of ordered elements can be compared, $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, where:

a = number of times $x_i = 1$ and $y_i = 1$

b = number of times $x_i = 0$ and $y_i = 1$

c = number of times $x_i = 1$ and $y_i = 0$

d = number of times $x_i = 0$ and $y_i = 0$

Given the sums a through d, and Table 16, distance measures for binarized data can be easily calculated and compared.

Table 16: Contingency Table Values for Binary Similarity and Distance Measures (Lourenco et al., 2004). Also referred to as the Operational Taxonomic Unit (OTU) expressions for binary instances (Choi et al., 2010).

		Object x		
		1	0	sum
Object y	1	a	b	a+b
	0	c	d	c + d
	sum	a + c	b + d	n = a + b + c + d

E.2 (Dis)Similarity measures

E.2.1 Simple Matching Similarity.

The simplest similarity measure is the Simple Matching, which is derived from the simple matching coefficient $|A \cap B|$, i.e., the number of shared, ordered, elements (van Rijsbergen, 1979). Formally, the Simple Matching similarity measure is the cardinality of the intersection of two ordered sets, A and B (Lourenco et al., 2004):

$$S_M(A, B) = n|A \cap B| \tag{45}$$

Equation (45) does not take into consideration the length of the ordered sets.

Simple Matching *similarity* measure for binarized data, s_{SM}, does take into consideration cardinality and the negative co-occurrence (value d in Table 16), and can be written as (Lourenco et al., 2004):

$$s_{SM} = \frac{a + d}{a + b + c + d}, \text{ Range: } [0, 1] \tag{46}$$

For the sake of consistency, the inverse, Simple Matching *distance* measure, d_{SM} can be written as:

$$d_{SM} = 1 - \left(\frac{a+d}{a+b+c+d} \right), \text{ Range: } [0,1] \tag{47}$$

E.2.2 Euclidean Distance.

(Deza and Deza, 2009) The Euclidean distance is the length of a straight-line between two points in Euclidean space. In Cartesian coordinates, if $\mathbf{x} = (x_1, x_2, ..., x_n)$ and $\mathbf{y} = (y_1, y_2, ..., y_n)$ are two points in Euclidean n-space, then the distance between $\mathbf{x}$ and $\mathbf{y}$ is calculated with the Pythagorean formula:

$$\begin{aligned} d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) &= \sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2 + \cdots + (y_n - x_n)^2} \\ &= \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2} \end{aligned} \tag{48}$$

When the Euclidean distance is applied to binarized Cartesian coordinates, where all $x_i, y_i \in \{0, 1\}$, then the Euclidean distance equation for binarized data, d_E, expressed in OTUs, can be written as (Choi et al., 2010):

$$d_E = \sqrt{b+c}, \text{ Range: } [0, \infty) \tag{49}$$

E.2.3 Hamming Distance.

The Hamming distance was originally defined for binary codes, as the number of bits that are different in two binary vectors. However, the Hamming distance can be applied to any ordered set of equal length. This distance measure being the sum of the total mismatches of the corresponding attributes of two objects (Gentle, 2009; Lourenco et al., 2004). Formally, (Lourenco et al., 2004):

$$d_H(x, y) = \sum_{i=1}^{n} \delta(x_i, y_i) \tag{50}$$

where

$$\delta(x_i, y_i) = \begin{cases} 0, \text{if } (x_i = y_i) \\ 1, \text{if } (x_i \neq y_i) \end{cases}$$

The Hamming distance function for binarized data, d_H, expressed in OTUs, can be written as (Lourenco et al., 2004):

$$d_H = b + c, \text{ Range: } [0, \infty) \tag{51}$$

Another popular distance measure, the Manhattan, or City Block, distance, d_M, is equal to the Hamming distance when applied to binarized data (Choi et al., 2010).

$$d_M = b + c, \text{ Range: } [0, \infty) \tag{52}$$

E.2.4 Qualia Space Distance.

The QS distance, d_Q, defined in Equation (18), expressed in OTUs, can be written as:

$$d_Q = \max\left(\left(\frac{c}{a+c}\right), \left(\frac{b}{a+b}\right)\right), \text{ Range: } [0, 1] \tag{53}$$

E.2.5 Comparing Standard Distance Measures.

Table 17 will be used to compare the distance measures discussed.

Table 17: A simplicial complex, which will be used to compare the different distance measures discussed.

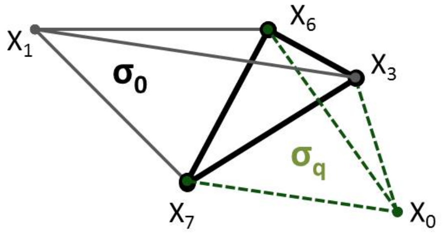
	Attribute Set										
	disease	hail		seed -discolor		precipitation			leaf -mildew		
	Simplex	present	absent	present	absent	lt normal	normal	gt normal	absent	upper-surface	lower-surface
ID	Label	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9
σ_0	bact.-blight	0	1	0	1	0	0	1	1	0	0
σ_1	bact.-blight	0	1	1	0	0	0	1	1	0	0
σ_2	cyst-nem.	1	0	1	0	1	0	0	0	0	1
σ_3	cyst-nem.	1	0	1	0	0	0	0	1	0	0
σ_4	herb.-injury	1	0	0	1	0	0	1	0	1	0
σ_q	$\langle query \rangle$	1	0	0	1	0	0	1	1	0	0

When categorical features are binarized, the Simple Matching (scaled), the Euclidean (squared), the Hamming and Manhattan distances provide the same results in terms of boundary delineation when applied to classification problems (Lourenco et al., 2004). However, the QS distance measure, d_Q, would result in a different boundary delineation, as illustrated in Table 18.

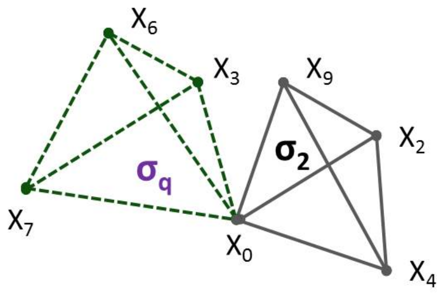
Table 18: Distance measures between each of σ_0 through σ_4, and the query, σ_q from Table 17. Distance measures are: Simple Matching (scaled), d_{SM}; Euclidean (squared), d_E; Hamming, d_H; and Qualia Space, d_Q;.

	Attribute Set										Distance (σ_i, σ_q)			
ID	X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	d_{SM}	d_E	d_H	d_Q
σ_0	0	1	0	1	0	0	1	1	0	0	2/10	$\sqrt{2}$	2	1/4
σ_1	0	1	1	0	0	0	1	1	0	0	4/10	$\sqrt{4}$	4	2/4
σ_2	1	0	1	0	1	0	0	0	0	1	6/10	$\sqrt{6}$	6	3/4
σ_3	1	0	1	0	0	0	0	1	0	0	3/10	$\sqrt{3}$	3	2/4
σ_4	1	0	0	1	0	0	1	0	1	0	2/10	$\sqrt{2}$	2	1/4
σ_q	1	0	0	1	0	0	1	1	0	0				

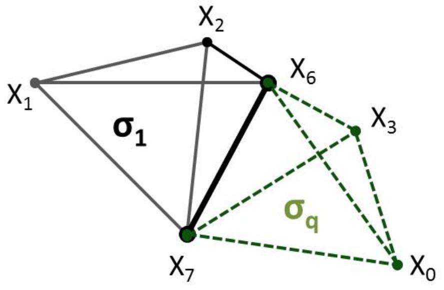
Figure 28 illustrates four of the d_Q distance measures from Table 18.



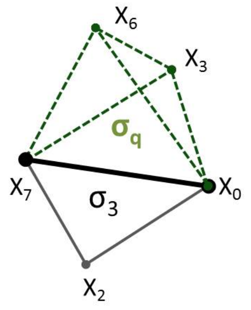
(a) Geometric relationship between σ_0 and σ_q, $d_Q(\sigma_0, \sigma_q) = 1/4$.



(b) Geometric relationship between σ_2 and σ_q, $d_Q(\sigma_2, \sigma_q) = 3/4$.



(c) Geometric relationship between σ_1 and σ_q, $d_Q(\sigma_1, \sigma_q) = 2/4$.



(d) Geometric relationship between σ_3 and σ_q, $d_Q(\sigma_3, \sigma_q) = 2/4$.

Figure 28: Four of the Distance Measures, d_Q, from Table 18, Represented Geometrically.

Benchmark example. Figure 29 is an artificially designed data set, widely used as a benchmark for distance and similarity measures in binary data sets, as referred to in (Lourenco et al., 2004), originally taken from (Ritter and Kohonen, 1989). In this original format missing elements cannot be captured in the purely binary variables, i.e., hooves, mane, and hunt. The other variables, such as, *Is small, medium, big*, i.e., *size*, are mutually exclusive and presumably exhaustive, therefore a missing element can be represented by 0 in each small, medium, big, for an animal.

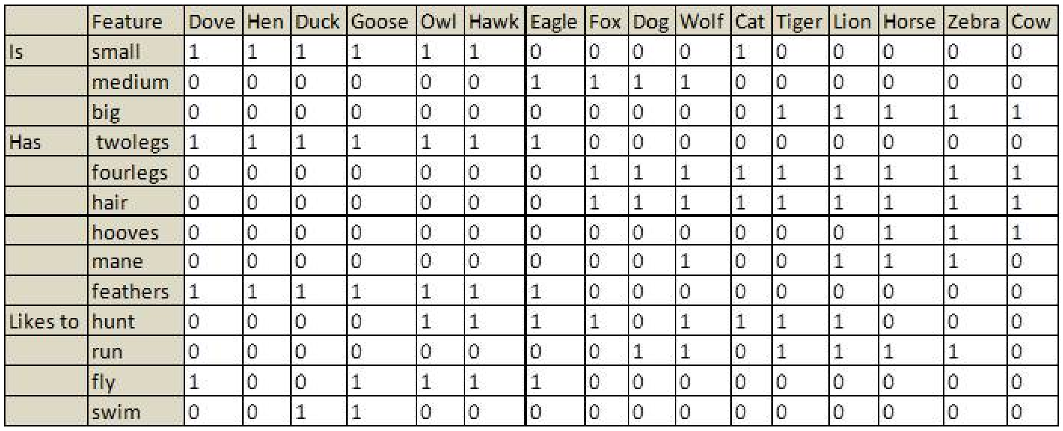
In the original benchmark data set, Figure 29, the following sample distances are:

d(Dove/Hen): $d_{SM} = 1/13$, $d_Q = 1/4$

d(Dove/Tiger): $d_{SM} = 9/13$, $d_Q = 5/5$

d(Dove/Cow): $d_{SM} = 8/13$, $d_Q = 4/4$

d(Cow/Tiger): $d_{SM} = 3/13$, $d_Q = 2/5$



	Feature	Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
Is	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Has	twolegs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	fourlegs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Likes to	hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 29: Original Benchmark for Distance and Similarity Measures in Binary Data Sets, as referred to in (Lourenco et al., 2004), Originally taken from (Ritter and Kohonen, 1989).

Figure 30 is a modified version of Figure 29, which has been expanded to support missing elements for any variable, and fits the hypernetwork theory formalism used in this research. However, there are no missing elements in this particular data set. Mutually exclusive attributes have been combined to create a variable, such as *Likes to run* and *Likes to fly*, were combined to create *Likes to travel by*, with three potential attributes: none, fly, run.

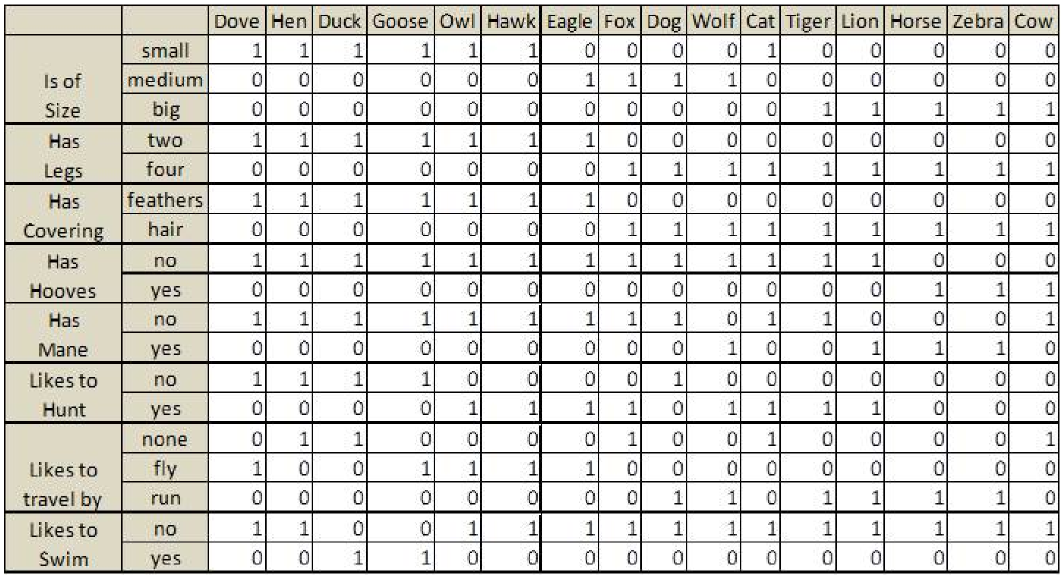
In the modified benchmark data set, Figure 30, the following sample distances differ from Simple Matching and the QS distance measures:

$$\text{d(Dove/Hen)}:\ d_{SM} = 2/18,\quad d_Q = 1/8$$

$$\text{d(Dove/Tiger)}:\ d_{SM} = 10/18,\ d_Q = 5/8$$

$$\text{d(Dove/Cow)}:\ d_{SM} = 11/18,\ d_Q = 5/8$$

$$\text{d(Cow/Tiger)}:\ d_{SM} = 5/18,\quad d_Q = 3/8$$



		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
Is of Size	small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Has Legs	two	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	four	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Has Covering	feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
Has Hooves	no	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
	yes	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
Has Mane	no	1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	1
	yes	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
Likes to Hunt	no	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0
	yes	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
Likes to travel by	none	0	1	1	0	0	0	0	1	0	0	1	0	0	0	0	1
	fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
Likes to Swim	no	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
	yes	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

Figure 30: Modified Benchmark for Distance and Similarity Measures in Binary Data Sets Which Allows for Missing Elements. This representation of the data allows for missing elements.

