

NORTHWESTERN UNIVERSITY

Statistical Relational Learning through Structural Analogy and Probabilistic Generalization

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Computer Science

By

Daniel T. Halstead

EVANSTON, ILLINOIS

December 2011

UMI Number: 3488474

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent on the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



UMI 3488474

Copyright 2011 by ProQuest LLC.

All rights reserved. This edition of the work is protected against unauthorized copying under Title 17, United States Code.



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

ABSTRACT

Statistical Relational Learning through
Structural Analogy and Probabilistic Generalization

Daniel T. Halstead

My primary research motivation is the development of a truly generic Machine Learning engine. Towards this, I am exploring the interplay between feature-based representations of data, for which there are powerful statistical machine learning algorithms, and structured representations, which are useful for reasoning and are capable of representing a broader spectrum of information.

This places my work in the emergent field of Statistical Relational Learning. I combine the two approaches to representation by using analogy to translate back and forth from a relational space to a reduced feature space. Analogy allows us to narrow the search space by singling out structural likenesses in the data (which become the features) rather than relations, and also gives us a similarity metric for doing unsupervised learning. In the process, we gain several insights about the nature of analogy, and the relationship between similarity and probability.

ACKNOWLEDGMENTS

THANK-YOU TO THE FOLLOWING INSTITUTIONS FOR FUNDING THIS RESEARCH:

The Air Force Office of Scientific Research, through the Massachusetts Institute of Technology
(#FA9550-05-1-0321, Computational Models for Belief Revision, Group Decision-Making, and Cultural Shifts)

The National Science Foundation's Knowledge Discovery and Dissemination Program
(#ITS-0325315, Analogy, Knowledge Integration, and Task Modeling Tools for Intelligence Analysts)



AS WELL AS:

Dr. Robert Piros of the Robert Piros Fellowship

SPECIAL THANKS GO TO:

Dr. Kenneth Forbus,

for administering the above support, and for his leadership and advisement
without which none of this could have happened.

Dr. Lawrence Birnbaum and Dr. Douglas Downey,
for serving on my committee and continuing to support me.

Dr. Michael Witbrock and Dr. Robert Kahlert of Cycorp,
for motivating and assisting in parts of this work.

FINALLY, I AM ESPECIALLY GRATEFUL TO:

My wife Lydia, who stuck by my side through it all.

AND:

My parents Lawrence and Carol, who have always believed in me.

TABLE OF CONTENTS

1	Introduction.....	6
2	Background.....	12
2.1	Knowledge Base and Input Representations	12
2.2	Statistical Relational Learning.....	15
2.3	Analogy	17
3	Generalization	22
3.1	Why Do Generalization?.....	24
3.2	How to do Generalization.....	31
3.2.1	A Generalization of Two Cases.....	31
3.2.2	Formal Algorithm for Generalization	33
3.3	Details and Analysis of the Generalization Process	35
3.3.1	Probability and Similarity	36
3.3.2	Threshold Values	43
3.3.3	Normalization and other Distance Metrics.....	45
4	Learning and Prediction.....	48
4.1	Approach to Learning	49
4.2	Flattening.....	53
4.2.1	Assigning Values to Propositions.....	55
4.2.2	Efficiency Analysis.....	59
4.2.3	Caching	64
4.3	Statistical Modeling and Prediction	66
4.3.1	Bayesian Networks	67
4.3.2	Rule Learning	70
5	The Whodunit Experiments	73
5.1	Early Whodunit Experiments	76
5.2	Answering Whodunit	79
5.2.1	Definition of the Whodunit Problem	80
5.2.2	The First Whodunit Experiment	82

		5
5.2.3	The Second Whodunit Experiment.....	87
5.3	Reduced Vocabularies.....	91
5.3.1	The Vocabularies	93
5.3.2	Translation.....	97
5.3.3	RRV Results	100
6	Comparison to other Algorithms.....	105
6.1	Comparing Generalization to IRMs.....	106
6.1.1	The IRM Experiments.....	107
6.1.2	Results of comparison to IRM.....	109
6.2	Comparing Generalization to MLNs and PRMs	112
6.2.1	The entity resolution procedure.....	113
6.2.2	The citation resolution experiments.....	115
6.2.3	The citation experiment results	118
6.2.4	Summary	125
7	Literature Review.....	126
7.1	Analogy and Similarity	126
7.1.1	Similarity and Probability	127
7.2	Relational Clustering.....	128
7.3	Model-Based Relational Learning	130
7.4	Induction-Based Learning	132
7.5	Other Works of Interest.....	132
8	Summary of Questions and Findings	133
8.1	Support for Claims.....	133
8.2	Other Conceptual Questions Raised	137
8.2.1	Generalization Questions	137
8.2.2	Feature Value Questions	139
8.3	Future Work.....	142
8.4	Conclusion.....	143

1 Introduction

Statistical relational learning has become an important problem in machine learning. It promises the ability to apply sound learning principles to the wide breadth of information that can be represented by systems of relations. We would like to be able to take preconstructed *cases* of arbitrarily structured facts and learn models for prediction from them. Yet learning from relational data has proven to be extremely difficult. Since relations operate over multiple objects at a time, enumerating all of the possible combinations of all of the possible objects quickly leads to exploding costs in real-world scenarios.

Hence, until recently, the statistical learning community has not strayed far from feature-based representations of data. A feature-based representation allows many cases to be represented in a uniform fashion. This makes it trivial to compare the different facets of a case set, something that is essential for learning. It also allows more accurate measures of uncertainty, such as probability, confidence, and significance. It also cannot be overlooked that there is a very large “toolbox” of algorithms for learning from feature-based representations.

However, structured representations are more useful when dealing with the wide range of information that people learn and manipulate. Plans, explanations, and arguments are among the kinds of important information that require explicit representation of relationships. Since such logical relationships are capable of representing these abstract thoughts, they are well-suited for reasoning, and for incorporating knowledge from other domains. Indeed, there is evidence

suggesting that the use of language and analogy to accumulate relational knowledge is why humans are so smart compared to other animals (Gentner, 2003).

The fundamental trade-off then, is uniformity for flexibility. This means that either we use some other, unknown representation of data which incorporates the best of both forms (the flexibility to succinctly represent all forms of knowledge with a guarantee of uniformity), or we use both forms simultaneously and have a means of mapping from one to the other. My proposal concerns the second option: designing a mapping.

The question then becomes, how can some uniform framework for learning be derived from a set of cases of arbitrary facts? Facts might link an object to a value, or link five objects together in some way; the order of the arguments may or may not matter; they may contain logical connectives and quantifiers such as *or* or even *thereExistsAtLeast*; they may even refer to an object which is completely hypothetical. Earlier techniques such as propositionalization are just not flexible enough to deal with such an array of possibilities.

Furthermore, how can a probability distribution be constructed at all? Probability requires some sense of uniformity: saying that “40% of dogs have fleas” requires recognizing that all dogs are equivalent, at least for the purpose of the comparison. However, few items or scenarios in the real world can be said to be equivalent in the perfect, mathematical sense. Rather, humans seem to make deductions from sets of things that are deemed to be "similar enough" for the context of the task at hand. Once these objects are grouped in some way, the individual differences can

then be expressed as probabilities. In contrast, traditional techniques in statistical relational learning examine probability distributions over relations. This requires enumerating every possible combination of objects for every relation, which quickly explodes in cost (De Raedt, 1998). Even more insidiously, multiple instances of the same relation over the same types of objects may play very different roles in their respective contexts. This makes probability distributions over relations expensive and inaccurate portrayals of what is truly going on.

We propose analogy as the solution for both of these problems. Analogy puts no constraints on the representation and it provides a means for assessing similarity. Moreover, analogy does not just posit that two cases are similar; it claims that two cases are similar *because* individual facts and objects in each case are *analogically equivalent*¹. That is, there are some elements of a case which, although seemingly different, actually indicate the exact same circumstances underlying both examples, based on shared structure. This allows us to generate probability distributions over analogous situations / structures in the data rather than individual relations.

For example, consider a simple analogy between a hockey and a baseball player. On the surface, there is no commonality between them. Yet students can regularly answer standardized questions such as: hockey : stick :: baseball : _____. The answer is found by digging beneath the surface, looking for a common relationship which unites them. A hockey player *swings* a stick (at a puck), while a baseball player *swings* a bat (at a ball). The relationship *swings* is

¹ A more standard term for this from the literature is a *correspondence*. However, I use analogical equivalence here because I will also need to refer to when items are analogically *different*. I also prefer to emphasize the notion of equivalence, since it is so closely tied to why we can compute probabilities.

identical in both cases (as well as its many corollaries: who is swinging what at what else and for what reason). Since their respective roles in the cases are identical, we can call the bat and the stick analogically equivalent.

This process of detecting analogical equivalence provides just the sort of equivalence notion we were looking for in order to achieve a uniform representation that is compact and useful, and to generate probabilities. By making a series of analogies across a sequence of cases, we can find the concepts which are common to most of them, even describing them probabilistically because of this found equivalence. Continuing with the example above, a person who had observed many different sports might begin to make some generalizations about sporting, such as “In most sports, points are scored by directing an object to some goal. About half the time, this is done by swinging another object at it. Opponents will usually try to prevent this.” Of course, words like “most”, “half the time”, and “usually” are just loose descriptors, which can be characterized more precisely by the probability of that particular concept occurring.

In exactly the same way, our system uses a sequence of analogical comparisons to learn probabilistic generalizations of a class of items. In effect, we are deriving probability from similarity. Similarity allows us to collect a certain class of examples, which we can then describe probabilistically. This runs counter to much of the existing literature (e.g. Wellman, 1994; Blok et al, 2003), in which similarity judgments are rendered based on probabilistic or even fuzzy calculations based on whatever dimensions they share. We claim that appropriate

dimensions could not even be found without first making some analogy between the cases to determine which concepts they share and which they do not.

Of course, if we want to do learning, we must also be able to do prediction. A learned model that cannot make predictions for novel data is of no use to anyone and cannot be said to have learned anything new. Additionally, prediction provides a criterion for the success of our approach. For evaluation, we examine four different techniques for doing prediction. Two of these serve as controls – one based on exemplar retrieval, and the other on generalization. The other two demonstrate our own contributions – one exploring probabilistic generalization, and the other exploring statistical models built from these generalizations. Crucially, we demonstrate that once the generalization has been constructed, any of the traditional feature-based statistical learning algorithms from the machine learning community can be applied, despite the relational structure of the original data.

The whole approach has been tested on five different domains (not counting others who have used probabilistic generalization with success in their own applications). Of these, the most exhaustive experimentation, in which all four techniques were examined, was done on the Whodunit Problem. The Whodunit Problem is to construct plausible hypotheses about the perpetrator of a new event (e.g., a terrorist attack), using a library of events where the perpetrator is known. Other domains such as unsupervised classification of animals (Kemp, et al. 2006) and modeling student success rates (Pazzani and Brunk 1991) serve to compare our approach to other

algorithms. We demonstrate how our approach can handle more highly structured data than other algorithms, thereby often requiring an order of magnitude fewer training examples.

In summary, this thesis demonstrates that analogy is an extremely powerful tool for generalizing, learning, and even making probabilistic predictions about the kind of highly structured information that the real world entails. In support of this, I make the following five theoretical claims:

§ 1. We can achieve the benefits of both feature-based and relational representations of data by constructing a mapping for transforming logical expressions into features and back again.

§ 2. It is more sensible in real-world scenarios to derive probability based on similarity rather than the other way around.

§ 3. With our techniques, we are able to apply any feature-based learning algorithm, despite the relational nature of the original data, without loss of information.

§ 4. Our classification algorithms are comparable in performance to existing algorithms, but they often² are more efficient and require fewer examples.

§ 5. Our approach is the only one we know of that can handle arbitrary relations and that requires no relational schema. This is in direct contrast to currently popular approaches such as Infinite Relational Models (Kemp et al, 2006), Probabilistic Relational Models (Getoor et al, 2001), and Markov Logic Networks (Richardson and Domingos, 2006).

² The conditions for this are qualified in section 4.2.2.

In the next chapter, I present some important background information to help readers understand the rest of the thesis. Then in Chapter 3, I describe my approach from beginning to end. After a high-level overview of the architecture as a whole, the first section of this chapter describes the algorithm for probabilistic generalization. This is followed by a section on statistical modeling. Both of these sections begin with a high-level overview of the technique before delving into more detail. In Chapter 4, I present the experiments we have performed, and discuss their results. Chapter 5 looks at related work and where this thesis fits into the existing literature. Finally, Chapter 6 summarizes the findings and the questions raised by this work. In this chapter, I provide support for the above claims, describe any unexpected questions and findings that were raised, discuss plans for future work, and summarize the research as a whole.

2 Background

We begin by first reviewing those background concepts which are relevant to a full understanding of our own algorithms.

2.1 Knowledge Base and Input Representations

A major goal of this research is to handle complex relational data. All of the input data is therefore represented by predicate calculus statements. These statements are expressed in LISP syntax, with the predicate first, followed by the arguments. The statements are organized topically into cases, and it is these cases which are fed into our learning algorithms. We do not

require any particular method for doing this case construction, as the choice of which facts to include in which cases is outside the scope of this research.

