

2015-05-11

Eliciting Knowledge Bases with Defeasible Reasoning: A Comparative Analysis with Machine Learning

Peter Keogh

Follow this and additional works at: <https://arrow.tudublin.ie/scschcomdis>



Part of the [Computer Engineering Commons](#)

Recommended Citation

Keogh, P. (2015) *Eliciting Knowledge Bases with Defeasible Reasoning: A Comparative Analysis with Machine Learning*. Thesis submitted in fulfilment of the requirements for the Degree of MSc in Computing (Advanced Software Development) in the DIT School of Computing.

This Dissertation is brought to you for free and open access by the School of Computer Science at ARROW@TU Dublin. It has been accepted for inclusion in Dissertations by an authorized administrator of ARROW@TU Dublin. For more information, please contact arrow.admin@tudublin.ie, aisling.coyne@tudublin.ie, vera.kilshaw@tudublin.ie.

DUBLIN INSTITUTE OF TECHNOLOGY

MASTERS THESIS

**Eliciting Knowledge Bases with
Defeasible Reasoning: A Comparative
Analysis with Machine Learning**

Author:

Peter KEOGH

Supervisor:

Dr. Luca LONGO

*A thesis submitted in fulfilment of the requirements
for the degree of MSc in Computing (Advanced Software Development)*

in the

DIT School of Computing

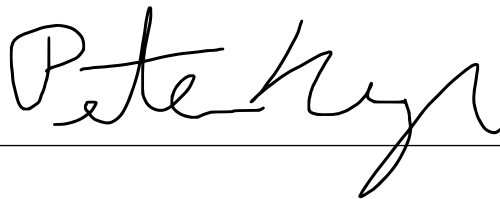
June 2015

Declaration of Authorship

I, Peter KEOGH, declare that this thesis towards the award of International MSc in Advanced Software Development titled, 'Eliciting Knowledge Bases with Defeasible Reasoning: A Comparative Analysis with Machine Learning' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a degree at this Institute.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this Institute or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:



Date:

24/6/2015

“Never trust a man who, when left alone in a room with a tea cozy, doesn’t try it on.”

Billy Connolly

DUBLIN INSTITUTE OF TECHNOLOGY

Abstract

Faculty Name

DIT School of Computing

MSc in Computing (Advanced Software Development)

Eliciting Knowledge Bases with Defeasible Reasoning: A Comparative Analysis with Machine Learning

by Peter KEOGH

This thesis compares the ability of an implementation of Defeasible Reasoning (via Argumentation Theory) to model a construct (mental workload) with Machine Learning. In order to perform this comparison a defeasible reasoning system was designed and implemented in software. This software was used to elicit a knowledge base from an expert in an experiment which was then compared with machine learning. The central findings of this thesis were that the knowledge based approach was better at predicting an objective performance measure, time, than machine learning. However, machine learning was better equipped to identify another object measure task membership. The knowledge base of the expert had a high concurrent validity with objective performance measures and a high convergent validity with existing measures of mental workload.

Acknowledgements

I couldn't have completed this masters without the support of a number of important people. I would like to take the time to acknowledge their contribution to this masters.

Firstly, I would like to thank my family for their continued support throughout this Masters. My mother and father, who's support was vital through out this masters. They helped keep me grounded during a number of crises by providing me with the perspective I needed. My sisters, Grace, Maria and Jenny for cup(s) of tea and keeping me informed about what you were going to wear to the wedding over Whatsapp. My grandparents; Granny for those late night chats, Coco for those amazing lunch time toasties, Papa and Granddad for their kindness and sense of humour. My dog Hudson, who made those early mornings much more tolerable.

I would like to thank my supervisor Luca Longo for taking the time out to meet with me, for having the patience to sit with me refining this thesis and for encouraging me when I lost faith. Sarah Jane Delany and Michael Collins for organising my experience in China. Without their help I wouldn't have met my great friends from China, Sweden and France or have the amazing memories I have now. I would also like to thank the other lecturers at DIT, from who I learned a lot.

I would like to thank my classmates, particularly from my Chinese classes at DIT, you guys were so much fun. All the dirt monkeys of the DIT mountaineering society for providing a much need escape. All the lads and lassies that I'm overdue a pint with. I would particularly like to acknowledge the work of Brain Dwyer, my PIC, for taking over my Web Development work for the last couple of weeks (enjoy Canada buddy).

Contents

Declaration of Authorship	i
Abstract	iii
Acknowledgements	iv
Contents	v
List of Figures	viii
List of Tables	x
Abbreviations	xi
1 Introduction	1
1.1 Background	2
1.2 Research Project	4
1.3 Research Objectives	5
1.4 Research Methodology	5
1.5 Scope and Limitations	6
1.6 Document Outline	6
2 Literature Review	8
2.1 Knowledge-Based Approaches	10
2.1.1 Expert Systems	12
2.1.2 Fuzzy Logic	12
2.1.3 Reasoning Systems	13
2.1.3.1 Case-Based Reasoning	13
2.1.3.2 Monotonic Reasoning	13
2.1.3.3 Non-Monotonic Reasoning	14
2.1.4 Defeasible Reasoning (Longo and Dondio (2014))	14
2.1.5 Argumentation Frameworks (Longo and Hederman (2013))	15

2.1.6	Construct Modeling	26
2.1.7	Evaluation of Knowledge-based Techniques	29
2.1.7.1	Concurrent Validity	30
2.1.7.2	Convergent Validity	30
2.2	Learning Based Approaches	31
2.2.1	Supervised Machine Learning	31
2.2.1.1	Naive-Bayes	32
2.2.1.2	Bayesian Networks	32
2.2.1.3	Decision Tables	33
2.2.1.4	K Star	34
2.2.1.5	Linear Regression	34
2.2.1.6	Logistic Regression	35
2.2.1.7	Artificial Neural Networks	35
2.2.2	Unsupervised Machine Learning	37
2.2.3	Evaluation of Machine Learning Techniques	37
2.2.3.1	Numeric Prediction Problems	38
2.2.3.2	Classification Problems	39
2.2.3.3	Minimum Description Length	41
2.3	Discussion	42
3	Design	45
3.1	Choice of Construct Longo and Kane (2011) Longo et al. (2012b)	46
3.2	Machine Learning Software	48
3.3	Defeasible Reasoning Software Design	48
3.3.1	Use cases	49
3.3.2	Defeasible Knowledge Base	49
3.3.3	Membership Functions	52
3.3.4	Additional Software Requirements	54
3.4	Experiment Procedure	55
3.5	Conclusions	57
4	Implementation	59
4.1	Introduction	59
4.2	Defeasible Reasoning Software Implementation	59
4.2.1	System Architecture	60
4.2.2	Application Front End	62
4.2.3	Application Back-End Implementation	69
4.3	Experiment Implementation	73
4.4	Conclusions	77
5	Evaluation of Results and Discussion	78
5.1	Results	78
5.1.1	Predictive Capacity - Numeric	78
5.1.2	Predictive Capacity - Task Membership	81
5.1.3	Concurrent Validity	83

5.1.4	Convergent Validity	83
5.2	Summary and Final Recommendations	84
5.2.1	First Hypothesis	84
5.2.2	Second Hypothesis	85
5.2.3	Third Hypothesis	85
5.2.4	Recommendations	86
6	Conclusions and Future Work	88
6.1	Problem Definition and Research Overview	88
6.2	Experimentation, Evaluation and Limitations	89
6.3	Contributions to the body of knowledge	91
6.4	Future Work and Research	91
A	Details of Experiment Data	93
	Bibliography	97

List of Figures

2.1	Chapter Overview - Top down approach (Blue boxes contain notions used in this thesis)	9
2.2	Example of an argumentation framework	17
2.3	A is reinstated since C attacks B	17
2.4	An example AF taken from Baroni et al. (2011)	18
2.5	Argument diagramming with Araucaria	20
2.6	The Aspartix web interface	22
2.7	Performance benchmarks taken from Bistarelli et al.	23
2.8	Performance benchmarks taken from Bistarelli et al. (2013)	24
2.9	A Carnedes argument graph showing more complex relations than considered in a Dung-style AF	25
2.10	An example of a Bayesian Network from Witten and Frank (2005)	33
2.11	Decision table example taken from Hoffer (1999)	34
2.12	An example of a multilayer perceptron taken from Shiffman et al. (2012)	36
3.1	Chapter Overview	46
3.2	A Mockup of the UI for entering an Argumentation Framework	51
3.3	Example membership functions and output function	54
3.4	A Mockup of the UI for entering an Argumentation Framework	55
4.1	Application Architecture	63
4.2	Examples of different attacks drawn using the interface	65
4.3	An interface that allows a user to ‘draw’ a membership function using Bezier curves	66
4.4	Examples of Bezier curves that can be drawn that are not functions	68
4.5	The fully developed tool for eliciting knowledge bases	68
4.6	A selection of data for the user to test their knowledge base with	68
4.7	A list of options for semantics that can be computed by the system	69
4.8	Results of computing the semantics on the framework. (Undefined values belong to mitigating arguments)	70
4.9	The Argumentation Framework developed by the expert using the tool	74
4.10	The Argumentation Framework developed by the non-expert using the tool	75
5.1	Mean Absolute Error for Time Prediction	80
5.2	Classified Instances - Task ID	81

5.3	Precision, Recall and F-Score	82
5.4	Area Under ROC Curve	82

List of Tables

2.1	The table of probabilities associated with the temperature node in figure 2.10	33
3.1	Membership function example results	54
4.1	Documentation of the Application REST routes	69
5.1	Mean Absolute Error Using K-Fold Cross Validation	80
5.2	Mean Absolute Error Varying Percentage of Split	80
5.3	Concurrent Validity - Pearson Correlation with Time	83
5.4	Pearson correlation of measures of MWL	84
A.1	Sample data: Columns 1 - 5	93
A.2	Sample data: Columns 6 - 10	93
A.3	Sample data: Columns 11 - 15	93
A.4	Sample data: Columns 16 - 20	94
A.5	Sample data: Columns 20 - 25	94
A.6	Explanation of data columns 1 - 14	95
A.7	Explanation of data columns 15 - 25	96

Abbreviations

RQ	Research Question
DSS	Decision Support Systems
AT	Argumentation Theory
ML	Machine Learning
DR	Defeasible Reasoning
MWL	Mental WorkLoad
ANN	Artificial NeuralNetwork

*For my dog Hudson. Without whom this assignment would
have been completed in half the time with half the fun.*

Chapter 1

Introduction

Modern organisations solve difficult problems that require complex decision making and reasoning involving partial and often conflicting information. Examples of such problems are found in many different contexts: in management at large companies, in medicine, in the command of military units or in investment banking. In companies, management must make decisions about next years products without knowing what products their competitors will release and without a full insight into why their current products are performing the way they are. Doctors may have to treat unconscious patients in emergency situations. Without access to a patient's medical records or being able to talk to them a doctor cannot access critical information about patient; for example, what medication the patient may currently be taking which could interfere with certain treatments. Without this crucial information a doctor must make assumptions, apply a reasoning process and prescribe a treatment based solely on what she can observe, with incomplete information. In the context of modern warfare, military strikes may put the lives of civilians at risk. The decision to attack is a difficult one made by military commanders. It is made more difficult in the case of guerrilla warfare, in which information is often conflicting as enemy combatants are embedded amongst ordinary citizens.

The rapid evolution of technology is increasing the amount of data that can be gathered about these problems. Product sentiment information can be gathered from social media; improved patient monitoring gives doctors better insight into a patient's condition;

government agencies gather information from emails and phone calls. The abundance of information associated with these problems make unaided decision making difficult and reasoning a non-trivial task. Decisions Support Systems (DSS) attempt to aid decision makers in solving these problems by presenting information in a manner that makes it easier to reason about. [Turban et al. \(2005\)](#) explain that the central purpose of Decision Support Systems is to support and improve decision making. Many Decision Support Systems make use of intelligent components to provide an improved understanding of the problem at hand to decision makers. *The purpose of this research study is to investigate two of the approaches used in the design of these intelligent components; learning based approaches and knowledge based ones.*

Learning based approaches modify their underlying models for data based on experience. Supervised machine learning makes predictions based on a labeled training set of data. Machine learning techniques can provide reasonable predictions based on data, however, they often fall short in presenting their process to the user. As machine learning is automatic it is unsuitable for the representation of complex constructs such as an expert's knowledge.

Knowledge based approaches focus on the acquisition and representation of knowledge as well as the modeling of processes of reasoning about that knowledge. Defeasible reasoning is one such approach. An expert in a field typically doesn't look at some variables and apply a function to them. She considers the information at hand and reasons about it in the context of what she already knows. Within her reasoning process she may make assumptions that get changed as a result of the reasoning process. This reasoning process is better reflected in Defeasible Reasoning which offers a solution to the short comings of the automatic process of ML.

1.1 Background

The process by which humans reason has typically been examined in the fields of Philosophy and Psychology but has been further investigated by the field of computer science in the last 30 years or so. This investigation began in order to develop "expert

systems”: systems that attempt to model and elicit the reasoning of an expert in a computer. In an expert system a knowledge engineer typically uses a number of knowledge elicitation techniques to compile expertise in a particular domain. The knowledge engineer inputs this knowledge into the system in the form of rule sets. These rule sets can be used by an inference engine to produce useful output based on input data.

Expert Systems have evolved in the last few decades with computer scientists looking to Psychology and Philosophy to develop a greater understanding of how reasoning works. Several mechanisms by which humans reason have identified.

On one hand is *deductive reasoning*; reasoning in which conclusions logically follow from a set of premises falls into a category known as *monotonic reasoning* (Baroni et al. (1997)). In monotonic reasoning proof of the conclusions is embedded in the premises and the conclusions remain true in the presence of new evidence.

On the other hand is *defeasible reasoning* (DR), defined by Pollock (1987) as reasoning in which “*the premises taken by themselves may justify accepting the conclusion, but when additional information is added, that conclusion is no longer justified.*” *Non-monotonic* reasoning is more suitable for modelling human reasoning, which is non-monotonic, and for implementation in DSS.

As a result of recent advances, it is possible to model defeasible reasoning using argumentation theory, a computational technique to model non-monotonic reasoning. Argumentation theory allows us to model arguments and the interactions between them. An overview of this technique is given in Chapter 2.

Knowledge based argumentation systems have many other characteristics that make their use in decision support systems advantageous. Argumentation systems can implement reasoning based on incomplete or corrupt data as well as conflicting information and explain how a reasoning process arrived at a conclusion. Through the process of visualising a knowledge base using an argumentation tool an expert may gain insight into his/her own ideas and refine the rules of his/her reasoning process.

Knowledge based approaches accrue their insights and advantages by having a human expert interacting with the system. These benefits are less prevalent in machine learning based systems.

On one hand, ML provides us with an approach that is automatable and can provide insights into data that it is unlikely any human would find sifting through large amounts of data by hand. ML is a better approach to solving classification problems such as natural language processing as well as image and handwriting recognition; tasks where it is difficult for a human to encode their understanding in a way that a machine can process. It is suitable for automatically extracting knowledge and rules from complex and unknown data.

On the other hand, ML is not always suited to modelling constructs, as human expertise is not accounted often for. Knowledge based approaches are better suited to representing ill defined concepts, constructs such as intelligence, mental workload and personality, that are difficult to measure and assess.

1.2 Research Project

This research project aims to provide a comparison between an implementation of DR (via argumentation theory) and a type of machine learning (supervised ML). The aim is to outline the strengths of both techniques and provide guidelines on what problems a particular approach may be suited for.

The comparison will build on previous research by evaluating the ability of the approaches to model a construct, to measure and assess it and determine the capacity of these assessments to be used in prediction tasks.

The research question (RQ) is formally defined:

“To what extent can an implementation of Defeasible Reasoning enhance the representation of a construct (mental workload) and support inference in comparison with Machine Learning?”

1.3 Research Objectives

The objective of this project is to provide an overview of the differences and similarities in learning based and knowledge based approaches and to investigate their strengths and weaknesses, in relation to *construct modelling*, assessment and prediction capacity.

The goals of the research project are outlined as follows:

1. To gain insight about knowledge-based inference approaches and learning based inference methods. The focus will be on defeasible reasoning, argumentation theory and supervised machine learning in relation to construct modelling.
2. To design a computable model of DR for construct representation as well as a structured experiment.
3. To implement the designed computational model as software in order to execute the experiments.
4. To evaluate the designed model.

1.4 Research Methodology

This project will use a mixed research methodology - a combination of both qualitative and quantitative methods. The study begins with secondary research (literature review) to develop gather understanding that supports the primary research (the design and implementation of an experiment).

1. A literature review is carried out to outline the strengths and weaknesses of the techniques mentioned in objective 1.
2. The output of the literature review is used to inform the theoretical solution and design of an experiment to tackle objective 2.

3. The theoretical solution will be implemented as a piece of software employing state-of-the-art technologies in the field of web development. This will be used to perform the designed experiments and accomplish objective 3.
4. The output of this experiment is quantitatively analysed using evaluation metrics as emerged from the literature review. Statistical methods are used on the results produced in the experiment to empirically demonstrate the predictive capacity of the designed model of DR against the one of selected supervised machine learning classifiers.

1.5 Scope and Limitations

- The project scope is that of a single construct. Mental Workload has been modeled as a defeasible construct in previous work by Dr. Longo who has provided his knowledge base for the purpose of the experiment.
- Further constructs could not be adopted due to limited available time for research.
- Similarly, although planned, it was not possible to interview other experts (in MWL) and translate their expertise in computable terms.
- The project is limited by the size of the data set that has been used and provided by Dr. Longo.
- The implemented software has not been fully documented according to best practices because this task is beyond the scope of the RQ.

1.6 Document Outline

This thesis is organised into the following sections:

-
- Chapter 2 provides a review of the literature relevant to defeasible reasoning, argumentation theory and machine learning including cutting-edge research in the area.
 - Chapter 3 outlines the design of the DR implementation and the experiment and the justification of those design choices in the context of the research methodology
 - Chapter 4 details the implementation of the software and the experiments; the implementation and deployment of relevant software artifacts, their use and the overall execution of the experiments are described as well.
 - Chapter 5 presents and critically discusses the results of the experiments highlighting the main findings, strengths and limitations.
 - Chapter 6 summarises the thesis, highlighting its main contribution to the body of knowledge. Future work and recommendations are suggested.

Chapter 2

Literature Review

This section introduces fields of study under investigation in this project, Machine Learning and Defeasible Reasoning. The existing literature on both areas is reviewed in order to provide the reader with an understanding of the context in which the research occurs. As the project is fundamentally a comparison between the two fields this literature review also gathers results and conclusions from other work that will help to inform this comparison.

This material will be used to support the arguments made for both approaches and will also inform the design of an experiment evaluating the approaches. At a high level this project examines two differing approaches to artificial intelligence, knowledge-base approaches and learning based approaches (machine learning).

[Mitchell \(2006\)](#) defines machine learning as a field of computer science that attempts to solve the question:

“How can we build computer systems that automatically improve with experience, and what are the fundamental laws that govern all learning processes?”

On a practical level it is an area of study that concerns itself with the design of techniques and algorithms that run uniquely on different data to achieve some aim without being explicitly programmed. A machine learning program initial takes a data as input which it learns from; this learning is then used on future inputs to make a prediction

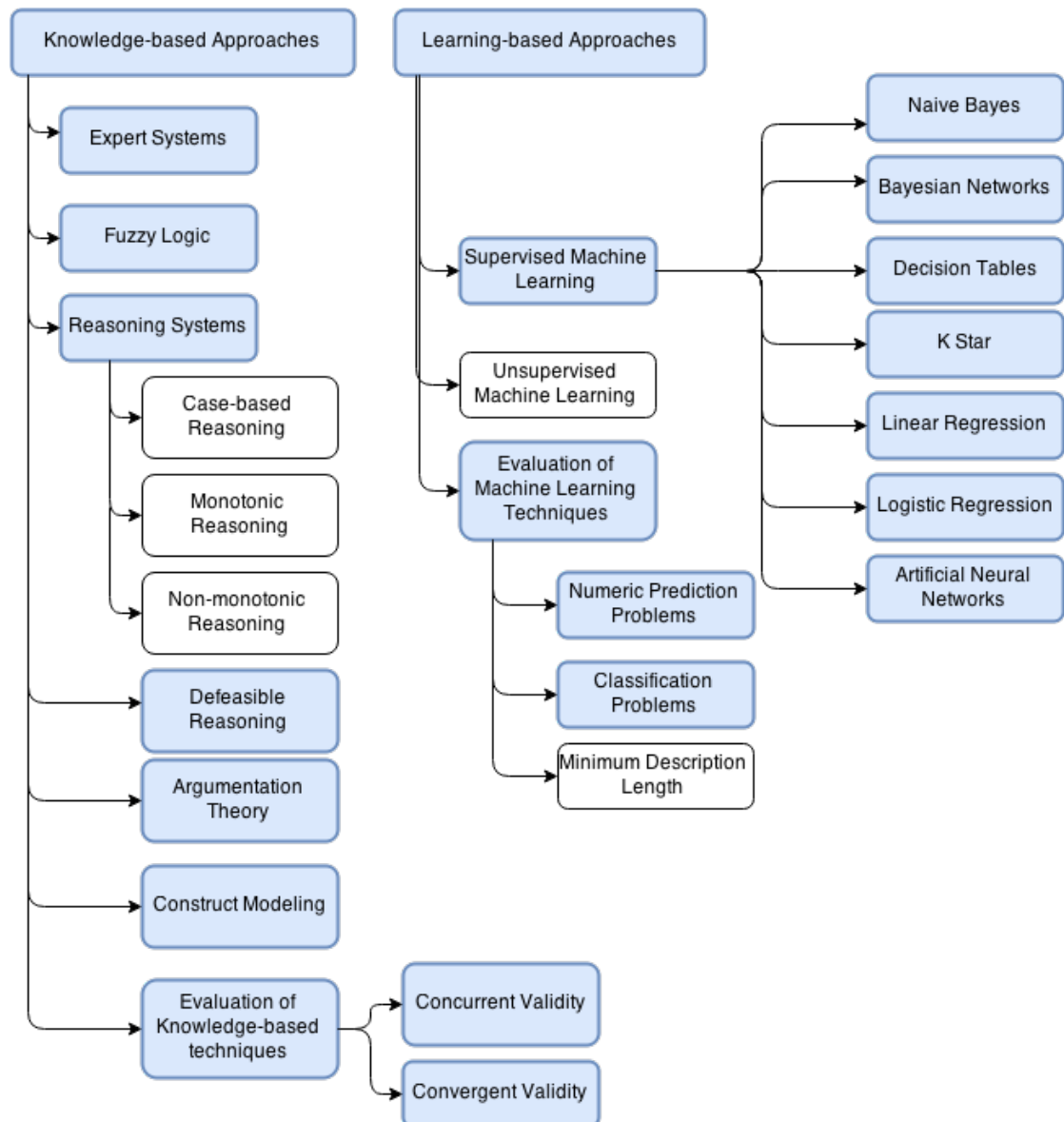


FIGURE 2.1: Chapter Overview - Top down approach (Blue boxes contain notions used in this thesis)

or to provide some understanding. Mitchell outlines that while there had been no successful commercial applications of machine learning as late as 1985, it has since been successfully applied in diverse fields such as speech recognition, computer vision, bio-surveillance, robot control and accelerating empirical sciences. Indeed in the last several years some of the biggest names in technology such as IBM, Google, Microsoft and Baidu have been making great strides in the development of advanced machine learning techniques and reaping the rewards.