Performance and Selection Criteria for binary distance measures. While no distance measure is optimal for all types of problems there are some criteria for distance measure selection, for example: accuracy, outlier detection and consistency of performance (Boriah et al., 2008).

Appendix F. Binary Similarity and Dissimilarity Measures Table

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Anderberg (1973)	$\frac{8a}{8a+b+c}$	$[0,1]$
Anderberg (1973) D	$\frac{\sigma-\sigma'}{2n}$ where $\sigma = \max(a,b) + \max(c,d) + \max(a,c) + \max(b,d)$ and $\sigma' = \max(a+c, b+d) + \max(a+b, c+d)$	$[0,1)$
Baroni-Urbani and Buser (1976)-I	$\frac{\sqrt{ad}+a}{\sqrt{ad}+a+b+c}$	$[0,1]$
Baroni-Urbani and Buser (1976)-II	$\frac{\sqrt{ad}+a-(b+c)}{\sqrt{ad}+a+b+c}$	[-1, 1]
Batagelj and Bren (1995)	$\frac{bc}{ad}$	$[0,\infty)$
Benini (1901)	$\frac{a-(a+b)(a+c)}{a+\min(b,c)-(a+b)(a+c)}$	$[1,2]$
Braun-Blanquet et al. (1932)	$\frac{a}{\max(a+b,a+c)}$	[0, 1]
Browsing (Köppen, 1884)	$a - bc$	$(-\infty,\infty)$
Clement (1976)	$\frac{a(c+d)}{(a+b)} + \frac{d(a+b)}{(c+d)}$	$(0,\infty)$
Cohen's κ (Cohen, 1960)	$\frac{2(ad-bc)}{(a+b)(b+d)+(a+c)(c+d)}$	[-1/2, 1]
Cole (1949)-I	$\frac{ad-bc}{\min((a+b)(a+c),(b+d)(c+d))}$	$[-1,\infty)$
Cole (1949)-II	$\frac{ad-bc}{(a+b)(b+d)}$	$[-1,1]$
Cole (1949)-III	$\frac{ad-bc}{(a+c)(c+d)}$	$[-1,\infty)$
Cosine (Choi, 2008) Ochiai (1957)-I Otsuka [Ochiai paper] Driver and Kroeber (1932) Fowlkes and Mallows (1983) Gower and Legendre (1986)	$\frac{a}{\sqrt{(a+b)(a+c)}}$	$[0,1]$
d Specific Agreement (Fleiss et al., 2003)	$\frac{2d}{2a+b+c}$	$[0,\infty)$

Continued on next page

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Dennis Dennis (1965)	$\frac{ad-bc}{\sqrt{n(a+b)(a+c)}}$	$[-1,\infty)$
Dice (1945)-I Wallace (1983) Post and Snijders (1993)	$\frac{a}{a+b}$	$[0,1]$
Dice (1945)-II Wallace (1983) Post and Snijders (1993)	$\frac{a}{a+c}$	$[0,1]$
Digby (1983)	$\frac{(ad)^{\frac{3}{4}}-(bc)^{\frac{3}{4}}}{(ad)^{\frac{3}{4}}+(bc)^{\frac{3}{4}}}$	$[-1,1]$
Dispersion (Morris, 2012)	$\frac{ad-bc}{(a+b+c+d)^2}$	$[-1/3, 1/3]$
Doolittle (1885) Pearson (1926)	$\frac{(ad-bc)^2}{(a+b)(a+c)(c+d)(b+d)}$	$[0,1]$
Eyraud (1938)	$\frac{n^2(na-(a+b)(a+c))}{(a+b)(a+c)(b+d)(c+d)}$	$(-\infty,\infty)$
Fager and McGowan (1963)	$\frac{a}{\sqrt{(a+b)(a+c)}}-\frac{1}{2\sqrt{\max(a+b,a+c)}}$	$[-1/2,1)$
Faith et al. (1987)	$\frac{a+0.5d}{a+b+c+d}$	$[0,1]$
Fleiss (1975)	$\frac{(ad-bc)[(a+b)(b+d)+(a+c)(c+d)]}{2(a+b)(a+c)(b+d)(c+d)}$	$(-\infty,1]$
Forbes (1907)-I	$\frac{na}{(a+b)(a+c)}$	$[0,\infty)$
Fossum (1966) Jones and Curtice (1967)	$\frac{n(a-0.5)^2}{(a+b)(a+c)}$	$(0,\infty)$
Gilbert (1884) (Ratio of Success)	$\frac{a-\frac{(a+b)(a+c)}{n}}{a+b+c-\frac{(a+b)(a+c)}{n}}$	$[-1/3,1]$
Gilbert and Wells (1966)	$\log a-\log n-\log(\frac{a+b}{n})-\log\left(\frac{a+c}{n}\right)$	$[0,\infty)$
Gini (1912)	$\frac{a-(a+b)(a+c)}{\sqrt{\left(1-(a+b)^2\right)\left(1-(a+c)^2\right)}}$	$[-4/3,0]$

Continued on next page

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Gleason (1920) Dice (1945) Sørensen (1948) (Coincidence Index) (Quotient Similarity) Czekanowski (1932) Nei and Li (1979) (Genetic Coefficient) Gower and Legendre (1986) VII	$\frac{2a}{2a+b+c}$	$[0,1]$
Goodman and Kruskal (1954) Max	$\frac{a+d-\max(a,d)-\frac{b+c}{2}}{1-\max(a,d)-\frac{b+c}{2}}$	[-1, 1]
Goodman and Kruskal (1954) Min	$\frac{2\min(a,d)-b-c}{2\min(a,d)+b+c}$	$[-1,1]$
Goodman and Kruskal (1954) Probability	$\frac{\max(a,c)+\max(b,d)-\max(a+b,c+d)}{1-\max(a+b,c+d)}$	[-1/3, 0]
Goodman and Kruskal (1954) Lambda	$\frac{\max(a,b)+\max(c,d)+\max(a,c)+\max(b,d)-\max(a+c,b+d)-\max(a+b,c+d)}{2-\max(a+c,b+d)-\max(a+b,c+d)}$	$[0,1]$
Goodman and Kruskal (1954) Tau	$\frac{\frac{(a-(a+b)(a+c))^2+(b-(a+b)(b+d))^2}{(a+b)}+\frac{(c-(a+c)(c+d))^2+(d-(b+d)(c+d))^2}{(c+d)}}{1-(a+c)^2-(b+d)^2}$	(-∞, -2]
Gower (1971)	$\frac{a+d}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$	$[0,1.5]$
Hamann (1961) Holley and Guilford (1964) Hubert (1977) Gower and Legendre (1986) IX	$\frac{(a+d)-(b+c)}{a+b+c+d}$	[-1, 1]
Harris and Lahey (1978)	$\frac{a((c+d)+(b+d))}{2(a+b+c)}+\frac{d((a+b)+(a+c))}{2(b+c+d)}$	$[0,\infty)$
Hawkins and Dotson (1973)	$\frac{1}{2}\left(\frac{a}{a+b+c}+\frac{d}{b+c+d}\right)$	$[0,1]$
Inner Product (Hamming (1950) Complement)	$a+d$	$[0,\infty)$
Intersection	a	$[0,\infty)$

Continued on next page

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Jaccard (1912) Gilbert (1884) (Ratio of Verification) Tanimoto (1957) (Cosine Coefficient) Gower and Legendre (1986) III	$\frac{a}{a+b+c}$	$[0,1]$
Jaccard-3W	$\frac{3a}{3a+b+c}$	$[0,1]$
Johnson (1967)	$\frac{a}{a+b}+\frac{a}{a+c}$	$[0,2]$
Kent and Foster (1977)-I	$\frac{-bc}{b(a+b)+c(a+c)+bc}$	[-1/3, 0]
Kent and Foster (1977)-II	$\frac{-bc}{b(c+d)+c(b+d)+bc}$	[-1/3, 0]
Köppen (1870)	$\frac{(a+b)(1-a-b)-c}{(a+b)(1-a-b)}$	$(-\infty,\infty)$
Köppen (1884)	$a+\frac{b+c}{2}$	$[0,\infty)$
Kuder and Richardson (1937) Cronbach (1951)	$\frac{4(ad-bc)}{(a+b)(c+d)+(a+c)(b+d)+2(ad-bc)}$	[-2, 1]
Kuhns (1965)	$\frac{2(ad-bc)}{n(2a+b+c)}$	[-1/2, 1]
Kuhns (1965) Proportion	$\frac{ad-bc}{n\left(1-\frac{a}{(a+b)(a+c)}\right)\left(2a+b+c-\frac{(a+b)(a+c)}{n}\right)}$	[-1/3, 1)
Kulczyński (1927)-I Gower and Legendre (1986) I	$\frac{a}{b+c}$	$[0,\infty)$
Kulczyński (1927)-II Driver and Kroeber (1932) Gower and Legendre (1986) X	$\frac{\frac{a}{2}(2a+b+c)}{(a+b)(a+c)}$	$[0,1]$
Loevinger (1947, 1948) H Forbes (1907)-II Mokken (1971) Sijtsma and Molenaar (2002)	$\frac{na-(a+b)(a+c)}{n\min(a+b,a+c)-(a+b)(a+c)}$	[-1, 1]
Maron and Kuhns (1960)	$\frac{ad-bc}{(a+b+c+d)}$	$(-\infty,\infty)$
Maxwell and Pilliner (1968)	$\frac{2(ad-bc)}{(a+b)(c+d)+(a+c)(b+d)}$	[-1, 1]
McConnaughey (1964)	$\frac{a^2-bc}{(a+b)(a+c)}$	[-1, 1]

Continued on next page

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Michael (1920)	$\frac{4(ad-bc)}{(a+d)^2+(b+c)^2}$	[-1, 1]
Gini (1912)	$\frac{a-(a+b)(a+c)}{1-\frac{\lvert b-c\rvert}{2}-(a+b)(a+c)}$	[0, 4/3]
Mountford (1962)	$\frac{a}{0.5(ab+ac)+bc}$	[0, 2]
Pearson and Heron (1913)-II	$\cos\left(\frac{\pi\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}\right)$	[-1, 1]
Pearson (1904)-I (Coefficient of Chi-square Contingency)	χ^2 where $\chi^2=\frac{n(ad-bc)^2}{(a+b)(a+c)(c+d)(b+d)}$	$[0,\infty)$
Pearson (1904)-II (Coefficient of Mean Square Contingency)	$\sqrt{\frac{\chi^2}{n+\chi^2}}$ where $\chi^2=\frac{n(ad-bc)^2}{(a+b)(a+c)(c+d)(b+d)}$	$[0,\sqrt{1/2})$
Pearson (1926)-III (Coefficient of Racial Likeness)	$\sqrt{\frac{\rho}{n+\rho}}$ where $\rho=\frac{ad-bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$	$[0,\sqrt{1/3})$
Peirce (1884)-I	$\frac{ad-bc}{(a+b)(c+d)}$	[-1, 1]
Peirce (1884)-II	$\frac{ad-bc}{(a+c)(b+d)}$	[-1, 1]
Peirce (1884)-III	$\frac{ab+bc}{ab+2bc+cd}$	[0, 1]
Phi Coefficient Yule (1912) Pearson and Heron (1913)-I (Fourfold point correlation) (binary version of Pearson (1904) Product Moment Correlation Coefficient) Gower and Legendre (1986) XIV	$\frac{ad-bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$	[-1, 1]
Relative Decrease of Error Probability	$\frac{\max(a,b)+\max(c,d)-\max(a+c,b+d)}{1-\max(a+c,b+d)}$	[-1, 0]
Rogers and Tanimoto (1960) Farkas (1978) Gower and Legendre (1986) VI	$\frac{a+d}{a+2(b+c)+d}$	[0, 1]

Continued on next page

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Rogot and Goldberg (1966)	$\frac{a}{(a+b)+(a+c)}+\frac{d}{(c+d)+(b+d)}$	$[0,1]$
Russel and Rao (1940) (dot product) (inner product) Gower and Legendre (1986) II	$\frac{a}{a+b+c+d}$	$[0,1]$
Scott (1955)	$\frac{4(ad-bc)-(b-c)^2}{(2a+b+c)(b+c+2d)}$	[-1, 1]
Simpson (1943) (Ecological Coexistence Coefficient)	$\frac{a}{\min(a+b,a+c)}$	$[0,1]$
Sokal and Michener (1958) (Simple Matching Coefficient) Rand (1971) Brennan and Light (1974) Gower and Legendre (1986) IV	$\frac{a+d}{a+b+c+d}$	$[0,1]$
Sokal and Sneath (1963)-I Gower and Legendre (1986) V	$\frac{a}{a+2b+2c}$	$[0,1]$
Sokal and Sneath (1963)-II Gower and Legendre (1986) VIII	$\frac{2(a+d)}{2a+b+c+2d}$	$[0,1]$
Sokal and Sneath (1963)-III	$\frac{a+d}{b+c}$	$[0,\infty)$
Sokal and Sneath (1963)-IV Gower and Legendre (1986) XI	$\frac{\frac{a}{(a+b)}+\frac{a}{(a+c)}+\frac{d}{(b+d)}+\frac{d}{(c+d)}}{4}$	$[0,1]$
Sokal and Sneath (1963)-V Ochiai (1957)-II Gower and Legendre (1986) XIII	$\frac{ad}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$	$[0,1]$
Sorgenfrei (1958) Cheetham and Hazel (1969) (Correlation Ratio)	$\frac{a^2}{(a+b)(a+c)}$	$[0,1]$
Stiles (1961)	$\log_{10}\frac{n\left(\lvert ad-bc\rvert-\frac{n}{2}\right)^2}{(a+b)(a+c)(b+d)(c+d)}$	$(-\infty,\infty)$
Stuart (1953) τ_c	$2\,(ad-bc)$	$(-\infty,\infty)$

Continued on next page

Table 19: Binary Similarity Measures (Morris, 2012).