The statements can be of arbitrarily high order, with any number of arguments (which may or may not be symmetric or commutative), connectives, quantifiers, and hypothetical or functional entities. Furthermore, no relational schema is needed to prescribe the possible arguments of each predicate (§5). For example, the statement below comes directly from one of our experiments, and demonstrates the degree of relational complexity that can be handled by our algorithms:

```
(thereExistAtLeast 2 ?X (and (isa ?X (CitizenFn Israel))
                              (injuredIn theAttack ?X)))
```

Some of our experiments were done without the presence of any knowledge base to draw from. These experiments instead used sets of facts provided by others in a test-bed. This was especially done when comparing our performance to other algorithms, in which case it was desirable to use the exact same inputs that were given to those other algorithms. However, the fact above comes from an experiment which drew its inputs from a massive, pre-existing knowledge base. The ResearchCyc knowledge base, developed by Cycorp contains over 36,000 concepts, over 8,000 relationships and over 5,000 functions, all constrained by 1.2 million facts. The use of this knowledge base highlights certain problems that would appear in a very long running, standalone generic learner.

For instance, since this particular knowledge base is so massive, it is useful to partition the facts within it by topic. This makes querying the knowledge base on a known topic much more efficient. It also provides scope for items like names and axioms, reducing contradictions. In the Cyc KB, these partitions are referred to as *microtheories*.

There are some problems with using microtheories though. For instance, there is no obvious way to generate them automatically. Furthermore, there is no method for determining where to place the partitions so that they would be the most useful, or even what the usefulness of a particular partition scheme would be. In the chapter on Future Work, we describe how generalization could provide a reasonable and automatic alternative for this partitioning of knowledge.

A second problem with using such a massive knowledge base is the aforementioned issue of case construction: how to determine which facts are relevant enough to be included in a given case description. Although we do not require any one method for doing this, we do use the method proposed by Mostek et al. (2000) in experiments where the cases were not preconstructed for us. I make no claims about the superiority of this approach over any other. In summary, we first restrict each case to be about a particular topic, such as one terrorist incident. We then include any fact from the knowledge base which mentions that topic. Finally, we also include attribute information (facts about collection membership) for any other items which those facts mentioned. This last step provides additional structure for the analogy algorithm to use when comparing cases.

2.2 Statistical Relational Learning

Statistical Relational Learning is a new subarea of Artificial Intelligence. It studies the question of how to perform machine learning over systems of relations such as those described in the previous section. This section briefly describes the most common techniques for accomplishing this and their various trade-offs. Typical approaches to statistical relational learning can be divided into two basic camps: those that take each relation separately, and those that try to model the whole system at once.

The first camp focuses on trying to somehow treat each relation as a separate feature that can be independently optimized. This is commonly done either by hill-climbing over the relations, such as Inductive Logic Programming (ILP) does, or by transforming the space to a flattened feature space, as propositionalization does. Both approaches have their drawbacks. ILP constrains the learner to a single model of learning: supervised rule-learning. It also typically relies on background knowledge to guide the hill-climber into less of a random walk. On the other hand, in (DeRaedt, 1998) it was shown that spatial transformation approaches suffer from having to make an expensive trade-off between either ignoring much of the important information contained in the shape of the original data, or requiring an unreasonably large search space to represent all of the structural possibilities.

The other and more recent camp tries instead to build a large, statistical, generative model of the entire domain at once. Examples of this approach include Kemp (2006), who tries to build a

large Bayesian generative mixture model (GMM) of the domain, Domingos (Richardson and Domingos, 2005), who tries to build an even larger Markov Logic Network (MLN), and Getoor (et al., 2001), who builds a Probabilistic Relational Model (PRM), which is essentially a more aggregative version of a Bayesian Network, specially tailored for relational learning problems. These approaches share many of the same limitations. For instance, they all are limited to doing supervised learning, and to using one particular statistical model. Some of them (MLNs in particular) also suffer under the same massive spatial requirements as propositionalization, by needing to account for every possible combination of formula groundings.

Furthermore, to our knowledge, none of the algorithms in either camp can be said to truly be able to handle all of the complexities that occur when dealing with relational data. For instance, none of them have methods designed for handling skolems (hypothetical entities) or non-atomic terms (functional entities). None of them can handle relations with an arbitrary number of arguments, such as *characters-in-word*, or in which the arguments are known to be symmetric, such as *sibling*. Domingos is one of the only ones who has attempted to deal with logical connectives and quantifiers, although they make the spatial requirements even larger. Getoor is the only one (including ourselves) who has attempted to deal explicitly with aggregate information, such as the mean grade in a class or the youngest age of a victim. And all of them require a relational schema to be laid out ahead of time, which means that new relations can never be introduced at run-time.

In contrast, our approach tries to attain the best of both camps. We retain much of the shape of the original data by using analogy to align entities and relations in a structurally optimal way (our definition of optimality is described in the next section), creating a single, structurally consistent generalization of the domain. This generalization may be used as a reductive model. However, it can also be used as a framework to efficiently propositionalize the domain in a way that retains the structural information contained in the generalization. This way, one can still apply any of the powerful statistical machine learning algorithms to the data without losing the most important structural information and without succumbing to an explosion in the size of the search space. This trade-off between structural information and search space will be shown in detail in the next chapter, when our procedure is described in full.

2.3 Analogy

Analogy forms the cornerstone of this work. While this thesis recommends several key extensions to the analogy process itself (e.g. sections 3.3.1 and 8.2.1), the basic SME algorithm (Falkenhainer et al., 1989) that we use to do analogy has existed for many years. In that time, it has been soundly tested for many uses in many domains and by many research groups. This section summarizes the essentials of structure-mapping needed to understand the rest of the thesis.

Analogy serves us in many ways. As described above, it finds correspondences which are indicative of an underlying structural likeness. These likenesses can be treated as random variables and therefore used to generate joint probabilities and learned from.

This means that analogy makes our relational learner more efficient than many others (§4).

Relations which can operate over many different types of objects often lead to quickly exploding costs, if they can be handled at all. Other relational learning algorithms, such as those described above, therefore require that all of the relations and their possible argument-types be laid out ahead of time in a relational schema, making online learning impossible. Even two groundings of the same relation on the same types of objects may play very different roles in their respective contexts, making probability distributions over these relations inaccurate portrayals of what is truly going on. Our approach bypasses many of the problems of exhaustive search, since analogy succinctly captures the notion of the role that each object plays from the context of the facts around it.

Finally, analogy also provides an excellent estimate of similarity, since the more role correspondences there are between two sets of facts, the more similar those sets will be. Thus it can be used as a distance metric when doing generalization, particularly in unsupervised learning.

To do analogical comparisons between cases, we rely on the Structure-Mapping Engine, or SME (Falkenhainer et al., 1989). SME is a computational model of similarity and analogy based on

Gentner's (1983) structure-mapping theory of analogy in humans. SME takes as input two cases: a base and a target. It finds all possible local correspondences between the entities, attributes, and relations in the two cases. Each correspondence hypothesizes some analogical equivalence. It then uses a greedy, polynomial-time algorithm (Forbus & Oblinger 1990) to combine as many consistent correspondences as possible into approximately optimal mappings between the cases.

For a real world example, consider the two cases about terrorist events, shown in Table 2-1. In the first case, the terrorist drives a van containing a bomb into the situation. In the second case, the bomb is planted in the victim's car. Both cases contain the exact same types of entities and relations. For instance, they both have a relationship describing a vehicle which contains a bomb, and both vehicles have a driver. However, the roles of the drivers are very different. A structural analogy between the cases captures this difference.

Table 2-1. A simplified but real-world example of an analogy between two terrorist events. (a) shows a subset of the facts in each case. In (b), correspondences have been proposed. Finally, (c) shows the maximally consistent set of top-level correspondences

a)	Base Case	Target Case
	(Location bomb1 van)	(Location bomb2 car)
	(Driver van agent1)	(Driver car agent3)
	(Terrorist agent1)	(Target agent3)
	(Target agent2)	(Terrorist agent4)
	(Desires agent1 (Hurt agent2))	(Desires agent4 (Hurt agent3))

b) Entity correspondences	bomb1:bomb2, car:van, agent1:agent3, agent1:agent4, agent2:agent3, agent2:agent4
Attribute correspondences	(Terrorist agent1) : (Terrorist agent4), (Target agent2) : (Target agent3), (Hurt agent2) : (Hurt agent3)
Relational correspondences	(Location bomb1 van) : (Location bomb2 car), (Driver van agent1) : (Driver car agent3), (Desires agent1 (Hurt agent2)) : (Desires agent4 (Hurt agent3))

c) Analogical Mapping		
(Location bomb1 van)	↔	(Location bomb2 car)
(Terrorist agent1)		(Terrorist agent4)
(Target agent2)		(Target agent3)
(Desires agent1 (Hurt agent2))		(Desires agent4 (Hurt agent3))

In the first stage (b), SME gathers all the potential correspondences by looking at pairs of statements which share the same predicate. In the last stage (c), the largest possible number of consistent correspondences is brought together into a mapping. Since items must be matched 1:1, a correspondence C between relations is consistent with a set of correspondences S as long as none of the correspondences between the arguments of C are contradicted by S. For example, the correspondence "(Driver van agent1) : (Driver car agent3)" cannot be added to the mapping in (c) because it matches agent1 to agent3. This would contradict the correspondences in the mapping which match agent1 to agent4, and agent2 to agent3. We therefore say that the two facts are analogically different.

In this way, the underlying concepts are distinguished by separating those facts in which the entities play a unique role, given the context of the other facts they appear in.

In addition to creating an analogy between the cases, SME computes the similarity between them based on the *systematicity* of the analogy. That is, each correspondence in the mapping contributes to the similarity score in proportion to the number of higher-level correspondences which invoke it:

Let x be any correspondence in the mapping, either of objects or relations, and call $s(x)$ the contribution to the overall similarity score that x makes. The base and target of x may appear as an argument in other relations in the mapping, which we will call $Parents(x) = \{R_x^1, \dots, R_x^n\}$. Then the equation for computing s is:

$$s(x) = a + b \sum_i (s(R_x^i))$$

where a and b are constants. The total similarity score for the analogy then is simply the sum of $s(x)$ over all correspondences x .

Finally, SME also provides a means for creating *candidate inferences*. That is, an expression which appears in the base but not in the target, and which is connected to the correspondences of the mapping, can be hypothesized to be true for the target as well by projecting its form onto the target. In a sense, it is inferred via analogy that the

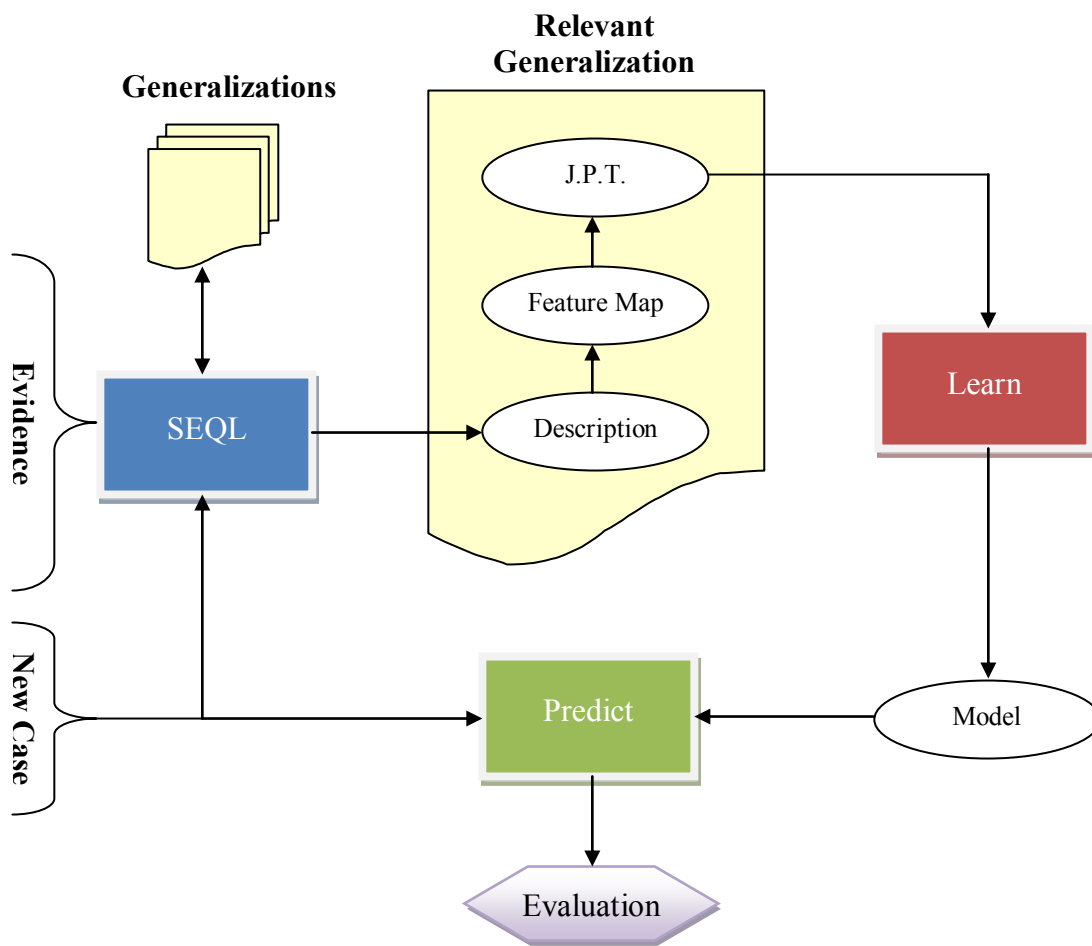
equivalent of the missing expression is also true for the target. When using analogy to construct a generalization, we do not use candidate inferences at all, since missing expressions are abstracted into the generalization with everything else only with a lower probability (described in the next chapter). However, when doing prediction, we are simply doing probabilistic inference via analogy. Thus, candidate inferences can be derived from comparing a case to a generalization rather than a single base case, thereby incorporating accurate estimates of the probability of the inference.