Knowledge-base approaches differ from learning approaches in that instead of learning

from labeled data, the software makes decisions based on 'knowledge'. The software is not explicitly programmed to perform operations on the data; a knowledge based approach takes a knowledge base and uses its contents to make inferences based on data. The most common example of a knowledge based approach is in expert systems where the knowledge base is that of an expert. The knowledge used in KB systems can have many forms such as natural language text, mathematical functions and raw data. Some knowledge based systems may infact learn from previous experience, for example, in CASE-based reasoning in which solutions for new cases may be added to the repository of cases used for inference.

2.1 Knowledge-Based Approaches

[Akerkar and Sajja \(2010\)](#) describe knowledge based systems as distinct from traditional information systems as they have an understanding of the data that they process. This understanding comes from a knowledge base made up of data, information and knowledge. Knowledge-base systems tend to be made up of a knowledge base and an inference engine, a program that can infer outcomes from the knowledge base given some input. Knowledge-base systems include expert systems and CASE-based reasoning systems.

Key issues in knowledge based systems include the acquisition of knowledge, the representation of knowledge and reasoning about that knowledge.

In order to develop expert systems a knowledge engineer must gather domain expertise. This knowledge is elicited in a number of ways. Typically the knowledge engineer is not an expert in the domain and will gather the knowledge from books, journals and other sources. It may be more efficient for the knowledge engineer to elicit a knowledge base directly from an expert by interviewing them. Although this approach is preferable it is not always possible as the expert is busy with other work and unavailable for the long periods of time required to build a knowledge base. According to

[Sagheb-Tehrani \(2009\)](#) in the early 1990's the most popular means of eliciting knowledge from an expert was the structured interview; other techniques include unstructured interviews, documentation and case study analysis, simulations and observation.

[Davis et al. \(1993\)](#) identifies five fundamental roles of knowledge representation. A knowledge representation is a surrogate; an imperfect stand in for something in the real world. Since it is imperfect, inferences made from it won't always be correct. The author discusses knowledge representation as an 'ontological commitment'; by choosing one knowledge representation strategy over another we become bound to it and the ontology will have an influence on how reality is interpreted. A knowledge representation is a 'fragmentary theory of intelligent reasoning'; the choice of knowledge representation influences what can be inferred from it and is opinionated with regards to what it means to reason intelligently. A knowledge representation is a medium for efficient computation; it doesn't gather every nuance of reality, if it did the problems it is attempting to solve would be computationally intractable. Lastly the author explains that a KR is a medium of human expression; like programming languages, KRs are used for communicating not just with the system but with other knowledge engineers as well.

[Petrik](#) provides descriptions of a number of knowledge representation strategies employed in expert systems examples of which include semantic networks, frames, logic, bayes networks and influence diagrams.

Rules are a form of KR in which the knowledge is encoded using if-then clauses activated with some heuristic function. Rules can be examined through the lens of the five roles proposed by [Davis et al. \(1993\)](#). As a surrogate rules are poor at modeling complex relationships. By choosing to use rules for KR in a system the ontology will be overly simplified. its theory of reasoning is that everything can be represented as an action to take in a given situation. Thanks to fast Rete matching algorithms it is an effective medium for computation for simple problems. As a medium of human expression it is desirable; as rules can be formulated in natural language, they are easy to understand and to create.

2.1.1 Expert Systems

[Todd \(1992\)](#) describes an expert system as one in which an inference mechanism is applied to the knowledge of an expert. Expert systems typically are employed in decision support systems in which they may provide assistance in tasks such as diagnosis. Expert systems typically have to reason with imprecise and incomplete information and various means have been adopted to deal with this information in expert systems. [Kandel \(1991\)](#) explains how two implementations CASNET and MYCIN handle these uncertainties; the later uses certainty factors while the former uses the most significant results of tests. Expert systems are limited by their ability to codify knowledge. Different knowledge bases vary in structure with more abstract knowledge bases typically more difficult to codify. They also often have trouble balancing knowledge bases involving conflicting goals. ([Cowan \(2001\)](#))

2.1.2 Fuzzy Logic

[Kandel \(1991\)](#) goes on to describe how fuzzy logic like that proposed by [Zadeh \(1965\)](#) has been utilised in expert systems to deal with uncertainty. In a nutshell fuzzy sets attempt to define the membership of a set by quantifying it. Membership is defined using membership functions which take an instance as input and quantify its membership as a real number in the range zero and one. If the output of the membership function is zero then the instance is not a member of the set; on the other hand if it is greater than zero then the input is a member of the set with some degree of truth (maximum 1). The values in between zero and one define the degree to which the instance is a member of the set. The strength of fuzzy logic is that it enables the modeling of vaguely defined rules (premises) but it is limited in its ability to model the interaction and relationships between those rules.

2.1.3 Reasoning Systems

Once knowledge has been acquired and represented it is reasoned about using an inference engine. Reasoning involves using some known knowledge to deduce logical consequences. [Singh and Karwayun \(2010\)](#) provides a comparison between different inference engines. Jess utilises rule based inference to make inferences. ([O'Connor and Das \(2012\)](#)) Hoolet translates its ontology to a collection of axioms which are reasoned about using first order logic. ([Bechhofer and Horrocks](#)) Pellet uses probabilistic means of inference. ([Parsia and Sirin \(2004\)](#)) All of these different engines attempt to make inferences based on imperfect data. In the last 20 years or so interest has increased in using defeasible reasoning and argumentation theory to make these inferences in the absences of consistent data. AT provides a model of reasoning that is both intuitive and computable.

2.1.3.1 Case-Based Reasoning

According to [Leake \(2003\)](#), the knowledge base of a case-based reasoning system consists of knowledge obtained in the past from solving problems (previous cases). When the system encounters a new situation not accounted for by the knowledge base, cases that may be relevant and useful are brought up and adapted to solve the new problem. The resulting new solution is then saved to the knowledge base for future use. Through the process of solving new problems the system effectively learns. CBR has the advantage of not requiring new solutions to be generated when familiar problems are encountered. CBR has applications in tasks such as interpreting the law, informing design and diagnosis.

2.1.3.2 Monotonic Reasoning

[Baroni et al. \(1997\)](#) explains that *monotonic* reasoning can be defined as follows: Given three sets A, B, C , if $A \vdash B$ then also $A \cup C \vdash B$. Informally, monotonic reasoning can be thought of a reasoning in which the conclusions for an argument are embedded in the premises and are not retracted in the light of new evidence. The truth that results from

one statement cannot be retracted as a result of new evidence in monotonic reasoning. Take the example “Tweety is a bird, birds can fly, therefore Tweety can fly.” Tweety being able to fly is inferred by default since Tweety is a bird. Premise A, that Tweety is a bird and premise B, that birds fly results in the conclusion that Tweety can fly. No matter what new evidence is discovered about Tweety or about birds we will continue to believe that Tweety can fly.

2.1.3.3 Non-Monotonic Reasoning

As a result of the limitations found in monotonic reasoning, researchers have formulated non-monotonic reasoning to better model human reasoning. In non-monotonic reasoning conclusions can be retracted in light of new evidence.([Baroni et al. \(1997\)](#)) In our day to day activities humans make assumptions based on past experience. If a person lives in a part of the world that often has sunny weather then they assume this to be true by default and generally behave as if it is going to be that way. If the weather forecast warns them of heavy rain they will retract their conclusion that today will be sunny and bring an umbrella. The logic of making assumptions is known as default logic. Default logic allows us to make assumptions with incomplete evidence based on what we believe to typically be the case. Non-monotonic reasoning allows us to override these defaults when presented with new information.

Referring to the example in the previous section, we discover that there is infact a subset of birds that do not fly. As mentioned, in monotonic reasoning this has no effect since we still believe that all birds fly. However with non-monotonic reasoning, we can revise our believe so, if Tweety is a penguin, we believe that Tweety cannot fly. Our new belief can be that “Birds fly, unless they are penguins”.

2.1.4 Defeasible Reasoning ([Longo and Dondio \(2014\)](#))

Argumentation theory has its roots in philosophy and is concerned with how issues are discussed and conclusions arrived at in the context of incomplete and conflicting

evidence. The ability to deal with inconsistent information through reasoning is what has motivated its use in AI.

[Reiter \(1980\)](#) recognised the need to make assumptions when presented with incomplete evidence and to change these assumptions when presented with new evidence. Reiter recognised that classical logic is insufficient for dealing with these situations and proposed a logic for default reasoning. Default reasoning is a formalisation of what we believe to be true in the absence of other evidence that makes the case exceptional; what we believe to be true by default.

Default logic is a non-monotonic logic. Reiter describes first order logic as monotonic - conclusions drawn in the presence of information A remain valid in the presence of new information B. Non-monotonic reasoning provides a mechanism for revising old beliefs in the presence of new information. Default logic takes into account that conclusions drawn in the case of A may not be true in the case of B. In first order logic we would still believe that Tweety can fly even though it is a penguin. Default logic allows us to revise our belief about what can fly.

[Pollock \(1987\)](#) recognised that while non-monotonic logic in AI is similar to how humans reason, it was also falling short in its recognition of the complexities of reasoning. Pollock introduced the concept of defeasible reasoning, similar to nonmonotonic reasoning, to AI to better model these complexities. Reasoning can be said to be defeasible if the premises taken in isolation can infer a conclusion, but that this conclusion can be defeated when additional information is added. An important element of Pollock's defeasible reasoning is the idea of warrant; he defined a proposition as being warranted if it would be believed by an ideal reasoner. The conditions that determine whether an argument is warranted are made explicit by his work and include notions such as defeat relations between arguments.

2.1.5 Argumentation Frameworks ([Longo and Hederman \(2013\)](#))

Arguably the most important work in the field is that of [Dung \(1995\)](#). Dung described his work as provided a bridge from "argumentation theory as a supporting analytic

tool for non-monotonic reasoning and the independent exploitation of argumentation models in wider AI contexts”. The work was concerned with modelling the fundamental mechanism humans use to argue so as to implement this model on computers. He summarised the basis for his work in the old saying “the one who has the last word laughs best.” In other words, in human typical human argumentation the last piece of evidence to be produced can nullify evidence produced earlier by opposing arguments winning the argument.

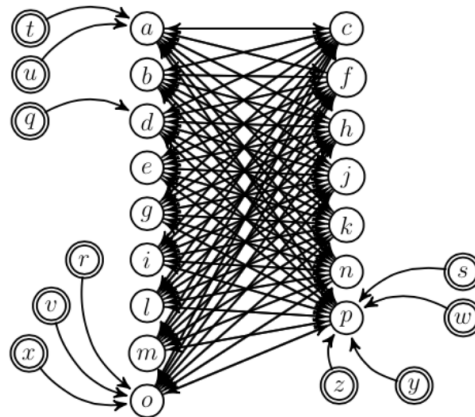
An objection to the argumentation approach produced by Dung is that the source of the information comes from one perceived rational entity. Argumentation naturally involves multiple rational agents; not one as in Dung’s framework. Ideas are exchanged and discussed. In Dung’s framework notions of fallacy are embedded in the defeat relations of the framework. This fails to take into account wider issues of fallacy.

Dung’s work can be divided into two important contributions. The first is the reduction of argumentation into a simplified abstract model, the Argumentation Framework. The second contribution is the techniques necessary to reduce an argumentation framework to a set of justified arguments. These two innovations have had a considerable influence of research in the area and are the fundamental basis for the defeasible reasoning implementation used in this project. What follows is a description of these ideas as previously explained in [Dung \(1995\)](#)’s work.

In a typical debate a participant may put forward an argument which is considered to be valid based on its premises. An opponent may put forward another argument that invalidates what the first participant has just stated. This cycle will continue throughout the debate and what we are left with is a collection of arguments and ‘attack’ relationships between those arguments. Dung modelled this interaction mathematically as a directed graph; with nodes representing the arguments and edges representing the attack relations between the arguments. This model of arguments and attack relations is known as an argumentation framework. The advantage of this representation is that it is relatively intuitive in comparison to other models of defeasible reasoning proposed previously and also relatively straightforward to implement in computers.

Formally, an argumentation framework AF can be defined as a pair $\langle AR, attacks \rangle$ where AR is the set of arguments and $attacks$, the set of attack relations between those arguments, $R \subseteq A \times A$. An attack, a attacks b , is defined $(a, b) \in R$. An example of an argumentation framework can be seen in figure 2.2.

FIGURE 2.2: Example of an argumentation framework



Once arguments have been formulated in an AF it must be determined which arguments are admissible. Figure 2.3 shows an typical interaction. Within an argumentation framework an argument A can become inadmissible if it is attacked by another argument B . However, if B is attacked by C and becomes inadmissible then A may be reinstated.

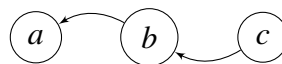


FIGURE 2.3: A is reinstated since C attacks B

As many arguments within an AF interact in this way, determining which arguments are admissible is difficult. Dungs solution is to use extension semantics to determine these arguments. These semantics contain conditions that a subset of arguments in an AF must satisfy in order to be collectively justified. Different semantics apply varying levels of strictness to the interpretation of what arguments are justified.

Semantics are based on a number of definitions given by Dung. A set of arguments is considered to be conflict free if no argument in the set attacks another argument in the set. A formal method to identify how conflicts are resolved is known as a semantic. (Baroni et al. (2011)) The literature defines two approaches to computing semantics: the labeling approach and the semantic approach. In a labeling approach arguments may

be defined as *out*(if they are attacked), *in* (if they receive no attacks or if the arguments attacking them are labeled out) or *undecided* (if a resolution cannot be immediately found). In the extension based approach the strategy is to look at groups of arguments that can win the conflict as opposed to individual arguments in the labeling approach. An argument A is acceptable with respect to the set S if every argument that attacks A is attacked by an argument in S . A set is considered admissible if every argument in it is acceptable with respect to the AF. From these definitions Dung defined the *preferred extension* as the maximum admissible set of arguments in the AF. A *stable extension* is defined as a conflict-free set of arguments that attacks every argument not in the set. The *grounded extension* of an AF is the smallest complete extension that exists within the set of admissible arguments. Each of the semantics interprets the AF in a different way, the preferred extension is more inclusive while the grounded extension is more skeptical.

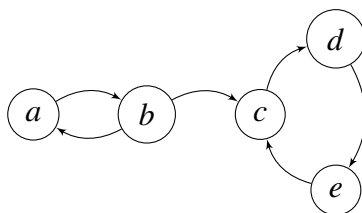


FIGURE 2.4: An example AF taken from [Baroni et al. \(2011\)](#)

In figure 2.4 the grounded extension is empty. There are two preferred extensions the set $\{a\}$ and the set $\{b, d\}$. The stable extension is $\{b, d\}$.

The implementation of a system based on defeasible reasoning and argumentation theory is a central focus of this project. This section has briefly outlined DR and AT at a high level for the reader. For a more detailed background in AT the reader is referred to [Bench-Capon and Dunne \(2007\)](#).

Implementations of argumentation theory based systems tend to fall into two broad categories. One branch takes advantage of the computational power of the techniques to solve problems in medicine, law and online behaviour. The other implementations focus on creating GUI tools in order to improve the reasoning process of their users.

Argument diagramming tools leverage visualisation techniques to aid users in reasoning about arguments. This has practical applications in several fields.

[Twardy \(2004\)](#) demonstrated the advantages of computer based Argument mapping systems in improving student critical thinking. Twardy measured improvement in student performance on the California Critical Thinking Skills Test across a semester. Students from an “introduction to critical thinking” class were divided into three groups, two groups receiving normal course tutorials, the third group using Reason!Able (argument mapping) software. Students who used argument mapping software scored results on average three times higher than their peers by the end of the semester. Twardy believes that the students’ critical thinking improved by using the software as it allowed them to distinguish between reasons and supporting premises.

In the same domain, [Reed and Rowe \(2001\)](#) designed Araucaria to make argument diagramming for undergraduates easier and also to support research activities. In addition, the authors developed AML (Argument Markup Language) an XML based syntax for describing the structure of arguments. They explain the strength of their software as its platform independence (it was developed in Java) and its interoperability with other tools. Araucaria represents arguments in a tree structure with the branches of this tree representing support relations. This is in contrast to Dung’s argumentation framework which uses edges to represent attack relations.

[Karacapilidis and Papadias \(2001\)](#) developed the HERMES system, an implementation of Argumentation Theory that allows users to collaboratively develop arguments online and support those arguments with data. HERMES also provides users with access to information from external databases to further justify their arguments. Arguments are represented using a labelling approach as opposed to a graph approach. Constraints are inserted into a discussion graph and when new constraints are introduced they are checked against existing ones.

The authors evaluated their system focusing on usability. A wide variety of users such as students, researchers and medical doctors were surveyed. The participants attempted to solve two problems collaboratively using the tool and then answered questions. These questions were both about the users overall opinion of the system (ease of use, enjoyable, intention to use again) and about effectiveness of the system (task clarity, easy to

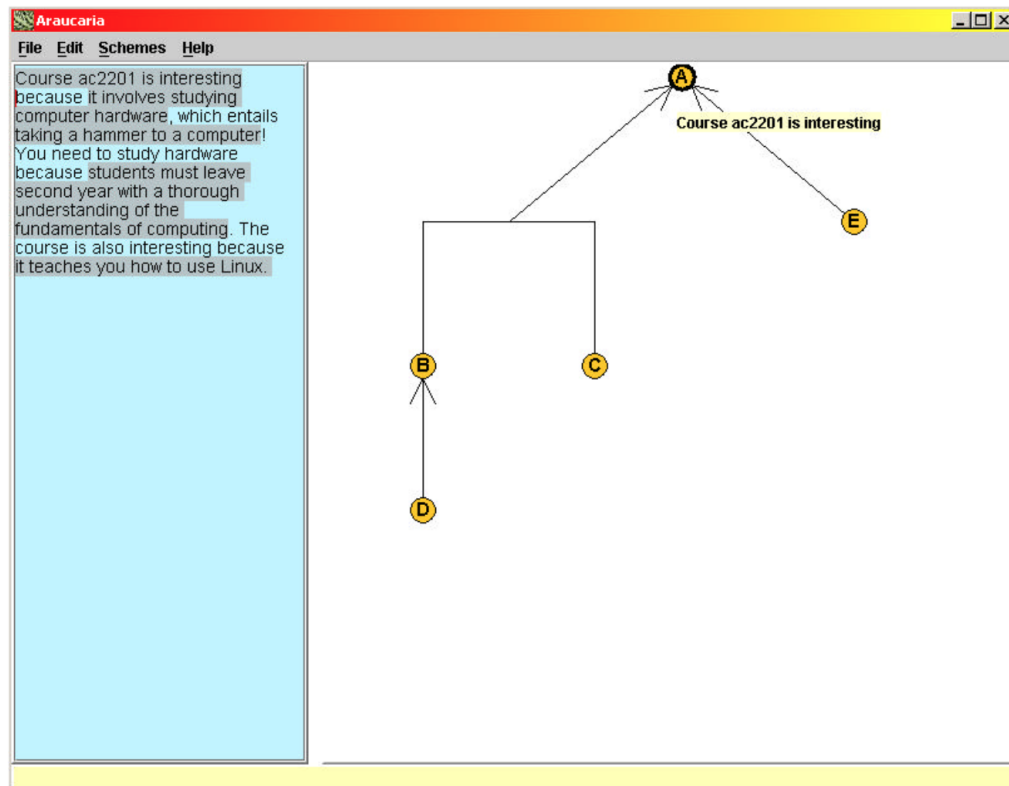


FIGURE 2.5: Argument diagramming with Araucaria

read, sufficiently informative). The tool focuses on collaborative decision making and not on automated output so there is no evaluation of task performance.

[Easterday et al. \(2009\)](#) surveyed many argument mapping tools, listed shortcomings common to the tools and developed requirements for argumentation tools from this list. The six requirements listed are correct visual representation, flexible construction, control over visual properties, automation of non-semantic operations, multiple diagrams for comparison and cross platform compatibility. As none of the available tools at the time satisfied all of these requirements, the authors developed a prototype, iLogos that would fulfill these requirements.

The fact that decision support systems allow users to aggregate evidence and make decisions based on that evidence lends makes them an obvious fit for medical practitioners. [Hunter and Williams \(2010\)](#) developed a framework for generating inference rules to argue for and against the benefits of medical treatments based on evidence. Their work

highlights the benefits of argumentation systems in abstracting away the complicated nature of medical evidence into a form more manageable for practitioners.

A review of defeasible reasoning implementations by [Bryant and Krause \(2008\)](#) highlight the need for well designed empirical evaluations of implementations and formal complexity analysis to justify the practical applicability of a reasoning engine. The paper also highlights the proprietary nature of successful argumentation theory based applications preventing researchers from peer reviewing the software.

[Dimopoulos et al.](#) provided one of the initial complexity analysis of the approach proposed by Dung. The authors found that while many believed that Dung's AF would simplify the implementation of non-monotonic logics, it turns out that the computational complexity of Dung's AF is greater than that of standard implementations of non-monotonic reasoning. The authors outline that in a particular worst case scenario, "Autoepistemic Logic" the computational complexity is two orders higher than in a standard implementation. In the discussion of their results the authors underline that for the computational complexity is only better in for sceptical reasoning and only in trivial cases that are almost equivalent to classical logic.

[Vreeswijk \(2006\)](#) opposed the authors approach of specifically targeting worst case scenarios. Vreeswijk takes a more pragmatic approach designing algorithms that produce grounded and admissible semantics for practical implementation in systems. The author presents average and best-case complexity as well as worst case complexity. He states that the admissible membership problem is NP-complete; for the worst case the complexity grows exponentially with the number of nodes. The algorithm developed by the author is an efficient and practical one and has been utilised in several implementations of AFs including the Dung-o-matic used in this project.

[Modgil and Prakken \(2014\)](#) presented a tutorial introduction to ASPIC+; a framework for specifying argumentation systems, rather than an implementation of a system. The authors claim that while Dungs calculus is indispensable, it provides little guidance for the development of a system based on his theories. ASPIC+ aims to provide guidance on how the constituent premises of an argument make up an attack. Within their framework arguments could be attacked in three ways: their uncertain premises, their

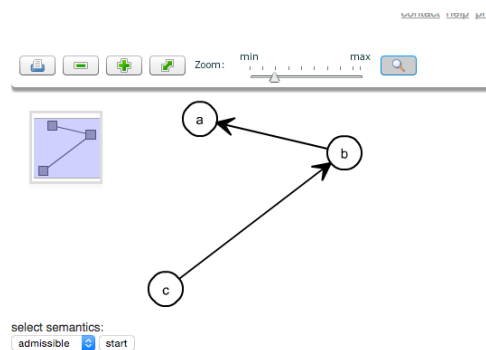
Specification of Input:

each argument "a" is defined via `arg(a)`,
and each attack from argument "a" to argument "b" is defined via `att(a,b)`.

```
arg(a).
arg(b).
arg(c).
att(b,a).
att(c,b).
```

load

(a) Input attacks and arguments



(b) Output graph and semantics

FIGURE 2.6: The Aspartix web interface

defeasible inferences or on the conclusions of their defeasible inferences. They suggest their approach as a best practice for the development of argumentation systems.