Similarity Measures (alternative names)	**Equation**	**Range**
Tarantula (Jones et al., 2001) Ample (Dallmeier et al., 2005)	$\frac{\frac{a}{(a+b)}}{\frac{c}{(c+d)}} = \frac{a(c+d)}{c(a+b)}$	$[0, \infty)$
Tarwid (1960)	$\frac{na-(a+b)(a+c)}{na+(a+b)(a+c)}$	[-1, 1)
Tversky (1977) (Feature Contrast Model)	$a - b - c$	$(-\infty, \infty)$
Warrens (2008)-I	$\frac{2a-b-c}{2a+b+c}$	[-1, 1]
Warrens (2008)-II	$\frac{2d}{b+c+2d}$	[0, 1]
Warrens (2008)-III	$\frac{2d-b-c}{b+c+2d}$	[-1, 1]
Warrens (2008)-IV	$\frac{4ad}{4ad+(a+d)(b+c)}$	[0, 1]
Warrens (2008)-V	$\frac{ad-bc}{\min((a+b)(a+c),(c+d)(b+d))}$	$[-1, \infty)$
Yule (1900) Q (Coefficient of Association) Montgomery and Crittenden (1977) Gower and Legendre (1986) XV	$\frac{ad-bc}{ad+bc}$	[-1, 1]
Yule (1912) Y (Coefficient of Colligation)	$\frac{\sqrt{ad}-\sqrt{bc}}{\sqrt{ad}+\sqrt{bc}}$	[-1, 1]

Table 20: Binary Dissimilarity Measures (Morris, 2012).

Dissimilarity measures (alternative names)	**Equation**	**Range**
Chord (Orloci, 1967)	$\sqrt{2\left(1 - \frac{a}{\sqrt{(a+b)(a+c)}}\right)}$	$[0, \sqrt{2}]$
Euclidean (Pythagorean metric)	$\sqrt{b+c}$	$[0, \infty)$

Continued on next page

Table 20: Binary Dissimilarity Measures (Morris, 2012).

Dissimilarity measures (alternative names)	**Equation**	**Range**
Hamming (1950) Squared-Euclidean Canberra (Lance and Williams, 1966) Manhattan CityBlock Minkowski	$b+c$	$[0,\infty)$
Hellinger (Rao, 1995)	$2\sqrt{1-\frac{a}{\sqrt{(a+b)(a+c)}}}$	$[0,2]$
Lance and Williams (1967) Bray and Curtis (1957)	$\frac{b+c}{(2a+b+c)}$	$[0,1]$
Mean Manhattan	$\frac{b+c}{(a+b+c+d)}$	$[0,1]$
Pattern Difference	$\frac{4bc}{(a+b+c+d)^2}$	$[0,1]$
Shape Difference Baulieu (1989)	$\frac{n(b+c)-(b-c)^2}{(a+b+c+d)^2}$	$[0,1]$
Size Difference Baulieu (1989)	$\frac{(b-c)^2}{(a+b+c+d)^2}$	$[0,1]$
Variance	$\frac{(b+c)}{4(a+b+c+d)}$	$[0,0.25]$
Yule (1900) Q dissimilarity	$\frac{2bc}{ad+bc}$	[-1, 1]

Binary Measures References

Anderberg, M. R. (1973). *Cluster Analysis for Applications, Monographs and Textbooks on Probability and Mathematical Statistics.* New York: Academic Press, Inc.

Baroni-Urbani, C. and Buser, M. W. (1976). Similarity of Binary Data. *Systematic Biology*, 25(3):251–259.

Batagelj, V. and Bren, M. (1995). Comparing Resemblance Measures. *Journal of classification*, 12(1):73–90.

Baulieu, F. B. (1989). A Classification of Presence/Absence Based Dissimilarity Coefficients. *Journal of Classification*, 6:233–246.

Benini, R. (1901). Principii di Demografia. *Manuali Barbera di Science Giuridiche Sociali e Politiche, Firenzi*, 29.

Braun-Blanquet, J. et al. (1932). *Plant Sociology. The Study of Plant Communities.* McGraw-Hill Book Co., Inc., New York.

Bray, J. R. and Curtis, J. T. (1957). An Ordination of the Upland Forest of the Southern Wisconsin. *Ecological Monographs*, 27(4):325–349.

Brennan, R. L. and Light, R. J. (1974). Measuring Agreement When Two Observers Classify People into Categories Not Defined in Advance. *British Journal of Mathematical and Statistical Psychology*, 27:154–163.

Cheetham, A. H. and Hazel, J. E. (1969). Binary (Presence-Absence) Similarity Coefficients. *Journal of Paleontology*, 43(5):1130–1136.

Choi, S. S. (2008). *Correlation Analysis of Binary Similarity and Dissimilarity Measures.* PhD thesis, Pace University, Seidenberg School of Computer Science and Information Systems, New York City.

Clement, P. W. (1976). A Formula for Computing Inter-Observer Agreement. *Psychological Reports*, 39(1):257–258.

Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and psychological measurement*, 20(1):37–46.

Cole, L. C. (1949). The Measurement of Interspecific Associaton. *Ecology*, 30:411–424.

Cronbach, L. J. (1951). Coefficient Alpha and the Internal Structure of Tests. *Psychometrika*, 16(3):297–334.

Czekanowski, J. (1932). Coefficient of Racial Likeness und Durchschnittliche Differenz. *Anthropologischer Anzeiger*, 9:227–249.

Dallmeier, V., Lindig, C., and Zeller, A. (2005). Lightweight Defect Localization for Java. In *Proceedings of the 19th European Conference on Object-Oriented Programming, ECOOP*.

Dennis, S. F. (1965). The Construction of a Thesaurus Automatically from a Sample of Text. In *Proceedings of the Symposium on Statistical Association Methods For Mechanized Documentation, Washington, DC*, pages 61–148.

Dice, L. R. (1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26(3):297–302.

Digby, P. G. (1983). Approximating the Tetrachoric Correlation Coefficient. *Biometrics*, pages 753–757.

Doolittle, M. (1885). The Verification of Predictions. *Amer. Meteor. J*, 2:327–329.

Driver, H. and Kroeber, A. (1932). Quantitative Expression of Cultural Relationships. *University of California Publications in American Archaeology and Ethnology*, 31:211–256.

Eyraud, H. (1938). Les Principes de la Mesure des Correlations. *Ann. Univ. Lyon Sect. A*, 1:30–47.

Fager, E. W. and McGowan, J. A. (1963). Zooplankton Species Groups in the North Pacific. *Science*, 140(3566):453–460.

Faith, D. P., Minchin, P. R., and Belbin, L. (1987). Compositional Dissimilarity as a Robust Measure of Ecological Distance. *Vegetatio*, 69(1-3):57–68.

Farkas, G. M. (1978). Correction for Bias Present in a Method of Calculating Interobserver Agreement. *Journal of Applied Behavior Analysis*, 11:188.

Fleiss, J. L. (1975). Measuring Agreement Between Two Judges on the Presence or Absence of a Trait. *Biometrics*, pages 651–659.

Fleiss, J. L., Levin, B., and Paik, M. C. (2003). *Statistical Methods for Rates and Proportions (3rd edition ed.)*. Hoboken: John Wiley & Sons, Inc.

Forbes, S. A. (1907). On the Local Distribution of Certain Illinois Fishes: An Essay in Statistical Ecology. *Bulletin of the Illinois State Laboratory for Natural History*, 7:273–303.

Fossum, E. G. (1966). *Optimization and Standardization of Information Retrieval Language and Systems*. Springfield, VA: Clearinghouse for Federal Scientific and Technical Information.

Fowlkes, E. B. and Mallows, C. L. (1983). A Method for Comparing Two Hierarchical Clusterings. *Journal of the American statistical association*, 78(383):553–569.

Gilbert, G. K. (1884). Finley's Tornado Predictions. *American Meteorological Journal*, 1:166–172.

Gilbert, N. and Wells, T. (1966). Analysis of Quadrat Data. *The Journal of Ecology*, 54(3):675–685.

Gini, C. (1912). Variabilità e Mutabilità. *Studi Economico-Giuridici dell'Universita di Cagliari*, 3:1–158.

Gleason, H. A. (1920). Some Applications of the Quadrat Method. *Bulletin of the Torrey Botanical Club*, 47(1):21–33.

Goodman, L. A. and Kruskal, W. H. (1954). Measures of Association for Cross Classifications. *Journal of the American Statistical Association*, 49(338):732–764.

Gower, J. C. (1971). A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, 27(4):857–871.

Gower, J. C. and Legendre, P. (1986). Metric and Euclidean Properties of Dissimilarity Coefficients. *Journal of classification*, 3(1):5–48.

Hamann, U. (1961). Merkmalsbestand und Verwandtschaftsbeziehungen der Farinosae: Ein Beitrag zum System der Monokotyledonen. *Willdenowia*, 2:639–768.

Hamming, R. W. (1950). Error Detecting and Error Correcting Codes. *Bell System technical journal*, 26(2):147–160.

Harris, F. C. and Lahey, B. B. (1978). A Method for Combining Occurrence and Nonoccurrence Interobserver Agreement Scores. *Journal of Applied Behavior Analysis*, 11(4):523–527.

Hawkins, R. P. and Dotson, V. A. (1973). Reliability Scores That Delude: An Alice in Wonderland Trip Through the Misleading Characteristics of Inter-Observer Agreement Scores in Interval Recording. *Behavior Analysis: Areas of Research and Application, E. Ramp and G. Semb, Englewood Cliffs, N. N.J., Prentice-Hall.*

Holley, J. W. and Guilford, J. P. (1964). A Note on the G Index of Agreement. *Educational and Psychological Measurement*, 24:749–753.

Hubert, L. J. (1977). Nominal Scale Response Agreement as a Generalized Correlation. *British Journal of Mathematical and Statistical Psychology*, 30:98–103.

Jaccard, P. (1912). The Distribution of the Flora in the Alpine Zone. *The New phytologist*, 11(2):37–50.

Johnson, S. C. (1967). Hierarchical Clustering Schemes. *Psychometrika*, 32(3):241–254.

Jones, J., Harrold, M. J., and Stasko, J. (2001). Visualization for Fault Localization. In *Proceedings of the Workshop on Software Visualization, 23rd International Conference on Software Engineering, Toronto, Ontario.*

Jones, P. E. and Curtice, R. M. (1967). A Framework for Domparing Term Association Measures. *American Documentation*, 18(3):153–161.

Kent, R. N. and Foster, S. L. (1977). Direct Observational Procedures: Methodological Issues in Naturalistic Settings. In Ciminero, A. R., Calhoun, K. S., and Adams, H. E., editors, *Handbook of Behavioral Assessment*, pages 279–328. New York, John Wiley & Sons.

Köppen, W. (1870). Die Aufeinanderfolge der Unperiodischen Witterungserscheinungen nach den Grundsatzen der Wahrscheinlichkeitsrechnung. In *Repertorium fur Meteorologie*, pages 189–238. Petrograd, Akademiia Nauk.

Köppen, W. (1884). Eine Rationelle Methode zur Prüfung der Wetterprognosen. *Meteorologische Zeitschrift*, 1:397–404.

Kuder, G. F. and Richardson, M. W. (1937). The Theory of the Estimation of Test Reliability. *Psychometrika*, 2:151–160.

Kuhns, J. (1965). The Continuum of Coefficients of Association. *Statistical Association Methods for Mechanised Documentation, (Edited by Stevens et al.) National Bureau of Standards, Washington*, pages 33–39.

Kulczyński, S. (1927). Die Pflanzenassociationen der Pienenen. *Bulletin International de L'Academie Polonaise des Sciences et des Letters, Classe des Sciences Mathematiques et Naturelles, Serie B, Supplement II*, 2:57–203.

Lance, G. N. and Williams, W. T. (1966). Computer Programs for Classification. *Proceedings of the ANCAC Conference, Canberra, Australia.*

Lance, G. N. and Williams, W. T. (1967). Mixed-Data Classificatory Programs I – Agglomerative Systems. *Australian Computer Journal*, 1(1):15–20.

Loevinger, J. A. (1947). A Systematic Approach to the Construction and Evaluation of Tests of Ability. *Psychological Monograph*, 61(4).

Loevinger, J. A. (1948). The Technic of Homogeneous Tests Compared with Some Aspects of Scale Analysis and Factor Analysis. *Psychological Bulletin*, 45:507–530.

Maron, M. E. and Kuhns, J. L. (1960). On Relevance, Probabilistic Indexing and Information Retrieval. *Journal of the ACM*, 7:216–224.

Maxwell, A. E. and Pilliner, A. E. G. (1968). Deriving Coefficients of Reliability and Agreement for Ratings. *British Journal of Mathematical and Statistical Psychology*, 21:105–116.

McConnaughey, B. H. (1964). The Determination and Analysis of Plankton Communities. *Marine Research, Special No, Indonesia*, pages 1–40.

Michael, E. L. (1920). Marine Ecology and the Coefficient of Association: A Plea in Behalf of Quantitative Biology. *Journal of Animal Ecology*, 8:54–59.

Mokken, R. J. (1971). *A Theory and Procedure of Scale Analysis.* The Hague, The Netherlands: Mouton.

Montgomery, A. C. and Crittenden, K. S. (1977). Improving Coding Reliability for Open-Ended Questions. *Public Opinion Quarterly*, 41:235–243.