3 Generalization

This chapter presents the first and most essential step in our overall procedure for doing learning and prediction: generalization. We describe the reasons we think generalization is needed and then describe the generalization algorithm itself. Finally, we will analyze the running time, threshold values, and heuristic calculations. We start with a brief overview of where generalization fits in the bigger picture.

As described in Section 2.1, we assume inputs to be descriptions of new situations or concepts, expressed by relationships between entities. These cases are fed into SEQL (Kuehne et al, 2000), a generalization algorithm which uses analogy to find relevant generalizations to the inputs, and builds one if none exist. In this way, it grows generalizations by progressively incorporating new cases into the most similar of them.

Figure 3-1. A simplified diagram of our approach. SEQL uses analogy to build generalizations of the evidence. A learner may learn a model of each generalization, and this can be used to do prediction for new cases.



Each generalization consists of three parts: a probabilistic summary of the shared relationships, expressed as facts; a joint probability table exploring the possible combinations of these relationships, expressed as features; and a mapping for converting from one to the other. This ability to convert freely from one representation to the other using the mapping is one of the major benefits of this approach (§1).

Optionally, a statistical model of the joint probabilities may then be learned. The model may be as simple as a list of rules, or a Bayesian network. It is built from a combination of background knowledge and statistical learning.

Finally, the generalization and/or model can then be used to do prediction for a new case, either by simply comparing the new case with the generalization, or by using the mapping to feed it into the statistical model.

3.1 Why Do Generalization?

Our approach to relational learning begins by building generalizations of the inputs.

Generalization is a process of moving from the specific descriptions of individual cases to an abstract description which aptly describes a set of related cases. Note that this entails three essential steps: determining which cases are actually related, determining the elements that those cases have in common, and finally abstracting those elements to build a generalized description which can be applied to all of the cases.

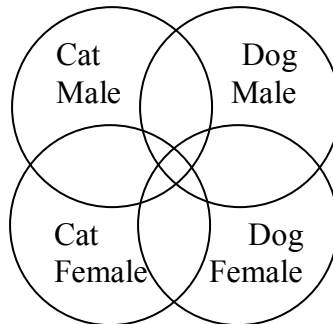
These three aspects to generalization give rise to the three most important reasons we have for doing it. First, in determining which cases are more related than others, generalization provides us with a means of doing unsupervised learning. This means that, unlike many other relational learners (e.g. MLNs, ILP), ours will be able to learn whether it is given labeled examples or not.

Secondly, in determining which elements the cases have in common (i.e. which entities and relations are analogous to which), the generalization process must decide on a single overarching set of correspondences to use. Deciding on a single mapping cuts out a lot of the search space, making it more efficient than many similar relational learners (e.g. FOIL and LINUS), and also leads to some interestingly human-like effects, such as a bias to the order of the inputs. Finally, the fact that we are able to abstract a common description from the bottom up means that we require no prior relational schema.

First, it is important to look at the issue of which cases should be generalized together.

Philosophically, there are several possible answers to this. The first answer, corresponding to a supervised learning scenario, is that a generalization should consist of all known instances of a given type. That is, some agent – whether a human advisor or another part of the system – assigns a label to a set of examples, and expects the learner to build appropriate generalizations. Of course, this is an excellent way of learning to distinguish between examples of one type from another. For example, if the system was instructed to build one generalization about dogs and another about cats, then a new animal could be compared to each generalization to see which was more similar. Recall from Section 2.2 that SME allows us to compute an analogical similarity score between any two cases. In the next section, we will show that it can also compute the similarity between a case and a generalization of cases. However, this approach would not help in distinguishing along a second dimension, such as gender (Figure 3-2).

Figure 3-2. It is often not clear which cases a generalization should cover and which it should not.



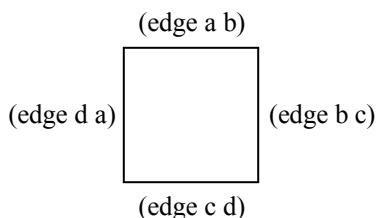
There is a second possible answer to the question of which cases to generalize over. Since SME provides a similarity score, this score could be used as a distance metric in an unsupervised learning scenario. That is, the learner could be left to its own devices to decide whether or not two cases belong in the same generalization, using the SME similarity metric. In fact, this is the default behavior of SEQL (Kuehne et al, 2000). It works by successively merging the cases which it deems to be similar enough into a generalization together. This is done by looking at each case in sequence, and merging it with either the first existing generalization or the first other case (making a new generalization) with which its similarity exceeds some threshold. This too has its problems. It is not certain that the generalizations the learner chooses will distinguish between male and female either, nor even between cat and dog. However, it is an excellent method for the discovery of new types.

A third idea is to build one larger generalization of all of the cases. Then, the generalization could serve as a unifying framework for statistically modeling all of the different trends that happen across those cases. These trends could then be used to predict any particular dimension of that generalization. Note that this could be done whether the learning was supervised or not.

This idea is explored further in the next chapter. Suffice it to say here that using generalization to do relational learning keeps all of these options available to us.

A second reason to use generalization is that in building a generalization, the computer is deciding on a single best way of matching up the entities and relations of one case with those of the next. This decision can dramatically speed up learning by seriously constraining the space of possible hypotheses to explore. For example, in their work on geometric analogy (Tomai et al, 2005), Tomai and Lovett attempted to use SME to determine the right answer on a series of classic Miller Geometric Analogies problems. These tests involve figuring out the analogical difference between diagrams in a series of SAT-like “A is to B as C is to what?” questions. In one such question, two of the diagrams include a basic square:

Figure 3-3. This simple square causes undue consternation for most relational learners.



It was shown how this square mapped equally well to every 90 degree rotation of itself. If the structure of argument order were ignored, it could match to still other transformations. Just to learn about such squares, a propositionalization algorithm would have to explore 12 different possible propositions, e.g. (edge W X), (edge W Y), (edge W Z), (edge X W), etc. Any corner in the first square could correspond with any corner from the second square, creating 12 different

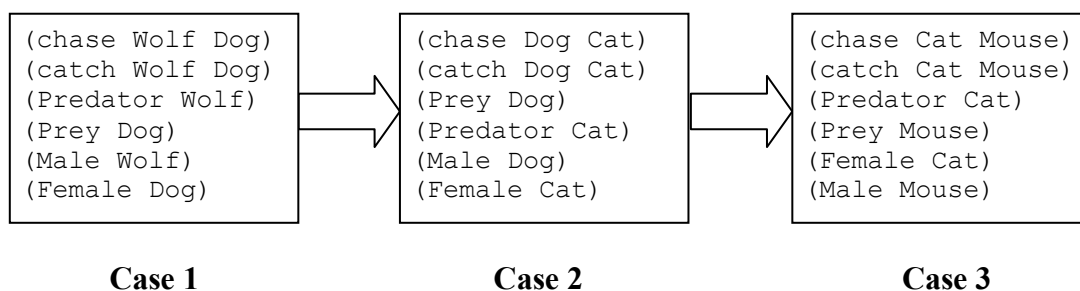
possible propositions. Of course, this explodes very quickly as the number of entities of a particular type increases. Newer implementations of propositionalization such as LINUS (Lavraç, Dzeraski, and Grobelnik, 1991) make it possible to ameliorate some of this overhead, but require customized background knowledge to do so. Other relational learners such as ILP and Domingos' MLNs suffer from the same problem.

Analogy, however, solves this problem. As shown in Section 2.2, analogy provides a means of choosing the overall best set of correspondences, called a *mapping*, to use. The rest of the possible correspondences are ignored and so do not consume valuable resources during search. In the truck-bomb example given in that section, there was one mapping that was clearly better than the other. As a result, a correspondence between the two drivers was thrown out in favor of a mapping which matched the two victims and the two agents. This mapping had a higher degree of shared relational structure, and therefore a higher similarity score. In the example above, where there is no such obvious mapping preference because all of the different rotations of the square are basically equivalent, SME returns an equal score for each mapping and so one of the mappings is chosen arbitrarily. This reduces the number of possible propositions from 16 to the much more sensible 4: one generalized proposition per edge. A more rigorous analysis of this speedup will be given in Section 3.3.

This approach is not without its problems. Undoubtedly, it is possible for the algorithm to simply choose the wrong mapping. In practice though, this is more often than not the result of poor representations in the input cases. A more interesting problem is that it introduces an order-

bias to the algorithm. The generalization algorithm functions by introducing one new case at a time, in sequence, to be compared to the whole. Each time a new case is deemed similar enough to be incorporated into the generalization, a best mapping for it is selected. If the input cases are given in an unlucky order, the generalization algorithm could be led astray into choosing a poor system of mappings for the generalization. For example, in the sequence of cases from Figure 3-4 below, a wolf is chasing a dog which is chasing a cat which is chasing a mouse. The wolf and cat are both predators, while the dog and mouse are both prey. This means that the roles of predator and prey are inverted in the second case. In the third case, the genders are inverted.

Figure 3-4. SEQL's bias for a single mapping can induce an order bias.



In this example, if SEQL were given all 3 cases in order, it would work out “correctly”. The animals doing the chasing would correspond in every match and the animals being chased would correspond in every match. However, if this sequence were run in reverse, it would go down the wrong path. In the first match, between cases 2 and 3, it finds it can line up four statements if it matches by the attributes predator/prey and male/female, and only two statements if it matches by chaser/chasee. Determining the former to lead to a higher similarity score, it chooses this mapping instead, and having already done so once, follows suit for the rest of the generalization.

The result becomes a generalization in which the males always correspond and the females always correspond but everything else is arbitrary. However, it is interesting that even humans are biased by such order effects when doing category learning (Elio & Anderson, 1984; Wattenmaker, 1993; Medin & Bettger, 1994). Furthermore, in previous studies it has been shown that the order-based effects of SEQL do actually match those of humans (Kuehne et al, 2000; Skorstad, Gentner, & Medin, 1988). We feel that a trade-off which yields an increase in learning efficiency for an increase in certain human-like biases is worth doing.

Finally, a third reason for using generalization in relational learning is that it works entirely bottom up. In contrast to other relational learning approaches, it does not need a relational schema to be laid out in advance in order to determine how to line everything up. Rather, it determines this during run-time by simply observing the structure of one case at a time. To our knowledge, it is the only structural relational learner that is capable of this. This comparison of structure rather than predicates also allows it to accept any kind of relational structure. This is in contrast to algorithms like LINUS and MLNs, which are severely restricted in the representations they can handle. For example, SEQL has no problem dealing with predicates of more than two arguments, or even a variable number of arguments, and can handle non-atomic terms and nested structure. One other advantage of this bottom-up approach is that it will never devote resources to learning about a combination of entities and relations that never occurs in the data.

Generalization allows us to do learning from structured data by putting the onus of pattern-finding on comparisons between the shapes of the structures themselves rather than combinations

of entity values. The process allows learning to be done whether in a supervised environment or not, from any structure of data, with no need of a relational schema. It assigns a single mapping to everything, which cuts down on both the size of the hypothesis space and the time needed to test a hypothesis (the latter will be shown in the next chapter). In short, structural generalization allows us to be both more efficient and more flexible than alternative approaches (§3).

3.2 How to do Generalization

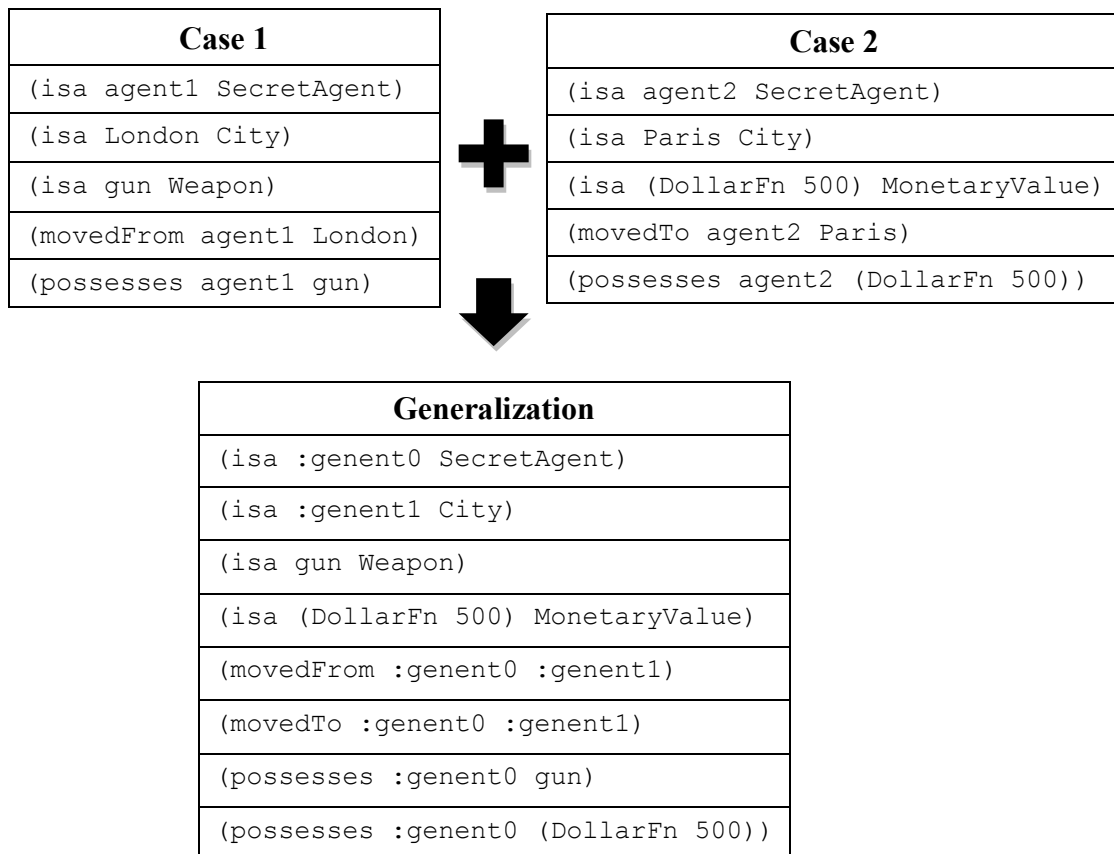
We build generalizations of the input cases while maintaining both their relational structure and their probabilistic differences. In order to understand how this is done, it is useful to first see an example of what a generalization actually looks like.