[Egly et al. \(2008\)](#) introduces Aspartix. Aspartix can compute the extensions that the authors consider to be most important from Dungs AF (admissible, preferred, stable, complete, and grounded). ASRARTIX uses answer set programming, a type of logic programming designed to solve intractable problems, in order to compute the semantics. The authors believe that implementing argumentation systems within this programming paradigm provides clarity not offered in other paradigms. It also offers a computational advantage since the computations that are initially intractable may be reduced into another language which already has efficient solutions implemented in it. In order to run Aspartix a computer must already have an answer set programming solver like Gringo installed on the system. Users also interact with a web hosted version of Aspartix¹.

Motivated by the desire to apply argumentation in multi-agent systems [Podlaskowski et al. \(2011\)](#) developed Argulab. Argulab is an attempt to provide a standard library of argumentation algorithms for use in multiple applications. The authors demonstrate the ability of Argulab to compute semantics, argument justification status, argument-based discussion and judgement aggregation.

Comparing the performance of programs to compute AF semantics is a challenging aspect of these new systems. [Cerutti et al.](#) explains that the lack of a large set of

¹<http://rull.dbai.tuwien.ac.at:8080/ASPARTIX/loadGraph.faces>

challenging AFs makes benchmarking the algorithms difficult and offers a solution in the form of AFBenchGen. AFBenchGen is a C++ program designed to generate random AFs for benchmarking. The program was able to efficiently generate large AFs with 5,000 nodes and 270,000 attacks. The authors explain that a more thorough test suite should contain AFs with different structures and determining how to generate these test cases is a subject for further study.

Two studies have built upon this work and benchmarked implementations designed to compute the semantics of Dung’s AFs. [Bistarelli et al.](#) provide a performance comparison of the computation of stable semantics in ASPARTIX, dynPARTIX, Dung-O-Matic, and ConArg2. The semantics were benchmarked across three different graph structures; the Erdos-Re nyi model, the Kleinberg small-world model, and the scale-free Barabasi-Albert model. By focusing on stable semantics they focus on one of the worst case scenarios as a stable semantic doesn’t always exist and its computation is NP-Hard. Out of all four Dung-o-matic performed least favourably, ASPARTIX performed well at solving Erdos-Renyi but overall ConArg2 was found to be the strongest implementation.

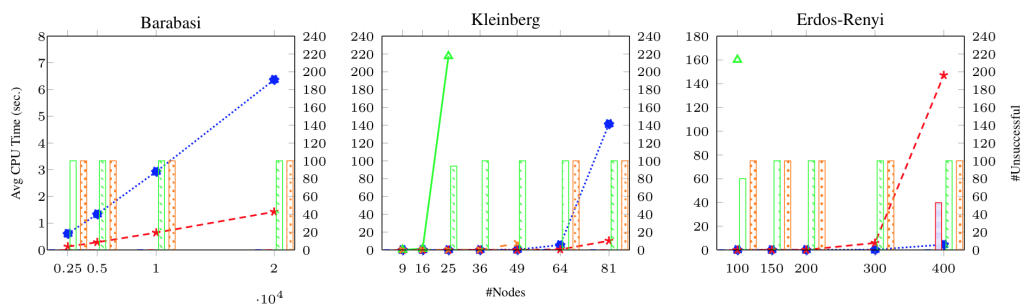


Figure 1. ASPARTIX ($\cdots\blacklozenge\cdots$), ConArg2 ($-*\cdot-$), Dung-O-Matic ($- \blacktriangle -$), dynPARTIX ($- \times -$), #Unsuccessful with ASPARTIX (\blacksquare), ConArg2 (\blacksquare), Dung-O-Matic (\blacksquare), dynPARTIX (\blacksquare).

FIGURE 2.7: Performance benchmarks taken from [Bistarelli et al.](#)

These findings are echoed in earlier work carried out by [Bistarelli et al. \(2013\)](#) that found ASPARTIX and ConArg to significantly out perform Dung-o-matic. Similarly in this work the authors attempted to compute the complete and stable extensions for the three random graphs given above and also for Watts-Strogatz graphs.

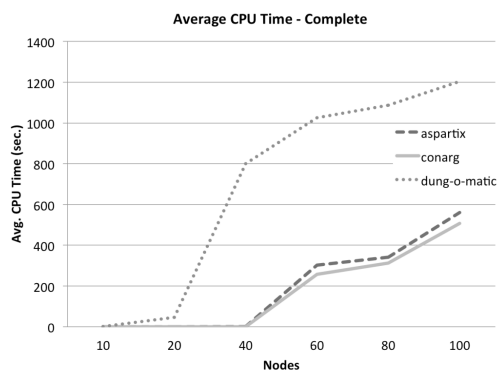


Fig. 2: Comparing complete extensions.

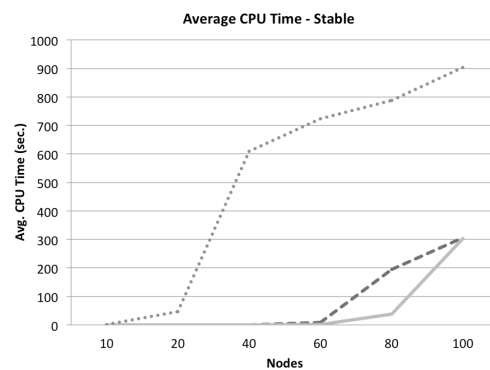


Fig. 3: Comparing stable extensions.

FIGURE 2.8: Performance benchmarks taken from [Bistarelli et al. \(2013\)](#)

An alternative to Dung's argumentation framework has been proposed by [Gordon et al. \(2007\)](#). Their model contrasts with Dung's AF by considering the internal structure of arguments in evaluating their feasibility. The model uses directed graphs to model the argumentation process as well, however, there are additional elements instead of simply arguments and attack relations. Nodes may be arguments or supporting information such as datum, claim, warrant, backing and exception. The relationships between nodes may be attack or support relations and are distinguished by using different arrow heads. An implementation of their framework exists known as Carneades.

[Van Gijzel and Nilsson \(2014\)](#) provide techniques for translating Carneades argumentation frameworks into Dung style AFs in order to use the more efficient implementations.

While the Carneades approach may be overly complex for practical purposes, it does underline the importance of considering the internal structure of arguments. Most of the implementations considered so far evaluate the semantics of an AF with respect to the framework as it is defined. However, whether arguments are relevant varies depending on the instance of data being considered.

A comparison between argumentation theory and machine learning was made by [Longo et al. \(2012a\)](#) in the domain of health informatics. The comparison specifically tackled the ability of both approaches to solve a classification problem; the recurrence of breast cancer. The authors developed a Dung style argumentation framework based on the knowledge base of a single health-care practitioner. The authors used this knowledge

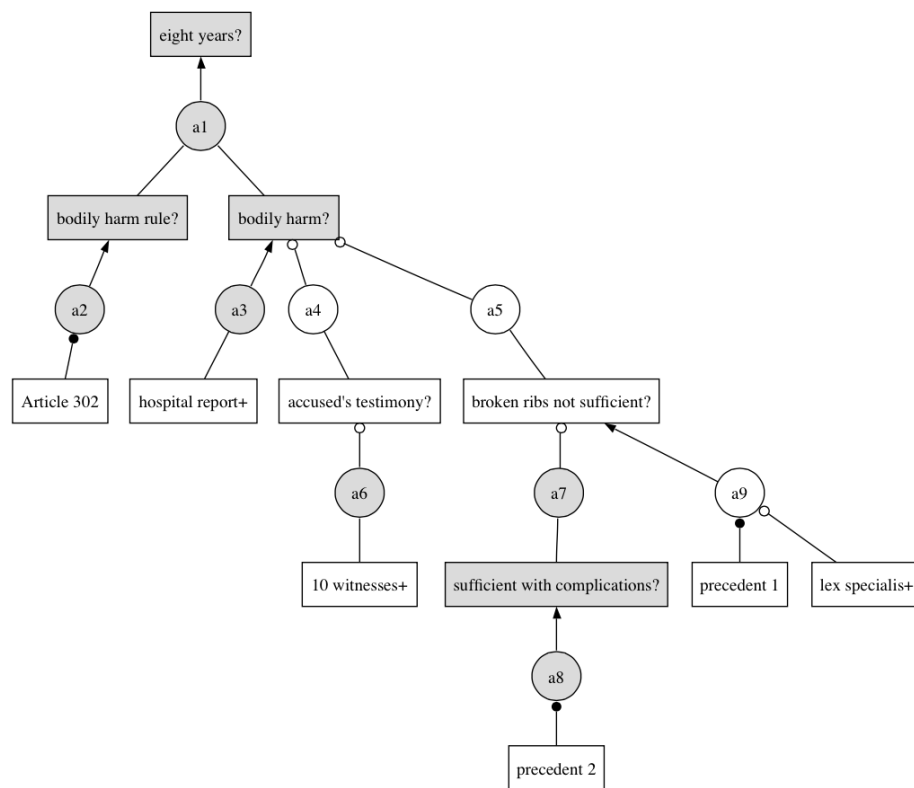


FIGURE 2.9: A Carnedes argument graph showing more complex relations than considered in a Dung-style AF

based to determine results for rows in a classic breast cancer data-set. In order to determine whether or not an argument in the framework was activated the authors used fuzzy membership functions. The authors then compared the results of this experiment with the predictive capacity of several machine learning algorithms (decision tables, bayesian networks, best-first and alternating decision tree, regression and multilayer perceptron) varying percentage of split and the number of folds. The results of their experiment showed that argumentation based systems could perform as well and in some cases better than machine learning algorithms. Moreover, as there was no training involved the approach was likely to perform equally well on a different data set, unlike the machine learning approaches.

Overall the advantages outlined by the authors highlight the ‘human’ element of the approach. AT is more intuitive and can provide users with greater understand of the problems they seek to solve. Experts can subjectively compare knowledge bases and

obtain explanations for results computed on the basis of their knowledge. The lack of the same human element in machine learning is what provides its advantages over AT; training is automatic and doesn't require a knowledge base to be elicited.

2.1.6 Construct Modeling

In order to answer the research question posed at the beginning of this dissertation the idea of a construct must be defined. [Price and Jhangiani \(2013\)](#) discusses measurement in the context of psychology. He explains that many variables are simple to determine such as height, age, sex. We can gather this information by simply observing it. By contrast, a construct can be considered to be as a non-physical thing that cannot be simply measured.²

Constructs are ideas people have about things, they may be idealised or may not really exist. Constructs are useful in science for improving our understanding of difference phenomena and techniques are often established in an attempt to quantify them. Examples of such constructs include intelligence (measured using IQ tests), personality (measured using surveys) and evolution (measured using changing characteristics over generations).

Representing constructs in a computable manner offers many potential benefits. By automating the representation of constructs we can free up the time of experts, for example, by automating diagnosis. We can reduce human error, for example in the case of reasoning. We can make predictions about certain situations and generally improve our understanding of phenomena.

Developing systems that can represent constructs accurately is not trivial. There are a number of underlying issues that make solving this problem hard. Evaluating the performance of such systems is challenging as we have no external measurement to compare the construct with. The same issues of implementation that occur in regular systems occur in these systems as well. There are a number of areas errors could arise:

²For the purpose of this experiment, time is not a construct. Time can be measured with a watch.

- Problems could occur as a result of errors in the data set
- the experts knowledge about the construct could be unsound
- the entire school of thought about the construct could be unsound
- the experts knowledge might be modeled incorrectly in the system
- the design of the system may be flawed drawing incorrect conclusions
- the implementation of the system may have underlying bugs that the software developer is unaware of.

In order to mitigate against these kinds of problems special care should be taken in gathering data for experiments. Experts should be selected carefully for modeling knowledge bases. There isn't much that the designer of a system can do to ensure that the knowledge of an entire field of study is correct. However, by interacting with a system for the evaluation of constructs, experts may gain a deeper understanding of the phenomena that they study. To ensure that an expert's knowledge base is modeled correctly user interfaces to the system should provide clear feedback and knowledge engineers should be trained to use them appropriately. Special care should be taken in the design of the system and appropriate test cases determined.

In order that a system can be evaluated using the techniques discussed in section [2.1.7](#) the data set must contain some representation of the construct being studied. One method for evaluating the representation of the construct would be to have an expert manually determine labels for each row in a data set. These labels can then be compared with the output of an experiment to determine the accuracy of the output.

This method is problematic as it is time consuming and may not yield accurate results. In order that a label is applied correctly an expert will need to study each row the data individually and take the time to assess it accurately. As this would need to be done for thousands of rows it will take up a lot of time. On top of this, the work is tedious so many experts will apply less scrutiny to each row resulting in less accurate labels.

Alternatively, the output can be compared with columns already present in the data but not used in the final result. It might be determined that there is a strong correlation between the construct and some other variable. If this variable is not included in the determination of the construct then we can measure the accuracy of our construct by seeing how it correlates to this variable.

Constructs are typically made up of interleaved ideas and variables which makes their representation in computer systems complex. The structure of constructs varies and as a result different models may be developed to measure them. Similarly, different computational techniques may be used to represent these models.

For example, if we take intelligence as a construct, there are a number of approaches we could use to measure it. Intelligence can be represented by IQ; by computing the sum of an individual's scores on questions across a number of domains. If intelligence is modeled in this manner then it is easy to derive the model using linear regression. If on the other hand a more complex scoring system was used then a different supervised machine learning technique could be used. Similarly an expert could communicate with a group of people in a room and give their opinion on which people they believe are intelligent. This opinion then be used as labels for a data set in which a (very bad) model of intelligence could be made by training a classifier based on age, weight, sex etc. In another case, we could run clustering algorithms on our data. If the algorithm returns two clusters and one cluster has all the Nobel prize winners we might conclude that is the intelligent cluster. An expert could model intelligence defeasibly. For example, if someone performs well on tests they are intelligent but if they cheat on tests they are not. We could run semantics on those defeasible representations to determine who is intelligent.

None of the approaches outlined above can objectively say that they truly measure intelligence. However, if they are useful they will give some indication of how a person will perform on a task. The purpose of this example is to underline the differences in models and how these models can be represented using computational techniques.

It is with this in mind that an experiment is designed to compare these techniques. To do an experiment this has to be done:

- A suitable construct must be chosen and access to an expert secured.
- Machine learning software must be procured and a number of machine learning techniques chosen for evaluation.
- A system based on defeasible reasoning must be designed.
- an experiment built on top of this must be chosen

2.1.7 Evaluation of Knowledge-based Techniques

As the overall aim of the research project is to compare an implementation of defeasible reasoning with machine learning, a framework for comparing the two must be developed. This is no easy task as the techniques vary considerably in the output they will produce. For supervised machine learning the output of the experiment will be a number or a classification. These numbers and classes will already be present in the training and test data sets. For defeasible reasoning this is not the case. The implementation outputs a value that is a representation of a construct. In the experiments performed for this project, that construct is mental workload. There are many ways to measure mental workload but there is no one definitive value that can be used for comparison with this value. There is no value already present in the dataset that we can compare the output of the defeasible reasoning system with.

The following section gives an overview of the various methods used to evaluate the techniques in work by previous authors.

As the aim of this project is to compare the ability of machine learning and argumentation theory to represent and predict the value of a construct, an investigation into construct validity is necessary. Constructs can be thought of as things that are not easy to observe. They are difficult to measure as they are often ambiguous, abstract and can be quite complex. Measuring constructs requires that they are defined precisely and a way to translate the construct from the abstract to the concrete is established.

Construct validity refers to the accuracy of this translation. Construct validity is not referred to in absolute terms, however, we can say that over time a certain measure has

shown strong construct validity³. Two measures used to determine construct validity are concurrent validity and convergent validity.

2.1.7.1 Concurrent Validity

Concurrent validity is used to establish that a particular measure may be used to predict some other outcome that is determined by the construct. The outcome to be verified against must be collected in the same experiment as the new measure being explored.⁴ The ability of the new construct to predict this outcome can then be determined by using a regression model. A notable flaw in concurrent validity is that the measure used to benchmark the new measure may itself be flawed. As a result of this it is rarely used alone but has been used with other measures to assess the validity of a wide variety of constructs such as social anxiety by [La Greca and Stone \(1993\)](#), marital satisfaction by [Schumm et al. \(1986\)](#) and depression by [Storch et al. \(2004\)](#).

2.1.7.2 Convergent Validity

Another means of determining the validity of a measure is the convergent validity of a measure. Concurrent validity was introduced by [Campbell and Fiske \(1959\)](#). If there already exists some accurate measure for the construct that is being examined, the new measure can be evaluated by looking at its relationship to the old measure (how the two converge). Convergent validity can be established by computing a correlation between the new measure and an old measure. This suffers from the same problems as concurrent validity since the measure used as a benchmark may be flawed. Convergent validity has also been widely used, for example in establishing measures for post traumatic stress disorder ([Neal et al. \(1994\)](#)), social desirability ([Stöber \(2001\)](#)) and social phobia ([Beidel \(1996\)](#)).

³See <http://dissertation.laerd.com/construct-validity.php>

⁴<https://explorable.com/concurrent-validity>

2.2 Learning Based Approaches

The literature makes the distinction between different learning scenarios: supervised and unsupervised.

2.2.1 Supervised Machine Learning

[Alpaydin](#) explains that ‘supervised learning’ happens in a scenario in which the task of the algorithm is to learn the mapping from some input X to an output Y . In a review of supervised learning classification techniques, [Kotsiantis et al. \(2007\)](#) explains that if “known labels (the corresponding correct outputs)” are used “then the learning is called supervised”. Examples ⁵ of questions supervised learning attempts to answer include:

- What is the probabilistic distribution of hourly rain given polarimetric radar measurements?
- What category does a particular product belong to given its features?
- Given a number of objective measurements, what will be a restaurant’s annual sales?

Supervised learning problems can be further divided into two categories; regression problems and classification problems. [Ng](#) explains that regression problems are problems where the output to be predicted is continuous. In a regression problem as specific numeric value is predicted, for example, a restaurant’s annual sales in the example above. Problems in which the output to be predicted is discrete are considered classification problems. Classification problems tend to be concerned with the sorting of data into correct categories, as in the second example above. Another such problem might be determining whether or not a particular email should be categorised as spam or not.

The following sections introduce the reader to supervised machine learning techniques relevant to this project from each of the different categories at a high level. Many of

⁵Taken from Kaggle.com; a website for data science competitions

the techniques outlined require a strong understanding of mathematics in order to fully understand and engage with them. This outline will avoid these details as explaining them sufficiently is beyond the scope of this project.

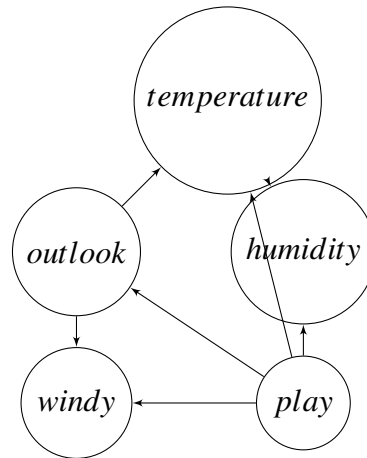
2.2.1.1 Naive-Bayes

[Kohavi et al. \(1997\)](#) introduces Naive-Bayes or Simple-Bayesian Classifier and then outlines techniques for its improvement. The learner is “built based on a conditional independence model of each attribute given the class.” Put simply the learner uses probability to classify the data. Bayes refers to Bayes rule, a mathematical rule for computing the probability that something happens given some other *a priori* condition. The technique is called Naive as the probabilities of the features in the model are assumed to be independent. The probability of each class is computed for a row in the data set. The class with the highest probability for that row is the class that is chosen for that instance of the data.

2.2.1.2 Bayesian Networks

[Heckerman \(1995\)](#) explains that Bayesian Networks have been increasingly employed in expert systems as an encoding for an experts knowledge. One of the strengths of Bayesian Networks is that they can still produce an output with incomplete data. Bayesian Networks provide methods for dealing with uncertainty by graphically modelling dependent relationships. Each attribute is modelled as a vertex in the graph with the relationships modeled as edges. Internally the nodes have a table containing the probability outcomes given conditions have occurred in their parent nodes. In figure 2.10, an example taken from [Witten and Frank \(2005\)](#) the probability of the temperature being hot, mild or cool depends on the values of play and outlook. The table appears as in table 2.1.

While Bayesian Networks are typically developed by mathematicians they can also be learned using ML techniques.

FIGURE 2.10: An example of a Bayesian Network from [Witten and Frank \(2005\)](#)

play	outlook	hot	mild	cool
yes	sunny	0.413	0.429	0.429
yes	overcast	0.455	0.273	0.273
yes	rainy	0.111	0.556	0.333
no	sunny	0.556	0.333	0.111
no	overcast	0.333	0.333	0.333
no	rainy	0.143	0.429	0.429

TABLE 2.1: The table of probabilities associated with the temperature node in figure 2.10

2.2.1.3 Decision Tables

[Kohavi \(1995\)](#) proposed decision tables as a representation for hypothesis in order to solve supervised machine learning problems. Decision tables are typically used in the development of rules in expert systems. The table consists of a list of conditions and actions to be taken depending on what conditions are met. An example decision table taken from [Hoffer \(1999\)](#) is given in figure 2.11.

[Kohavi \(1995\)](#)'s supervised technique uses an induction algorithm to develop a decision table whose default is the majority class in the data set (the decision table majority). The induction algorithm build a decision table based on training data. When the model is presented with a new instance of test data it checks the decision table for matches. If there are no matches it returns the majority class of the training data. If there are several matches with different classes the class that makes up the majority of the classes is

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs							
	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

FIGURE 2.11: Decision table example taken from [Hoffer \(1999\)](#)

chosen. Through experimentation the author determined that the technique performed with prediction accuracy comparable with C4.5, a decision tree algorithm.

2.2.1.4 K Star

[Cleary et al. \(1995\)](#) describes K*, an instance based learner that uses entropy as a distance measure. An instance based learner classifies an instance based on a database of labeled examples. K* uses entropy in order to determine which instance in the database the current instance is most like. The entropy can be determined by taking the Kolmogorov distance (the length of the shortest string) between the two instances. The author reports that through experimental validation K8 performed better than another instance based learner, the 1R algorithm.

2.2.1.5 Linear Regression

Simple linear regression is a fundamental technique in supervised learning used to solve regression problems. In simple linear regression the technique takes a data set with one explanatory variable and one dependent variable and attempts to fit a simple line to it. According to [Ng](#) this line can be defined as $h_{\theta}(x) = \theta_0 + \theta_1 x$. The parameters θ that

define the function are tweaked in order to minimise a cost function. An efficient way to do this is using a technique known as gradient descent.