Morris, J. F. (2012). A quantitative methodology for vetting dark network intelligence sources for social network analysis. Technical report, DTIC Document.

Mountford, M. D. (1962). An Index of Similarity and Its Application to Classificatory Problems. In *Progress in Soil Zoology*, pages 43–50. London, Butterworths.

Nei, M. and Li, W.-H. (1979). Mathematical Model for Studying Genetic Variation in Terms of Restriction Endonucleases. *Proceedings of the National Academy of Sciences*, 76(10):5269–5273.

Ochiai, A. (1957). Zoogeographic Studies on the Soleoid Fishes Found in Japan and its Neighbouring Regions. *Bull. Jpn. Soc. Sci. Fish*, 22(9):526–530.

Orloci, L. (1967). An Agglomerative Method for Classification of Plant Communities. *Journal of Ecology*, 55(1):193–205.

Pearson, K. (1904). Mathematical Contributions to the Theory of Evolution, XIII: On the Ttheory of Ccontingency and Its Relation to Association and Normal Correlation. In *Draper's Company Research Memoirs, Biometric Series II.* Cambridge University Press.

Pearson, K. (1926). On the Coefficient of Racial Likeness. *Biometrika*, 9:105–117.

Pearson, K. and Heron, D. (1913). On Theories of Association. *Biometrika*, 9:159–315.

Peirce, C. S. (1884). The Numerical Measure of the Success of Predictions. *Science*, 4:453–454.

Post, W. J. and Snijders, T. A. (1993). Nonparametric Unfolding Models for Dichotomous Data. *Methodika*, 7:130–156.

Rand, W. (1971). Objective Criteria for the Evaluation of Clustering Methods. *Journal of the American Statistical Association*, 66:846–850.

Rao, C. R. (1995). A Review of Canonical Coordinates and an Alternative to Correspondence Analysis Using Hellinger Distance. *Questiio (Quaderns d'Estadistica i Investigacio Operativa)*, 19:23–63.

Rogers, J. S. and Tanimoto, T. T. (1960). A Computer Program for Classifying Plants. *Science*, 132:1115–1118.

Rogot, E. and Goldberg, I. D. (1966). A Proposed Index for Measuring Agreement in Test-Retest Studies. *Journal of Chronic Disease*, 19:991–1006.

Russel, P. F. and Rao, T. R. (1940). On Habitat and Association of Species of Anopheline Larvae in South-Eastern Madras. *Journal of Malaria Institute India*, 3:153–178.

Scott, W. A. (1955). Reliability of Content Analysis: The Case of Nominal Scale Coding. *Public Opinion Quarterly*, 19:321–325.

Sijtsma, K. and Molenaar, I. W. (2002). *Introduction to Nonparametric Item Response Theory*. Thousand Oaks: Siege.

Simpson, G. G. (1943). Mammals and the Nature of Continents. *American Journal of Science*, 241:1–31.

Sokal, R. R. and Michener, C. D. (1958). A Statistical Method for Evaluating Systematic Relationships. *University of Kansas Science Bulletin*, 38:1409–1438.

Sokal, R. R. and Sneath, P. H. (1963). *Principles of Numerical Taxonomy.* San Francisco: W. H.

Sørensen, T. (1948). A Method of Stabilizing Groups of Equivalent Amplitude in Plant Sociology Based on the Similarity of Species Content and its Applications to Analyses of the Vegetation on Danish Commons. *Kongelige Danske Videnskabernes Selskab Biologiske Skrifter*, 5:1–34.

Sorgenfrei, T. (1958). *Molluscan Assemblages From the Marine Middle Miocene of South Jutland and Their Environments.* Copenhagen: Reitzel.

Stiles, H. E. (1961). The Association Factor in Information Retrieval. *Journal of the Association for Computing Machinery*, 8:271–279.

Stuart, A. (1953). The Estimation and Comparison of Strengths of Association in Contingency Tables. *Biometrika*, 40:105–110.

Tanimoto, T. T. (1957). IBM Internal Report.

Tarwid, K. (1960). Szacowanie Zbieznosci Nisz Ekologicznych Gatunkow Droga Oceny Prawdopodobienstwa Spotykania Sie Ich W Polowach. *Ecologia Polska, Series B*, 6:115–130.

Tversky, A. (1977). Features of Similarity. *Psychological Review*, 84(4):327–352.

Wallace, D. (1983). A Method for Comparing Two Hierarchical Clusterings: Comment. *Journal of the American statistical association*, 78(383):553–569.

Warrens, M. J. (2008). *Similarity Coefficients for Binary Data*. Leiden, Netherlands.

Yule, G. U. (1900). On the Association of Attributes in Statistics: With Illustrations from the Material of the Childhood Society. *Philosophical Transactions of the Royal Society, Series A*, 194:257–319.

Yule, G. U. (1912). On the Methods of Measuring Association Between Two Attributes. *Journal of the Royal Statistical Society, Series A*, 75(6):579–652.

Appendix G. Malware Data Set

G.1 Introduction

Classifying malicious software (malware), and identifying its behavior, is a necessary step in mitigation and helps to identify new and emerging threats (Szor, 2005). Malware metadata provides additional information about the malware, including potentially identifying the source, which can help analysts recognize persistent threats and take appropriate actions to protect the network, as well as support forensic investigations (Sikorski and Honig, 2012).

The goal of the data set creation process, discussed in this appendix, was to create a reasonably large data set of malware labels (type-classification), and an attribute set of behavior and metadata. In order to ensure an accurate (type-classification) label, in this highly contested environment, for each sample collected at least three, of four, domain experts from well-respected Antivirus (AV) software vendors had to agree on the type-classification. This process was additionally complicated by the fact that AV vendors have different naming conventions.

The results of this process is a data set of 2088 samples, consisting of 9 malware types, 46 behavioral and 3 metadata attributes, exported from the VirusTotal (2016b) malware repository. See Table 21 for a summary of the malware types selected for the data set, number of samples of each type and the benchmark AV vendors. See Appendix G.4.1 through Appendix G.4.9 for descriptions of the individual types, and Table 22 for a description of the behavior and metadata attributes selected for the data set.

G.2 Data Source — VirusTotal

VirusTotal is a website for malware analysts, currently owned and managed by Google, Inc. VirusTotal aggregates (at the time of this research) contributions from 57 AV vendors, 60+ scanning engines and data sets, and 17 file characterization tools and data sets (VirusTotal, 2016a).

Anyone can submit a suspect file to VirusTotal for analysis. Each contributing AV vendor has the opportunity to individually evaluate the malware instance and determine if it is, by their analysis, malicious, and, if so, classify it according to their, often unique, type-classification ontology. Suspect files submitted are executed in a controlled instantiation of Cuckoo Sandbox (Guarnieri et al., 2016) environment. The behavior, captured by Cuckoo, are recorded in order to give the analyst a high level overview of what the sample is doing, but does not provide a (malware) type-classification or recommend mitigation steps (VirusTotal, 2015a,b).

Individual files are identified by their sha256 hash. This creates a considerable amount of redundancy in the VirusTotal database, as, even a one-bit difference between files, which are effectively identical in behavior, creates a new instance. Obviously, malicious actors use this fact to defeat traditional signature-based AV tools.

For the malware research accomplished in this dissertation, a private (premium) VirusTotal API was obtained. The private Application Programmer Interface (API) allowed access to VirusTotal related metadata (first seen and last seen dates, number of submissions, submission file names, etc.), file tool information (sigcheck, packer information, Portable Executable (PE) structure, sandbox analysis, etc.). This additional level of access provided behavioral features and metadata contributing to the data set presented here (VirusTotal, 2015b).

G.3 Malware Type Selection Process

Selecting and downloading a data set of behavioral features and metadata from VirusTotal as a test set for the Qualia Modeling Agent (QMA) model was a complex, and somewhat convoluted, task. Malware type-classification is a contested, non-standardized environment, on many levels. The malicious actors intentionally obfuscate the information being searched. There is no industry-standard naming convention for malware identification, despite some well-publicized efforts. Notably, Computer Anti-Virus Researchers' Organization (CARO) in the early 1990's developed a standard naming schema, and in the early 2000's Common Malware Enumeration (CME) initiative by Mitre attempted to address the differences in AV naming conventions by establishing vendor-neutral identification numbers for vendors to reference in addition to their own assigned names. With the explosion of new and emerging malware, the administrative overhead of support the CME soon because in feasible (Dube, 2011; Szor, 2005).

The data set needed to contain malware types that are not so dissimilar as to make differentiation trivial. Therefore, the scope was limited to currently active, as of May 2015, Portable Executable (PE)s, *type:peexe*, whose intended target is a variant of the Microsoft Operating System (OS).

Avira, Kaspersky, Symantec and McAfee were selected as benchmark AV vendors because they maintain comprehensive current type-classification documentation, provide freely available descriptions of malware, and sometimes, on a case-by-case basis, provide cross references to other vendors.

In order to accomplish the following data gathering and parsing tasks a series of Python scripts were developed to execute *http get request methods* against the VirusTotal databases, and parse the files returned. The data gathering process was accomplished in May 2015.

Malware type-classification collection tasks:

1. Initially manually searched Kaspersky Lab's Virus Watch web site Kaspersky (2015) for current threats, which had been identified in the last few weeks as PE, a significant threat, and whose intended target is a variant of the Microsoft OS. For example, Trojan.Win32.Autoit.
2. Submit a query (http get request method) to the VirusTotal repository for instances of this particular variant. Each VirusTotal query returns up to 300 hashes (instances) meeting the query requirements. If a significant number of hashes (200+) were returned then that is an indicator that there may be adequate number of instances to include in the data set.
3. Subsequent queries are submitted, which download the individual detailed Reports on each of the 300 hashes.
4. The results were evaluated to identify type-classifications between vendors that tended to be coincident, and the cross-references between vendor identifications were manually searched, if available. For example: With the Kaspersky classification of Trojan.Win32.Autoit, Avira tended to be classified as DR/AutoIt.Gen, Symantec tended to be classified as WS.Reputation.1 and McAfee tended to be classified as Artemis!
5. Submit a subsequent query to the VirusTotal repository for instances where the classifications for the select AV vendors matched the expected classifications. For example: Kaspersky classification was Trojan.Win32.Autoit, Avira was DR/AutoIt.Gen, Symantec was WS.Reputation.1 and McAfee was Artemis!
6. This process resulted in trial and error, and several malware variants were dismissed as there were not enough agreement among the vendors as to a consistent classification. Often the malware was too new to obtain 200+ instances.

7. Next the JavaScript Object Notation (JSON) files were checked for behavior data, indicating the instance had been sandboxed. If instanced had not yet been sandboxed, then they are removed from the analysis, because there will be no behavioral attributes to analyze.

These steps resulted in the data set summarized in Table 21.

G.4 Malware Types

Table 21: Malware Type-Classifications, Sample Count and Original AV Vendor Names. *Type* is the label given to the malware variance in this dissertation.

Type	**#**	**Avira**	**Kaspersky**	**Symantec**	**McAfee**
Autoit	125	DR/AutoIt.Gen	Trojan.Win32. Autoit	WS. Reputation.1	Artemis!
MBR-Ransomware	285	TR/Spy.Banker	Trojan-Banker.Win32	Infos-tealer.Bancos	W32/ VirRansom
Neshta-Virus	261	W32/Neshta or W32/Delf	Virus.Win32. Neshta	W32.Neshta	W32/HLLP
Sality	194	W32/Sality	Virus.Win32. Sality	W32.Sality	W32/Sality. gen.z
Spy-Banker	270	TR/Crypt. XPACK.Gen	Virus.Win32. PolyRansom	W32. Ransomlock.A	PWS-Banker
VikingAT	209	W32/Viking.AT	Virus.Win32. Qvod.a	W32. Wapomi!inf	W32/Fujacks.be
WORM-Brontok	170	WORM/ Brontok	Email-Worm. Win32. Brontok	W32. Rontokbro	W32/ Rontokbro.gen
Worm-Ramnit	282	W32/Ramnit	Virus.Win32. Nimnul	W32.Ramnit	W32/Ramnit.a
WormViking	292	TR/Kryptik or TR/Cryptic	Worm.Win32. Viking	WS. Reputation.1	Obfuscated-FHH!

G.4.1 Autoit.

Autoit is a family of Trojans, also known as a dropper. Activities this family of Trojans/Droppers may attempt in order to accomplish the goal:

1. Propagation – Autorun
2. Propagation – Peer-to-peer (P2P)
3. Propagation – mapped network drives
4. Downloads files
5. Establishes network connection (connect external domain)
6. Drops malicious files
7. Lowers security settings
8. Registry modification in order to run the processes after reboot. (maintain persistence)
9. Queries registry keys to retrieve shared P2P folders and propagate
10. Checks for internet connections
11. Creates Mutex

(AVIRA, 2015, . . . ?sq=DR%2FAutoIt)

G.4.2 MBRRansomware.

MBRRansomware is a family of Trojans, which drops ransom ware. Activities this family of Trojans/Droppers may attempt in order to accomplish the goal:

1. Drops a malicious file
2. Registry modification
3. Steals information
4. Pricetrap function – user is fooled into making a costly subscription via web page
5. Enumerates many system files and directories.
6. Enumerates process list
7. Process attempts to call itself recursively

(AVIRA, 2015, . . . ?sq=TR%2FCrypt.XPACK.Gen)

G.4.3 NeshtaVirus.

Activities this virus may attempt in order to accomplish the goal:

1. propagation – through infected files

2. Registry modification
3. Code integration – The virus merges its own code with the host program's code. This is a complicated process that requires completely disassembling and reassembling of the host file.

(AVIRA, 2015, …?sq=W32%2FNeshta)

G.4.4 Sality.