3.2.1 A Generalization of Two Cases

Fundamentally, a generalization is simply a more abstract case description. In our method, any formula whose form appears in any of the member cases of the generalization will appear in the generalization³. Furthermore, every *analogically different* grounding of the formula will appear uniquely in the generalization. Each unique occurrence of each formula is an expression grounded with new, generic entities which share the same constraints as their analogical counterparts in the original data.

³ To save memory, we currently cull the least frequently occurring formulae. This should eventually be changed to something which at least culls only the least significant formulae. However, this is difficult to calculate on the fly.

Figure 3-5. An example of a generalization.



For example, in the generalization above, the two statements that agent1 and agent2 are secret agents gets generalized into a more abstract statement because they are deemed to be analogically equivalent. The original entities are replaced by the more abstract placeholder :genent0. Such placeholders are referred to as *generalized entities*. This replacement is done so that :genent0 shares the same constraints as before: there are still facts describing that :genent0 moved and that :genent0 possesses something.

However, the facts about what the agent possesses were deemed to be *analogically different*. Despite appearing to be similar, the system made a distinction between owning \$500 and owning

a gun. As a result, the facts were not abstracted together: they remain unique expressions in the generalization.

It is crucial to understand that each generalized fact actually represents some set of grounded facts which were found to be analogically equivalent. This means that each of the original groundings were actually just different instantiations of the same underlying relational structure, with each entity playing an identical role in the case. The corollary of this is that each separate generalized fact in the generalization represents a fact which was structurally unique in some way, indicating some new role in the context of the case (§1).

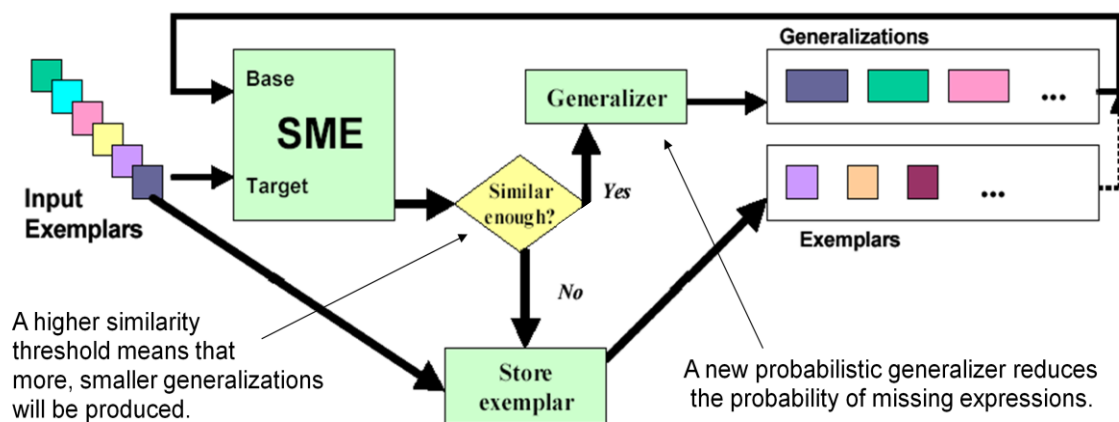
3.2.2 Formal Algorithm for Generalization

We use the SEQL algorithm to produce generalizations incrementally from a stream of examples. Given a new example, it uses SME to compare the new example to a pool of prior generalizations and examples. If the new example is sufficiently close to one of the generalizations, it is assimilated into that generalization. Otherwise, if it is sufficiently close to one of the prior exemplars, it is combined with it to form a new generalization. By "sufficiently close", we mean that the structural evaluation score⁴ exceeds a pre-set threshold. The process of assimilating an exemplar into a generalization (or of combining two exemplars into a new generalization) consists of taking the overlap between the two descriptions, as found via SME, as the new description. Matching entities that are identical are kept in the generalization, non-

⁴ In order to account for differences in the sizes of the descriptions, the score is normalized. This procedure is described below.

identical entities are replaced by new entities that are still constrained by all of the statements about them in the overlap. Should no sufficiently close match be found, the exemplar is simply added to the pool of exemplars. This process is shown in Figure 3-6.

Figure 3-6. A diagram of the SEQL algorithm, which we use to construct generalizations.



More formally, given an initially empty set of generalizations G and set of exemplars X , a threshold ρ , and a function $\text{Score}(x,y)$ which calculates the similarity (i.e. the normalized structural evaluation score) between two descriptions, SEQL operates as follows:

1. Receive an exemplar x .
2. Look for a generalization $y \in G$ s.t. $\text{Score}(x,y) > \sigma$. If one is found, proceed to step 5.
3. Look for an exemplar $y \in X$ s.t. $\text{Score}(x,y) > \sigma$. If one is found, proceed to step 5.
4. Add x to X . Repeat from step 1.
5. Remove y and add $\text{Generalization}(x,y)$ to G . Repeat from step 1.

SEQL provides an interesting tradeoff between traditional symbolic generalization algorithms like EBL (Dejong & Mooney, 1986) and statistical generalization algorithms, such as connectionist systems (Seidenberg & Elman, 1999; Altmann & Dienes, 1999; Shastri & Chang, 1999). Like EBL, it operates with structured, relational descriptions. Unlike EBL, it does not require a complete or correct domain theory, nor does it produce a generalization from only a single example. Like most connectionist learning algorithms, it is conservative, only producing generalizations when there is significant overlap. However, SEQL has been shown to be substantially faster than connectionist algorithms when compared head-to-head (Kuehne *et al.*, 2000). Moreover, this was done using the same input representations as the connectionist models, and the SEQL-based simulation continued to work when given noisy versions of the data.

3.3 Details and Analysis of the Generalization Process

The previous section was a high-level view of how generalization is done. In contrast, this section provides a low-level, formal analysis of the technique, including the incorporation and effect of probability and similarity, available parameters and their optimal values, and running time. The first subsection on Probability and Similarity is especially important since it relates directly to the second claim of this thesis.

3.3.1 Probability and Similarity

One contribution of this thesis is the extension of analogical generalization to incorporate probabilities. The original SEQL algorithm made no use of probability at all. We utilize it to make generalizations more accurate. Previously, SEQL would simply throw away any information that did not occur in every member case of the generalization. While this made it very efficient and tolerant to noise, it also meant that any generalizations of more than a few cases began to all look the same – it was throwing away useful, discriminating information along with the noise.

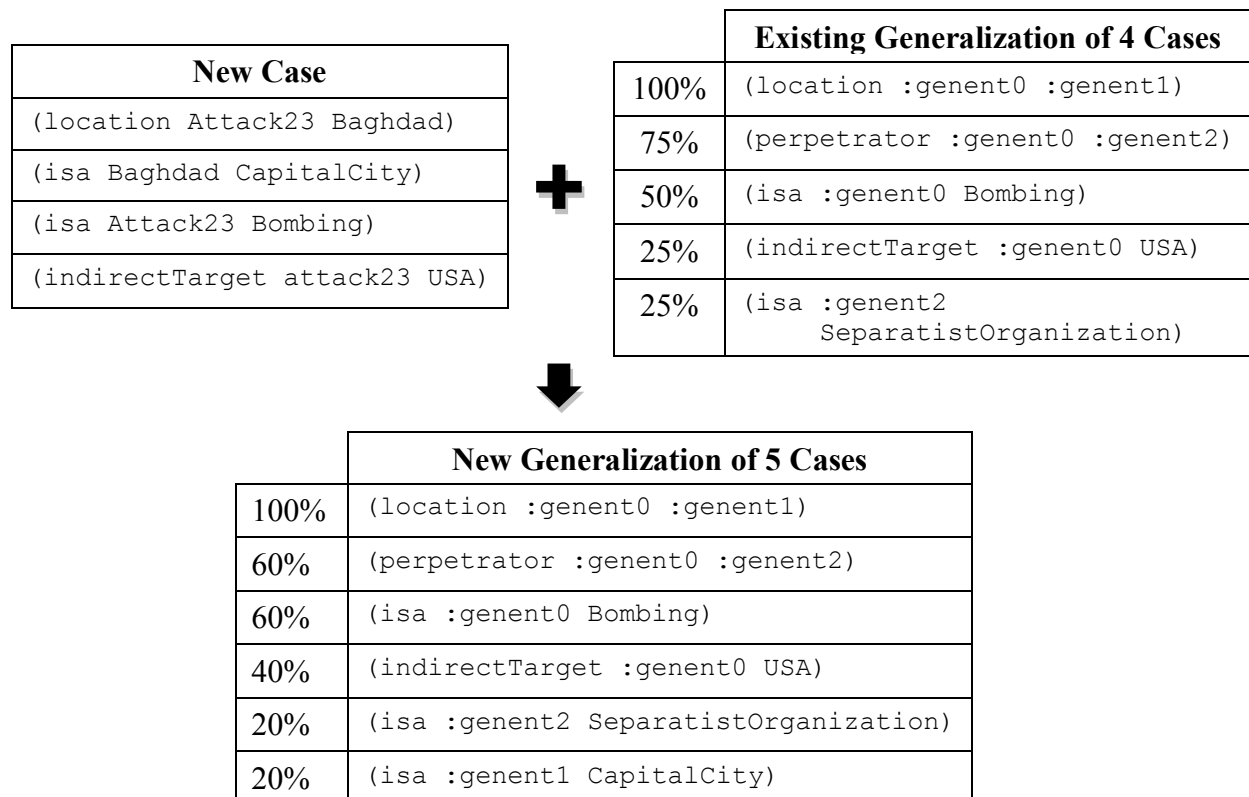
Furthermore, by keeping records of the joint probabilities of all the evidence used to build a generalization, we can build a causal model of the generalization (§3), rather than the reductive prototype or exemplar model that is usually built when relying on symbolic representations. Such a causal model has allowed us to do things such as improved classification, probabilistic inference, and anomaly detection (§4). It could also provide a deeper understanding of the domain by detecting unknown dependencies between expressions.

As argued above, analogy is a natural method to use when trying to calculate probabilities over highly structured assertions in order to ensure that the facts going into the probability computation correspond to the same concept in every case. Generalization simply helps us define a method for performing analogy over many examples at once. Therefore, whenever one

wants to calculate the probability over multiple cases, it follows that generalization is a useful tool.

We calculate the top-level (non-joint) probability of a particular concept occurring in a generalization by simply counting the number of times a grounded analog of it occurs in the member cases. For example, given a match between two cases with no probabilities in them, if a statement exists in one case but not in the other, then it should have a probability of .5 in the ensuing generalization. A more complex example, from matching a new case to an existing generalization, is shown in Figure 3-7 below.

Figure 3-7. Expanding an existing generalization to encompass a new case.



Note that the probability of the Bombing and IndirectTarget facts increased, since they were in the match (occurred in both the case and the old generalization), while the probability of the perpetrator fact decreased since it was not. Just like before, many of the entities have been replaced by constrained variables labeled :genent, but the USA entity remains intact since so far it has been the only value to be in that slot.

Of course, now that probabilities are computed and represented in the descriptions for the generalizations, those probabilities must be taken into account when performing analogy between two such descriptions, i.e. during SME matches. Otherwise, SEQL would consider far too many things in the mapping and end up not matching strongly to anything (or, if the disjuncts are just ignored, it will have the same problem as before, where it will successfully match the generalization to everything once most of the facts have fallen out of it).

Since probability doesn't affect the structure of the facts, it doesn't affect which correspondences the analogy will draw. However it does affect the similarity score calculated for those correspondences. For example, matching to a generalization where the corresponding fact occurs in only 1 out of 1000 cases should certainly not be as good as matching it to a generalization where it occurs in all 1000 cases.

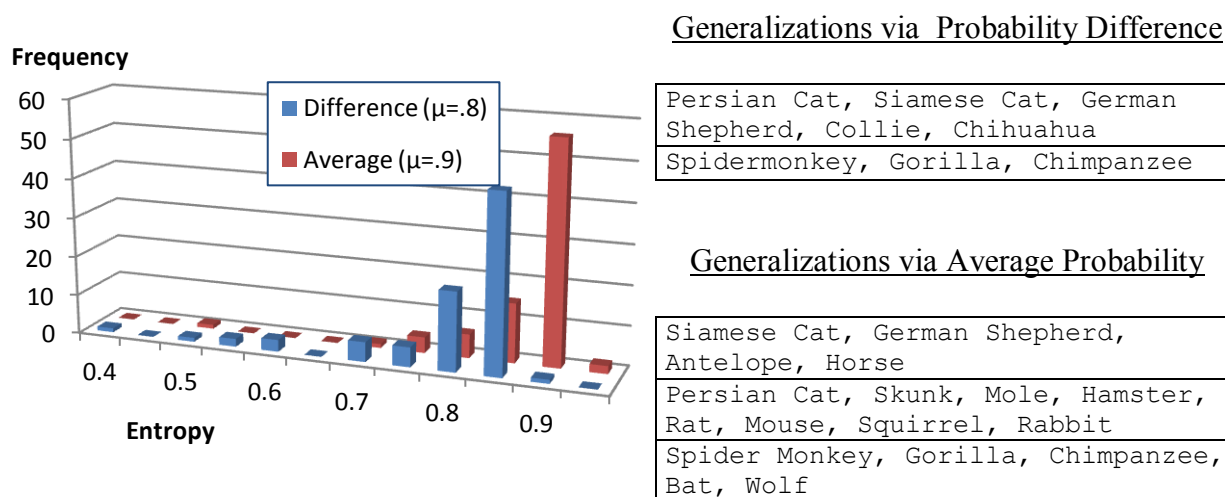
However, it is not so clear what should happen if the fact occurs in 1 out of 1000 cases in both the base and the target. Should this correspondence be considered a weak one, since their probabilities are so low, or a strong one, since their probabilities are matching?