Linear regression can be used with many explanatory variables; in this case it is known as multivariate linear regression. The same basic principles that apply to simple linear regression apply here but the function we are trying to optimise is $h_{\theta}(x) = \theta_0 + \theta_1x_1 + \theta_2x_2 + \dots + \theta_nx_n$. In order to compute the large number of multiplications that need to occur the calculations are often formulated as linear algebra problems. The advantage of formulating these learning algorithms as linear algebra problems is that many programming languages contain libraries with optimised methods for computing matrix operations.

2.2.1.6 Logistic Regression

The regression techniques described above can be slightly modified to classify labeled instances of a data set. Ng gives the example of classifying an email as spam (assigned the value 1) or not spam (assigned the value 0). In this case logistic regression computes the probability that an email is spam. This probability is given by the hypothesis function:

$$h_{\theta}(x) = \frac{1}{1 + e^{-x\theta^T}}$$

If there are multiple labels that need to be accounted for in the data set then logistic regression can be still be used in a ‘one-vs-all’ fashion. Each label has its own unique hypothesis function that can be used to determine whether or not to apply that label to the data.

2.2.1.7 Artificial Neural Networks

Shiffman et al. (2012) provides an introduction to Neural Networks. Neural networks attempt to model the way learning occurs in the brain by simulating neurons and axions. The fundamental unit of a neural network is a perceptron; similar to one neuron. A single perceptron takes a number of weighted input values, computes their sum and

applies a sigmoid function (in a similar manner to logistic regression) to the sum. It then compares this to a label and computes the error. This error value is used to adjust the weights of the input values; this feedback process is how the perceptron ‘learns’.

A single perceptron can only compute linearly separable hypotheses. In order to compute more complex hypotheses the perceptrons are linked in what is called a multi-layer perceptron. In a multilayer perceptron an input layer of perceptrons take the inputs and then passes their output to another ‘hidden’ layer of perceptrons. There may be multiple hidden layers that the output propagates through until it eventually reaches the output layer. The error for the network is then computed and applied to the weights of each perceptron using a technique known as back-propagation.

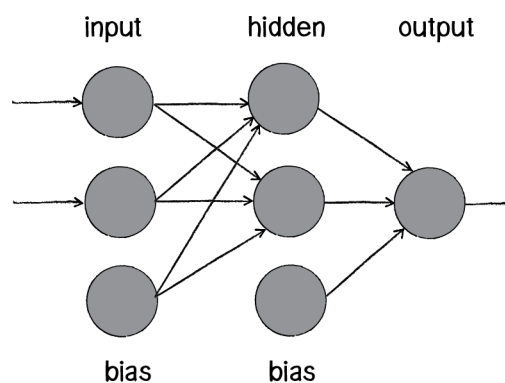


FIGURE 2.12: An example of a multilayer perceptron taken from [Shiffman et al. \(2012\)](#).

The state of the art in machine learning is “deep” learning, currently being utilised by Google, Microsoft, IBM and others. [Arel et al. \(2010\)](#) outline how deep learning overcomes the exponential growth in learning complexity associated with an increase in data dimensionality. Deep learning focuses on the development of computational models that represent information in a fashion similar to the neocortex. Convolutional Neural Networks are described as being the first successful approach to learning many dimensions in a complex manner. Deep belief networks are “probabilistic generative models”; provide a different solution to the problem of deep learning by providing probabilities associated with observations and labels bidirectionally.

2.2.2 Unsupervised Machine Learning

Supervised learning can be contrasted with unsupervised learning; described by [Mohri et al. \(2012\)](#) as problems in which “the learner receives unlabelled training data and makes predictions for all unseen points.” The learning is called unsupervised because the techniques don’t attempt to make predictions based on a specific labeled output. Instead unsupervised learning techniques are used to perform tasks such as identifying clusters in data, anomaly detection and dimensionality reduction. Examples of practical applications of unsupervised learning (taken from [Ng](#)) are:

- Organizing computer clusters.
- Social network analysis.
- Identification of market segments.

Unsupervised ML is sometimes known as class discovery ([Gentleman and Carey \(2008\)](#)), referring to the ability of the techniques to uncover groupings not already made explicit through labeling. The literature refers to two approaches to clustering: hierarchical clustering and partitioning. In hierarchical clustering the algorithms work by building up small clusters or breaking large ones down in a hierarchical fashion. The advantage of this technique is that the number of clusters can be specified after the algorithm has run. With partitioning the number of clusters is specified before hand and this number of elements are chosen at random to define clusters around based on distance. These clusters are then refined over a number of iterations. An example of a commonly used partitioning technique is K-Means.

2.2.3 Evaluation of Machine Learning Techniques

There are a number of methods typically used for evaluation of machine learning techniques. Numeric predictions are evaluated similarly to many scientific experiments using statistical values such as correlation and mean absolute error. Machine learning techniques for solving classification problems are often more detailed as the costs of

false positives and false negatives may be different depending on the problem. Take for example the prediction of cancer recurrence. A false positive (that it is predicted cancer will recur when it will not) will result in a patient being examined by a doctor; on the other hand a false negative would result in the missed opportunity of early diagnosis, potentially costing a life.

Witten and Frank (2005) explain that machine learning techniques are typically evaluated first developing a model using a labeled ‘training set’ of data. Once the model has been developed it can be run on a labeled ‘test set’ of data. The performance of the technique can then be measured by comparing the values predicted by the model with the labels in the test set. The exception to this is with unsupervised machine learning as the data is unlabeled. As the purpose of unsupervised machine learning is to come up with a kind of ‘theory’ about the data, good techniques “make everything as simple as possible, but not simpler”. This idea is the foundation of the Minimum Description Length Principle.

Typically the amount of data used for training and testing is limited and so techniques such as cross validation and percentage of split may be used to test and validate the model. In percentage of split the data set is divided into two groups, one for training and one for testing. Typically a higher percentage is used for training the data than testing it. In cross validation a data set is divided into groups. One group is used at a time for testing the model while the rest of the data in the set is used for training it. This technique is often referred to as N-fold cross validation where N is the number of groups the data is divided into.

The following is a brief description of a number of measures used when evaluating the results of machine learning techniques.

2.2.3.1 Numeric Prediction Problems

Witten and Frank (2005) explain that for practical situations the best numeric prediction method tends to perform well across all performance measures. Most performance measures tend to give an overall value for the difference between the predicted and actual

value in a test set. Examples of such measure include mean-squared error, root mean-squared error and mean absolute error. As mean squared error squares the difference from the mean it tends to punish large errors more than the other measures.

Mean-squared error can be defined:

$$\frac{\sum_{i=1}^n (p_i - a_i)^2}{n}$$

Where p are the predicted values and a are the actual values.

Another performance measure used for numeric prediction problems is Pearson's Correlation Coefficient. This measures the linear correlation between the predicted value and the actual value. It differs from the other measure in that it ignores the differences in the scale of the two values.

$$\frac{S_{PA}}{\sqrt{S_P S_A}}$$

where:

$$S_{PA} = \frac{\sum_i (p_i - \bar{p})(a_i - \bar{a})}{n - 1}$$

$$S_P = \frac{\sum_i (p_i - \bar{p})^2}{n - 1}$$

$$S_A = \frac{\sum_i (a_i - \bar{a})^2}{n - 1}$$

2.2.3.2 Classification Problems

Evaluating the performance of a classifier is not as straight forward as in assessing the performance of regression techniques. Regressive techniques are often not expected to predict an exact value, but a good approximation of the value. As classification

problems deal with discrete answers, there is a well defined notion of whether or not an answer is correct. As a result of this there are a number of extra considerations that are made when choosing classifiers. The cost of incorrectly classifying an instance (a false positive) must be weighed against the cost of not classifying an instance correctly (false negative). Take for example the cancer recurrence problem. If a patient is a false-positive the patient will have to pay medical bills for extra tests. On the other hand a false-negative is more costly potential costing a life. In order to capture these ideas a number of measures are used in the literature.

Common measures for evaluating the performance of a classifier include precision, recall, F-score and area under the ROC curve. (Powers (2011))

- Recall refers to the ratio of the number of true-positives versus the total number of real-positives. Recall is useful in determining how much of what we are interested in has been identified. However, alone it is a poor performance measure; a classifier that returns the whole data set including false positives can be said to have 100% recall.
- Precision balances this by offering a measure of number of correct positives as a whole of what is returned. Precision is defined as the ratio of true-positives against the total number of classified positives. Both of these measures say nothing about how the classifier deals with negative cases.
- The same is true of F-score which is simply the harmonic mean of precision and recall ($F_1 = \frac{precision \times recall}{precision + recall}$).
- The area under the ROC curve is a technique taken from electrical engineering that better captures the tendencies of a classifier report false positives. A plot of true positives against false positives is made for varying by varying the classifier threshold. The area under this curve can be used to evaluate the classifier. It can be calculated as follows:

$$\begin{aligned}
 AUC &= \frac{TPR - FPR + 1}{2} \\
 &= \frac{TPR + TNR}{2} \\
 &= 1 - \frac{FPR + FNR}{2}
 \end{aligned}$$

Where AUC is Area Under the Curve, TPR is true positive rate, FPR is false positive rate, TNR is true negative rate and FNR is false negative rate.

2.2.3.3 Minimum Description Length

[Grünwald \(2005\)](#) gives an overview of how the Minimum Description Length Principal is used to solve the problem of model selection. Overfitting is a problem that occurs frequently with ML techniques. Overfitting occurs when a learning technique puts too much emphasis on fitting the data exactly. As a result data points like outliers can skew the model and result in it classifying future instances of the data wrongly. A model that overfits the data will classify the data set used to train it with very little error, however, when used on a test set will perform poorly. A simpler model may not classify the training data perfectly but will perform better on test data.

As observed earlier, a simpler theory is one that allows data to be encoded using fewer bits. This idea is born from information theory. For example, the SVG file type encodes a red circle as `<circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />`. This is far simpler than sending every single bit used to describe circle in raw data. From the perspective of programming MDL can be compared to Kolmogorov Complexity; the simplest theory is the one that produces the shortest computer program that will print the data.

MDL views learning as data compression; [Grünwald \(2005\)](#) defines this formally: “for a given set of hypotheses H and data set D , we should try to find the hypothesis or combination of hypotheses in H that compresses D most.” The best point hypothesis H to explain the data D is the one which minimizes the sum $L(H) + L(D|H)$, where

- $L(H)$ is the length, in bits, of the description of the hypothesis; and
- $L(D|H)$ is the length, in bits, of the description of the data when encoded

with the help of the hypothesis.

The underlying idea of that can be derived from the principle is that the simple representation of constructs should be considered favourable to one that is complex. An informal approach to computing the MDL of a model is to describe a model in terms of the number of lines of code it would take to output the data. The greater the number of lines of code, the greater the MDL.

2.3 Discussion

This chapter began with an exploration of two broad domains; knowledge base systems and machine learning. Knowledge based techniques were explored and in particular argumentation theory and defeasible reasoning were introduced. In particular the fundamental work of [Dung \(1995\)](#) was introduced. This lays the foundation for the defeasible reasoning implementation of this experiment. Implementations for the computation of AF semantics have been reviewed in order to be integrated in this design. The results of other work using argumentation theory have been explored. Within machine learning several supervised learning techniques were explored; bayesian networks, naive bayes, artificial neural networks, kstar and various regression techniques.

The implementation of argumentation theory in knowledge based approaches is advantageous as it allows for default reasoning, while simultaneously being able to derive solutions in the case of conflicting data. With a reasoning based approach data can be evaluated with greater scrutiny that can't be captured automatically through learning. Experts can provide the insight into why a particular outcome occurred, this could be modeled defeasibly in a way that wouldn't be captured by simply plugging data into an equation.

Machine learning on the other hand can in some cases be more convenient than knowledge based systems. Linear regression is intuitive for developers and researchers as fitting a line to data is a familiar task. Linear regression is like default reasoning in that the line attempts to model things that happen typically. Outliers can be ignored as deviations from the norm and good predictions can be made some of the time. Linear regression considers outliers to be noise in the data and attempting to accommodate these outliers results in a model that overfits the training data. A more complex machine learning technique such as an ANN could account for an outlier, however, it would need to be provided with enough instances of similar outliers that is appropriately labeled. A data set large enough to train a classifier sufficiently to accommodate for these outliers may not exist. Once this hypothetical classifier has been trained, the resulting representation may not be easy for an expert to interpret. The automatic nature of machine learning has advantages that could not be obtained using a knowledge based approach. For example it would be difficult and time consuming for an expert to codify all of the different ways a person might write the letter 't'. An artificial neural network can be trained to recognise the letter if provided with enough labeled training data.

Both knowledge based approaches and learning based approaches fall in the field of artificial intelligence. Despite this there are relatively few works that provide comparison between the two. Within each approach exist many variations in techniques and algorithms with more being created as years go by. To specialise in one domain takes researchers years of focus; many would rather focus on advancing their research further than taking a step back to look at what research has been done in other fields. This is reasonable as it can take months of study to become familiar with even the basic concepts in an entirely different domain. A comparison made at this stage would likely be biased and would fail to account for the subtleties of one approach or another. Researchers that investigate both fields tend to do so to create advanced hybrid approaches (for example [Gómez and Chesnevar \(2004\)](#)) rather than provide a comparison for the benefit of the research community. This research project attempts to fill that gap by providing such a comparison. The comparison of the two domains is a non-trivial task and it is unreasonable to expect it to be without bias. This project will attempt to provide

some value to readers interested in both domains, however, it is to be considered by no means exhaustive.

A number of argumentation theory implementations were surveyed. Many of these implementations focused on aiding users in their reasoning. This focus has likely come about because it is a domain in which argumentation theory has an obvious immediate application. Graphical tools that incorporate the argument diagramming and the computation of results are less likely to be developed as they require several design issues to be resolved such as argument activation by data. It is also difficult for designers to know before implementation that such a system will fulfill a practical purpose.

As a result of the gaps identified in the literature, a problem has been identified to be the subject of research. The problem is the shortage of available comparisons between knowledge based and learning based approaches, particularly in the area of construct modeling. In order to solve this problem a research question is defined:

“To what extent can an implementation of Defeasible Reasoning enhance the representation of a construct (mental workload) and support inference in comparison with Machine Learning?”

Several techniques for assessing the predictive capacity of the techniques have been identified. Measures for establishing the validity of the construct measures have been determined. These measures will be valuable when assessing the techniques in a quantitative experiment as they will provide some objective way to compare results. None of these measures are perfect. Error values obtained during experiments of this nature may be misleading as the data used to train classifiers comes from the same set of data that is used to validate those classifiers. The measures being used to validate the representation of constructs are similarly imperfect. These measures rely on a correlation between the new measure and a previously established measure of the construct. If the previously established measure in fact is a poor representation of the construct then we may have simply created another poor representation. These factors will need to be considered in order to thoroughly evaluate the results of the experiment.

Chapter 3

Design

In order to answer the research question posed in Chapter 1, an experiment was designed. The research question is restated here:

“To what extent can an implementation of Defeasible Reasoning enhance the representation of a construct and support prediction capacity in comparison with Machine Learning?”

In order to answer this question a number of hypotheses to be tested were defined as follows:

- The defeasible measure of a construct created by an expert is better at predicting objective performance values than machine learning approaches.
- The construct measure of an expert will have a high concurrent validity with objective measures related to the construct.
- The construct measure of an expert will have a high convergent validity with existing measures for the construct.

In order to test these hypotheses the following experiments are conducted.

- An argumentation system is developed in software and used to elicit a knowledge base from an expert.

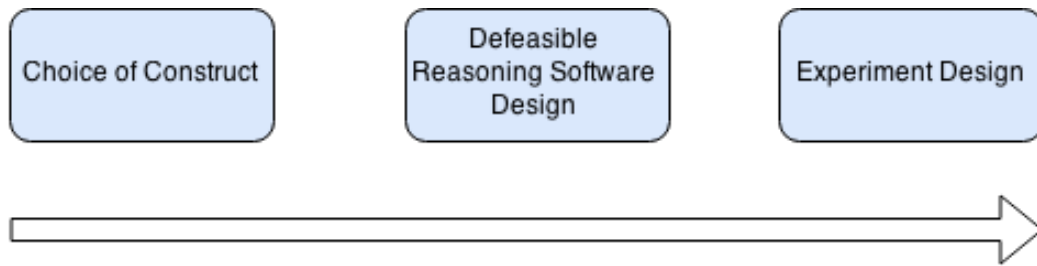


FIGURE 3.1: Chapter Overview

- A number of ML classifiers (using both classification and regression algorithms) are trained to using a partition of the experiment data set.
- The predictive ability of the classifiers and the knowledge based system are tested using a subset of the data set.

The rest of this chapter deals with the design associated with these experiments. The choice of construct to be examined is justified, the design of the software is outlined and the experiment procedure is defined.

3.1 Choice of Construct [Longo and Kane \(2011\)](#) [Longo et al. \(2012b\)](#)

In order to perform the experiment a subject must be chosen. Access to an expert in the field of this subject needs to be secured as well as a data set that can be used to evaluate the representation of the construct. Suitable constructs don't have a conclusive easy to assess for of measurement such as cancer, intelligence or personality.

For this experiment Human Mental Workload was chosen as the construct. Mental Workload can loosely defined as the amount of effort it takes a user to perform a task. Researchers across a wide variety of domains study MWL, particularly those with interests in human performance and machine usability. [Meshkati and Hancock \(2011\)](#) outlines the problems that are associated with defining, quantifying and measuring MWL. Generalising MWL is difficult as it is a multifaceted phenomenon that varies depending on context. [Meshkati and Hancock \(2011\)](#) defines MWL as “the operator’s evaluation

of the attentional load margin (between their motivated capacity and the current task demands) while achieving task performance in a mission-relevant context.” Possible ways of measuring MWL include objectively (using task time, physiological indicators or task success) or subjectively (by having the participant answer introspective questions about their perceived state while completing the task).

One possible way of measuring MWL is using the NASA-Task Load Index (TLX). NASA-TLX is an introspective questionnaire in which participants subjectively assess their mental demand, physical demand, temporal demand, performance, effort and frustration after completing the task. Participants are then asked to compare the importance of these factors in their completion of the task.

Another possible way to determine that MWL was high for a given task is based on the task completion time. If all variables are equal then task time can help us determine which tasks were most difficult.

Dr. Luca Longo from the DIT School of Computing is an expert in the area of MWL having completed his doctoral dissertation in formulating MWL as a defeasible construct. He has kindly volunteered his knowledge base to be used within the experiment.

The data set was obtained in an experiment performed by Dr. Longo that aimed to gather MWL information from participants based on their used of different web based interfaces. The participants had to complete 11 web based tasks using popular web applications including Google Search, Youtube and Wikipedia. 40 participants between ages 20 and 35 were split into two groups. In a given task, one group used the original application interface while the other group received a version in which the presentation of the page had been altered. Which group received the original or modified interface varied by task. It was believe that the changes to the structure of the interface would impose a greater mental workload on the participants. After each task was completed the participants answered a questionnaire designed to assess their subjective mental workload. The data set contains the results of the experiment. Sample rows are available in the appendix along with the details of each column.(Longo (2014))

3.2 Machine Learning Software

In order to implement the experiment a suite of machine learning software needed to be procured. Designing and implementing a suite of ML algorithms is a time consuming process. Moreover, a naive implementation of ML software will result in software that performs poorly. For these reasons, it was decided to use an existing ML tool set. Proprietary options include Oracle Data Miner or SAS Enterprise Miner while open source alternatives include the R programming language or Rapid Miner. It was decided that WEKA would be a good fit for the project.

WEKA (Waikato Environment for Knowledge Analysis) is an open source ML workbench developed at the University of Waiko. WEKA is widely used in both academia and business. It has a simple user interface which provides feedback related to model performance in a clear and concise format. It contains implementations of machine learning algorithms across a range of typologies which allows for a large comparison with defeasible reasoning. As it is an open-source project it is possible to investigate the source code to gain a greater understanding of the underlying techniques. WEKA has a low barrier to entry; this is important to save time as a significant portion of the project will involve the implementation of defeasible reasoning software.

3.3 Defeasible Reasoning Software Design

In order to demonstrate how DR can model a complex construct, software was designed that would allow an expert to input their knowledge base as directed graph. There is no standard way to implement a system based on defeasible reasoning. Section ?? outlines approaches taken by others in the past to implement such systems. By drawing on these designs and identifying use cases for this project a new DR implementation can be designed.

3.3.1 Use cases

The users of the system are both the experiment administrator and the expert/knowledge engineer. In the context of the experiment the role of knowledge engineer is played by the experimenter.

- “As an expert/KE I want to input my/a knowledge base as a defeasible construct.”
- “As an expert/KE I want to model my/an expert’s knowledge base in a way that will produce a numerical output given some numerical input.”
- “As a system user I want to be able to save my work and retrieve work I have done previously.”
- “As a system user, when I open the program what I was last working on should be displayed or a new project should open.”
- “As a system user I should be able to retrieve data to test my knowledge base with. I should be able to investigate that data and investigate the results of running my knowledge-base on that data.”
- “As an experimenter I should be able to collect results from running an experts knowledge-base on a full set of data.”

By investigating each of these user stories a system can be designed suitable for the purposes of the experiment.

3.3.2 Defeasible Knowledge Base

The expert or knowledge engineer must be able to model their knowledge base as a defeasible process. This process is being modeled as an Argumentation Framework as proposed by [Dung \(1995\)](#). To reiterate, an argumentation framework is a set of arguments and attack relations between those arguments. In the case of MWL, an example

of an argument is “if the user’s effort is low, this implies that the user’s mental workload is low”. Another is “if the user’s performance is low, this implies that the user’s mental workload is high”. It can be said that the former attacks the later since if the user doesn’t make any effort the performance will be low. In the argumentation framework $S = \langle A, R \rangle$ A is the set of arguments $A = \{a, b, c, d\}$ and R is the set of attacks $R = \{(a, b), (b, c), (c, d)\}$ These arguments and attacks relations must be input into the system in a format that allows them to be processed and that allows the user to easily modify and reason about what they have input.

One method that could be adopted in designing the interface is a text based approach. The user enters their knowledge base in a text editor as a list of nodes and attack relations according to a format specified by the system designer that can be parsed by the software. An example of a JSON based format would be the following:

```
"knowledge_base" {
  "arguments": [
    "Low Effort->Low MWL",
    "High Effort->High MWL",
    "Low Performance->High MWL",
    "High Performance->Low MWL"
  ],
  "attacks": [
    ["Low Effort->Low MWL", "Low Performance->High MWL"]
  ]
}
```

Using this approach it is easy to implement logic to parse the AF. This lightweight approach is preferable for designing test cases for the system as it is trivial for a technical user to modify and copy.

This approach is unsustainable for regular users and large (real) knowledge bases. It is time consuming and cumbersome for a user to have to type out an argument every time that they want to create an attack. It is also intimidating for non-technical users and error-prone. As they are stored separately it can be difficult for the user to keep track of the nodes and attacks. The user would need to establish a strong naming convention to ensure that the correct arguments are attacking each other. If the user needs to store

more information about the arguments and their relationships the resulting file format will grow increasingly complex.

For this reason the software has been designed to utilise a Graphical User Interface that allows a user to draw a directed graph. A user creates a new argument by clicking on an empty space on the graph. The user can name this node in order to keep track of what it represents. Once nodes have been created the user can then model the attack relations between the arguments by dragging from one node to another.