Activities this virus may attempt in order to accomplish the goal:

1. propagation – through infected file from:
   - Local network
   - Mapped network drives
2. Drops a malicious file, or type:
   - Windows Executables (*.exe)
   - Windows Dynamic Link Libraries (*.dll)
   - HyperText Markup Language (*.htm/ *.html)
3. Appender – The virus main code is added at the end of the infected file. The last section of the file is modified to include the virus code.
4. This direct-action infector actively searches for files.

(AVIRA, 2015, …?sq=sality)

G.4.5 SpyBanker.

The goal of this malware is to steal financial information allowing for access to financial accounts, subsequently transferring money to unauthorized accounts. Displays a fake banking web browser display in order to trick the user into entering username and password information.
Activities this family of Banker Trojan may attempt in order to accomplish the goal:

1. Connect to a domain that may pose a security risk.
2. Enumerate system files and directories.
3. Send data or commands via HTTP
4. Send stolen information via Email
5. Add or modify browser cookies
6. Modify registry keys, in order to run after reboot (maintain persistence)
7. Begins logging keystrokes after a particular banking site is visited
8. Downloads files

9. Steals information (exfiltration information)
10. Mitigation techniques (covering its tracks): remove registry entries, remove malware, clear cookies

(AVIRA, 2015, . . . ?sq=TR%2FSpy.Banker)

G.4.6 VikingAT.

This is a worm, which self-propagates. Activities this worm may attempt in order to accomplish the goal:

1. Infect a *.exe file
2. Appender – The virus main code is added at the end of the infected file. The following section is added to the infected file: Dbt
3. Drops a malicious file
4. Disable Safe boot and Network boot modes.
5. Registry modification in order to run the processes after reboot. (maintain persistence)
6. Install a rootkit
7. Create new named pipes to communicate with the lsasvc.dll and the rootkit component.
8. Disables show hidden files.
9. copy iteself to network shares using known default and common passwords

(McAfee, 2015, . . . ?id=141204)

G.4.7 WORMBrontok.

This is a worm, which self-propagates. Activities this worm may attempt in order to accomplish the goal:

1. Propagation – Email
2. Drops a malicious file
3. Lowers security settings
4. Registry modification in order to run the processes after reboot. (maintain persistence)
5. Overwrites autoexec.bat

(AVIRA, 2015, . . . ?sq=Brontok)

G.4.8 WormRamnit.

> Virus:Win32/Viking.G is a virus that can infect other executable files. It may also spread to other computers in the network by copying itself to network shares. It may terminate other security-related software and download files from certain web sites
> (Microsoft, 2016, . . . /entry.aspx?Name=Win32%2fRamnit).

G.4.9 WormViking.

> This is a worm that spreads via removable drives and network shares. It can terminate security-related processes, relocate certain Windows files, drop other malware, modify the HOSTS file and Internet files, infect certain files, and connect to a remote server
> (Microsoft, 2016, . . . /entry.aspx?Name=Worm:Win32/Viking.NA).

G.5 Feature (i.e., Attribute) Selection Process

For each of the 2088 malware samples collected, VirusTotal provides two files: [SHA256]Report.txt, and [SHA256]Behaviour.txt. The Report file contains static information, including metadata and Dynamic-Link Library (DLL) imports (declared at compile time). The Behaviour file includes the dynamic information gathered, primarily API calls, collected via Cuckoo Sandbox analysis.

Ideally, all of the features provided in these two files will be used to classify the select malware types. However, many of the API calls are standard among non-malicious PEs, and some of the larger malware instances may incorporate hundreds of API calls.

Therefore, the task described here, was used to identify a small set of high-level behaviorial features, imports and metadata, which were used to classify malware types.
Malware attribute collection tasks:

1. Specific API calls, imports and metadata, were selected for inclusion in an initial

attribute set based on analysis, provided by domain experts (AV vendors), listed in Appendix G.4.

2. An initial set of 93 attributes was used in the QMA formalism and classified malware with accuracy comparable to the results reported in Chapter IV. The computational overhead of such a large set of attributes was, however, a problem.

3. In order to reduced the number of attributes in the data set, while maintaining classification accuracy, statistical methods of multivariate attribute correlation were employed. The analysis resulted in identifying several attributes that were highly correlated.

4. Subsequently 44 attributes were removed from the data set, leaving 49. The QMA still classified malware with significant accuracy, as reported in Chapter IV.

These steps resulted in the attribute set summarized in Table 22

Table 22: Malware Type, Behavior, Features, and Descriptions. Malware type, 34 Application Programmer Interface (API) calls, 9 imported DLLs, 3 communications protocols and 3 (multivalued) metadata.

Type	**Description**	**Index**
Type-Classification	Multivalued: Autoit, MBRRansomware, NeshtaVirus, Sality, SpyBanker, VikingAT, WORMBrontok, WormRamnit, WormViking	0
API Call (binary)	**Description**	**Index**
CopyFileExW: C:\Program Files \Microsoft \DesktopLayer.exe	Possibly infected executable file. (Microsoft, 2016a, Search: DesktopLayer.exe)	31
CreateFileW: \.\MountPointManager	Mount Point Manager – Windows 7 Service Driver responsible with maintaining persistent drive letters and names for volumes (Batcmd.com, 2016, . . . /windows/7/services/mountmgr/)	3
CreateFileW: \.\PIPE\lsarpc	The lsarpc interface is used to communicate with the Local Security Authority (LSA) subsystem. Used by Windows Remote Procedure Call (RPC) services. (Herve Schauer Consultants, 2016, Select: lsarpc interface)	4
CreateFileW: c:\autoexec.bat	Executed on startup of all windows systems.	1
CreateFileW: C:\cmt.exe	Supports remote access to system files and resources. (Microsoft, 2016b, . . . /ee391643(v=vs.85).aspx)	18
CreateFileW: C:\Program Files \Microsoft\px1.tmp	Randomly named portable executable temp file. (TRENDMicro, 2016, . . . /troj_ramnit.smc)	30
CreateFileW: C:\WINDOWS\system32 \drwtsn32.exe	Dr. Watson for Windows program error debugger tool. (Microsoft, 2016c, . . . /kb/308538)	22

Continued on next page

Table 23: Malware Type, Application Programmer Interface (API) calls, Imported DLLs, Communications Protocols and MetaData.

API Call (binary)	Description	Index
CreateFileW: C:\WINDOWS\system32 \winsock.dll	"WINSOCK.DLL is a dynamic-link library that provides a common API for developers of network applications that use the Transmission Control Protocol/Internet Protocol (TCP/IP) stack. This means that a programmer who develops a Windows-based TCP/IP application, such as an FTP or Telenet client, can write one program that works with any TCP/IP protocol stack that provides Windows Socket Services (WINSOCK.DLL). Other applications that depend on a Windows Socket provider include Eudora (a mail package) and Mosaic (a browser for the Internet World Wide Web) (Microsoft, 2016c, .../kb/122982)."	21
LoadLibraryA: advapi32.dll	The process known as Advanced Windows (version 32 Base API) belongs to software Windows Management Instrumentation Driver Extensions (Wmi). Some malware camouflages itself as advapi32.dll. (file.net, 2016, .../process/advapi32.dll.html), (DLL Information, 2016a, .../advapi32_dll.html)	23
LoadLibraryA: C:\WINDOWS\system32 \mswsock.dll	Library which supports windows sockets. The socket function creates a socket that is bound to a specific transport service provider. (Microsoft, 2016b, .../windows/desktop/ms740506(v=vs.85).aspx)	10
LoadLibraryA: comctl32.dll	Support for common controls is provided by ComCtl32.dll, which all 32-bit and 64-bit versions of Windows include. Supports common Graphical User Interface (GUI) displays – used to receive user input. (Microsoft, 2016b, .../windows/desktop/hh298349(v=vs.85).aspx)	5
LoadLibraryA: DNSAPI.dll	This library support Domain Name System (DNS) functions. (Microsoft, 2016b, .../windows/desktop/ms682058(v=vs.85).aspx)	11
LoadLibraryA: kernel32.dll	over 1500 functions supporting kernel (i.e., root) system functionality. (Chappell, 2016)	17
LoadLibraryA: ntdll.dll	Windows NT kernel functions, which run prior to Windows OS instantiation. (DLL Information, 2016a, .../ntdll_dll.html)	28
LoadLibraryA: ole32.dll	a library which contains core Object Linking and Embedding (OLE) (ProcessLibrary.com, 2016, .../ole32/23128/)	8
LoadLibraryA: oleaut32.dll	The oleaut32 module contains functions for application developers. (Microsoft, 2016b, .../ms923851.aspx)	9

Continued on next page

Table 23: Malware Type, Application Programmer Interface (API) calls, Imported DLLs, Communications Protocols and MetaData.

API Call (binary)	Description	Index
LoadLibraryA: rasadhlp.dll	Remote Access AutoDial Helper (DLL Information, 2016a, . . . /rasadhlp_dll.html)	20
LoadLibraryA: rpcrt4.dll	Remote Procedure Call Runtime used by Windows applications for network and Internet communication. (ProcessLibrary.com, 2016, . . . /rpcrt4/23580/)	24
LoadLibraryA: Secur32.dll	Provides required components of windows security — Secure Socket Layer (SSL). (Microsoft, 2016b, . . . /ms913708(v=winembedded.5).aspx)	13
LoadLibraryA: sensapi.dll	"Library to support System Event Notification Service System Event Notification Service (SENS). The sensapi.dll library is used by windows applications when performing synchronization with mobile devices using SENS. Sensapi.dll is required for synchronization (using SENS) to function correctly (Microsoft, 2016b, . . . /windows/desktop/aa377589(v=vs.85).aspx)."	16
LoadLibraryA: SHELL32.dll	"SHELL32.dll is not essential for the Windows OS and causes relatively few problems. SHELL32.dll is [should be] located in the C:\Windows\System32folder. The program has a visible window and provides file access support and webpage opening support to caller applications. The process is loaded during the Windows boot process. SHELL32.dll is able to record keyboard and mouse inputs . . . Some malware disguises itself as SHELL32.dll, particularly when not located in the C:\Windows\System32 folder. (file.net, 2016, . . . /process/shell32.dll.html)"	14
LoadLibraryA: shlwapi.dll	"Shell Light-Weight API DLL — contains the functions for Universal Naming Convention (UNC) and Uniform Resource Locator (URL) paths, registry entries, and color settings is the shlwapi.dll module. The functions can be structured into four categories: string manipulation, path manipulation, registry access and miscellaneous (Microsoft, 2016b, . . . /windows/desktop/bb776779(v=vs.85).aspx)." Wrapper functions that provide limited Unicode functionality for user32.dll, kernel.dll, advapi32.dll, shell32 functions. (Microsoft, 2016b, . . . /windows/desktop/bb759845(v=vs.85).aspx)	26

Continued on next page

Table 23: Malware Type, Application Programmer Interface (API) calls, Imported DLLs, Communications Protocols and MetaData.

API Call (binary)	**Description**	**Index**
LoadLibraryA: VERSION.dll	Version Checking and File Installation Libraries. (DLL Information, 2016b)	27
LoadLibraryA: WS2_32.dll	Library which supports windows sockets. The socket function creates a socket that is bound to a specific transport service provider. (Microsoft, 2016b, . . . /windows/desktop/ms740506(v=vs.85).aspx)	12
LoadLibraryW: comctl32.dll	Unicode version of LoadLibraryA: comctl32.dll (ASCII library)	6
LoadLibraryW: rpcrt4.dll	Unicode version of LoadLibraryA: rpcrt4.dll (ASCII library)	25
LoadLibraryW: RTUTILS.DLL	"Routing Utilities, a module that contains functions used by a tracing API that provides a uniform mechanism for generating diagnostic output for the Microsoft Windows NT/Windows 2000 Routing and Remote Access Service (RRAS) components. The RRAS supports Internet Protocol version 4 (IPv4) and Internet Protocol version 6 (IPv6) network routing and remote user or site-to-site connectivity by using Virtual Private Network (VPN) or dial-up connections (ProcessLibrary.com, 2016, . . . /rtutils/18866/), (Microsoft, 2016, . . . /Dd469790.aspx)."	19
LoadLibraryW: SHELL32.dll	Unicode version of LoadLibraryA: SHELL32.dll (ASCII library)	15
OpenMutexW: ShimCacheMutex	This Mutex cache supports synchronization of shared resources between processes. (Davis, 2012)	29
OpenServiceW: RASMAN	"rasman.exe is a Windows service which is used to dial phone numbers from the phone book. This program is important for the stable and secure running of your computer and should not be terminated (liutilities.com, 2016, . . . /products/wintaskspro/processlibrary/rasman/)."	2
RegCreateKeyExW: Software\Microsoft\Windows NT\CurrentVersion \Winlogon	creates registry key for login shell. (Microsoft, 2016b, . . . /windows/desktop/ms838576(v=winembedded.5).aspx)	7

Continued on next page

Table 23: Malware Type, Application Programmer Interface (API) calls, Imported DLLs, Communications Protocols and MetaData.

API Call (binary)	**Description**	**Index**
RegOpenKeyExA: HKEY_LOCAL_MACHINE	Open the registry key tree that contains detailed information about the local computer. (Microsoft, 2016, . . . /Cc959046.aspx)	34
RegOpenKeyExW: HKEY_LOCAL_MACHINE	Unicode version of RegOpenKeyExA: HKEY_LOCAL_MACHINE	33
VirtualAllocEx: Cmgr.exe	VirtualAllocEx: reserves, commits of changes state of memory in virtual address space for specified file, Cmgr.exe. Cmgr.exe is possibly an infected PE file. (Microsoft, 2016b, . . . /windows/desktop/aa366890(v=vs.85).aspx), (SystemExplorer.net, 2016, . . . /file/cmgr-exe)	32

Table 24: Malware Type, Application Programmer Interface (API) calls, Imported DLLs, Communications Protocols and MetaData.