To answer this, we tried two different metrics, designed to reflect each hypothesis. The first weighted the contribution of a correspondence to the overall similarity by the average probability of the base and target. The second weighted it by the square of their difference. After some experimentation, we found the second metric to work better:

$$Score \leftarrow Score * (1 - |p_{base} - p_{target}|)^2.$$

We believe this is because the above metric makes a better distinction between the roles of similarity and probability. That is, two descriptions will not be judged similar simply because the expressions within them are common, but because the expressions which match also have matching probabilities. This seems to make better matches and hence clearer generalizations.

Figure 3-8. Two metrics for weighing similarity by probability produce different degrees of clarity in the generalizations.



For example, consider the generalizations in Figure 3-8, taken from an experiment on animal classification (described in section 5.5). When the similarity is weighted by average probability,

the two types of cat appear in two different generalizations, and the primates appear mixed together with bats and wolves. However, when we use the difference in probabilities instead, the classes seem more correct. A more rigorous analysis can be made by looking at the entropy with which each predicate is distributed among the generalization descriptions. An entropy of 1 for a predicate would mean it appears with the same frequency in every generalization, and so a high entropy for every predicate would mean that every generalization appears the same. The histogram shows that weighing the similarity by difference in probability provides the lowest entropy, and hence clearest differences in generalization descriptions.

There is also an interesting question as to whether this probability-weighting should occur before or after the trickle-down takes place. Trickle-down is how SME implements the structure-mapping preference for *systematicity*, i.e., for mappings where larger interconnected systems of relations are matched. Each match hypothesis is given an initial score. This initial score is then augmented by a weighted recursive sum, starting with the highest-level nodes, recursing down to the level of correspondences between entities.

If the probability occurs after trickle-down, then the score of no sub-expression would ever be different from what it would have been if probability were ignored altogether (since they always have a probability of 1). Therefore the effect of probability would be to only weight the scores of the very top-level expression correspondences. Since the structural evaluation score is computed by summing the score of every expression correspondence, even non-top-level ones, and since the lower level correspondences will tend to experience the largest scores because of

trickle-down, then the effects of having a top-level expression with a probability other than 1 would be miniscule.

However, if the probability-weighting occurs before trickle-down, then the probability of the higher-level correspondences are trickled down as well. For example, the score of an entity that occurred in only one place, two levels down from the top-level expression with probability p would look like:

$$\lambda_2 p t^2 + \lambda_1 t + \lambda_0$$

(Here, λ_i is the local score for the correspondence at level i and t is the trickle-down parameter.)

Note how in this situation, if p drops by 50%, then since $t > 1$, the drop in the sum of correspondence-scores of all sub-expressions will drop by something close to 50% as well. This seems to be a more reasonable behavior, given that the structural evaluation score is computed in the end by a simple linear combination of these correspondence scores.

Some others have looked at the problem of assigning probability to structural relations. They are essentially descendants of evidential reasoning (Pearl, 1987). They require a causal model to start with, and then infer probabilities based on observations and simulation from the model. For example, Koller and Pfeiffer (1997) use a maximum-likelihood approach of fitting probabilities to rules, given the observed data. They require “rule skeletons” to be given first. Only then can they do Knowledge-Based Model Construction (KBMC) to build a Bayes net of the causality of the domain, and then use that causality to find the set of probabilities which give the greatest

likelihood to the observations. In contrast, our approach learns the prior probabilities of the relational structures themselves, and can then learn the appropriate models for those probabilities. That is, we try to both infer the probabilities and induce the model from the evidence alone.

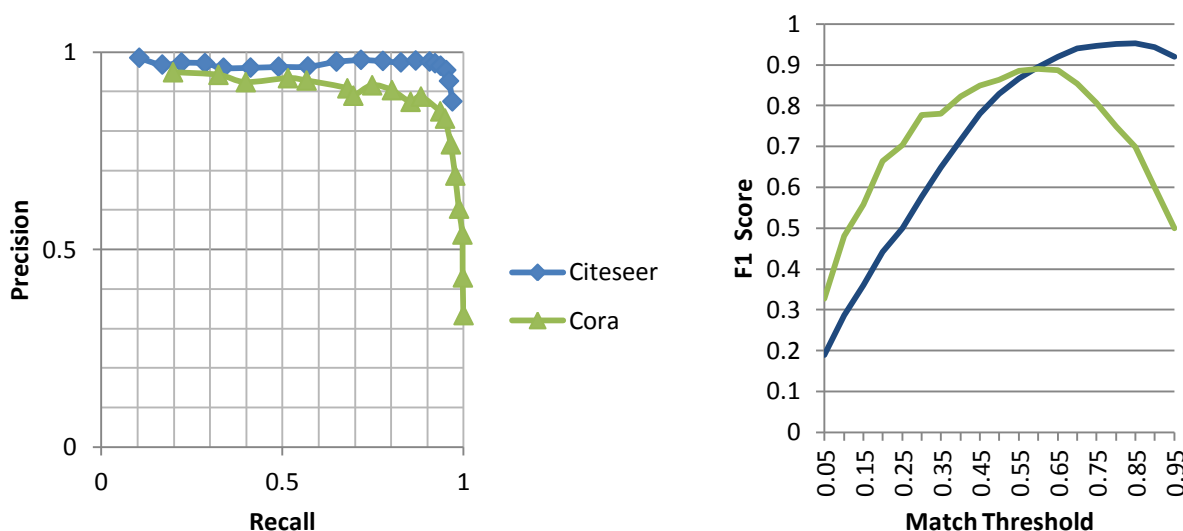
There is almost no other literature on deriving probability from similarity assessments. The literature which does look at the problem either looks at it purely theoretically (Wellman, 1994), or uses similarity to affect rough estimates of probability only. For example, Blok et al (2002) present a heuristic for estimating the probability that a fox has trichromatic vision, based on the probabilities that a dog and a goldfish might have it, and their respective similarities to the fox. We would instead build a probabilistic model of our generalization of animals, and use it to predict the probability for foxes. Thus, our probability calculations never use a heuristic, and are carefully grounded in the reality of observed commonalities.

In summary, the probabilities in a generalization are based on shared relational structure, indicating the *roles* of the objects being described rather than the relational predicates themselves. This means that the probabilities are less likely to be conflated over multiple real-world roles (such as the two drivers in Table 2-1), making them much more meaningful. It also means we are deriving probability based on similarity rather than the other way around, an approach which is quite novel in the literature. Yet we believe the only way to go from real-world relational data, with an arbitrary language describing arbitrary objects, to something generalized and probabilistic, is by using similarity to find the commonalities in that data (§2).

3.3.2 Threshold Values

Recall that there are two parameters for controlling the behavior of SEQL. The first is σ , a threshold which the similarity between the base and target must surpass in order to be accepted and incorporated into a generalization. It is tempting, given our goals of creating a feature description of the whole domain, to set σ to be as low as possible, so that as much of the domain as possible will be incorporated into the generalization. However, in practice, we have found that a low σ actually makes the matches so poor that the features constructed from them are meaningless. A value of 0.8 to 0.9 for σ (out of a maximum of 1, since the similarity score is normalized) performs much better in this regard.

Figure 3-9. The effects of the match threshold value on two similar datasets. (a) Plots the Precision-Recall curve over varying threshold values. (b) Uses these curves to compute the F1 score at every threshold.

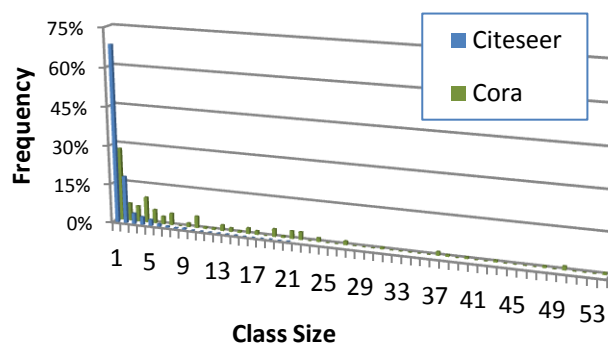


In general, we find that a value of 0.6 works well in the most situations. We have never had an experiment where the optimal threshold was less than 0.6, nor where the success rate at 0.6 was

very far from the optimal rate. Occasionally, there are domains where the algorithm can perform slightly better by moving the threshold slightly higher, as shown in Figure 3-9.

There are factors to indicate when a higher threshold is likely to help. For instance, when there is more noise or the differences between correct classes are more subtle, then a higher threshold may help, since more finesse may be needed to distinguish between the proper classes. However, for the most part, we have found that the

Figure 3-10. The difference between the datasets is the distribution of sizes for the correct classes.



greatest factor is simply the size of the correct classes. When fewer, larger classes are desirable, then a threshold of 0.6 is likely to be best. When more, smaller classes are called for, then a higher threshold such as 0.8 may do better. This was the case in the two datasets above, which both consisted of the same type and shape of data from the same domain (described in section 4.5). However, one dataset was distributed into larger classes than the other, as shown in Figure 3-10.

Secondly, when our generalizations grew very large (including many infrequent statements), we have found it useful to invoke a probability cutoff ρ . This means that any statements with probability less than ρ are ignored during the primary match process for the sake of speed, efficiency, and to reduce the chance of over-fitting. However, they are kept around for a certain

number of rounds, inversely proportional to the cutoff itself, in case they should ever rise above the cutoff again. When ρ is low, a great many expressions are included in the generalization which only occur a very few number of times throughout the case descriptions and there is a high risk of over-fitting the data; however, when ρ is high, we risk losing the significant low-frequency information that is crucial for feature discrimination.

We have found a value of 0.2 to behave reasonably in our experiments. This can be adjusted based on how important it is to the user to retain low-frequency information and how much memory is available for use.

3.3.3 Normalization and other Distance Metrics

In order to account for cases of different sizes, SEQL performs a normalization operation on the similarity scores. If this was not done, then two large cases which matched perfectly would get a higher score than two small cases which matched perfectly, because the two large cases would share a greater amount of relational structure. This in turn would lead to an unfair bias, over-fitting towards cases that simply contained more information.

The way we usually account for this is to do normalization to one of the two cases. That is, the similarity score between any two cases is divided by the score of matching one of the cases to itself (called a *self-match score*). The question that remains then is which one of the two cases to use. The default is to always simply use the base case. One reason for this is that under normal

conditions⁵, SEQL never compares two generalizations, and when it does include a generalization in the comparison, it is always the base case. Hence, normalizing to the base score ensures that we normalize to the generalization if there is one. Since the generalization is usually the larger of the two cases, it usually has the higher self-match score. This essentially limits the normalized score to a range between 0 and 1, as one would expect.

There is however, one potential problem with this scheme. As the generalizations grow larger and larger from embodying more and more cases, the normalized score will tend to drop. This may artificially constrain the growth of a generalization, since fewer normalized scores would then exceed the threshold σ . In situations like this, one could resort to normalizing to the target score instead. An advantage is that, for the default SEQL algorithm, the target case is always the same for a given iteration. In practice though, the irregularity that stems from often having normalized scores greater than 1 leads to poor generalizations. Furthermore, in domains where normalizing to the base did not work, normalizing to the target actually has the opposite problem: generalizations become more and more accepting of new cases as they grow larger.

A third option is to have the computer always automatically normalize to whichever case has the either the lower or higher self-match score. The advantage of this is that it will be slightly more regular than matching to the base or target, since the normalized score will always be either smaller or greater than 1. The disadvantage though is serious: in a series of tries matching a given case to each of a sequence of other cases, the highest score is no longer certain to be the

⁵ There are some alternative algorithms that SEQL may use to do generalization, described in the Appendix. However, the algorithm given in this chapter is far and away the most commonly used.

best match. For example, the scores between matching case A to case B and matching case A to case C cannot be compared, since one might be normalized to case A and not the other. Since SEQL's default behavior is indeed to match a given case to each of a series of others (although it does so greedily), it suffers from this loss of a consistent frame of reference. Therefore, this is only really recommended for alternative generalization algorithms, such as where the comparisons are all done at once and globally and so no such frame of reference could be found anyway.

There are many other ideas which have not yet been tried. One could always match to the target, since it is constant for a given iteration, and use the inverse of the score to get a normalized score less than 1. Or one could adjust σ through normalization too, to reduce the pull towards larger or smaller generalizations. Unfortunately, there are only so many novel ideas that one can invent and try in the timespan of a single thesis.

Similarly, there are actually other choices for a distance metric besides those that account for probability in Section 3.2.2.1. For example, suppose that there is a generalization G of cases X_1, \dots, X_n , which we would like to match to a new case Y . In the above section, we describe how one can simply use SME to compare the structure of G to the structure of Y , weighing the strength of each correspondence by its probability. However, one could also use some of the more traditional, aggregative methods for clustering on a non-cartesian space. That is, one could compare Y to each of the member cases X_1, \dots, X_n individually, and then report the highest, lowest, or mean score.

Many of these ideas have been tried, particularly during the citation database experiments reported in Chapter 6. However, the design space is large enough that there is still room for plenty of new ideas to come.