This approach has many advantages in comparison with the first approach. A non technical user will be more comfortable using a GUI than using the text based approach. When the knowledge base is input using text the user must take special care to ensure that the attack relations are correct.

There are additional requirements that need to be satisfied by the software in order to correctly model an AF. The first is the concept of rebutting attacks. In order to model rebutting attacks the user must be able to draw edges in the graph with arrows on both ends. The second is modeling the concept of mitigating arguments. Some arguments weigh in on the final evaluation of the semantic without actually contributing to the value of the construct. These arguments must be taken into consideration but have their output ignored in the final value.

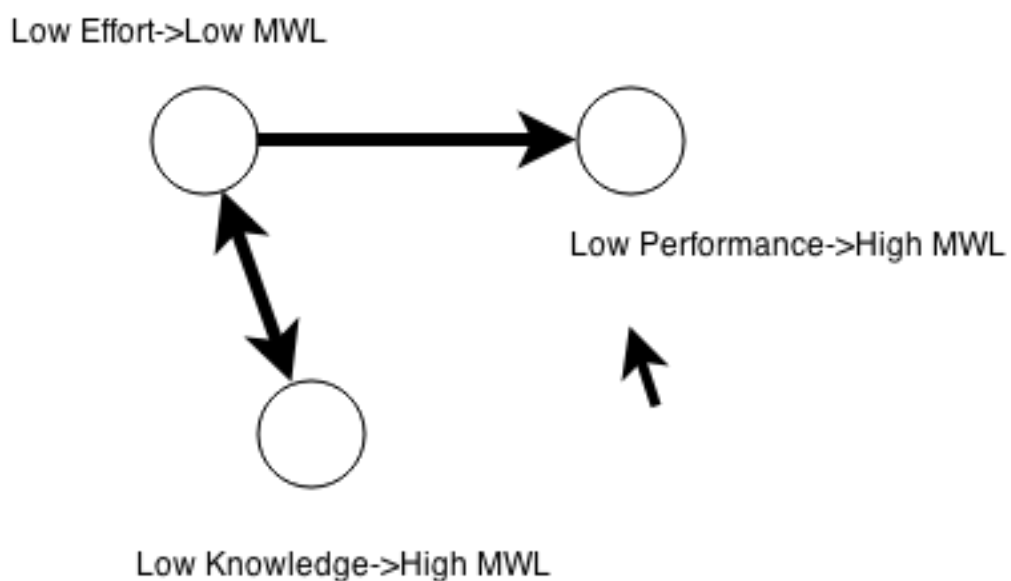


FIGURE 3.2: A Mockup of the UI for entering an Argumentation Framework

3.3.3 Membership Functions

As the system is currently defined it will provide an expert with the ability to visualise their knowledge base in the form of a directed graph. The only information that can be obtained about an argument is its relationship with other arguments and its label (a natural language statement that allows the user to identify the argument and that may in some way describe the nature of the argument).

In order that the argumentation framework can be used to compute results a number of other concepts need to be designed into the system. The notion of whether or not an argument is activated or not needs to be modeled. Argument activation allows us to consider which attack relations to take into consideration and which to discard for a particular tuple in the data set. Arguments are based on one or more premises and each premise corresponds to one column in the data set. An argument is activated if all of its premises are relevant to a particular instance in the data.

Example 3.3.1. If an instance had a value of 0 for effort then an argument with the premise “Low Effort” would be activated and an argument with the premise “High Effort” would be discarded. For a value of 0 effort and 0 motivation an argument with two premises “Low Effort and Low Motivation” would be activated. Given this same input, the argument “High Effort and Low Motivation” would be discarded as only one of its premises are satisfied.

The process of activating and discarding arguments results in a sub-graph of arguments relevant to the row in the database. In order to further reduce the sub-graph argumentation semantics are run on it which take into account the attack relations between the arguments. This results in a set of possible sub-graphs that are applicable to that instance in the data-set.

From the remaining arguments in each sub-graph a value for the construct being measured must be determined. These values can then be averaged for a particular graph to give an overall value for the construct for that instance.

Example 3.3.2. If the argument is “low performance \rightarrow high MWL” then for a simple mapping a performance value of 0 will result in a MWL value of 100. This can be repeated for every argument and averaged for a value of MWL.

In order to determine whether or not an argument is activated we will use fuzzy sets in a manner similar to Longo and Hederman. Fuzzy Sets were defined by Zadeh (1965) as “a class of objects with a continuum of grades of membership.” These sets are characterised by membership functions; functions that take a value and map it to a number between 0 and 1 (where 0 indicates absence of the value in the set and 1 indicates its presence.) This allows us to take a vague statement such as “High Performance” and determine to what extent a value of performance is considered to be high.

For the purposes of the experiment we consider a premise to be relevant if the input value falls between the bounds of its membership function. If all associated values satisfy the membership functions of an argument, even with a very small degree of truth, then that argument is taken into consideration when evaluating the semantics of the AF for that row. If even one value associated with an arguments premise falls outside the membership function then the argument is disregarded.

By taking a value from a column the degree of truth of the premise as applied to the row can be determined using a membership function. We can then determine the overall degree of truth for an argument by computing the average of the degrees of truth of the premises. The degree of truth for the argument is then used as the input for an argument output function which determines value associated with the construct for that argument.

The following is an example of this process. Taking the argument labeled “Low Effort & Low Performance \rightarrow Low MWL”; two premises can be identified: “Low Effort” and “Low Performance” and an output function “Low MWL”. The premises and output function could be modeled as in figure 3.3. Table 3.1 shows example input and output for the function.

It is possible that membership functions could be input by the user in the form of mathematical functions. This would require the user has sufficient mathematical proficiency that they can express their beliefs in as mathematical functions. A more user friendly

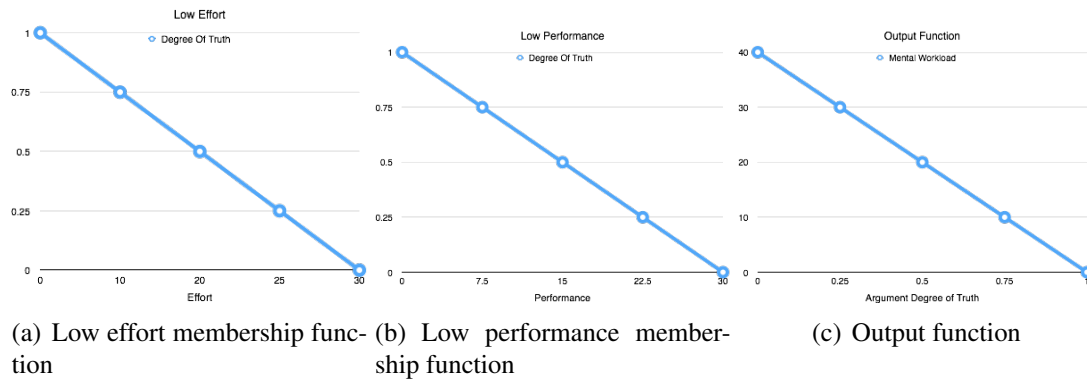


FIGURE 3.3: Example membership functions and output function

Effort	Performance	Result
0	0	The argument's degree of truth is 1 overall, the value for MWL is 0.
30	30	The argument's degree of truth is 0 overall, its output is considered since the input's are in range. its value for MWL is 40.
50	15	The argument is discarded as the value for Effort is out of the range of the membership functions.
20	15	The argument's degree of truth is .5 overall, the value for MWL is 20.

TABLE 3.1: Membership function example results

method of eliciting membership and output functions from the user is to have them draw the functions by hand using the software. We can then use the data associated with this drawing to determine the appropriate output for a given premise given a tuple in the data set.

3.3.4 Additional Software Requirements

Additional requirements specified in the user stories involve being able to retrieve and save the knowledge bases they have been working on. This is to be implemented by serializing and deserializing the AF into a JSON file similar in format to that described in section 3.3.2. This serialization will also be used to retrieve whatever the user was last working on when they utilise the system.

In order that a user can test their knowledge base the user will be given access to a data set. This will be displayed as a table with the user able to compute results for individual rows.

The complete design for the regular user stories has been achieved. A mock up of the resulting UI is given in figure 3.4.

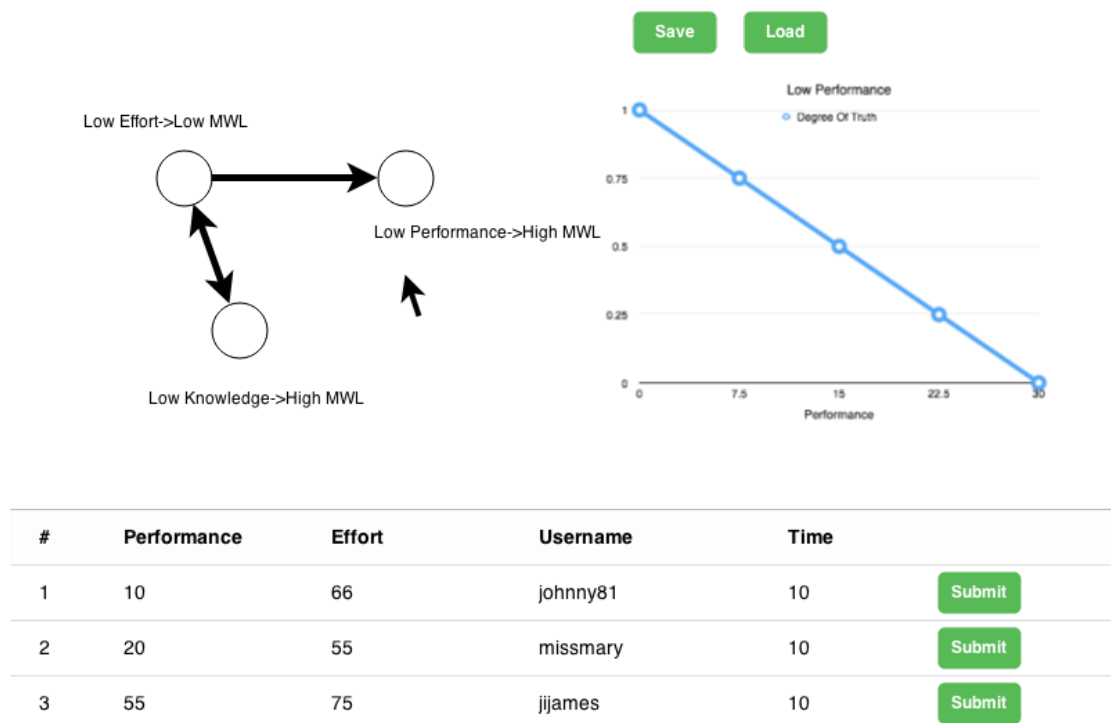


FIGURE 3.4: A Mockup of the UI for entering an Argumentation Framework

In order that the experiment administrator can compute results for many knowledge bases and collect additional information, the business logic for the application is encapsulated in a class, `FrameworkRunner`. This allows it to be called from both the GUI and a command line interface with additional options.

3.4 Experiment Procedure

Now that the necessary software for the experiment has been designed a formal procedure for undertaking the experiment can be defined. The research question referred to at the beginning of this chapter needs to be reinterpreted in the context of mental

workload. With respect to predictive capacity an attempt will be made to answer the following question: “Given data associated with an individual undertaking a task can we predict their mental workload?”

We will attempt to answer this question by performing four sub-experiments. Each experiment will examine a different technique’s ability to model and predict MWL based on the data set that has been provided. An experiment will be performed for Supervised ML continuous techniques, Supervised ML discrete techniques, Unsupervised ML and Defeasible Reasoning. All of the techniques will be required to make their inferences based on the following data set columns: mental, temporal, psychological, performance, effort, central, response, visual, auditory, spatial, verbal, manual, speech, arousal, bias, intention, knowledge, parallelism, skill, difficulty.

What is important to note, as explained in section 3.1 is that there is no concrete definition of what MWL is. Each technique will have a different interpretation of what MWL really “means”. As a result there is no yardstick we can objectively measure the results against and say that one technique is inconclusively better than the other. The results will be evaluated both quantitatively and qualitatively in the context of the material gathered in the literature review.

In order to test ML techniques for predicting numerical outputs, some measure of a construct must already be present in the data used for training and testing. For this reason MWL is interpreted to be equivalent to task time. Time on task is often used in usability experiments to determine the usability of a computer interface. This approach has several flaws which will be the subject of further discussion in the evaluation chapter.

Similarly for the classification techniques, the construct must be interpreted as some label in the data. There is a relationship between MWL and the task ID, some tasks were designed to produce greater MWL than others. Thus in this experiment task ID is interpreted as MWL.

The unsupervised ML techniques will produce subsets of the data. These can be compared against the results of the other experiments to yield some insight into MWL.

For each of the ML techniques mentioned a number of common algorithms of that category are chosen. Each algorithm is run on the data using Weka. The output from each iteration of the experiment are the models developed by the algorithm and statistics about the performance of the model. All of the supervised ML algorithms are trained and evaluated on the dataset using 10-fold cross validation. The results of each iteration are collected and stored for analysis later.

The last experiment to be conducted involves determining the defeasible reasoning software implementation's capacity to model and predict a construct. The software is implemented and then used by to elicit the knowledge base of both an expert and a lay person with regards to MWL. The participant evaluates their knowledge base using a portion of the data set. The results of running the knowledge base using DR techniques are then collected by the experiment implementer. Within this experiment a completely new value for MWL is developed based on the participants beliefs. The experiment also returns a degree of truth for MWL value and some performance characteristics of the software. These results may then be evaluated with respect to each other and the results from the other experiments.

3.5 Conclusions

This chapter presented the experiment being undertaken to evaluate the hypothesis of this research project. The experiment will consist of a test of machine learning methods and defeasible reasoning techniques. In order to establish how the research question will be tested the idea of a construct was defined. The construct for the experiment was chosen to be mental workload and an experiment for determining the ability of different techniques to model this construct was designed.

A necessary step in performing this experiment is the implementation of a defeasible reasoning system. The design of this software is influenced by the implementation created by [Longo et al. \(2012a\)](#); however it build upon this work by offering a GUI that allows the implementation to be used for modeling multiple phenomena as defeasible processes. A software design is outlined order to undertake the defeasible reasoning

portion of the evaluation. The design of the software includes the requirements for the user to be able to draw an argumentation framework that models their knowledge base and determine activation of the arguments within that framework by drawing membership functions.

Chapter 4

Implementation

4.1 Introduction

This chapter describes the implementation of an experiment undertaken to answer the research question proposed by this thesis. A critical first step in performing that experiment is the implementation of a defeasible reasoning system. This is taken as a starting point for this chapter.

The application architecture and programming decisions that were made are explained and justified with respect to how they support the experiment. The challenges associated with implementing a system of this nature are presented along with the solutions pursued to overcome those challenges.

The implementation of the experiment is then discussed. Specific details of how events unfolded are mentioned where they are relevant to discussion in the evaluation section.

4.2 Defeasible Reasoning Software Implementation

The research question of this dissertation deals specifically with an implementation of defeasible reasoning. The design of the system was outlined in chapter 3 without any specific implementation details discussed.

It was decided to implement the software as a web application. In the last 10 years web browser technology has improved vastly. Advances such as HTML5 APIs, powerful Javascript engines and mobile technology allows fully featured applications to be developed and run in the browser.

Developing the application as a web based one offers a number of practical advantages to the experiment. Participants in the experiment can access the application remotely from anywhere. The software is platform independent and can run on any web browser with Javascript enabled. No software needs to be installed on different machines. No software needs to be updated locally, just once on the application server. All of the data associated with the DR experiment is all located and stored on the server allowing it to be retrieved easily for analysis.

4.2.1 System Architecture

In chapter 3 the required system functionality was designed and relevant components identified. The functionality of eliciting the knowledge base (the AF and membership functions) and the verification of this knowledge base is implemented as a Javascript client application. This communicates with a back-end server that provides a RESTful Web service written in PHP (using the Slim Framework¹).

The Web Service back-end provides the basic functions outlined for the application. The back-end saves a knowledge base to disk as a JSON file. The Javascript client can request a list of knowledge bases and from this list retrieve a knowledge base previously created by the user. Knowledge bases were saved as JSON files on disk rather than in a database. This allowed structure of the application to change more fluently during development. It also supports knowledge bases to be examined and serialized in their complete form without needing to assemble it with queries from the database. The server allows the client to access data stored as CSV files so that an expert can evaluate

¹www.slimframework.com

their knowledge base against the data. The last critical component of the server application is to take a row from the data set and compute the value of the construct using the knowledge-base.

The application is served from an apache server running on a virtualised Ubuntu instance provided by the Okeanos project ².

In order to speed up the development process a number of open source frameworks and libraries have been utilised in the software implementation. The CSS framework Bootstrap³ and the Javascript library JQuery⁴ have been used for presentational aspects of the site. Bootstrap provides a number of useful components such as modal boxes that allow the software to present information to the user in a clear manner. JQuery provides wrappers around native browser functionality such as DOM manipulation and AJAX networking facilities that abstract away the inconsistencies between browser implementations.

The graphical input of argumentation frameworks and membership functions is achieved using the D3 library⁵ developed by Bostock et al.^{Bostock et al. (2011)} D3.js is a data visualisation library that allows developers and designers to interact with data in the browser. Data can be loaded from urls in multiple formats such as JSON and CSV. Data points can be “attached” to DOM elements which provides useful functions. The styling of the DOM element can be linked to the value of the data and the data can be manipulated by user interactions. D3 is most commonly used with SVG⁶ (Scalable Vector Graphics) a markup language for implementing vector graphics. SVG has been an open standard for more than 10 years now, as a result it has been widely implemented and there is currently more support for it than other graphics alternatives such as HTML5 Canvas. Two key features of D3 that are used in the implementation of the experiment software are its implementation of Bezier curves and its force directed graph implementation.

²okeanos-global.grnet.gr

³getbootstrap.com

⁴jquery.com

⁵d3js.org

⁶www.w3.org/Graphics/SVG/

The server implementation requires Argumentation semantics to be computed based on the input of a Dung Argumentation Framework. Computing these semantics efficiently requires a deep understanding of argumentation theory, graph theory and algorithms. This implementation is specialised and a number of libraries have been explored in the literature review for its implementation. Dung-o-matic, an implementation of these algorithms has been made available by the University of Dundee under the Apache License, Version 2.0. The Dung-o-matic⁷ was found not to be the most efficient implementation for the computation of semantics by a number of studies. However, its source code is freely available and can be run on any platform provided the platform has Java installed. Moreover, the advantage of using Java is that as one of the world's most popular programming languages there is abundant documentation available to assist in its integration. The source code for both dynPARTIX and ConArg2 are unavailable and cannot be used for implementation in this project. It could be possible to integrate ASPARTIX for a more efficient implementation, however, this requires an answer set programming solver to be installed. It is anticipated that the integration of such a solver could be laborious and time consuming without adding much value to the project. For these reasons the Dung-o-matic was chosen despite performance concerns.

The technologies outlined in this section are manifested in the application architecture in figure 4.1. With these underlying technologies established a number of implementation challenges remain. The solutions to these challenges are the focus of the following chapters.

4.2.2 Application Front End

The two crucial features of the application front-end are the interface for drawing argumentation frameworks and the interface for drawing membership functions. The implementation of these features is discussed here.

The implementation leverages D3's force-directed layout⁸, a built in layout that solves the problems posed when visualising graph data structures. Typical data visualisation

⁷www.arg-tech.org/index.php/projects/dung-o-matic/

⁸<https://github.com/mbostock/d3/wiki/Force-Layout>

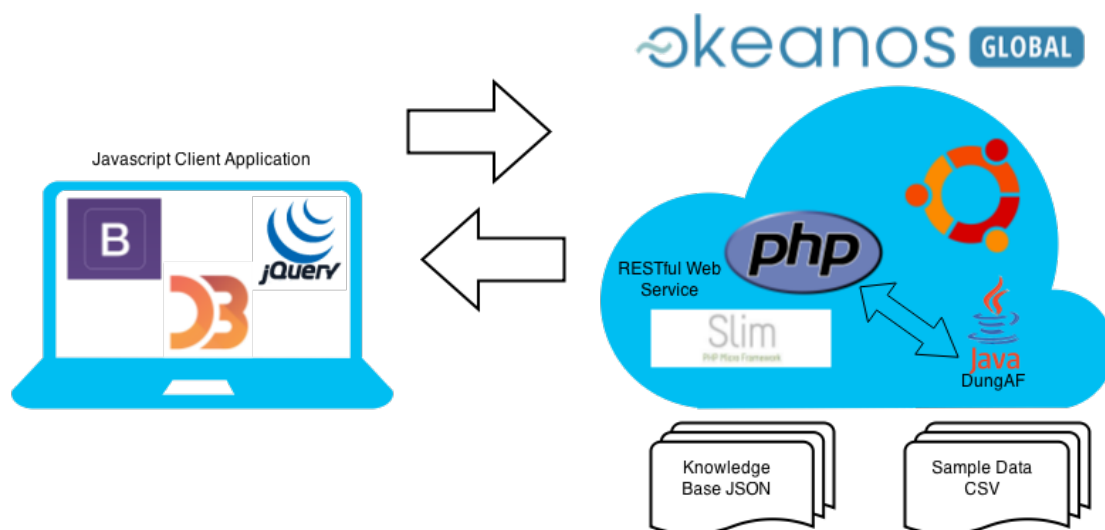


FIGURE 4.1: Application Architecture

techniques take values and parse them one by one, drawing a marker in a position based on each value. This is not the case with graph data structures. With graph data structures, it is generally preferable to have vertices that have common edges close together and to have those without common edges far apart. D3's force directed layout takes list of vertices and edges and generates positions for vertices using simulation inspired by physics. Similarly to sub-atomic particles, nodes are given charges that repel other nodes in the graph and are kept from drifting apart by the links in the graph. There is also a force at the center of the visualisation that prevents any of the nodes from drifting outside the view port.

By utilising this layout and D3's event helpers (listeners for mouse actions such as click and drag) an interface can be implemented that allows a user to input a directed graph, or in the case of the experiment, an argumentation framework. The user can create a node by clicking on an empty space on the graph and can create links between two nodes by dragging from one node to another. When the user creates a node they are prompted to give the node a label. This results in a knowledge base being represented as in listing 4.1.

```
knowledge_base {
  nodes : [
    {
      id: 0
    },
    {
```

```
    id: 1
  },
  {
    id: 2
  }
],
lastNodeId : 2,
links : [
  {source: nodes[0], target: nodes[1], left: false, right: true },
  {source: nodes[1], target: nodes[2], left: false, right: true }
]
}
```

LISTING 4.1: JSON data structure for argumentation framework

D3 expects the data as an array of node objects and an array of links which contain references to the nodes. It adds x and y values to the nodes to track their position in the viewport and updates these values at a fixed interval in order to animate the graph. The id of the last node added to the viewport is stored in the `lastNodeId` variable. This is important for creating new nodes as nodes are tracked based on their IDs, not their array indexes.