IMPORTS (binary)	**Description**	**Index**
Import comctl32.dll	see *LoadLibraryA: comctl32.dll*, above	38
Import ole32.dll	see *LoadLibraryA: ole32.dll*, above	39
Import oleaut32.dll	see *LoadLibraryA: oleaut32.dll*, above	40
Import WS2_32.dll	see *LoadLibraryA: WS2_32.dll*, above	41
Import SHELL32.dll	see *LoadLibraryA: SHELL32.dll*, above	42
Import advapi32.dll	see *LoadLibraryA: advapi32.dll*, above	43
Import shlwapi.dll	see *LoadLibraryA: shlwapi.dll*, above	44
Import VERSION.dll	see *LoadLibraryA: VERSION.dll*, above	45
Import ntdll.dll	see *LoadLibraryA: ntdll.dll*, above	46
Communication Protocol (binary)	**Description**	**Index**
DNS	Domain Name System (DNS) comminations protocol is the (Internet, Uniform Resource Identifier (URI)) name resolution protocol for TCP/IP networks. (Microsoft, 2016, . . . /dd197470(v=ws.10).aspx)	35

Continued on next page

Table 25: Malware Type, Application Programmer Interface (API) calls, Imported DLLs, Communications Protocols and MetaData.

Communication Protocol (binary)	**Description**	**Index**
HTTP	Hypertext Transfer Protocol (HTTP) is the communication protocol used to exchange information between a client system and a Web server across a TCP/IP connection. (Microsoft, 2016, . . . /cc780570(v=ws.10).aspx)	36
TCP	Transmission Control Protocol (TCP) comminations protocol which provides a connection-based, reliable service to applications. (Microsoft, 2016, . . . /cc940037.aspx)	37
MetaData	**Description**	**Index**
CharacterSet	The character set used to write the executable code. Multivalued, examples: ASCII, Unicode, Windows_Latin1, Windows_Chinese_Simplified, Windows_Turkish, Windows_Cyrillic, Windows_Taiwan_Big5, Windows_Korea_Shift_KSC_5601 (Harvey, 2016)	47
Language Code	The human language extracted from the executable code. Multivalued, examples: Chinese_Simplified, Chinese_Traditional, English_British, English_U.S., Russian, Korean, Neutral, Spanish_Modern, Turkish, Japanese, German_Austrian, Xhosa, Portuguese_Brazilian. (Harvey, 2016)	48
Subsystem	The environment that the executable runs in. Multivalued, examples: Windows_command_line, Windows_GUI, Console, Native, POSIX. (Harvey, 2016), (Microsoft, 2016b, . . . /windows/desktop/fcc1zstk.aspx)	49

Malware References

AVIRA (2015). AVIRA Support Virus Lab. Retrieved from `https://www.avira.com/en/support-virus-lab`. Last accessed December 2015.

Batcmd.com (2016). Batch Command Services. Retrieved from `http://batcmd.com/windows/7/services/`. Last accessed March 2016.

Chappell, G. (2016). KERNEL32 Functions. Retrieved from `http://www.geoffchappell.com/studies/windows/win32/kernel32/api/`. Last accessed March 2016.

Davis, A. (2012). Leveraging the application compatibility cache in forensic investigations. *Retrieved from* *`https://dl.mandiant.com/EE/library/Whitepaper_ShimCacheParser.pdf`*.

DLL Information (2016a). Windows 7 DLL File Information. Retrieved from `http://www.win7dll.info/`. Last accessed March 2016.

DLL Information (2016b). Windows 8 DLL File Information. Retrieved from `http://www.nirsoft.net/dll_information/windows8/version_dll.html`. Last accessed March 2016.

Dube, T. E. (2011). *A Novel Malware Target Recognition Architecture for Enhanced Cyberspace Situational Awareness*. Dissertation, Air Force Institute of Technology.

file.net (2016). file.net. Retrieved from `http://www.file.net/`. Last accessed March 2016.

Guarnieri, C., Tanasi, A., Bremer, J., and Schloesser, M. (2016). Cuckoo Sandbox. Retrieved from `http://www.cuckoosandbox.org/`. Last accessed January 2016.

Harvey, P. (2016). Exiftool: Read, Write and Edit Meta Information! Retrieved from http://web.mit.edu/graphics/src/Image-ExifTool-6.99/html/index.html/. Last accessed March 2016.

Herve Schauer Consultants (2016). Well-known MSRPC named pipes. Retrieved from http://www.hsc.fr/ressources/articles/win_net_srv/well_known_named_pipes.html. Last accessed March 2016.

Kaspersky (2015). Kaspersky Virus Watch Lite. Retrieved from http://www.kaspersky.com/viruswatchlite. Last accessed December 2015.

liutilities.com (2016). liutilities, Free resource libraries. Retrieved from http://www.liutilities.com/. Last accessed March 2016.

McAfee (2015). McAfee Threat Intelligence. Retrieved from http://www.mcafee.com/threat-intelligence/malware/default.aspx. Last accessed December 2015.

Microsoft (2016a). Microsoft Community. Retrieved from http://answers.microsoft.com/en-us/protect/forum/mse-protect_scanning/. Last accessed March 2016.

Microsoft (2016b). Microsoft Software Developer Network (MSDN) Library. Retrieved from https://msdn.microsoft.com/en-us/library/. Last accessed March 2016.

Microsoft (2016c). Microsoft Support. Retrieved from https://support.microsoft.com/en-us/. Last accessed March 2016.

Microsoft (2016). Microsoft Threat Encyclopedia. Retrieved from http://www.microsoft.com/security/portal/threat/encyclopedia/. Last accessed March 2016.

Microsoft (2016). TechNet online technical support. Retrieved from `https://technet.microsoft.com/en-us/library/`. Last accessed March 2016.

ProcessLibrary.com (2016). Process Library. Retrieved from `http://www.processlibrary.com/en/directory/files`. Last accessed March 2016.

Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands–On Guide to Dissecting Malicious Software*. No Starch Press.

SystemExplorer.net (2016). System Explorer File Database. Retrieved from `http://systemexplorer.net/file-database/`. Last accessed March 2016.

Szor, P. (2005). *The art of computer virus research and defense*. Pearson Education.

TRENDMicro (2016). TREND Micro. Retrieved from `http://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/`. Last accessed March 2016.

VirusTotal (2015a). VirusTotal Behavioural Information. Retrieved from `http://blog.virustotal.com/2012/07/virustotal-behavioural-information.html`. Last accessed November 2015.

VirusTotal (2015b). VirusTotal Private API. Retrieved from `https://www.virustotal.com/en/documentation/private-api/`. Last accessed November 2015.

VirusTotal (2016a). VirusTotal Credits and Acknowledgements. Retrieved from `https://www.virustotal.com/en/about/credits/`. Last accessed March 2016.

VirusTotal (2016b). VirusTotal Malware Repository. Retrieved from `https://www.virustotal.com/`. Last accessed January 2016.

Appendix H. Glossary of Acronyms, Cognitive and Technical Terms

abductive (reasoning) An inference to the best explanation (Henson et al., 2012). Abduction is inferring a case (particular abstract relationship) from a rule (abstract, general claim) and a result (empirical observation) (Shanahan, 1996). Generating a series of competing plausible explanatory hypothesis, and choosing the best based on some set of criteria (Henson et al., 2012). 3, **94**

abstraction Abstractions are a conceptualization of a set of related concepts or a recurring pattern (Henson et al., 2012). **23**, 40, 91, 94, 105, 171

ACT–R Adaptive Control of Thought–Rational (ACT–R). A modern Cognitive Modeling Architecture (CMA) based on the cognitive theory originally published by Anderson and Lebiere (1998). 8, 33ff., 39, 56, 85f., 90, 104, **104**, 105ff., 116, 170

activation level "A state of memory traces that determines both the speed and the probability of access to a memory trace (Anderson, 2005, 183)". 19, 27, 33, 90, **107**, 116

agent An agent is a physical or virtual entity, with some capability of acting or reasoning (Ferber, 1999). 5f., 10f., 21, 29, 33, 61ff., 78, 80, 87, 90, 169

agent-centric From the perspective of an individual agent, as opposed to the term subjective which implies exclusively a human agent. 20, 176

AI Artificial Intelligence (AI). The field of AI incorporates four broad areas of computational sciences: thinking humanly, thinking rationally, acting humanly and acting rationally (Russell and Norvig, 2009). 1

algorithmic The conscious, *algorithmic mind*, is is responsible for sequencing behavior and cognitive decoupling (Stanovich and Evans, 2013).. 6f., 11, 13, 16, 27, 30, 86

ANOVA analysis of variance (ANOVA). xiv, 71f.

API Application Programmer Interface (API). 151, 158

ASCII American Standard Code for Information Interchange (ASCII). 163

autonomous Refers to cognitive Type I processing. 5–8, 11, 13, 15f., 18f., 27, 30, 33, 38, 56, 76, 79, 89, 116, 179

AV Antivirus (AV) software. 150–153, 159

bit-reduced Depending on the context, bit-reduced indicates the reduction in the amount of memory required to represent knowledge or the reduction in the amount of information transfer required. 21

CARO Computer Anti-Virus Researchers' Organization (CARO). 152

chunk A single declarative unit of knowledge. In some cognitive theories, including Adaptive Control of Thought–Rational (ACT–R), elements of declarative knowledge are called chunks (Anderson et al., 2004). 34, 39, 56, 105, **105f.**, 107f.

CL Common Lisp (CL) programming language. 34f., 86, 104

CMA Cognitive Modeling Architecture (CMA). A cognitive architecture which has been implemented in a computer-based system. 29, 33, 35, 104, **104**, 169ff.

CME Common Malware Enumeration (CME) initiative by Mitre Corporation. 152

CMN Computational Models of Narrative (CMN). 85, 87, 89

cognitive architecture A cognitive architecture is both a theory and the representation of that theory in a computer-based modeling tool, referred to as a Cognitive Modeling Architecture (CMA). The theory is "The fixed (or slowly varying) structure that forms the framework for the immediate processes of cognitive performance and learning (Newell, 1990)." No CMA claims to fully model cognitive processes. 76, **104**, 170

cognitive decoupling "Cognitive decoupling appears to be the central cognitive operation accounting for individual differences in fluid intelligence and, because of its role in [mental] simulation and hypothetical thinking, cognitive decoupling is a crucial mental capacity. Decoupling operations help us carry out cognitive reform: the evaluation of our own beliefs and the critique of our own desires. Decoupling secondary representations from the world and then maintaining the decoupling while simulation is carried out is a Type 2 processing operation (Stanovich, 2009)". 13ff., 37, 89, 169

concept drift Concept drift refers to a learning problem that changes over time. In particular, the statistical properties of the target variable, which the model is trying to predict, change over time in unforeseen ways (Žliobaitė, 2010). iv, 1, 43, **43**, 49, 54, 60, 62ff., 78, 83f., 89, 92, 116, **116**, 117

conceptual knowledge A theory of the way episodic memories are abstracted and recorded into Long-term Memory (LTM) is *conceptual knowledge*. Also referred to as aggregate, abstract or blended knowledge, or **conceptual memory**. This knowledge is primarily *declarative*, in that we are aware of we know and can usually describe to others, therefore it may also be referred to as *declarative aggregate knowledge*. When experiences are represented in memory, every detail is not captured. Details, perceived as unimportant, are dropped from memory. Specific episodes are abstracted to general categories of experiences. This abstraction creates conceptual knowledge (Anderson, 2005). 10, 14ff., 18, 23–26, 29–32, 116

conscious See consciousness. 6, 11, 13

consciousness The experiencing of qualia (Cowell, 2001). 5f., 10, 13, 16–23, 26, 29, 171

context "That subset of the complete state of an individual that is used for reasoning about a given goal (Giunchiglia, 1993)". 94

cuckoo Cuckoo Sandbox is a malware analysis system (Guarnieri et al., 2016). 151, 158

deductive (reasoning) A general rule is applied to a specific case, i.e., from the general to the specific. Deductive reasoning is the only way to achieve a provable, logical, solution (Svennevig, 2001). 94, **94**

DLL Dynamic-Link Library (DLL). 158, 160–165

DM Declarative Memory (DM). Also called declarative knowledge, corresponds to things we are aware we know and can usually describe to others. declarative memory (DM) is generally used to represent episodic memory and semantic memory (Anderson, 2007). 106, 116, 172

DNS Domain Name System (Server) (DNS). 161, 164

DPT Dual–Process Theory of Higher Cognition (DPT). A widely accepted view that most creatures, especially humans, have two distinct cognitive processes. One is fast and automatic, such as reactively ducking when an object comes too close to ones face. This is referred to as Type I, the autonomous/reactive mind. Type I processing is often credited with intuitive behavior, such as a babies ability to suck at birth or the physical reaction to a foul odor or loud noise. The second form of cognition is when a being consciously deliberates over a decision, such as choosing a menu option, or when a squirrel learns how to defeat the expensive anti-squirrel bird feeder. This is referred to as Type II, the deliberative/reflective mind. It is believed that these two minds work together to provide the cognitive processing for survival, problem solving as well as creativity in humans (Stanovich and Evans, 2013; Stanovich, 2009).. 5, 8, 10f., 16, 19, 25

DT decision tree (DT). 69ff., 74, 76, 79, 82, 86f., 109f., 112ff.

eccentricity Hypernetwork Theory: Eccentricity is an asymmetric measure of connectivity between two simplices (Johnson, 2013). 46f., 88, 102

FDR fully disjunctive reasoning (FDR). 15, 23, 27, 34, 36, 84, 86, 89f.