4 Learning and Prediction

The generalization process itself is a form of learning. Many experiments have already demonstrated the ability of the SEQL algorithm to do very well at classification problems. For example, Kuehne et al. (2000) showed that SEQL performed at humanlike levels in categorizing stories. More recently, SEQL has also been successful at classifying sketches as well as music, which both will be discussed briefly in Chapter 6.

Furthermore, SEQL is flexible in that it can perform either supervised or unsupervised learning. Recall from Section 3.1 that a generalization can consist either of all the instances of a given type, or only the most similar ones. In the former case, the learning is supervised: the type is assigned, and all cases of that type are formed into the same generalizations. In the latter case, the learning is unsupervised: cases are clustered according to similarity and each cluster may or may not correspond to its own novel type.

A common machine learning paradigm is to make observations, and then find a hypothesis which explains those observations. This chapter examines the difference between making hypotheses based on whole-case comparisons, as SEQL does, and hypotheses based on

individual propositions and entity values. We explain how our approach is motivated by a combination of the two. The second section describes our technique for propositionalization. The third section examines some traditional machine learning techniques we can then invoke on our propositionalized relational data.

4.1 Approach to Learning

Our approach to learning is model-based. That is, we build up an abstraction (i.e., a model) of the evidence, and generate predictions about novel/hypothetical instances by comparing them to the model. Although generalization is a model and a means of learning in and of itself, it is a form of *reductive* learning. That is, the model supports comparison between the cases as a whole, but not between individual characteristics of them. Although it is possible to do prediction with a reductive learner, such as by candidate inference, this is always based on a comparison between cases (and/or generalizations) as a whole. This comparison (for which we use structural analogy) is essential for relational representations of data, since it is important to figure out how the information in the two cases is aligned, and by no means do we wish to replace it. However, once the structure of the data is aligned, it is then possible to do more than this first comparison, using some of what has been achieved in the field of proposition-based learning where this alignment is taken for granted because it is inherent to the representation of the data. The key ability we will gain is the ability to use not just the structure but *also* the values of some arguments when making predictions.

The primary motivation in reductive learning is usually to answer whether or not a novel instance is a typical case for the model. In our implementation this means determining whether the similarity score between the new case and the generalization is high enough that the case could be included into the generalization. This focus on typicality of the case for the model makes reductive learners excellent at classification.

However, reductive learners are not always so good at making other predictions about the data. Recall the example about dogs and cats from Section 3.1. In that example, the system might correctly make a generalization about cats and a generalization about dogs. These models may be very good at answering whether a new instance is a dog or not. However, any other question about the new case, such as its gender, would not be modeled very well.

As a concrete example from our research, consider the following case, taken from the Whodunit experiments⁶ (Section 5.2):

Table 4-1. A difficult example for a reductive learner.

Case: August 1998 bombing in Omagh, Ireland
(numberOf person killed omagh-attack 55)
(minimumNumberOf civilian killed omagh-attack 2)
(someNumberOf civilian wounded omagh-attack)
(location omagh-attack ireland)
(isa omagh-attack (attackUsingFn bomb-roadvehicledelivered))

⁶ Some of the facts have been simplified for presentation, and the insignificant ones have been omitted.

The task is for the learner to decide who might have been behind a terrorist attack, given information about previous attacks. Of course, it is possible for the reductive learner to come up with an answer to this question. If it knows that 20% of the terrorist attacks it has seen were committed by Al-Qaeda, it could give the reductive answer that there was a 20% chance that the attack was done by Al-Qaeda. However, it will be outperformed by a system that can observe that, say, the attack occurred in Ireland, and 90% of terrorist attacks in Ireland are done by the IRA. This requires a higher order of learning, modeling not just the differences across cases as a whole, but also between individual characteristics of those cases such as location and perpetrator.

It is still possible for a reductive learner to get around this limitation. One solution is to generate a separate classifier for each question that might be asked. One could imagine making a separate generalization for each perpetrator and relying on the similarity between the novel case and the generalization to determine the likelihood it was done by that particular perpetrator. Of course, one problem with this approach is that all of the relevant questions must be known beforehand. Another problem is all the extra overhead it would take to make distinct sets of generalizations for each possible question. For this example, one could imagine having to make separate sets of generalizations by perpetrator, outcome, method, target, number of casualties, and location, just to start with. Also, suppose that the original set of inputs was simply to be disasters. It is unclear then whether these new generalizations by each characteristic should only be across terrorist incidents or across all disasters. When generalizing by location, it probably makes sense to partition by terrorist attacks first, since most terrorist attacks will be in the Middle East. On the other hand, when trying to answer the number of casualties, it might make more sense to

generalize by casualties first, since this may be more a function of the type of government and quality of health care available than whether the disaster was a terrorist incident vs. a hurricane.

There is yet another possible way for the reductive learner to get around its limitations and be successful at answering arbitrary questions. Instead of organizing everything into predefined categories, it could do unsupervised learning, and try to determine the best abstract classes on its own. The problem with this approach is that all the information for answering all the questions must then be compressed into a single dimension: the class of the new incident. Certainly, one dimension is not enough. Even in a perfectly deterministic world with no hidden information, in order to make the correct answer to any query from only one input dimension, the system would have to assign a different class to every non-identical case, which is over-fitting the problem rather than learning. It is too much to hope then that many questions can reliably be answered by a single dimension of classification.

Of course, one could also choose not to frame it as a classification problem at all. MAC/FAC is an approach which our lab has shown works very well under many circumstances (Gentner & Forbus, 1991; Law et al., 1994; Forbus et al., 1995). The idea is to find the most similar cases to the test case, and project their values onto it as possible answers via candidate inference (described in section 2.2). The hope would be that the most similar cases to this bombing were also carried out by the IRA. However, this process has problems too. Here, the most similar cases to this one turn out to all be car-bombings from the Middle East.

The key to solving this particular problem lies not in the structure of the case as a whole, but in one particular fact: that the location was Northern Ireland. This one slot value alone should set off all kinds of bells that the IRA was somehow involved. The information needed for doing this higher-order learning is already present in the generalizations. It would seem silly not to use it.

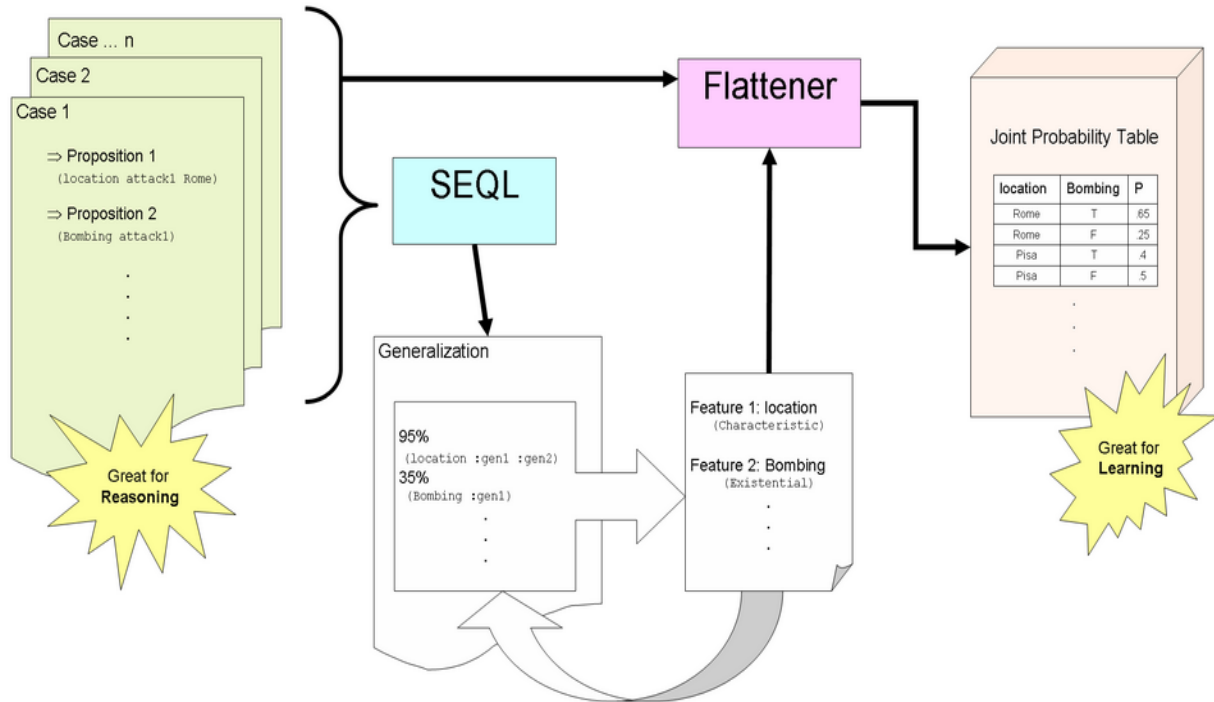
4.2 Flattening

Once the facts in the original cases have been uniformly sorted via analogy into sets of structurally equivalent facts, they can be treated as propositions and reasoned with probabilistically. Each of the generalized facts can be treated as a unique random variable, since it represents some concept that has an analogical equivalent in the input cases and yet is analogically different from the other abstract expressions. Thus, there is a 1:1 mapping between the abstracted facts in the generalization and random variables, aka *propositions*.

The importance of creating this mapping cannot be emphasized enough. The recognition that analogically equivalent facts can be reasoned with probabilistically is the key insight to this thesis, from which the others derive (§1).

Although each expression in the generalization can be represented by a single proposition, different expressions have different pieces of information that are important to convey. For this reason, we distinguish between two types of propositions, which we refer to as *existential propositions* and *characteristic propositions*.

Figure 4-1. A mapping is generated which can *flatten* relational knowledge into propositions and probabilities, or *unflatten* them back again.



The fact `(isa gun Weapon)` is an example of what would become an existential proposition. In this instance, the principal piece of information to convey is simply whether the fact occurred in each input case or not. It is essentially a Boolean proposition, which takes the value True when the corresponding fact exists in an input case, or False when it does not. On the other hand, the fact `(possesses agent1 (DollarFn 500))` is an example of what would become a characteristic proposition, capable of answering the question “how much money does the agent have?” Here, it is not enough to know that the agent possesses some dollar amount. Rather, we would like to know how much money the agent has. Thus, the values of a characteristic proposition correspond to some particularly interesting slot in the expression. In this case, the value for this proposition is `(DollarFn 500)`.

4.2.1 Assigning Values to Propositions

This begs the question of how to determine which slot of a proposition, if any, is interesting enough to be the value of the proposition. For example, in the expression

`(numCasualtiesOfType 4 Engineer attack)`, the best value to extract is the number 4 from the first argument. Not only are there many possible values for that slot, providing more grist for a careful learner, but the other two slots are probably not as important to know. In the example that this is drawn from, the entity *attack* is the case-entity; since the whole case is a description of it, knowing its value provides no new information at all. The entity *Engineer* may occur elsewhere in the case, or at least have other structural clues pertaining to its value. However, the second argument, containing the value 4, is functional in the other arguments: it has *one* value, which occurs only in relation to the other two arguments.

We currently make the decision on which value to extract with a similar mixture of meta-knowledge and heuristics. For example, we call a particular slot the value-slot of a proposition if it is known to be functional over the other slots for that predicate. Unfortunately, the only way for the system to know this for certain is with meta-knowledge. Hence, we check for the Cyc-specific predicate `FunctionalInArg`, which is intended to convey when an argument slot never has more than one value for any combination of the non-`FunctionalInArg` slots. Unfortunately, this predicate is not used very often in the Cyc KB, and its equivalent does not always exist in other knowledge bases, so it may have to be manually asserted. Since we would like to rely on such manual intervention and background knowledge as little as possible, we also have two other

heuristics that can be used: we determine a particular slot to be the value-slot if the only other slot contains the case-entity (such as in `(eventOccursAt attack Baghdad)`), or if the value of the slot is always numeric or always non-atomic. Although this solution works on the domains we tried, it may not be a satisfactory solution for a long-running generic agent.

Note that in assigning only one value to each proposition, we are making an implicit assumption that either enough information to convey the other slots will appear elsewhere, or else that the values of those other slots can be ignored. We have not found this assumption to be a problem for two reasons. First, every slot does play some role during the generalization process, which in turn determines which propositions will be created. Thus, if two values of a given slot play different enough roles across the input cases, then this may be represented automatically by splitting the corresponding proposition into two when SME refuses to match them together. Then the values of one proposition will correspond to one value of the slot, and the values of the other proposition will correspond to the other value of the slot. In effect, this creates a kind of two-dimensional proposition. A simple example of this is seen in Figure 3-5, where the proposition `(possesses ?x ?y)` is split into two because of a correct failure to match. Secondly, if the value really were important enough to warrant recording two different values for the proposition, then the input representation can account for this by either representing the value in another fact or in combination with the other value in a non-atomic term. Furthermore, an advantage of making this simple assumption is that it does cut out some additional search space, as will be shown in the next section.

While addressing value assignment, it is also important to address the question of closed world assumptions. That is, when the fact that corresponds to a particular proposition is missing from the case, what value should be assigned? We handle this problem differently for characteristic propositions than for existential ones. When a characteristic proposition's value is missing, we simply assign it the value `:no-value`. In existential features though, the `:no-value` is represented by the value `False`. Yet there is a nuance. We label an existential feature as `false` only if some of the entities that it mentions are also missing from the case. When all of the entities that the feature mentions do appear elsewhere, then we label it as a `:missing-value` instead. Missing values do not count for or against anything during learning – they are treated during counting as if the case didn't exist at all. This reduces non-causal dependencies in the data, since otherwise the values of all features containing the same entity would be identical whenever that entity was not present.