Three types of attack relation must be modeled by the interface for correct implementation of the system. These attacks are undercutting attacks, rebutting attacks and mitigating arguments. Undercutting attacks are implemented simply as an arrow from the attacking node to the attacked node. In the data structure this is modeled as a link with the value for either ‘right’ or ‘left’ equal to true. In a rebuttal attack both arguments attack each other. This link is visualised as an arrow with two ends. In the data structure this is represented by setting both ‘right’ and ‘left’ equal to true. Examples of these arguments visualised using the tool are shown in figure 4.2. Mitigating arguments are modeled the same as undercutting arguments but are represented in the system by labeling their output functions as “mitigating arguments”.

By selecting a node in the graph a user can then define membership functions representing the internal premises of the argument and an output function representing the contribution of the truth of this argument to the overall value of the construct. It was decided in the system design phase that the user should be able to draw these functions

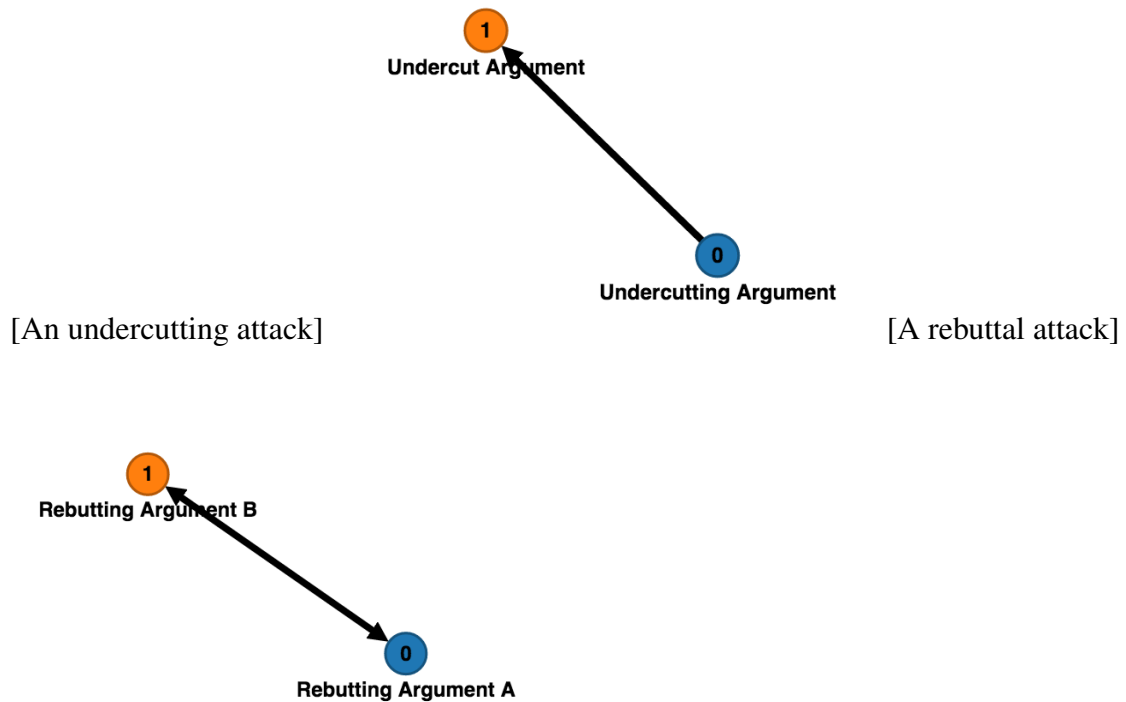


FIGURE 4.2: Examples of different attacks drawn using the interface

using the interface. It was decided to utilise Bezier curves to achieve this in a manner that is simple to implement, usable and that would facilitate easy representation, storage and serialisation of membership functions.

According to [Farin et al. \(2002\)](#), Bezier curves provide a “geometric-based method for describing and manipulating polynomial curves and surfaces.” Bezier curves are parametric curves in which each point on the curve is a function of the parameter t . Bezier curves are defined by a number of control point that they interpolate. The curves begin at their first control point $x(0)$ and end at their last control point $x(1)$.

Bezier curves may be defined recursively as draw on for their implementation in the system. Given a Bezier curve $\mathbf{B}_{\mathbf{P}_0\mathbf{P}_1\dots\mathbf{P}_n}$ with points $\mathbf{P}_0\mathbf{P}_1\dots\mathbf{P}_n$ the recursive definition of a curve is:

$$B(t) = B_{P_0 P_1 \dots P_n}(t) = (1-t)B_{P_0 P_1 \dots P_{n-1}}(t) + tB_{P_1 P_2 \dots P_n}(t) \quad (4.1)$$

where $B_{P_0}(t) = P_0$ and $B_{P_n}(t) = P_n$.

D3 provides methods for manipulating SVG, which defines its ‘paths’ (Bezier curves) using control points. By providing the user with a collection of control point the can drag it is possible for them to define curves however they please. It also provides the advantage of allowing membership functions and output functions to be defined simply as control points in the data model. As bezier curves exist within the range 0 and 1 on the x and y axis the associated values must be scaled to the users desires. Users can input minimum and maximum values that are used to scale the graph. These are stored with the membership function in the JSON data structure for later processing. From this a single argument can be defined in JSON as in listing 4.2. In order to allow users to create membership functions quickly functionality was implemented to allow the user to save previously used functions as template functions as can be seen in figure 4.3.

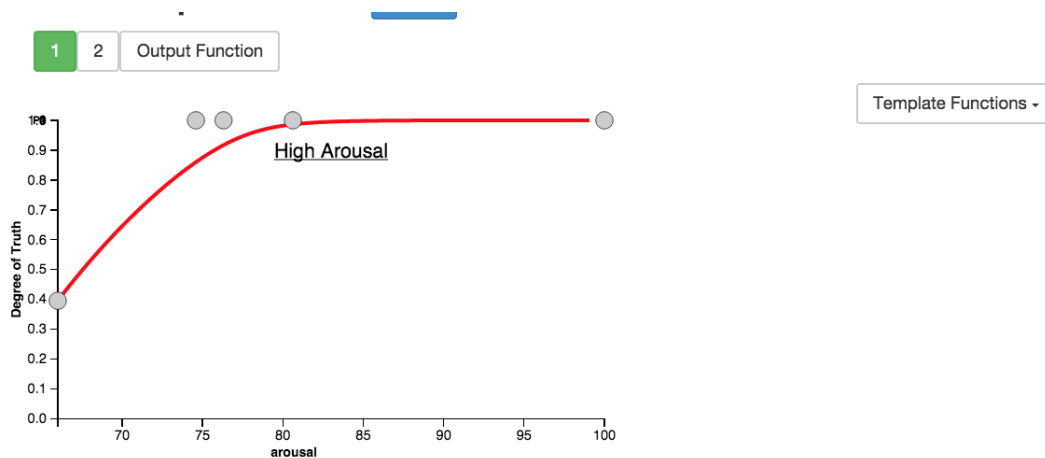


FIGURE 4.3: An interface that allows a user to ‘draw’ a membership function using Bezier curves

```
{
  id: 0,
  "name": "MD1",
  membership_functions: [{
    title: "Low Effort->Low MWL",
    points: [{x: 1, y: 25}, {x: 0, y: 0}, {x: 10, y: 0},
              {x: 20, y: 25}, {x: 22, y: 12}],
    xLabel: "Effort",
    yLabel: "Degree of Truth",
```



```
xMin: 0,
xMax: 50,
yMin: 0,
yMax: 50,
},
{
  title: "Low Performance->High MWL",
  points: [{x: 10, y: 140}, {x: 30, y: 0}, {x: 140, y: 0},
           {x: 200, y: 150}, {x: 125, y: 125}],
  xLabel: "Performance",
  yLabel: "Degree of Truth",
  xMin: 0,
  xMax: 250,
  yMin: 0,
  yMax: 300,
}
],
"output_function": {
  "title": "Underload",
  "xLabel": "Degree of Truth",
  "yLabel": "Mental Workload",
  "xMin": 0,
  "xMax": 1,
  "yMin": 0,
  "yMax": 33,
  "points": [
    {"x": 0.00454, "y": 33}, {"x": 0.222, "y": 24.849},
    {"x": 0.46136, "y": 16.954}, {"x": 0.65681, "y": 10.3458}, {"x": 1, "y": 0}
  ]
},
}
```

LISTING 4.2: JSON data structure a node including its fuzzy membership functions

There is a flaw in this approach that should be noted. In order to compute the users beliefs accurately the membership function should follow the strict mathematical definition of a function; that it should only output a single value for any input. Bezier curves do not obey this rule as they are parametrised by t and as a result it is possible for users to draw functions like in figure 4.4. It is also possible for users to define functions with no corresponding output for a given figure. If a user does the system will compute an incorrect value for the premise and the result will be compromised. Participants in the experiment are made aware of this before undertaking the experiment.

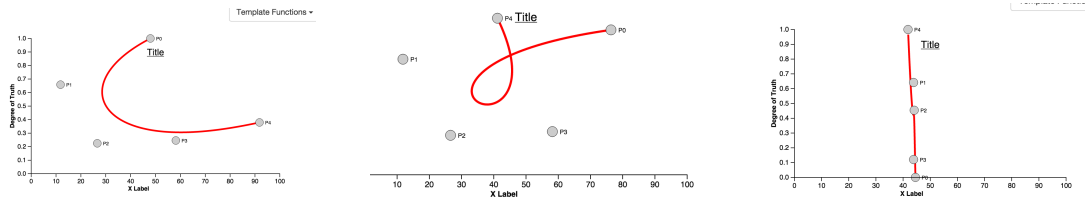


FIGURE 4.4: Examples of Bezier curves that can be drawn that are not functions

With the argumentation framework and membership function finally implemented the web interface appears as in figure 4.5.

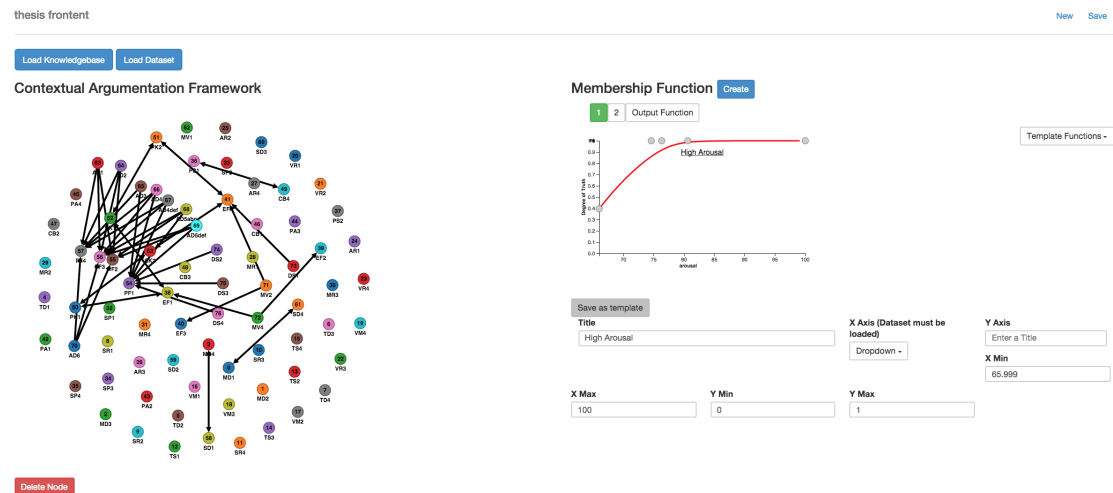


FIGURE 4.5: The fully developed tool for eliciting knowledge bases

Once a knowledge base has been elicited from a user they can test their with the system by computing results. They can download a sample data set which is displayed as in figure 4.6.

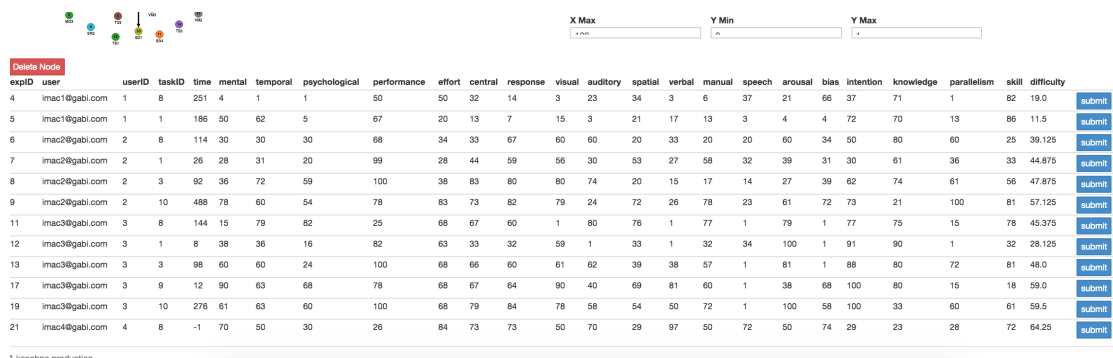


FIGURE 4.6: A selection of data for the user to test their knowledge base with

For a single row in the data the user can choose from a selection of semantics to run on the data (see figure 4.7).

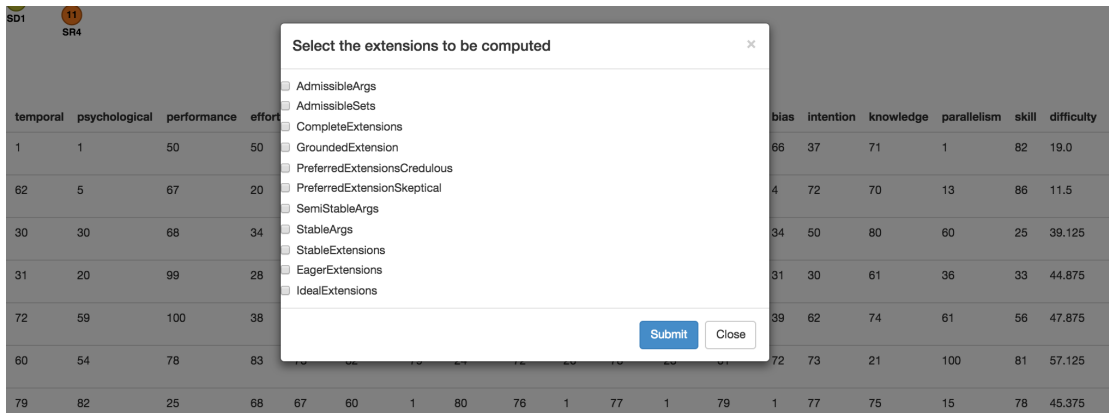


FIGURE 4.7: A list of options for semantics that can be computed by the system

URL	Request Method	Functionality
/	POST	Compute the Knowledge Base result for a given row of data
/knowledgebases/	GET	Return a list of Knowledge Bases saved on the server
/knowledgebases/:filename	GET	Retrieve a specific knowledge base named :filename
/knowledgebases/:filename	POST	Save a knowledge base in a JSON file named :filename
/datasets/	GET	Return a list of data sets saved on the server
/datasets/:filename	GET	Retrieve a specific data set csv file named :filename

TABLE 4.1: Documentation of the Application REST routes

The results of running the semantics are then presented to the user in the form given in figure 4.8. The results present an overall value for the construct as well as the total degree of truth of the semantic. Each argument that contribute to the semantic is returned as well as a degree of truth value and the value of the construct computed for that value. This extra information allows the user to scrutinise their knowledge base and its associated results further.

4.2.3 Application Back-End Implementation

The application back-end implements a REST architecture that allows the front-end to retrieve data via AJAX requests. A summary of the REST API is outlined in table 4.1.

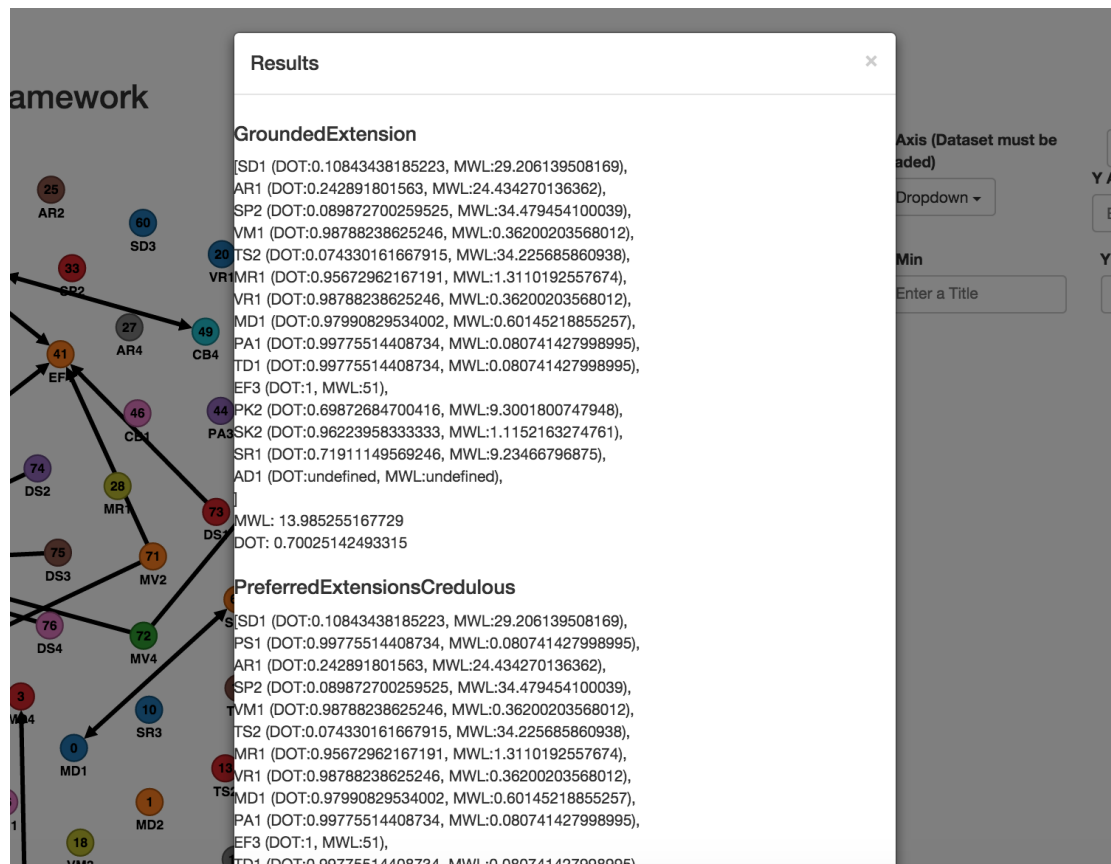


FIGURE 4.8: Results of computing the semantics on the framework. (Undefined values belong to mitigating arguments)

In order to compute the result for a single row in the data set the front end sends a POST request containing the knowledge base nodes and links, the row in the data and a list of semantics to be computed. A PHP class `FrameworkRunner` copies these values into a local list of arguments and attacks.

The first step in the process is to determine which arguments are activated. This is done by looking at each argument and determining if each of its premises are satisfied by the data in the row. For an individual premise the value of data associate with its x label is retrieved. If this value is between the premise's x_{Min} and x_{Max} values then the premise is relevant. If the value falls outside these bounds then the parent argument and its associated attack relations are discarded.

The sub-graph of activated arguments and their attack relations remains in the class to compute semantics for. In order to compute the semantics the Dung-o-matic Java class is wrapped in another class that allows it to read a list of arguments and attacks as a

string and return the results in a similar format. The `FrameworkRunner` class produces a string containing the IDs of the nodes in the format expected by this Java class. The Java class has been exported as an executable JAR file which is then called from PHP using the `exec()` function and passing this simple string as an argument. The Java code returns JSON object with the name of the computed semantic as keys and the sub-graphs as an array of arrays associated with those keys.

Once the semantics have been computed for an Argumentation Framework the values associated with each argument can be computed. For a given row in the data set, the results for an argument remain the same no matter what the configuration of the framework is. This allows us to cache the output value of an argument in memory rather than having to recompute the same values for each semantic. The results are computed on the server using a Bezier curve implementation.

The Bezier curve computation is implemented in a PHP class `Bezier`. This class contains methods to compute the X and Y values for a point on the line given a list of control points and a value for t . It provides two other methods `yFromX` and `xFromY` that compute a value for Y given X and vice versa. The Bezier curve implementation is based on the recursive definition given in equation 4.2.2. Its implementation is given in Algorithm 1.

Algorithm 1 Computing a point on the curve for a given value of t

Data: A list of control points $L = \mathbf{P}_0\mathbf{P}_1 \dots \mathbf{P}_n$ and a value t

Result: A point $P = (X, Y)$ corresponding to t

Algorithm `bezier(L, t)`

```

if there is only one control point in L then
  | return  $\mathbf{P}_0$ 
else
  |  $\mathbf{P}_0 \leftarrow \text{bezier}(\mathbf{P}_0\mathbf{P}_1 \dots \mathbf{P}_{n-1}, t)$ 
  |  $\mathbf{P}_1 \leftarrow \text{bezier}(\mathbf{P}_1\mathbf{P}_2 \dots \mathbf{P}_n, t)$ 
  |  $X \leftarrow (1-t)\mathbf{P}_{0X} + t\mathbf{P}_{1X}$ 
  |  $Y \leftarrow (1-t)\mathbf{P}_{0Y} + t\mathbf{P}_{1Y}$ 
  | return  $(X, Y)$ 
end

```

For a given membership function with fixed values for its control points any point on the curve can be described by a value t between 0 and 1. By passing t into the function we obtain a value for x and y . As it is not possible to simply pass in an x value and obtain a corresponding y value we must search for y by varying the parameter t . This is done efficiently using a binary search algorithm. For each iteration we pass in two values of t to get two values of x and compare them with our target x value. We continue to search a space closer and smaller to x until we arrive at a value that is within a threshold we consider to be satisfactory. From this point we can obtain the Y value. The pseudocode for this algorithm is given in Algorithm 2.

Algorithm 2 Obtaining a value for Y given an X value

Data: A list of control points $L = \mathbf{P}_0\mathbf{P}_1 \dots \mathbf{P}_n$, a tolerance T and a value X

Result: A value for Y corresponding to the input X

$t_{lower} \leftarrow 0$

$t_{upper} \leftarrow 1$

while *The computed X values are outside the tolerance T of X* **do**

$P_{lower} \leftarrow \text{bezier}(L, t_{lower})$

$P_{upper} \leftarrow \text{bezier}(L, t_{upper})$

if P_{lowerX} is closer to X than P_{upperX} **then**

$t_{upper} \leftarrow t_{upper} + t_{lower}$

else

$t_{lower} \leftarrow t_{upper} + t_{lower}$

end

end

This calculation is performed for each membership function in the node and an average of the values of the output is produced. This corresponds to the overall degree of truth for the argument. This value is then used as the input to the output function which computes an overall construct value for that argument. If the output function of an argument is labeled as a mitigating argument then that argument is ignored.

Once the all of the argument construct values and degrees of truth are taken the overall results for the semantic can be computed. An overall construct value and degree of truth for the semantic is computed by taking the average of these values for each argument in

the semantic. The client presents this information to the user in the evaluation interface previously presented.