GUI Graphical User Interface (GUI). 161

Hausdorff distance "Given two sets of points, ...the maximum of the distance from a point in any of the sets to the nearest point in the other set (Rote, 1991)". 47, 88, 96, 99, **99**, 102

HTTP Hypertext Transfer Protocol (HTTP). 165

hypernetwork theory aka Hypernetworks, a theory which extends network theory to multidimensional hypernetworks for modeling multi-element relationships, in particular systems in nature, society and cognition. Also referred to as Polyhedral dynamics in earlier literature (Casti, 1977; Johnson, 2013; Wang et al., 2010). 8, 32–35, 39, 41f., 47f., 54, 61, 64, 87ff., 96

IIT The Integrated Information Theory (IIT) of Consciousness equates consciousness with integrated information (Tononi, 2012). iv, 1, 8, 21ff., 26, 28–31, 84–87, 89, 175

incidence matrix "The incidence matrix of a graph gives the $(0, 1)$-matrix which has a row for each vertex and column for each edge, and $(v, e) = 1$ iff vertex v is incident upon edge e (Wolfram, 2015)". 99, 101

inductive (reasoning) A generalization is inferred given a specific case, i.e., from the specific to the general. In induction the conclusion is not assured to be correct

(Svennevig, 2001). 94, **95**

ISR Intelligence, Surveillance, and Reconnaissance (ISR). 3

JSON JSON, (JavaScript Object Notation). 154

kNN k Nearest Neighbors. 69ff., 74, 76–79, 82, 86f., 91f., 109–112

LSA Local Security Authority (LSA). 160

LTM Long-term Memory (LTM). Relatively permanent storage of memory. "Information must be rehearsed before it can be moved into a relatively permanent Long-term memory (Anderson, 2005, 176) ". 171

malware malicious software (malware). 3, 50, 54, 56f., 60, 62f., 69, 150–153, 158f.

metric "A formal way to view the notion of distance between [elements] in a set. (Peeler, 2011)". 97, 102

metric space A precise formal measure of distance given any set. A abstracted notion of Euclidean spaces applied to any set (Peeler, 2011). 97ff.

ML Machine Learning (ML). iv, 1, 3, 10, 29, 33, 41, 43, 49, 54, 59f., 62, 69f., 74, 76, 78f., 83, 89, 115ff., 119f., 178

mpd marginal probability distribution (mpd). 65ff., 119

Mutex Mutex, Short for mutual exclusion object. A Mutex is a program object that allows multiple program threads to share the same resource, such as file access, but not simultaneously. (Webopedia, 2015) (/term/m/mutex.html)). 155, 163

OLE Object Linking and Embedding (OLE). 161

OOM order of magnitude (OOM). xiv, 76f., 80, 87, 93, 110, 112, 114

OS Operating System (OS). 152f.

OTU Operational Taxonomic Unit (OTU). **123**, 124f.

P2P P2P, (Peer-to-peer). 155

packer Packers are wrappers put around pieces of software to compress and/or encrypt their contents (Bulletin, 2015).. 151

PCT Parsimonious Covering Theory (PCT), primarily used for medical diagnosis, PCT uses domain-specific background knowledge to determine the best explanation (e.g., diagnosis) for a set of observations (Henson et al., 2012). **94**

pdf Probability density function (pdf). 117

PE Portable Executable (PE). 151ff., 158, 164

PM Procedural Memory (PM), also referred to as *procedural knowledge*. “An implicit memory [a memory without conscious awareness] involving knowledge about how to perform tasks (Anderson, 2005, 238)” Procedural knowledge is knowledge which we display in our behavior but of which we are not conscious (Anderson, 2005). **106**

polyhedral dynamics Polyhedral dynamics (aka Hypernetwork, aka Q-Analysis) developed from set theory, and is entirely compatible with network theory, as a methodology to address the analysis of large-scale systems represented in multidimensional arrays, the relationship between qualitative data, psychological and social relations (Casti, 1977; Johnson, 2013; Empowerment, 2008). 32, 173

QMA Qualia Modeling Agent (QMA). iv, xiiff., 1, 29ff., 36ff., 49f., 56, 60, 62ff., 67, 69ff., 74, 76–87, 90ff., 99, 107f., 110, 114, 116, 152, 159, 176

QS Qualia Space (QS). A component of the Integrated Information Theory (IIT) of Consciousness which equates consciousness with integrated information. “Qualia space (QS) is a space where each axis represents a possible state of the complex, each point is a probability distribution of its states, and arrows between points represent the informational relationships among its elements generated by causal mechanisms (connections). Together, the set of informational relationships within a complex constitute a shape in QS that completely and univocally specifies a particular experience (Tononi, 2008)". iv, xii, 1, 21–26, 31f., 40f., 59, 66, 76f., 84, 88ff., 106, 109, 114, 116, 121, 125f., 130

quale Plural: qualia. The abstracted, agent-centric, context dependent internal representations of evoked experiences based on perceived or predicted sensor data. The mental representations used throughout cognitive processing through which high-level perception, chaotic environmental stimuli are organized into the mental representations (Chalmers et al., 1992; Hubbard, 1996). xii, 20, **20**, 21, **21**, 22, 24, 26, 30, 32, 171

Query Element In QMA: The Query Element is the specific element that is to be inferred. An observation/experience can have multiple unobserved elements, but only one (at a time) can be a Query Element.. **43**, 48, **54**, 55–59, 65, 88, 103, 176

Query Simplex In QMA: A simplex with one Query Element. 48f., **54**, 55, 57f., 88, 103

reflective The conscious, *reflective mind*, is explicit, effortful, pattern-completion (i.e., hypothetical) decision making which supports slow deliberation, and uses working memory (WM) (Anderson, 2005; Kahneman, 2011; Stanovich and Evans, 2013).. 6f., 11, 13–18, 23, 27, 30, 34f., 39

RPC Remote Procedure Call (RPC). 160

RRAS Routing and Remote Access Service (RRAS). 163

schemata (aka Schematic memory) "Existing knowledge providing a framework within which new knowledge is integrated (Young, 1998)." F.C. Bartlett's work (Bartlett, 1932) (as cited in M. J. Young, 1998) "An active organization of past experiences and reactions that shapes a person's response to new stimuli (Young, 1998)." J. M. Mandler's work (as cited in M. J. Young, 1998) "A modern definition of the schema concept defines a schema as a temporally or spatially organized structure whose components are a set of variables or slots that are filled or instantiated by values. Mandler proposes that schema are formed on the basis of proximities experienced in space or time (Young, 1998)". xii, 23–26, 30ff.

semantic memory Semantic memories reflect general knowledge of the world, and is typically viewed as a non-instance based representation. Although we have all encountered the fact that 2+2=4 hundreds of times in our lives, we might only have one memory representation of this fact (Anderson, 2005, 240), (Sims and Gray, 2004). 172

SENS System Event Notification Service (SENS). 162

sigcheck Sigcheck is a command-line utility that shows file version number, timestamp information, and digital signature details, including certificate chains (Russinovich, 2016).. 151

simplex "A simplex is the generalization of a tetrahedral [or greater (Johnson, 1995)] region of space to n dimensions. The boundary of a k-simplex has $k+1$ 0-faces (vertices), $k(k+1)/2$ 1-faces (edges) and $\binom{k+1}{i+1}$ i-faces, where $\binom{n}{k}$ is a binomial coefficient (Wolfram, 2015)". 44, 47f., 88, 100ff.

simplicial complex "a set of simplices with all of their faces (Johnson, 2013)". 101

SSL Secure Socket Layer (SSL). 162

Stanovich's framework An explanation for human behavior by describing the interaction of the three minds or cognitive levels (Stanovich, 2009). iv, xii, 1, 5, 7f., 10f., 15f., 19, 22, 25, 27, 29ff., 37, 61f., 84f., 89

supervised machine learning "Supervised machine learning is the search for algorithms that reason from externally supplied instances to produce general hypotheses, which then make predictions about future instances. In other words, the goal of supervised learning is to build a concise model of the distribution of class labels in terms of predictor features. The resulting classifier is then used to assign class labels to the testing instances where the values of the predictor features are known, but the value of the class label is unknown (Kotsiantis et al., 2007)". **118**

target variable In QS: The variable, which is the target of the inference/pattern-completion formalism, whose value is to be inferred. The term *target variable* is also referred to as *class* or *category* in machine learning (ML) terminology (Pang–Ning et al., 2006). iv, 1f., 5, 10f., 29, 33, **43f.**, 49f., 53f., 56–59, 61–67, 69f., 77f., 80, 82ff., 87, 92f., 109, 171, 179

TCLI Tightly Compiled Learned Information (TCLI), which is knowledge generated in WM that has become "…tightly compiled and available to the autonomous mind due to overlearning and practice (Stanovich, 2009)." TCLI is retained in the autonomous mind [Type I] and provided to the reflective and algorithmic minds [Type II] for production (Stanovich, 2009). 13, 16–19, 27, 33f., 90

TCP Transmission Control Protocol (TCP). 165

TCP/IP Transmission Control Protocol/Internet Protocol (TCP/IP). 161, 164f.

TL Transfer Learning (TL) In the research field of ML, transfer learning is the ability of a system to apply knowledge or skills learned in previous tasks to subsequent tasks or new domains, which are similar in some way (Pan and Yang, 2010). 43, **43**, 60, 64–67, 83, 92, **119**

transductive (reasoning) Transductive reasoning is inferring from one specific experience to another specific case (Vapnik, 2006). **95**

Type I Refers to autonomous cognitive processing. 170, 172

Type II Refers to reflective cognitive processing. 172

type-classification The classification of potentially malicious software (malware) into categories based on various attributes, such as, spreading mechanism, destructive behavior, system dependencies, specific target variables, etc...(Szor, 2005). 151ff.

UCI University of California, Irvine (UCI). xiv, 49, 51, 69, 84

UNC Universal (or Uniform, or Unified)[file] Naming Convention (UNC). 162

unconscious See autonomous. 5, 11, 13

URI Uniform Resource Identifier (URI). 164

URL Uniform Resource Locator (URL). 162

VirusTotal VirusTotal. 152

VPN Virtual Private Network (VPN). 163

WM Working memory (WM), is the knowledge that is currently available in memory for working on a problem. That knowledge may consist of a combination of various forms: e.g., declarative, conceptual, procedural (Anderson, 2005; Stanovich, 2009). 5–8, 11–18, 21, 23, 25, 27, 30f., 33ff., 39, 84ff., 89, **176**, 178

Bibliography

Aleksander, I. and Gamez, D. (2011). Informational Theories of Consciousness: A Review and Extension. In *From Brains to Systems*, pages 139–147. Springer.

Almaksour, A. (2011). *Incremental Learning Of Evolving Fuzzy Inference Systems: Application To Handwritten Gesture Recognition.* PhD thesis, INSA de Rennes.

Almanac, C. Q. (1984). 98th congress. *2nd session*, 40.

Anderson, J. R. (2005). *Cognitive Psychology and its Implications.* Macmillan, sixth edition.

Anderson, J. R. (2007). *How Can the Human Mind Occur in the Physical Universe?* Oxford University Press.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2004). An Integrated Theory of the Mind. *Psychological review*, 111(4):1036.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2012). *ACT–R 6.0 Tutorial.* Carnegie Mellon University.

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C., and Qin, Y. (2015). Adaptive Control of Thought–Rational (ACT–R) @ONLINE. Retrieved from `http://act-r.psy.cmu.edu/`. Last accessed May 2015.

Anderson, J. R. and Lebiere, C. (1998). *The Atomic Components of Thought.* Lawrence Erlbaum Associates Publishers.

Anderson, R. C. and Pearson, P. D. (1988). A Schema–Theoretic View of Basic Processes in Reading Comprehension. *Interactive approaches to second language reading*, pages 37–55.

Apostol, T. M. (1974). *Mathematical Analysis.* Addison Wesley Publishing Company.

Arrabales, R., Ledezma, A., and Sanchis, A. (2009). CERA-CRANIUM: A Test Bed for Machine Consciousness Research. *International Workshop on Machine Consciousness.*

Arrabales, R., Ledezma, A., and Sanchis, A. (2010). Towards the Generation of Visual Qualia in Artificial Cognitive Architectures. *Brain Inspired Cognitive Systems (BICS).*

Babbie, E. R. (1998). *The practice of social research*, volume 112. Wadsworth publishing company Belmont, CA.

Bação, F. (2002). Data Mining Geo-Espacial. *Instituto Superior de.*

Balduzzi, D. and Tononi, G. (2009). Qualia: The Geometry of Integrated Information. *PLoS computational biology*, 5(8):e1000462.

Banks, A. P. (2013). The Influence of Activation Level on Belief Bias in Relational Reasoning. *Cognitive science*, 37(3):544–577.

Bartlett, F. C. (1932). Remembering: An Experimental and Social Study. *Cambridge: Cambridge University.*

Boriah, S., Chandola, V., and Kumar, V. (2008). Similarity Measures for Categorical Data: A Comparative Evaluation. *red*, 30(2):3.

Bothell, D. (2015). *ACT-R 6.0 Reference Manual.*

Brook, A. (2013). Kant's View of the Mind and Consciousness of Self. The Stanford Encyclopedia of Philosophy, fall 2013 edition.

Bulletin, V. (2015). Virus Bulletin. Retrieved from `https://www.virusbtn.com/resources/glossary/packer.xml`. Last accessed December 2015.

Casti, J. (1977). Polyhedral Dynamics–I: The Relevance of Algebraic Topology to Human Affairs.

Chalmers, D. J., French, R. M., and Hofstadter, D. R. (1992). High–Level Perception, Representation, and Analogy: A Critique of Artificial Intelligence Methodology. *Journal of Experimental & Theoretical Artificial Intelligence*, 4(3):185–211.

Choi, S., Cha, S., and Tappert, C. C. (2010). A Survey of Binary Similarity and Distance Measures. *Journal of Systemics, Cybernetics and Informatics*, 8(1):43–48.

Chong, H., Tan, A., and Ng, G. (2007). Integrated Cognitive Architectures: A Survey. *Artificial Intelligence Review*, 28(2):103–130.

Churchland, P. S. (2002). *Brain-wise: Studies in neurophilosophy.* MIT press.

Cornuéjols, A. (1993). Getting Order Independence in Incremental Learning. In *Machine Learning: ECML-93*, pages 196–212. Springer.