To continue with the example from Figure 3-5, examine the facts of the form `(isa ?x Weapon)` and `(possesses :genent0 ?x)`. The latter fact is split into two propositions since they do not match each other: `(possesses :genent0 gun)`, an existential features with values of `True` or `False`, and `(possesses :genent0 :genent1)`, a characteristic feature whose possible values are all of the dollar amounts observed for `:genent1`. For this generalization, which contains only one agent, simply knowing that the proposition `(isa :genent1 Weapon)` is true means that the agent carries a weapon, and hence `(possesses :genent0 gun)`. It also gives information about the proposition `(possesses :genent0 :genent1)`, since it will only have a dollar value if it is not a weapon and vice-versa. This splitting of facts of the form `(possesses :genent0 ?x)` into

two propositions is not arbitrary, but is done because the two usages of the form carry very different meanings – something which SME is able to discern even though their forms are identical, because the structure they participate in is not.

Behaviors like missing values and split propositions mean that it is possible to save search space even further through dimension reduction. The three propositions from the example could be reduced to a single proposition (`possesses :genent0 :genent1`) which conveyed the same information as the three if it had the value `False` when the possession was a weapon, or a dollar amount when it was not. This is a simple example and more complicated dimension reduction scenarios would not have a single, known propositional form to which they can be reduced. This sort of dimension reduction is not an optimization that we chose to pursue. However, the caching mechanism that we use, described in section 4.2.3, does take advantage of such redundancies to reduce the time and space needed to test hypotheses and count joint probabilities.

Note also that characteristic propositions could be further subdivided into continuous propositions (such as how much money the agent possesses), and discrete propositions (such as where the agent is going). In this work, we handle continuous propositions simply by discretizing their range into a set of three buckets using K-Means. We recognize that there are many ways for doing automatic discretization, such as the CARVE system (Paritosh, 2004), and would like to employ these at some point. However, in our work to date, discretization has not

been a frequent or severe enough problem to warrant the investment needed to take a more rigorous approach.

4.2.2 Efficiency Analysis

We have claimed that doing generalization with SEQL has certain advantages in run-time over other relational learners. In order to quantify this savings, we can make a comparison between the number of facts needed for a generalization by SEQL to describe a domain, and the number of propositions/terms that a propositional or ILP learner will need. In both cases, this is directly related to the amount of effort needed for a learner to search for a correct hypothesis. First, we must quantify certain parameters of the learning task. Let r be the number of relations/predicates in the domain, and let e be the maximum number of entities in a case.

Each relation will usually correspond to more than one possible proposition. The number of propositions needed will be the number of different combinations of entity assignments that are possible for that relation. We can determine that this will be at most e^k , where k is the number of entity assignments (i.e. joins) that are needed. This means that the number of possible propositions needed to describe a domain is bounded above by $r * e^k$.

This can be seen in the example of a square given in section 3.1. The example had only one relation, `edge`, with an arity of 2. However, there were 4 possible entities (corners) to choose from as the arguments. This gives an upper bound for the number of propositions as $1 * 4^2 = 16$,

which includes the possibility of having an edge that is reflexive (the two arguments are identical).

However, in generalization, all of the variable assignments have already been done. Each generalized entity of each fact corresponds to a set of entities that is predetermined, based on the single best mapping that the generalization has found. Therefore, $k = 1$ and the generalization will need to contain only $1 * 4^1 = 4$ generalized facts.

What seems like a small savings on a toy problem can grow quickly as the problem grows larger. In the next chapter, we will show other savings in efficiency due to using generalization. In particular, the fact that there is a known mapping with all of the generic entities assigned allows us to use caching to significantly speed up hypothesis testing.

The term e^k then, is the maximum number of “copies” of a proposition needed to represent all the possible entity combinations for a given relation. In general, k is equal to the maximum number of joins / variable assignments; however, in our implementation $k=1$ thanks to the mapping between entities that is built up through structural analogy.

From this, we can determine the size of the hypothesis space, i.e. the number of possible hypotheses in the domain. Suppose that we want to count the number of hypotheses which contain at most T terms (where each term consists of a proposition and a value). Each hypothesis will then have to cover some choice of up to T of the P possible propositions. Thus, there are

, or $O(P^T) = O((r^*e^a)^T)$ possible combinations of propositions. However, we can bound this a little bit more tightly by noting that only one of the e^k propositional copies of a relation will appear in any case, and so only one is needed in any hypothesis. This means that the e^k can be factored out of the exponent, leaving us with $O(r^T e^k)$ possible combinations of terms.

In order to generate a hypothesis, each of the (at most) T terms in one of these combinations must be assigned a value. The number of possible hypotheses then becomes the product of the number of possible combinations of propositions with the number of possible combinations of values. If we call the maximum number of possible values of a proposition V , then we can calculate this second combination the same way as the first. I.e., there are V^T , or $O(V^T)$ possible combinations of value assignments. The final equation for the size of the hypothesis space becomes:

$$O(r^T V^T e^k)$$

Again, in our implementation, we can substitute $k=1$. We can also be a little more precise by calling the maximum arity of a proposition a . Then the number of entities to choose from for a hypothesis with T propositions, each of which contains at most a entity slots, is $e=aT$. This means the number of hypotheses for our implementation is:

$$O(r^T V^T aT)$$

Note the similarity between this equation and the equation given by DeRaedt (1998) for the size of a hypothesis space needed to do relational learning through propositionalization:

$$O(r^T V^{aT} (aT)^k)$$

There are two differences between this equation and our own. The first is in the value of k . As described above, the construction of a single analogical mapping allows us to make the savings in efficiency from setting $k=1$. The second difference is that the number of value assignments to choose increased by a factor of a . This corresponds to the space that would be needed to assign a value for each of the a entity slots in a term. Although we make a simplifying assumption in assigning only one value per proposition, we describe in the previous section how structural analogy makes this a reasonable assumption. Taking the quotient of the two equations, we see that altogether, the use of structural analogy gives us a savings on the order of $O(V^{T(a-1)}(aT)^{k-1})$ and reduces the hypothesis space by $O(r^T V^{aT} (aT)^k - O(r^T V^T aT)) = O(r^T V^{aT} (aT)^k)$.

SME has been shown to run in quadratic time, i.e. $O(N^2)$ where N is the number of facts in the base/target cases (Forbus and Oblinger, 1990). SEQL on the other hand, needs to find the best generalization for each of M cases, and each time the number of generalizations to choose from will be less than the time after it. It therefore operates in $O(M \log M)$ time. Although note that if it is run with a threshold of 0 to create one large framework for learning, then it runs in $O(M)$ time. At worst then, when taking the time for SME into account, SEQL operates in $O(N^2 M \log M)$. Therefore, given a learner that is linear with respect to the number of hypotheses, we can expect to experience savings in run-time when $N^2 M \log M \ll r^T V^{aT} (aT)^k$. We might therefore expect to outperform other algorithms when either the amount of input data (M and N) is smaller,

or when the amount of relational structure (a and k) is higher. This is borne out in the experimental results.

In summary, we have found a sensible and concise way to incorporate probabilities into relational knowledge through a generalization process. A smaller number of possible hypotheses leads to a savings in the time needed to explore the space of possibilities and select the best hypothesis for explaining the observations. However, this is just one of two very important efficiency considerations to make during learning. The first, described above, is the number of hypotheses that need to be tested; and the second, described below, is the time that it takes to test them. We have clearly demonstrated the savings for the first element of this duo. In the next section, we will show how caching saves even more in efficiency on the second element.

Taken together, these algorithms allow us to do learning from relational knowledge in a way that handles noise while still being representative and discriminative after a great variety of training examples (§1). The probabilities are based on shared relational structure, indicating the *roles* of the objects being described rather than the relational predicates themselves, which we believe are more meaningful dimensions. The results of our research in chapters 5 and 6 support this belief. Moreover, as the analysis in this chapter and the next indicate, finding common structure is both more efficient and useful than simply finding common predicates (§3).

4.2.3 Caching

In order to increase efficiency in learning, it is important not only to reduce the number of possible hypotheses, but also to reduce the time that it takes to test each one. There are two basic approaches for hypothesis testing. Algorithms such as ILP which are designed to run on top of a large, optimized knowledge base or database will often do this by simply running a separate query for each hypothesis. Other approaches rely on counting and storing the relative frequency with which each hypothesis occurs ahead of time. This can be done either in a very large count table or in some other more efficient data structure, and it can be done either partially or exhaustively, caching the relative count (i.e. joint probability) for every possible combination of terms. We use the latter technique, exhaustively caching every count in a very efficient data structure. The reduced size of the hypothesis space that structural analogy provides us also means that this caching will require much less space, making it more viable.

That is, we build what is effectively one large joint probability table for the set of propositions by counting the combination of values present in each case. This allows us to query for the probability in the observed data of any possible combination of analogically different formula groundings. For example, it would allow us to very quickly calculate the percentage of attacks that occurred in Ireland, for each possible perpetrator. This table is all that is needed to invoke any statistical model in existence (§1).

Since even with analogy and generalization to reduce the size of the problem, a complete joint probability table might still be unreasonably large, we also use an AD-Tree (Moore and Lee, 1998 and Anderson and Moore, 1998) to cache the sufficient statistics in a more efficient way. The AD-Tree reduces the needed size immensely by recognizing that many of the combinations don't need to be stored at all. It organizes the combinations into a tree of alternating *adnodes* (which vary the attribute/proposition being counted) and *vnodes* (which vary the value). Each adnode also stores the number of times that the combination of propositions in the beam from the root to the node has occurred.

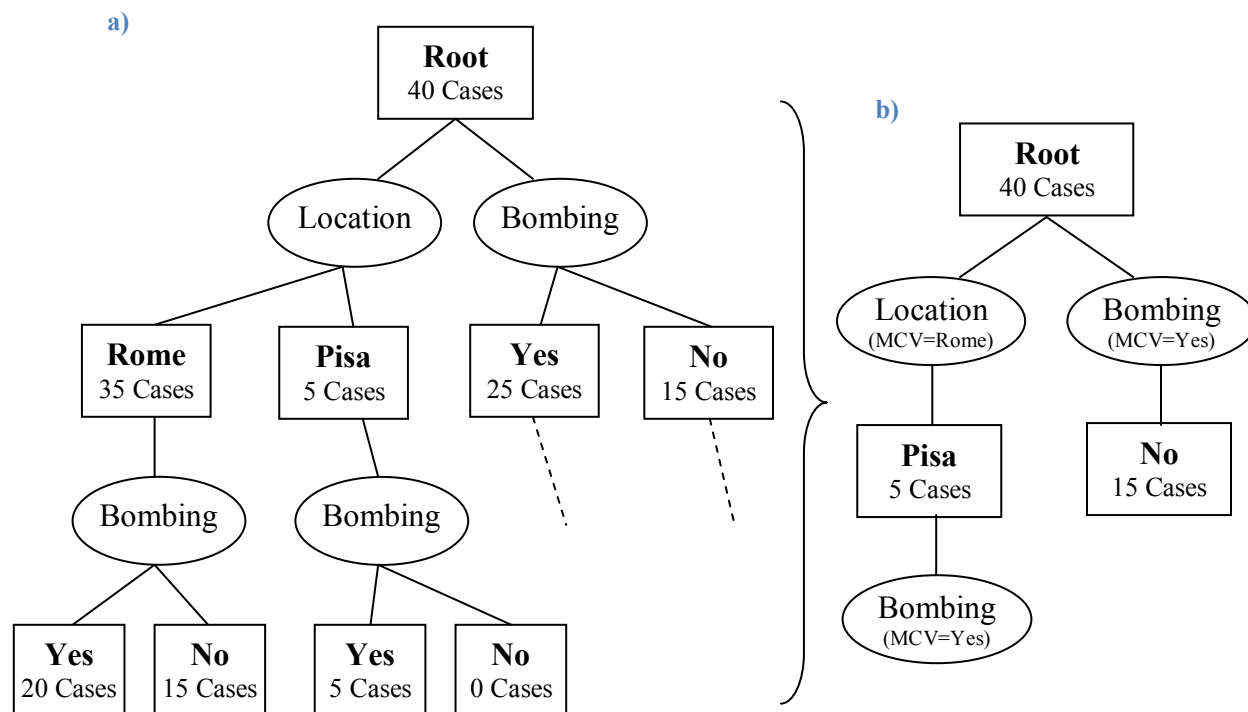
Additionally, the AD-Tree makes the following optimizations to reduce the required space:

1. Propositions are sorted, and an earlier proposition is never a child of a later proposition. This prevents permutations from being stored separately.
2. Once the count is down to 0, a NIL is stored instead of a subtree.
3. At points in the tree where it would make more sense to just store the subset of matching records (instead of a subtree), it does so.
4. The biggest space-saver of all is made by recognizing that for any proposition with n possible non-zero values, only $n-1$ of them ever needs to be stored, since the count for the other value can be found by subtraction. E.g. a binary proposition will only ever need to store the count for one value. Furthermore, to reduce the needed space as much as possible, the value that isn't stored is always the most common value (MCV).

The AD-Tree is a way to accurately count any combination of propositions in the data in constant time with respect to the number of cases, and it provides orders of magnitude savings in

both time and memory. Together with analogy and generalization, it makes statistical modeling of relational data very easy to do.

Figure 4-2. An AD-Tree caches the sufficient statistics for calculating joint probabilities. Several optimizations can be made to reduce the space needed for the tree from (a) to (b).



4.3 Statistical Modeling and Prediction

Although any statistical model of the generalization could conceivably be used with our technique, we tried it with two of them in particular: Bayesian networks, and association rules. Of these, we have put the greatest focus on the learning of association rules. Rules were a little

simpler to incorporate into our existing framework, worked fine for the domains we looked at, which only required predicting one or two values, and were more straightforward to adapt for incremental learning, which was an earlier goal of this work.