The software was manually validated in order to determine that it was functioning correctly. Initially simple test cases were developed using the ASPARTIX online interface⁹ in order to validate that the argument evaluation was working correctly. Once the interface was working for small test cases it was evaluated by hand using an existing knowledge base. A number of bugs were discovered and fixed at this stage. The process continued iteratively until the implementation was considered to be of a quality appropriate to the experiment.

4.3 Experiment Implementation

In order to provide a comparison of the ability of the defeasible approach to model a construct the knowledge bases of both an expert and a lay person were modeled using the tool developed in this chapter. The two knowledge bases provide a contrasting result and it is expected that the knowledge base of the expert will perform better than that of the lay person. These knowledge bases were saved on the server and evaluated in order to determine that they matched the expert's expectations.

⁹<http://rull.dbai.tuwien.ac.at:8080/ASPARTIX/index.faces>

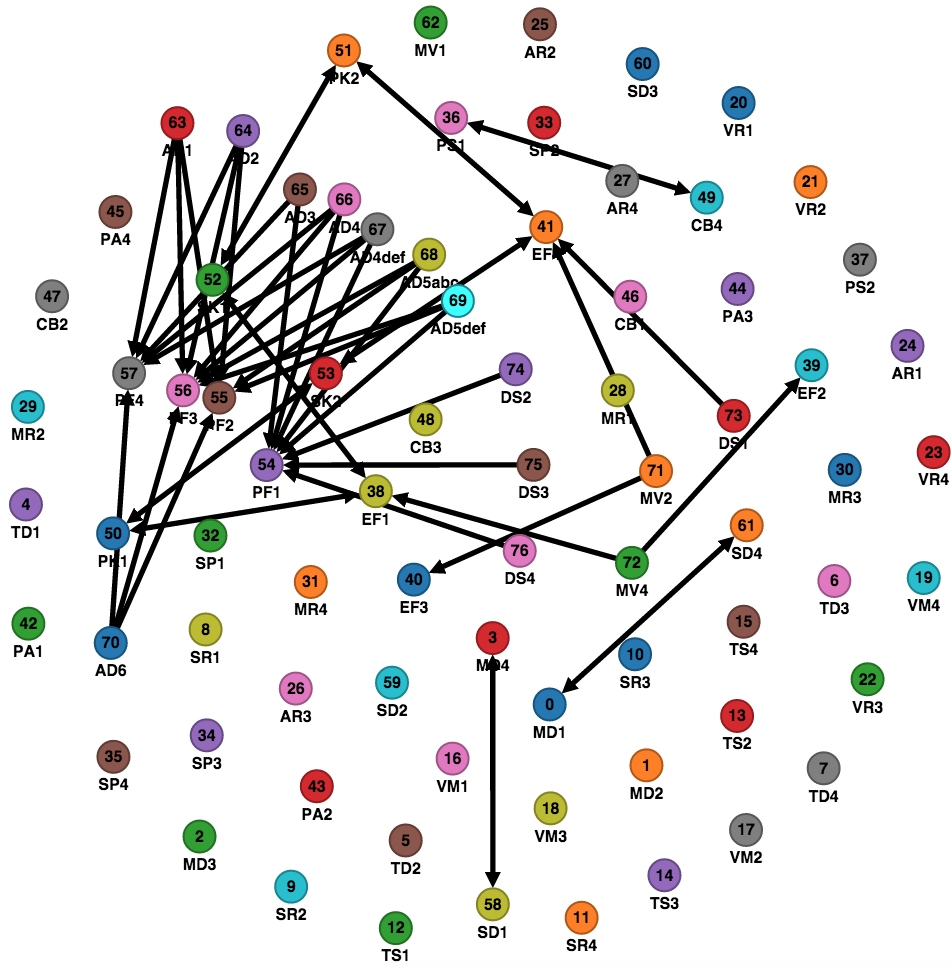


FIGURE 4.9: The Argumentation Framework developed by the expert using the tool

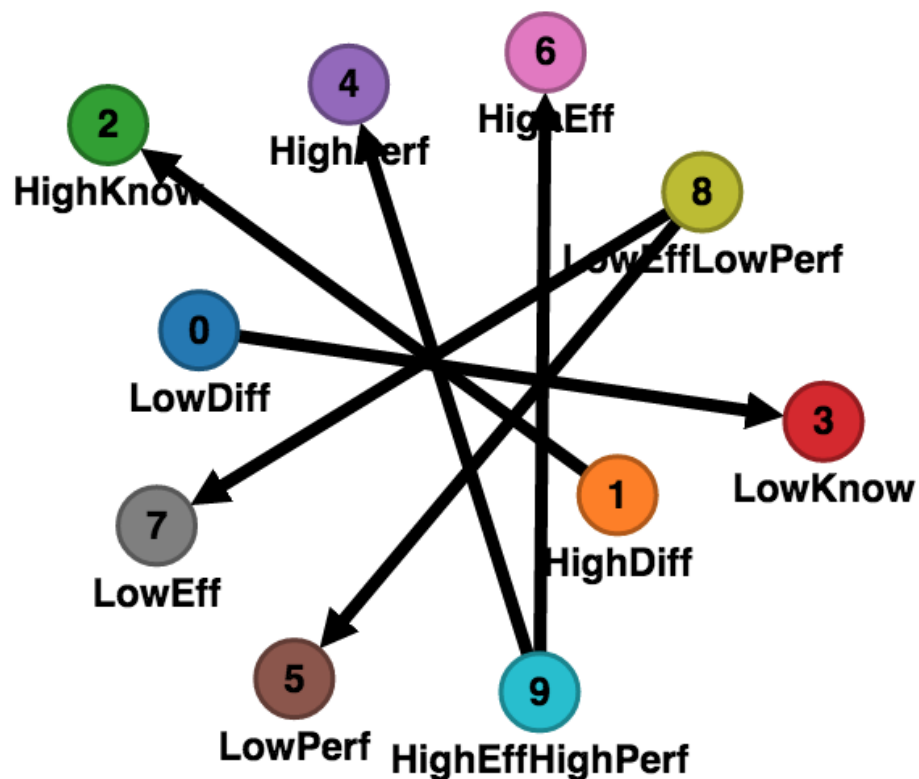


FIGURE 4.10: The Argumentation Framework developed by the non-expert using the tool

As computing the semantics for an AF is an NP complete problem computing the results for a whole data set carries a large overhead for time. The time is considerably larger than the time of a typical HTTP request-response cycle so it was not feasible to compute all of the results for the data set in the web application format.

In order to generate the results of running a knowledge base on a whole data set the `FrameworkRunner` tool was wrapped in a command line tool written in PHP. Results were computed for the ‘grounded’ and ‘preferred’ extensions of the argumentation frameworks. The overall values for MWL for the extensions were collected for each row in the data. The time taken to compute these results was also collected in order to have an objective measure of the performance of the technique.

The second part of the research question was then answered by running machine learning algorithms on the experiment data using Weka. In order to assess regression algorithms task time was chosen as the dependant variable. Classification algorithms were testing using task ID as the dependent variable.

The regression algorithms used were additive regression, k-star, linear regression, multilayer perceptron, regression by simple linear regression and additive regression. These results were collected for each regression algorithm: correlation coefficient, mean absolute error, root means squared error, relative absolute error and root relative squared error.

In order to run classification algorithms on data, Weka requires the dependent variable to be in a string format. A simple python script was written to create a new column in the data set that would have the task numbers represented as letters ($1 \leftarrow A, 2 \leftarrow B, \dots$). The classification algorithms used were bayes net, decision table, logistic regression, naive bayes and multilayer perceptron. The following metrics for these algorithms were collected: percentage of correctly classified instances, percentage of incorrectly classified instances, Kappa statistic, mean absolute error, root mean squared error, relative absolute error and root relative squared error. Confusion Matrices and a breakdown of accuracy by class (including TP Rate, FP Rate, Precision, Recall, F-Measure and ROC Area) was also collected for each model.

In order to determine the concurrent validity of the defeasible MWL models two regressions needed to be run on each model. The first regression is a linear regression using the values for MWL against time. The second is a logistic regression using the values for MWL against task number. Lastly in order to determine the convergent validity of the MWL values the Pearson co-relation of the values with other measured of MWL was determined.

4.4 Conclusions

This chapter outlined the deployment of a piece of software and an experiment to implement in practice the theoretical model designed in Chapter 3. Additionally, an experiment that uses this software has been executed. The process of realising a tool used to elicit and perform computations on a defeasible knowledge base was not a trivial task. Several challenges were encountered:

- The implementation of membership functions required that functions could be defined graphically. This required that the view rendering logic be decoupled from the client data. An implementation of Bezier curve calculations were required on the server.
- As time and resources (implementations of semantic computation) were limited the project required that Java and PHP were integrated in the same application. This required writing wrappers for the Java code and a more complex deployment process.
- As the application initially performed poorly when computing a number of semantics, performance bottlenecks were identified. One was the computation of Bezier curves. This was improved by caching the previously computed results for the curves.

The chapter briefly discussed the implementation of an experiment using the software and the machine learning work-bench Weka. Finally, the collection of extra data to analyse the experiment results was briefly discussed. The analyses of the information that has been gathered is the subject of the next chapter.

Chapter 5

Evaluation of Results and Discussion

This chapter discusses the ability of an implementation of defeasible reasoning to model a construct in comparison with machine learning. The chapter presents and discusses the results of the experiment performed in the previous chapter. A comparative analysis of both techniques is performed using the evaluation techniques gathered in the literature review.

5.1 Results

The research question posed at the start of this project is focused on the ability of two techniques to represent a construct and make predictions based on this representation. The predictive capacity of the techniques is presented here. In order to determine how these new measures perform, the concurrent and convergent validity of the new values is calculated using existing measures of mental workload.

5.1.1 Predictive Capacity - Numeric

In order to compare the predictive capacity of the approaches each technique was used to compute the value of an objective performance measure: time. The mean absolute error for time prediction was taken for each technique. In order to gather a wider comparison

the size of the data used for training and testing was adjusted by varying the number of folds and percentage of split. Figure 5.1 displays the results of the experiment. Immediately obvious in this graph is the poor performance of Artificial Neural Networks. According to [Silvert and Baptist \(2000\)](#) ANNs typically require large amount of data for training. The performance of the ANN could be improved by pre-processing the data. It is also possible that the ANN has overfit the training data. This could be determined by evaluating the minimum description length of the model, however, this is beyond the scope of the thesis as it is not possible to compare that measure with the knowledge based approach.

The performance of the machine learning algorithms vary dramatically. It is interesting to note that simple linear regression using one variable doesn't out perform the regression based on the non-expert knowledge base which is also based on one variable. Decision table, kstar and additive regression have error rates higher than the expert knowledge base. Decision table and kstar don't correlate well with time although additive regression performs nearly as well as the expert knowledge base. A linear regression outperforms all of the techniques and can predict task time with the strongest correlation. With the exception of linear regression, the machine learning techniques all show a large variability based on the amount of data that is available for training. KStar and additive regression are outperformed by linear regression.

Table 5.1 and table 5.2 gives a breakdown of the results that provides more clarity than the previous figures. For different numbers of folds we can see that the regressions based on the expert's knowledge base perform best of all which validates the first hypothesis of the experiment. What is interesting is that the predictions based on the knowledge of the non-expert perform better than that of the expert for varying percentage of split. As the knowledge base of the non-expert contains fewer nodes it is possible that it weighs some variable heavily that contributes greatly to task time.

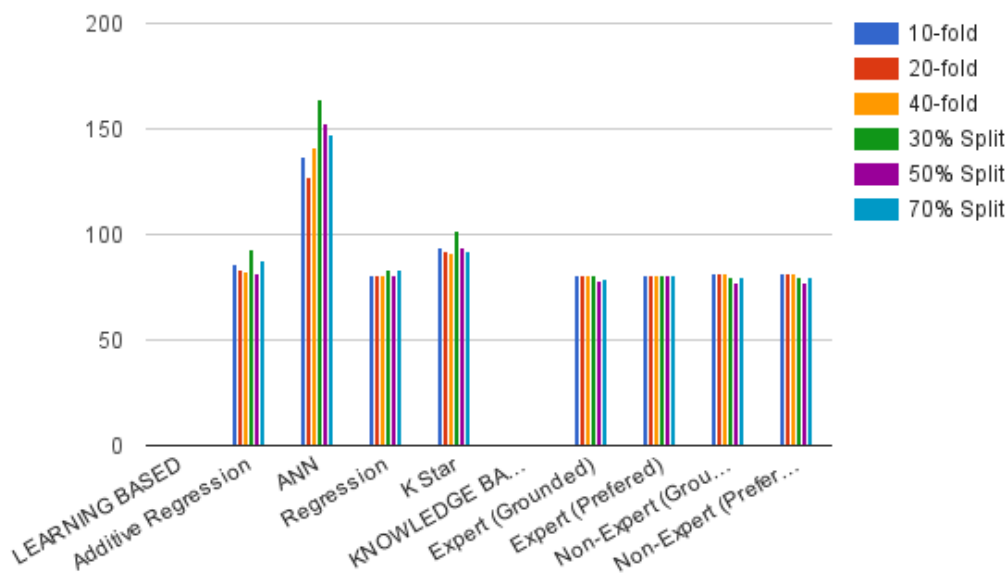


FIGURE 5.1: Mean Absolute Error for Time Prediction

	10-fold	20-fold	40-fold
ANN	137.238	126.9808	141.3834
K Star	93.6672	92.1658	91.1066
Additive Regression	85.8072	83.5995	82.0641
Regression	80.8231	80.9963	80.2697
Expert (Grounded Extension)	80.6007	80.5986	80.5241
Expert (Preferred Extension)	80.4185	80.4475	80.3192
Non-Expert (Grounded Extension)	81.5565	81.6743	81.7295
Non-Expert (Preferred Extension)	81.5565	81.6743	81.7295

TABLE 5.1: Mean Absolute Error Using K-Fold Cross Validation

	30% Split	50% Split	70% Split
ANN	163.7684	152.5288	147.3721
K Star	101.9482	93.8966	92.0756
Additive Regression	93.3442	81.2395	87.8533
Regression	83.1717	80.868	83.589
Expert (Grounded Extension)	81.0625	78.0558	78.7381
Expert (Preferred Extension)	80.4185	80.4475	80.3192
Non-Expert (Grounded Extension)	79.5426	77.2909	79.9484
Non-Expert (Preferred Extension)	79.5426	77.2909	79.9484

TABLE 5.2: Mean Absolute Error Varying Percentage of Split

5.1.2 Predictive Capacity - Task Membership

A comparison of the techniques ability to predict task membership was also performed. The performance of each measure was evaluated simply using 10-fold cross validation. This was chosen as there are a larger number of evaluation metrics for classifiers that are being considered here than for the numeric prediction task. The first metric that is examined is the number of correctly and incorrectly classified instances which is shown in figure 5.2. This metric shows that ANN, Naive Bayes and Logistic Regression perform best in terms of correctly classified instances. This nullifies the hypothesis that the knowledge base of the expert predicts task membership better than machine learning. In this case the knowledge base of the expert actually performed worse than the knowledge base of the non-expert.

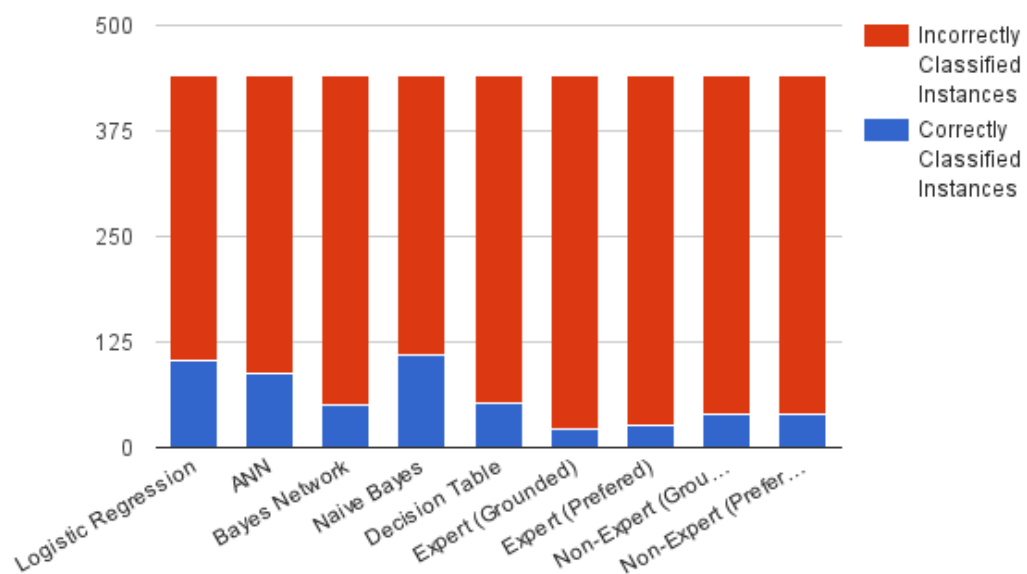


FIGURE 5.2: Classified Instances - Task ID

By examining the precision and recall of the classifiers we can gain further insight into their performance. The best performing machine learning tasks have about the same precision and recall. We can see a large variation in the precision and recall of the expert and non-expert. The recall of the non-expert is higher than that of the expert resulting in a greater number of correctly classified instances.

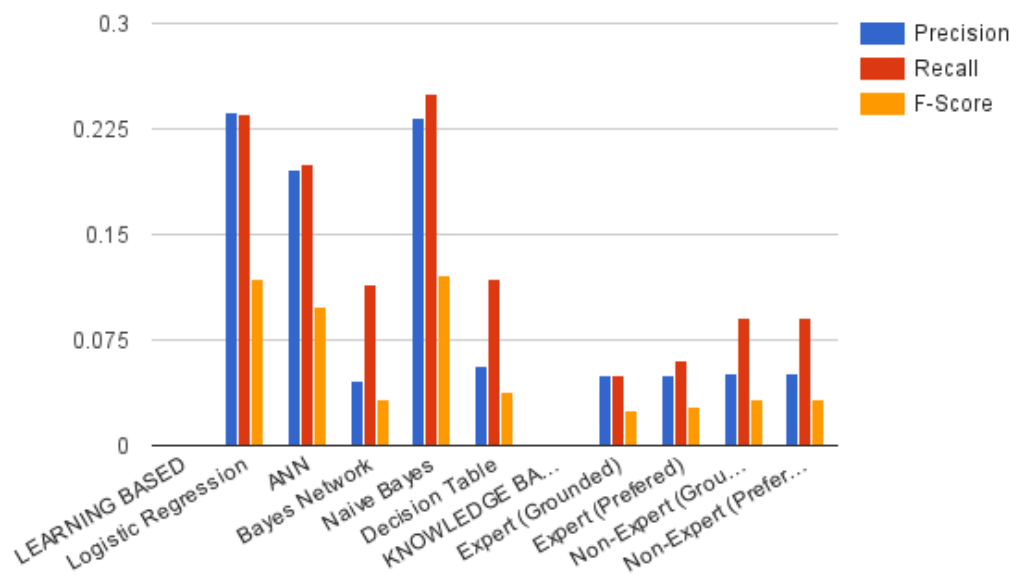


FIGURE 5.3: Precision, Recall and F-Score

The area under the ROC curve provides further clarification of the performance of the classifiers as it takes into account the number of false positives. We can see that while the non-expert may have classified more instances correctly, the expert's knowledge base provides a greater balance between true positives and false positives.

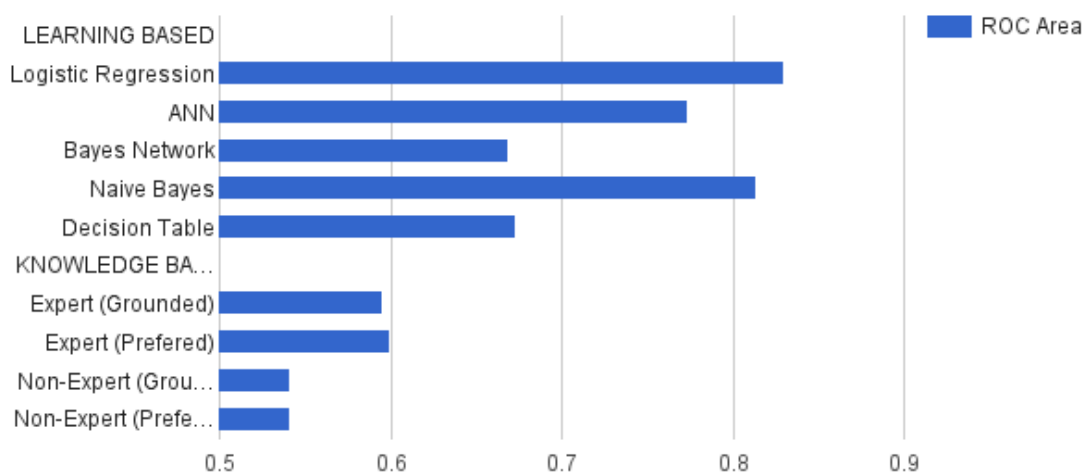


FIGURE 5.4: Area Under ROC Curve

It is interesting that the machine learning approach was more effective in determining task membership than the defeasible reasoning approach. This could be because the tasks don't vary MWL strongly enough. Another possibility is that the knowledge bases are failing to take into account some aspect of mental workload that could determine task membership better.

5.1.3 Concurrent Validity

The concurrent validity of the defeasible constructs was measured by performing regressions against time (tables 5.3). Time is an objective performance measure that is similar to MWL. The Pearson correlation provides an understanding of the performance of each knowledge base. The results show a much stronger correlation between the knowledge base of the expert and time than the knowledge base of the non-expert. This provides some confirmation that the implementation is modeling defeasible knowledge bases correctly as it is expected that the knowledge base of the expert should more accurately represent MWL than that of the non-expert. The preferred extension of the knowledge base of the expert performed better than the grounded extension. The preferred extension and the grounded extension of the non-expert were determined to be equivalent by the software and so show the same results.

	Grounded Extension	Preferred Extension
Expert	0.3362	0.3414
Non-Expert	0.2046	0.2046

TABLE 5.3: Concurrent Validity - Pearson Correlation with Time

5.1.4 Convergent Validity

In order to determine how well the constructs actually modeled MWL the convergent validity of these measures was determined. The convergent validity was determined by computing the correlation of each measure with existing measures of MWL. These existing measures are the NASA Task Load Index and the Workload Profile.

It can be seen that the expert's knowledge base correlates strongly with the other measures for MWL (table 5.4). There is a moderate correlation between other measures of MWL and the non-expert's knowledge base. This strong correlation reinforces the belief that defeasible reasoning can accurately model a construct.

	NASA	WP
expertGroundedExt	0.7233711	0.8593995
expertPref1	0.7247067	0.8499664
non-expertGroundedExt	0.5518035	0.5648483
non-expertPref1	0.5518035	0.5648483

TABLE 5.4: Pearson correlation of measures of MWL

5.2 Summary and Final Recommendations

The results presented in this chapter were collected in order to test the designed hypotheses:

1. The defeasible measure of a construct created by an expert is better at predicting objective performance values than machine learning approaches.
2. The construct measure of an expert will have a high concurrent validity with objective measures related to the construct.
3. The construct measure of an expert will have a high convergent validity with existing measures for the construct.