Cowell, C. W. (2001). *Minds, Machines and Qualia: A Theory of Consciousness.* PhD thesis, University of California.

Dennett, D. (1991). Consciousness explained (london: Allen lane, 1992). *Darwin's Dangerous Idea: Evolutions and Meanings of Life.*

Deza, M. M. and Deza, E. (2009). *Encyclopedia of Distances.* Springer.

Dhurandhar, A. and Dobra, A. (2013). Probabilistic characterization of nearest neighbor classifier. *International Journal of Machine Learning and Cybernetics*, 4(4):259–272.

Duda, R. O., Hart, P. E., and Stork, D. G. (2001). Pattern classification. 2nd. *Edition. New York*.

Edelman, G. M. (1989). *The Remembered Present: A Biological Theory of Consciousness*. Basic Books.

Empowerment, P. (2008). Polyhedral Empowerment of Networks through Symmetry: Psycho–social implications for organization and global governance. Retrieved from `http://www.laetusinpraesens.org/docs00s/polynet.php/`. Last accessed November 2015.

Faghihi, U., Estey, C., McCall, R., and Franklin, S. (2015). A Cognitive Model Fleshes out Kahneman's Fast and Slow Systems. *Biologically Inspired Cognitive Architectures*, 11:38–52.

Ferber, J. (1999). *Multi–Agent Systems: An Introduction to Distributed Artificial Intelligence*, volume 1. Addison–Wesley Reading.

Gamez, D. (2008). Progress in Machine Consciousness. *Consciousness and Cognition*, 17(3):887–910.

Gardner, A., Kanno, J., Duncan, C., and Selmic, R. (2014). Measuring distance between unordered sets of different sizes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 137–143.

Geng, X. and Smith-Miles, K. (2009). Incremental Learning. In *Encyclopedia of Biometrics*, pages 731–735. Springer.

Gentle, J. E. (2009). *Computational Statistics*, volume 308. Springer.

Giunchiglia, F. (1993). Contextual Reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, 16:345–364.

Goodman, L. A. and Kruskal, W. H. (1954). Measures of Association for Cross Classifications. *Journal of the American Statistical Association*, 49(338):732–764.

Guarnieri, C., Tanasi, A., Bremer, J., and Schloesser, M. (2016). Cuckoo Sandbox. Retrieved from `http://www.cuckoosandbox.org/`. Last accessed January 2016.

Hélie, S. and Sun, R. (2014). Autonomous Learning in Psychologically Oriented Cognitive Architectures: A Survey. *New Ideas in Psychology*, 34:37–55.

Henson, C., Sheth, A., and Thirunarayan, K. (2012). Semantic Perception: Converting Sensory Observations to Abstractions. *Internet Computing, IEEE*, 16(2):26–34.

Hubbard, T. L. (1996). The Importance of a Consideration of Qualia to Imagery and Cognition. *Consciousness and Cognition*, 5(3):327–358.

Imandoust, S. B. and Bolandraftar, M. (2013). Application of k-nearest neighbor (knn) approach for predicting economic events: Theoretical background. *International Journal of Engineering Research and Applications*, 3(5):605–610.

Jacobs, D., Weinshall, D., and Gdalyahu, Y. (2000). Class representation and image retrieval with non-metric distances. *IEEE Trans. Pattern Anal. Mach. Intell*, 22(6):583–600.

Johnson, J. (1995). The Multidimensional Networks of Complex Systems. In *Networks in Action*, pages 49–79. Springer.

Johnson, J. (2013). *Hypernetworks in the Science of Complex Systems*, volume 3. Imperial College Press London.

Kahneman, D. (2011). *Thinking, Fast and Slow*. Macmillan.

Keinosuke, F. (1990). Introduction to statistical pattern recognition. *Academic Press Inc.*

Kohavi, R. et al. (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Ijcai*, volume 14, pages 1137–1145.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.

Lakoff, G. and Narayanan, S. (2010). Toward a computational model of narrative. In *AAAI Fall Symposium: Computational Models of Narrative*.

Leslie, A. M. (1987). Pretense and representation: The origins of" theory of mind.". *Psychological review*, 94(4):412.

Lichman, M. (2013). UCI Machine Learning Repository. Retrieved from `http://archive.ics.uci.edu/ml`, University of California, Irvine, School of Information and Computer Sciences. Last accessed November 2015.

Lourenco, F., Lobo, V., and Bacao, F. (2004). Binary–Based Similarity Measures for Categorical Data and their Application in Self-Organizing Maps.

Lynott, D. and Connell, L. (2009). Embodied Conceptual Combination. *Embodied and grounded cognition*, page 79.

Maimon, O. and Rokach, L. (2005). Introduction to supervised methods. In *Data Mining and Knowledge Discovery Handbook*, pages 149–164. Springer.

Maloof, M. A. and Michalski, R. S. (2004). Incremental Learning with Partial Instance Memory. *Artificial intelligence*, 154(1):95–126.

Mandler, J. M. and Ritchey, G. H. (1977). Long-Term Memory for Pictures. *Journal of Experimental Psychology: Human Learning and Memory*, 3(4):386.

McCune, B., Grace, J. B., and Urban, D. L. (2002). *Analysis of ecological communities*, volume 28. MjM software design Gleneden Beach, OR.

Michalski, R. S. and Chilausky, R. L. (1980). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2):125–161.

Mitchell, T. M. (1997). Machine learning.

Monk, J. D. (1974). Connections Between Combinatorial Theory and Algebraic Logic. *Studies in Math*, 9:58–91.

Mooney, R. J. (2016). Machine Learning Research Group, Univerity of Texas. Retrieved from `http://www.cs.utexas.edu/~ml/publications/area/125/transfer_learning`. Last accessed May 2016.

Morris, J. F. (2012). A quantitative methodology for vetting dark network intelligence sources for social network analysis. Technical report, DTIC Document.

Munkres, J. (1993). Simplicial Complexes and Simplicial Maps. *Elements of Algebraic Topology*, pages 7–14.

Narasimhamurthy, A. M. and Kuncheva, L. I. (2007). A framework for generating data to simulate changing environments. In *Artificial Intelligence and Applications*, pages 415–420.

Newell, A. (1990). Unified Theories of Cognition. *Cambridge, MA: Harvard University.*

Pan, S. J. and Yang, Q. (2010). A Survey on Transfer Learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359.

Pang–Ning, T., Steinbach, M., Kumar, V., et al. (2006). Introduction to Data Mining. In *Library of Congress.*

Patterson, R. E. (2016). Intuitive Cognition and Models of Human–Automation Interaction. *Human Factors, in press.*

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Peeler, L. (2011). Metric Spaces and the Contraction Mapping Principle.

Peirce, C. S. (1974). *Collected Papers of Charles Sanders Peirce*, volume 5. Harvard University Press.

Reggia, J. A. (2013). The rise of machine consciousness: Studying consciousness with computational models. *Neural Networks*, 44:112–131.

Ritter, H. and Kohonen, T. (1989). Self-organizing semantic maps. *Biological cybernetics*, 61(4):241–254.

Rogers, S. K., Kabrisky, M., Bauer, K. W., and Oxley, M. E. (2003). Computing machinery and intelligence amplification. *Computational Intelligence, The Experts Speak (Chapter 3).*

Rogers, S. K., Sadowski, C., Bauer, K. W., Oxley, M. E., Kabrisky, M., Rogers, A., and Mott, S. D. (2008). The life and death of ATR/sensor fusion and the hope for resurrection. *Automatic Target Recognition XVIII*, 6967.

Rote, G. (1991). Computing the minimum hausdorff distance between two point sets on a line under translation. *Information Processing Letters*, 38(3):123–127.

Rumelhart, D. E., Ortony, A., et al. (1976). *The representation of knowledge in memory.* Center for Human Information Processing, Department of Psychology, University of California, San Diego.

Russell, S. and Norvig, P. (2009). Artificial Intelligence: A Modern Approach.

Russinovich, M. (2016). Windows sysinternals, sigcheck v2.2. Retrieved from `https://technet.microsoft.com/en-us/sysinternals/bb897441.aspx`. Last accessed January 2016.

Samsonovich, A. V. and Nadel, L. (2005). Fundamental principles and mechanisms of the conscious self. *Cortex*, 41(5):669–689.

Searle, J. R., Dennett, D. C., and Chalmers, D. J. (1997). *The mystery of consciousness.* New York Review of Books.

Shanahan, M. (1996). Robotics and the common sense informatic situation'. In *ECAI*, pages 684–688. PITMAN.

Sikorski, M. and Honig, A. (2012). *Practical Malware Analysis: The Hands–On Guide to Dissecting Malicious Software.* No Starch Press.

Sims, C. R. and Gray, W. D. (2004). Episodic versus semantic memory: An exploration of models of memory decay in the serial attention paradigm. In *ICCM*, pages 279–284.

Stanovich, K. E. (2009). *What intelligence tests miss: The psychology of rational thought.* Yale University Press.

Stanovich, K. E. and Evans, J. S. B. T. (2013). Dual–Process Theories of Higher Cognition Advancing the Debate. *Perspectives on Psychological Science*, 8(3):223–241.

Sternberg, S. (2010). *Dynamical systems.* Courier Corporation.

Svennevig, J. (2001). Abduction as a methodological approach to the study of spoken interaction. *Norskrift*, 103:1–22.

Szor, P. (2005). *The art of computer virus research and defense.* Pearson Education.

Thirumuruganathan, S. (2010). A detailed introduction to k–nearest neighbor (knn) algorithm. *Retrieved March*, 20:2012.

Tononi, G. (2008). Consciousness as integrated information: a provisional manifesto. *The Biological Bulletin*, 215(3):216–242.

Tononi, G. (2012). Integrated Information Theory of Consciousness: An Updated Account. *Arch Ital Biol*, 150(2-3):56–90.

Tye, M. (2015). Qualia. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy.*

Uzgalis, W. (2015). John locke. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy.* Summer 2015 edition.

Van Gulick, R. (2014). Consciousness. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy.* Spring 2014 edition.

van Rijsbergen, C. J. (1979). *Information Retrieval.* Department of Computing Science, University of Glasgow.

Vapnik, V. (2006). Empirical inference science. *Estimation of Dependencies Based on Empirical Data, second edition, Springer.*

Vaughan, S. L., Mills, R. F., Grimaila, M. R., Peterson, G. L., Oxley, M. E., Dube, T. E., and Rogers, S. K. (2015). Quest for malware type–classification. In *SPIE Defense+ Security.* International Society for Optics and Photonics.

Vaughan, S. L., Mills, R. F., Grimaila, M. R., Peterson, G. L., and Rogers, S. K. (2014). Narratives as a Fundamental Component of Consciousness. In *5th Workshop on Computational Models of Narrative.*

VirusTotal (2016). VirusTotal Malware Repository. Retrieved from `https://www.virustotal.com/`. Last accessed January 2016.

Wang, J., Rong, L., Deng, Q.-H., and Zhang, J. (2010). Evolving hypernetwork model. *The European Physical Journal B-Condensed Matter and Complex Systems*, 77(4):493–498.

Wanner, E. (1974). *On remembering, forgetting, and understanding sentences: A study of the deep structure hypothesis*. Number 170. Mouton De Gruyter.

Wason, P. C. (1966). New horizons in psychology.

Webopedia (2015). Webopedia. Retrieved from `http://www.webopedia.com`. Last accessed December 2015.

Wolfram (2015). Wolfram mathworld. Retrieved from `http://mathworld.wolfram.com/`.

Young, M. J. (1998). *Computational Modeling of Memory: The Role of Long–Term Potentiation and Hebbian Synaptic Modification in Implicit and Schema Memory*. Dissertation, Miami University.

Žliobaitė, I. (2010). Learning under Concept Drift: An Overview. *arXiv preprint arXiv:1010.4784*.

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704–0188*
The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704–0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202–4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**			
1. REPORT DATE *(DD–MM–YYYY)* 15–09–2016	**2. REPORT TYPE** PhD Dissertation		**3. DATES COVERED** *(From — To)* Oct 2012 — Sep 2016
4. TITLE AND SUBTITLE A Novel Machine Learning Classifier Based on a Qualia Modeling Agent (QMA)		**5a. CONTRACT NUMBER**	
		5b. GRANT NUMBER	
		5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Vaughan, Sandra L., Civilian		**5d. PROJECT NUMBER** JON16G337-NS	
		5e. TASK NUMBER	
		5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765			**8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT-ENG-DS-16-S-016
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/Sensors Directorate 2241 Avionics Circle, Bldg 600 WPAFB OH 45433-7318 DSN 798-8200, COMM 937-528-8200 Email: catherine.griffith.3.ctr@us.af.mil			**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RY
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.			
13. SUPPLEMENTARY NOTES This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.			

14. ABSTRACT

This dissertation addresses a problem found in supervised ML classification, that the target variable, i.e., the variable a classifier predicts, has to be identified before training begins and cannot change during training and testing. This research develops a computational agent, which overcomes this problem. The QMA is modeled after two cognitive theories: Stanovich's framework, which proposes learning results from interactions between conscious and unconscious processes; and, the IIT, which proposes that the fundamental structural elements of consciousness are qualia. By modeling the informational relationships of qualia, the QMA allows for retaining and reasoning-over data sets in a non-ontological, non-hierarchical QS. This novel computational approach supports concept drift, by allowing the target variable to change *ad infinitum* without re-training while achieving classification accuracy comparable to or greater than benchmark classifiers. Additionally, the research produced a functioning model of Stanovich's framework, and a computationally tractable working solution for a representation of qualia, which when exposed to new examples, is able to match the causal structure and generate new inferences.

15. SUBJECT TERMS

Machine Learning, Supervised Classifier, Qualia, Cognitive Modeling

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. Robert F. Mills, AFIT/ENG
U	U	U	UU	204	**19b. TELEPHONE NUMBER** *(include area code)* (937) 255-3636, x4527; Robert.Mills@afit.edu