5.2.1 First Hypothesis

The first hypothesis was examined by comparing the capacity of the techniques to make predictions of two objective values: task time and task ID.

In the case of task time, the knowledge base of the expert proved to be the best for prediction. This supports the hypothesis. On the other hand the knowledge based approach performed worse for the prediction of task membership. These contradictory results

require further investigation. The time prediction experiment should be further refined by including a greater number of knowledge bases and learners. A more statistically valid comparison of the task ID and MWL value could be performed, for example by examining the distribution of MWL values with respect to the task ID.

As the expert and non-expert knowledge bases perform similarly, it is difficult to tell why the approaches perform the way they do. It is possible that as regressions are performed, the use of one independent variable is more useful than using many. Future work might include a comparison between knowledge bases that use intentionally bad premises to model MWL and expert ones.

5.2.2 Second Hypothesis

The second hypothesis was examined by determining the concurrent validity of the measures of mental workload with time, an objective performance measure. It was found that the knowledge base of the expert had a statistically significant correlation with time, however, it was not a very strong one. This can be explained by considering the construct of MWL as a defeasible phenomenon. The time spent on a task could be high, however, the participant may not be very motivated or may be distracted. In this case the MWL would in fact be low. A short coming of machine learning is highlighted here. If we do not believe time to accurately model MWL in all cases we must create a new model. This would require an expert to apply labels to the data in a time consuming procedure.

5.2.3 Third Hypothesis

The third hypothesis was examined by determining the convergent validity of the measures of MWL with other existing measures of MWL: NASA-TLX and Workload Profile. It was found that the construct measure developed by the expert had a high convergent validity with existing measures of MWL. It was also found that the construct measure of the expert had a higher convergent validity than that of a non-expert. This

reinforces our belief that the defeasible modeling of the construct is working effectively. Again, it should be pointed out that this modeling is not possible in supervised machine learning without applying labels to the data. An underlying flaw in this approach is that it is assumed that the existing measures correctly model the construct. These measures may be proven invalid in the future and so if such a system were to be adopted it is important to consider this. The accuracy of the diagnoses made by the knowledge based system are entirely influenced by the knowledge used. The system cannot automatically detect that the mappings from premises to conclusions are incorrect. On the other hand, it is hoped that through using the system and obtaining visual feedback a user might improve their understanding in their area of expertise.

5.2.4 Recommendations

These findings support the case for the implementation of defeasible reasoning as an alternative to learning based approaches.

The main advantage of learning based systems is that the learning is automatic. If given enough data and enough computing power machine learning can tackle many problems well. However, the available data sets are often not comprehensive enough or representative of the wider picture. This can result in models that are too localised and unable to predict exceptional cases. Data often must be cleaned and labeled, a time consuming and expensive process. Knowledge based approaches offer an alternative to this. The results of the experiments suggest that knowledge based approaches offer a viable alternative for prediction. While the time to input a knowledge base is non-trivial, it is less than the time required for labeling in many cases.

One of the shortcomings of using convergent validity for assessment is that the existing measures of the construct must also model the construct accurately. This means that while in this particular instance we have shown a strong concurrent validity with existing constructs, this may not always be the case. In order to further verify that the system is working as intended more experiments need to be performed with a wider variety of knowledge bases and data sets. If it is then verified that the system works as

intended it may then be used to develop alternative measures for constructs that deviate from previous measures.

What has not been examined is the role that the system feedback could play in improving an expert's understanding. The machine learning techniques typically represent their findings as numeric and mathematical models. In the DR system it is possible to represent these visually. Comparing these representations and how helpful they are in assisting an expert could be the subject of future work.

Chapter 6

Conclusions and Future Work

This chapter revisits the aims of the projects. The key results of the research are summarised and their significance to answering the research question is discussed. The contribution of the research is made clear. Finally, areas for future research are outlined.

6.1 Problem Definition and Research Overview

The central motive of this project is to compare and contrast machine learning with an implementation of defeasible reasoning. This research was motivated by a lack of comparison between knowledge based approaches and learning based approaches. Specifically, the project examined the ability of the two techniques to measure a construct; in this case mental workload. Constructs by their nature are abstract and difficult to measure, so evaluating the performance of the techniques empirically is also a challenge. In order to determine that the techniques performed well two measures were taken which are widely used to assess the accuracy of measures; concurrent validity and convergent validity. Neither of these measures can conclusively tell us that the results are correct; they can only suggest to us that we are on the right path.

6.2 Experimentation, Evaluation and Limitations

Although there was a limited amount of data to train and evaluate the techniques with, the results from the experiment can still provide us with some insight into the question posed at the start of the project. A core finding of this thesis is that supervised machine learning techniques are limited to learning to make predictions based on labeled training data. The only way to predict construct measures is to choose a field in the data and decide that this accurately measures the construct we are interested in. In the experiment, it was decided that MWL could be modeled using an objective performance measure, time. This approach was flawed, as it was shown that time had a poor convergent validity as a measure for MWL. From a practical point of view, this would suggest that if machine learning is to be used in an application to predict a construct, the data should be labeled or a column with a high convergent validity with other measures of the construct should be used.

The convergent validity of the experts knowledge base was very high with respect to an existing measure of MWL. The non-expert's knowledge base had a high convergent validity with the existing measure of MWL as well, however, it was not nearly as strong as the expert's. This suggests that the implementation was providing a good representation of the construct and demonstrates the strength of defeasible reasoning for use in this situation.

The concurrent validity of the approaches was measured by their ability to predict an objective performance measure, time, and another objective value, task membership. It was believed that MWL would vary consistently across tasks as some tasks were designed to be more difficult than others. Predictions of task time made based on the MWL values computed from the expert's knowledge base were better than those of the amateur and all but one machine learning approach. Over all, linear regression performed best at predicting task time. Predictions of task membership based on values for MWL from the knowledge based approach were found to be inaccurate. In this case logistic regression and naive bayes performed best. It is possible that task membership has a weak relationships with MWL. Unfortunately, as the task IDs are discrete values it is not possible to determine their convergent validity with other MWL measures.

The performance of machine learning was likely hindered significantly by the limited amount of data available for training. It would be interesting to perform the experiments again with a larger data set. This highlights another strength of the defeasible reasoning approach, no training based on data is required. It is noted that the ability to verify a knowledge base against a data set is useful in the implementation of such a system as in this project. This requires a significantly smaller data set than that required to train a machine learning model. One criticism of knowledge based approaches is that they allow predictions to be made based on anecdotal evidence and not hard data. This is alleviated somewhat by the verification step in our approach.

It was noted that this implementation of defeasible reasoning was particularly slow; implementation reasons for this have been identified. This approach may not be ready for real world implementation today, however, a hybrid approach using AT for data set labeling could be adopted as outlined in Appendix B. One of the main culprits of performance issues was the Dung-o-matic inference engine. It has been noted that other faster implementations exist but were not ready or available for integration in this project. This highlights a greater need within this argumentation theory community for an open source implementation of a library for computing argument semantics. This sentiment is shared by other authors who's work has been highlighted in the literature review.

The implementation of defeasible reasoning used in this project is (to my knowledge) the first implementation to adopt a graphical diagramming approach that is also used for computation of results. This has several possible advantages that could be investigated as the subject of future research. An expert could be trained in using the tool to elicit their knowledge base graphically which is likely more intuitive than inputting their knowledge base using some domain specific language. It is possible that in a similar fashion to the findings of [Twardy \(2004\)](#), experts could improve their critical reasoning around their domain through interaction with the tool. It is also likely that through visualisation the comparison of knowledge bases will be made easier.

6.3 Contributions to the body of knowledge

The contributions of this work are outlined as follows:

- This project has highlighted a shortcoming of machine learning, the measurement of constructs, that can be better accomplished using a DR based approach.
- A key strength of argumentation theory over machine learning has been highlighted; that AT can allow experts to develop and evaluate constructs in an intuitive way.
- The project provides a generalisable implementation of defeasible reasoning that is user friendly. It is the first implementation (to my knowledge) that integrates an argument diagramming approach with the computation of results.
- This implementation and its design can provide guidance to engineers and academics that seek to integrate these approaches into applications or their research.

6.4 Future Work and Research

- In order to provide greater insight into the problem at hand the experiment could be carried out again with larger data sets across using different constructs.
- Further steps should be taken to optimised the DR implementation, including possible research into the efficient computation of argument semantics.
- This project focused exclusively on supervised machine learning. An exploration of construct representation with unsupervised machine learning might provide new insights that have been ignored in this research.
- The application developed in this project could itself be the subject of further research. The usability of the interface could be examined and whether it provides advantages in the elicitation of knowledge bases. In a similar manner to the work conducted by [Twardy \(2004\)](#) it could be interesting to examine whether or not

an expert's understanding of their domain was improved through the knowledge elicitation process.

Appendix A

Details of Experiment Data

The data set has the following columns (shown with example rows):

expID	user	userID	taskID	time
4	imac1@gabi.com	1	8	251
5	imac1@gabi.com	1	1	186
6	imac2@gabi.com	2	8	114
7	imac2@gabi.com	2	1	26

TABLE A.1: Sample data: Columns 1 - 5

mental	temporal	psychological	performance	effort
4	1	1	50	50
50	62	5	67	20
30	30	30	68	34
28	31	20	99	28

TABLE A.2: Sample data: Columns 6 - 10

central	response	visual	auditory	spatial
32	14	3	23	34
13	7	15	3	21
33	67	60	60	20
44	59	56	30	53

TABLE A.3: Sample data: Columns 11 - 15

verbal	manual	speech	arousal	bias
3	6	37	21	66
17	13	3	4	4
33	20	20	60	34
27	58	32	39	31

TABLE A.4: Sample data: Columns 16 - 20

intention	knowledge	parallelism	skill	difficulty
37	71	1	82	19.0
72	70	13	86	11.5
50	80	60	25	39.125
30	61	36	33	44.875

TABLE A.5: Sample data: Columns 20 - 25

expID	An ID that uniquely identifies the experiment instance
user	The participant's email address
userID	The participant's unique ID
taskID	The experiment task ID undertaken by the participant
time	The time taken by the participant to complete the task
mental	The mental demand of the task, the answer to the question "How much mental and perceptual activity was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.)? Was the task easy (low mental demand) or complex (high mental demand)?" on a scale from 0 - 100
temporal	The temporal demand, the answer to the question "How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely (low temporal demand) or rapid and frantic (high temporal demand)?" on a scale from 0 - 100
psychological	The frustration felt while completing the task, the answer to the question "How secure, gratified, content, relaxed and complacent (low psychological stress) versus insecure, discouraged, irritated, stressed and annoyed (high psychological stress) did you feel during the task?"
performance	The effort expended completing the task, the answer to the question "How successful do you think you were in accomplishing the goal of the task? How satisfied were you with your performance in accomplishing the goal?" on a scale from 0 - 100
effort	The effort expended completing the task, the answer to the question "How much conscious mental effort or concentration was required? Was the task almost automatic (low effort) or it required total attention (high effort)?" on a scale from 0 - 100
central	The answer to the question "How much attention was required for activities like remembering, problem-solving, decision-making and perceiving (eg. detecting, recognizing and identifying objects)?" on a scale from 0 - 100
response	The answer to the question "How much attention was required for selecting the proper response channel and its execution?(manual - key- board/mouse, or speech - voice)" on a scale from 0 - 100
visual	The effort expended completing the task, the answer to the question "How much attention was required for executing the task based on the information visually received (through eyes)?" on a scale from 0 - 100
auditory	The effort expended completing the task, the answer to the question "How much attention was required for executing the task based on the information auditorily received (ears)?" on a scale from 0 - 100

TABLE A.6: Explanation of data columns 1 - 14

spatial	Spatial workload, the answer to the question “How much attention was required for spatial processing (spatially pay attention around you)?” on a scale from 0 - 100
verbal	Verbal workload, the answer to the question “How much attention was required for verbal material (eg. reading or processing linguistic material or listening to verbal conversations)?” on a scale from 0 - 100
manual	Manual effort, the answer to the question “How much attention was required for manually respond to the task (eg. keyboard/mouse usage)?” on a scale from 0 - 100
speech	The effort expended through speech, the answer to the question “How much attention was required for producing the speech response(e.g. engaging in a conversation or talk or answering questions)?” on a scale from 0 - 100
arousal	The degree to which the participant was aroused, the answer to the question “Were you aroused during the task? Were you sleepy, tired (low arousal) or fully awake and activated (high arousal)?” on a scale from 0 - 100
bias	The context bias, “How often interruptions on the task occurred? Were distractions (mobile, questions, noise, etc.) not important (low context bias) or did they influence your task (high context bias)?” on a scale from 0 - 100
intention	The effort expended completing the task, the answer to the question “Were you motivated to complete the task?” on a scale from 0 - 100
knowledge	The knowledge of the user before the experiment, the answer to the question “How much experience do you have in performing the task or similar tasks on the same website?” on a scale from 0 - 100
parallelism	The degree to which a user multi-tasked, the answer to the question “Did you perform just this task (low parallelism) or were you doing other parallel tasks (high parallelism) (eg. multiple tabs/windows/programs)?” on a scale from 0 - 100
skill	The effect of the participants skill level on completing the task, the answer to the question “Did your skills have no influence (low) or did they help to execute the task (high)?” on a scale from 0 - 100
difficulty	$\frac{1}{8}$ ((solving/deciding) + (response) + (task/space) + (verbal material) + (visual resources) + (auditory resources) + (manual response) + (speech response))

TABLE A.7: Explanation of data columns 15 - 25

Bibliography

- Akerkar, R. and Sajja, P. (2010). *Knowledge-based systems*. Jones & Bartlett Publishers.
- Alpaydin, E. Introduction to machine learning. 2004. *Cover, Copyright Page, Table of Contents for*, pages 1–327.
- Arel, I., Rose, D. C., and Karnowski, T. P. (2010). Deep machine learning—a new frontier in artificial intelligence research [research frontier]. *Computational Intelligence Magazine, IEEE*, 5(4):13–18.
- Baroni, P., Caminada, M., and Giacomin, M. (2011). An introduction to argumentation semantics. *The Knowledge Engineering Review*, 26(04):365–410.
- Baroni, P., Guida, G., and Mussi, S. (1997). Full non-monotonicity: a new perspective in defeasible reasoning. In *European Symposium on Intelligent Techniques*, pages 58–62.
- Bechhofer, S. and Horrocks, I. Hoolet: An owl reasoner with support for rules (2004). URL <http://owl.man.ac.uk/hoolet>.
- Beidel, D. C. (1996). Assessment of childhood social phobia: Construct, convergent, and discriminative validity of the social phobia and anxiety inventory for children (spa-c). *Psychological Assessment*, 8(3):235.
- Bench-Capon, T. J. and Dunne, P. E. (2007). Argumentation in artificial intelligence. *Artificial intelligence*, 171(10-15):619–641.
- Bistarelli, S., Rossi, F., and Santini, F. Benchmarking hard problems in random abstract afs: The stable semantics.
- Bistarelli, S., Rossi, F., and Santini, F. (2013). A first comparison of abstract argumentation systems: A computational perspective. In *CILC*, pages 241–245.

- Bostock, M., Ogievetsky, V., and Heer, J. (2011). D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*.
- Bryant, D. and Krause, P. (2008). A review of current defeasible reasoning implementations. *The Knowledge Engineering Review*, 23(03):227–260.
- Campbell, D. T. and Fiske, D. W. (1959). Convergent and discriminant validation by the multitrait-multimethod matrix. *Psychological bulletin*, 56(2):81.
- Cerutti, F., Giacomini, M., and Vallati, M. Generating challenging benchmark afs.
- Cleary, J. G., Trigg, L. E., et al. (1995). K*: An instance-based learner using an entropic distance measure. In *Proceedings of the 12th International Conference on Machine learning*, volume 5, pages 108–114.
- Cowan, R. (2001). Expert systems: aspects of and limitations to the codifiability of knowledge. *Research Policy*, 30(9):1355–1372.
- Davis, R., Shrobe, H., and Szolovits, P. (1993). What is a knowledge representation? *AI magazine*, 14(1):17.
- Dimopoulos, Y., Nebel, B., and Toni, F. Finding admissible and preferred arguments can be very hard.
- Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in non-monotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357.
- Easterday, M. W., Kanarek, J. S., and Harrell, M. (2009). Design requirements of argument mapping software for teaching deliberation. *Online deliberation: Design, research, and practice*, pages 317–23.
- Egly, U., Gaggl, S. A., and Woltran, S. (2008). Aspartix: Implementing argumentation frameworks using answer-set programming. In *Logic Programming*, pages 734–738. Springer.
- Farin, G. E., Hoschek, J., and Kim, M.-S. (2002). *Handbook of computer aided geometric design*. Elsevier.
- Gentleman, R. and Carey, V. (2008). Unsupervised machine learning. In *Bioconductor Case Studies*, pages 137–157. Springer.

- Gómez, S. A. and Chesnevar, C. I. (2004). A hybrid approach to pattern classification using neural networks and defeasible argumentation. In *FLAIRS Conference*, pages 393–398.
- Gordon, T. F., Prakken, H., and Walton, D. (2007). The carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10):875–896.
- Grünwald, P. (2005). A tutorial introduction to the minimum description length principle.
- Heckerman, D. (1995). A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research.
- Hoffer, J. A. (1999). *Modern Systems Analysis and Design, 6/e*. Pearson Education India.
- Hunter, A. and Williams, M. (2010). Argumentation for aggregating clinical evidence. In *ICTAI (1)*, pages 361–368.
- Kandel, A. (1991). *Fuzzy expert systems*. CRC press.
- Karacapilidis, N. and Papadias, D. (2001). Computer supported argumentation and collaborative decision making: the hermes system. *Information systems*, 26(4):259–277.
- Kohavi, R. (1995). The power of decision tables. In *Machine Learning: ECML-95*, pages 174–189. Springer.
- Kohavi, R., Becker, B., and Sommerfield, D. (1997). Improving simple bayes.
- Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques.
- La Greca, A. M. and Stone, W. L. (1993). Social anxiety scale for children-revised: Factor structure and concurrent validity. *Journal of Clinical Child Psychology*, 22(1):17–27.
- Leake, D. B. (2003). Case-based reasoning.
- Longo, L. (2014). *Formalising Human Mental Workload as a Defeasible Computational Concept*. PhD thesis, Trinity College.

- Longo, L. and Dondio, P. (2014). Defeasible reasoning and argument-based medical systems: an informal overview. In *27th International Symposium on Computer-Based Medical Systems, New York, USA*, pages 376–381. IEEE.
- Longo, L. and Hederman, L. (2013). Argumentation theory for decision support in health-care: a comparison with machine learning. In *International Conference on Brain Informatics, Maebashi, Japan*, pages 168–180. Springer.
- Longo, L. and Kane, B. (2011). A novel methodology for evaluating user interfaces in health care. In *Proceedings of the 24th IEEE International Symposium on Computer-Based Medical Systems, 27-30 June, 2011, Bristol, United Kingdom*, pages 1–6.
- Longo, L., Kane, B., and Hederman, L. (2012a). Argumentation theory in health care. In *Computer-Based Medical Systems (CBMS), 2012 25th International Symposium on*, pages 1–6. IEEE.
- Longo, L., Rusconi, F., Noce, L., and Barrett, S. (2012b). The importance of human mental workload in web-design. In *8th International Conference on Web Information Systems and Technologies, Porto, Portugal*, pages 403–409. SciTePress.
- Meshkati, N. and Hancock, P. (2011). *Human mental workload*. Elsevier.
- Mitchell, T. M. (2006). *The discipline of machine learning*. Carnegie Mellon University, School of Computer Science, Machine Learning Department.
- Modgil, S. and Prakken, H. (2014). The aspic+ framework for structured argumentation: a tutorial. *Argument & Computation*, 5(1):31–62.
- Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of machine learning*. MIT press.
- Neal, L. A., Busuttil, W., Rollins, J., Herepath, R., Strike, P., and Turnbull, G. (1994). Convergent validity of measures of post-traumatic stress disorder in a mixed military and civilian population. *Journal of traumatic stress*, 7(3):447–455.
- Ng, A. Cs229 lecture notes.
- O’Connor, M. J. and Das, A. (2012). A pair of owl 2 rl reasoners. In *OWLED*, volume 849.
- Parsia, B. and Sirin, E. (2004). Pellet: An owl dl reasoner. In *Third International Semantic Web Conference-Poster*, volume 18.

- Petrik, M. 'knowledge representation for expert systems. Citeseer.
- Podlaszewski, M., Caminada, M., and Pigozzi, G. (2011). An implementation of basic argumentation components. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1307–1308. International Foundation for Autonomous Agents and Multiagent Systems.
- Pollock, J. L. (1987). Defeasible reasoning. *Cognitive science*, 11(4):481–518.
- Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.
- Price, P. and Jhangiani, R. (2013). *Research Methods in Psychology: Core Concepts and Skills*. BC open textbook collection.
- Reed, C. and Rowe, G. (2001). Araucaria: Software for puzzles in argument diagramming and xml. *Department of Applied Computing, University of Dundee Technical Report*.
- Reiter, R. (1980). A logic for default reasoning. *Artificial intelligence*, 13(1):81–132.
- Sagheb-Tehrani, M. (2009). A conceptual model of knowledge elicitation. *Proc CON-ISAR*, page v2.
- Schumm, W. R., Paff-Bergen, L. A., Hatch, R. C., Obiorah, F. C., Copeland, J. M., Meens, L. D., and Bugaighis, M. A. (1986). Concurrent and discriminant validity of the kansas marital satisfaction scale. *Journal of Marriage and the Family*, pages 381–387.
- Shiffman, D., Fry, S., and Marsh, Z. (2012). *The nature of code*. D. Shiffman.
- Silvert, W. and Baptist, M. (2000). Can neuronal networks be used in data-poor situations? In *Artificial Neuronal Networks*, pages 241–248. Springer.
- Singh, S. and Karwayun, R. (2010). A comparative study of inference engines. In *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*, pages 53–57. IEEE.
- Stöber, J. (2001). The social desirability scale-17 (sds-17): Convergent validity, discriminant validity, and relationship with age. *European Journal of Psychological Assessment*, 17(3):222.

- Storch, E. A., Roberti, J. W., and Roth, D. A. (2004). Factor structure, concurrent validity, and internal consistency of the beck depression inventory—second edition in a sample of college students. *Depression and anxiety*, 19(3):187–189.
- Todd, B. S. (1992). *An Introduction to Expert Systems*.
- Turban, E., Aronson, J., and Liang, T. (2005). *Decision Support Systems and Intelligent Systems*. Prentice-Hall International.
- Twardy, C. (2004). Argument maps improve critical thinking. *Teaching Philosophy*, 27(2):95–116.
- Van Gijzel, B. and Nilsson, H. (2014). A principled approach to the implementation of argumentation models.
- Vreeswijk, G. A. (2006). An algorithm to compute minimally grounded and admissible defence sets in argument systems. In *COMMA*, pages 109–120.
- Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.
- Zadeh, L. A. (1965). Fuzzy sets. *Information and control*, 8(3):338–353.