4.3.1 Bayesian Networks

A Bayesian network is a probabilistic graphical model that represents a set of propositions and their probabilistic dependencies. Beyond the usefulness of the representation itself, a Bayesian network has a variety of applications. It can be used to induce the probabilistic dependencies of a domain in the first place (using algorithms for finding the best fit of a network's structure to the domain observations), to find cases which are anomalous to the domain as represented by this dependency structure, to determine which questions are important to ask next when given partial information, and to do probabilistic inference.

Learning the structure (aka shape) of a Bayesian network is an excellent way to do learning on a novel domain as a whole. Rather than be driven by predicting a particular output proposition as a goal, it seeks to model the dependencies between all of the propositions of the domain at once⁷.

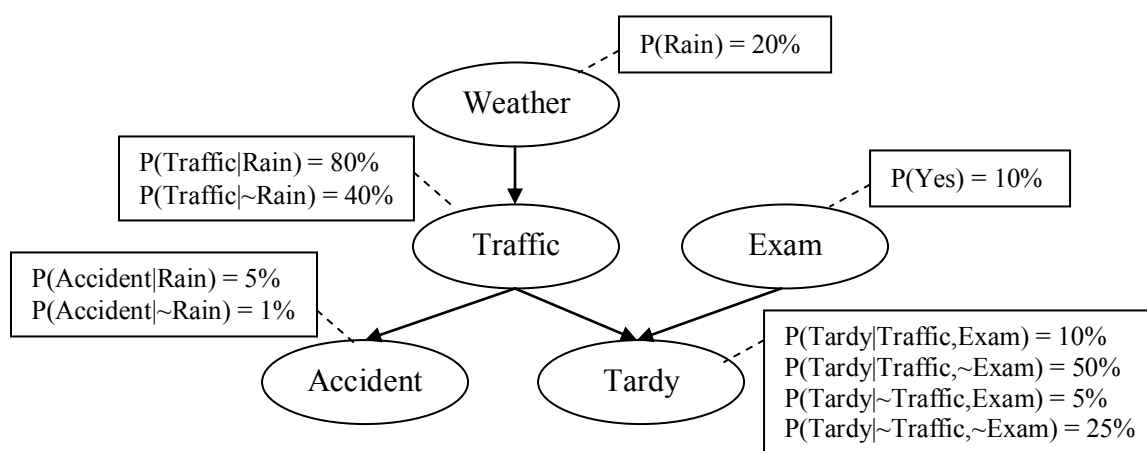
Our structure learner takes a hill-climbing approach. To be specific, we compute the shape of a Bayes net using simulated annealing with random restarts. It uses operators such as adding/removing a parent of a node, and swapping two nodes, until it finds the optimal shape.

⁷ To be accurate, a Bayesian network actually searches for those propositions which are *conditionally independent*, and hopes to accurately model the dependencies by extension.

Optimality is defined by the shape which best fits the data seen so far, as evaluated by a scoring heuristic. The heuristic we used is a Bayes Information Criterion estimate (Friedman and Yakhini, 1996).

Once a good shape has been found, the Bayes net can act as a kind of probabilistic reasoner for doing probabilistic inference. For example, consider the following Bayes net:

Figure 4-3. An example of a Bayes Net.



The network says that the chance that a student is tardy depends on both the traffic, and whether there's an exam scheduled for that day. Traffic also depends on the weather, and affects the chance of having an accident. The lack of an arrow between the traffic and the exam nodes means that these two statements are *independent*: knowing the value of one does not help us in determining the value of the other. Similarly, the lack of an arrow between the tardiness and the accident nodes indicate *conditional independence*: once we know the traffic, it doesn't help to

know either the tardiness or the chance of an accident to predict the other. Each node summarizes the probability of it taking a certain value, given the possible values of its parents.

To do probabilistic inference, one can tell the network various facts that are known, leaving the unknowns unasserted. Then upon asking for one of the unknowns, the network can simply look up the corresponding probabilities. For example, suppose the user asserted that there was an exam on Monday, and then asked for the probability that Eric (a student) would be late on Monday. Despite not knowing anything about the traffic, the machine can still return an answer by simply adding the correct probabilities. Here, it would add up the probabilities for $P(\text{Tardy}|\text{Traffic,Exam})$ and $P(\text{Tardy}|\sim\text{Traffic,Exam})$, weighted by $P(\text{Traffic})$. Since it also does not know anything about the weather, it would also do a weighted sum to determine $P(\text{Traffic})$. That is:

1.
$$P(\text{Traffic}) = P(\text{Traffic}|\text{Rain}) * P(\text{Rain}) + P(\text{Traffic}|\sim\text{Rain}) * P(\sim\text{Rain})$$

$$= .8 * .2 + .4 * .8 = .48$$
2.
$$P(\text{Tardy}) = P(\text{Tardy}|\text{Traffic,Exam}) * P(\text{Traffic}) + P(\text{Tardy}|\sim\text{Traffic,Exam}) * P(\sim\text{Traffic})$$

$$= .1 * .48 + .05 * .52 = .074$$

Thus, the system would estimate that Eric had a 7.4% chance of being late on Monday. If we then told it that the weather was rainy, then $P(\text{Traffic})$ would rise to 80% (simple lookup), and so $P(\text{Tardy})$ would rise to 9%.

Note that in our implementation, each node is actually a proposition that can be expressed as a fact about constrained variables. So the Tardy node actually would correspond to (tardy :genent1 :genent2), where :genent1 is whatever matches analogically to the student in the generalization. Also, since our implementation relies on analogy, it takes a case of facts as input rather than a set of facts drawn directly from a working memory of assertions (although it is a simple enough step to construct a case based on the contents of working memory). To continue with the example above, the arguments to the Bayes Net inference engine might be:

Case: ((isa Eric Student) (hasExam Eric Monday) (weather Monday Rainy)).

Ask: ((tardy Eric Monday)).

To answer this, the computer would add the tardy fact into the case, and then flatten it via analogy to the generalization. This would indicate which nodes corresponded to which statements. It would then go through the simple sums above, and unflatten the answer back:

(withProbability .09 (tardy Eric Monday)). This prediction is based not just on how well the structure matches the generalization (which may be used to choose which generalization and model to use in the first place), but also on the values of slots such as Rainy which it believes to have some causal relationship to the concept it has been queried about.

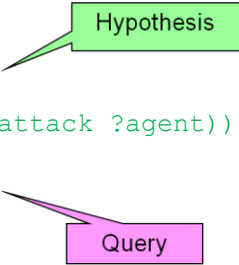
4.3.2 Rule Learning

We also have applied association rule learning algorithms to do prediction. In particular, we learn a rule-list for each possible output value of the proposition we are interested in.

Our definition of an association rule follows from that of Agrawal, et al. (1996). That is, we use it to mean a conjunction of literals which implies another conjunction of literals. By literal, we mean a proposition-value pair such as `(<Proposition 12: (location :attack ?value)> . Baghdad)`. (Examples in this section are taken directly from the Whodunit experiments described in the next chapter, which used rule-learning extensively.) Thus, given a set of all possible literals L and a query $Q \subset L$, the goal of the rule learner is to find a hypothesis $H \subset L$ which best fits $H \Rightarrow Q$.

Example:

Figure 4-4. A learned rule



The diagram shows a learned rule in Prolog-like syntax. The first part, `(implies (and (location ?attack Philippines) (agentCaptured ?attack ?agent))`, is enclosed in a green callout box labeled "Hypothesis". The second part, `(perpetrator ?attack MoroIslamicLiberationFront))`, is enclosed in a pink callout box labeled "Query".

```
(implies (and (location ?attack Philippines) (agentCaptured ?attack ?agent))
  (perpetrator ?attack MoroIslamicLiberationFront))
```

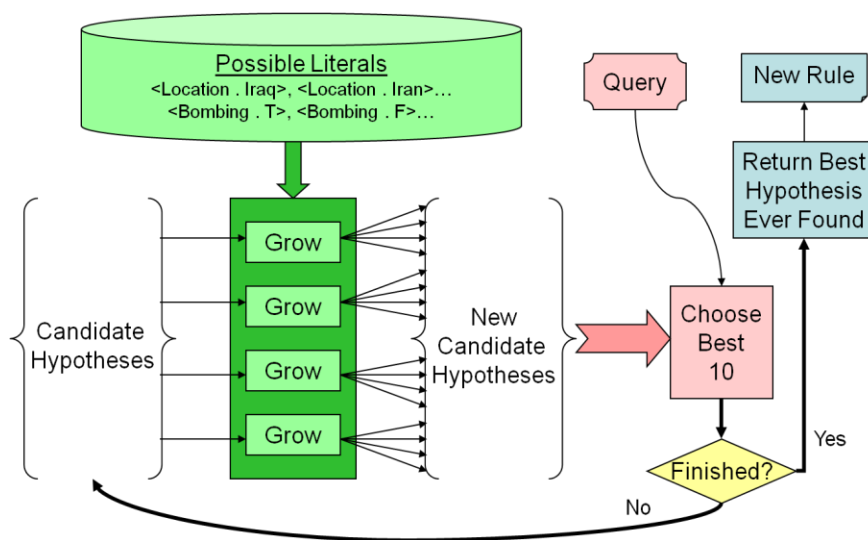
As is typical, we define the confidence in a rule as $\text{count}(H \wedge Q) / \text{count}(H)$. We also define the *relative* support of a rule as $\text{count}(H \wedge Q) / \text{count}(Q)$. This is equivalent to the traditional definition of support, since $\text{count}(Q)$ does not vary within a single invocation of the rule learner on a given query, and it enables us to use the more flexible rule performance metric described below.

For this application, since we prefer to be generic to the choice of domain, and since we are generating so many rules at once, then rather than use minimal thresholds for support and confidence as is customary, we instead prefer the rule which maximizes the minimal value

between relative support (reduces false negatives) and confidence (reduces false positives). Ties are broken by the rule with the maximal value between them. As will be shown in Chapter 6, we have found this approach to work the best for the most situations.

Similar to the goal of the single-rule learner, the goal of the rule-list learner is to find a small set of hypotheses $\{H\}$ which, taken disjunctively, provide maximal support for Q with minimal loss of confidence.

Figure 4-5. The Rule Learner



The actual rule learning is done by starting with a list of candidate hypotheses H , initially empty, and performing a breadth-first search with a beam-width of 10 over the space of possible hypotheses. On each iteration of the search, we further specify each hypothesis in H by adding literals from the set of possible literals L . We then re-evaluate each hypothesis, choose the best

10 again, and re-iterate until H is empty or the rules are 5 terms long. The beam-width of 10 was chosen simply because it was sufficiently exhaustive, and the rules were capped at 5 terms to prevent over-fitting.

The rule-list learner is simply a recursive implementation of the rule learner. We start by learning a single rule for $H_1 \Rightarrow Q$. On the next iteration, we try to learn a rule for $H_2 \Rightarrow Q \wedge \neg H_1$. This process continues until we reach a maximum of 5 rules (again to prevent over-fitting) or until no more rules can be found.

After unflattening the literals found by the rule learner back into their relational structure, the result is a set of logical axioms for each generalization which can be reasoned with and used to make predictions about any case which sufficiently matches the generalization. Again, these predictions will be based not just on how well the structure matches the generalization, but also on the values of certain slots which it believes to have some causal relationship to the concept it has been queried about.

5 The Whodunit Experiments

A great deal of the research and analysis found in this dissertation was devoted to experiments in counter-terrorism. These experiments are the only ones in which we tested both probabilistic generalization (section 4.1) and statistical modeling (section 4.2) against two controls: non-probabilistic generalization and simple exemplar retrieval. They are also the only experiments in

which we have a real performance analysis of the statistical modeling step. In short, these experiments are central to the arguments made in this thesis.

The experiments all used data from the Terrorist Knowledge Base, an extension of the Cyc Knowledge Base containing terrorist incidents that was provided to us by Cycorp, Inc. Each incident was hand-entered and checked by domain experts.

Basic case construction was performed upon each entity representing a terrorist attack, as laid out in section 2.1. That is, for each attack, facts mentioning the attack as well as attributes of the entities from those facts were put together into a single case. The result was that each case provided a relatively informative description of each terrorist attack. Examples of these cases can be seen in Table 5-1.

Some of the facts in the knowledge base contained meta-knowledge, such as the author of an incident or a natural language string describing it. These were filtered out according to predicate. Furthermore, in their original form, some facts contained Rule Macros: predicates which Cycorp used to simplify and compress certain relational structures. These facts were all expanded into their full form, in order to provide more structure for the analogical matching algorithm to latch onto. For example, `(RelationInstanceExistsCount 3 Person organismKilled TerroristAttack-March-1985-Rome-Italy)` was expanded into `(thereExistExactly 3 ?X (and (Person ?X) (organismKilled TerroristAttack-March-1985-Rome-Italy ?X)))`.

Table 5-1. Two examples of case representations for terrorist attacks.

TerroristAttack-March-1985-Rome-Italy	TerroristAttack-March-20-2000-Basilan-province-Philippines
(thereExistExactly 3 ?X-7034 (and (Person ?X-7034) (animalWoundedIn TerroristAttack-March-1985-Rome-Italy ?X-7034)))	(thereExistExactly 22 ?X-7002 (and ((InstanceNamedFn "school children" PersonTypeByOccupation) ?X-7002) (agentCaptured TerroristAttack-March-20-2000-Basilan-province-Philippines ?X-7002)))
(thereExistExactly 0 ?X-7033 (and (Person ?X-7033) (organismKilled TerroristAttack-March-1985-Rome-Italy ?X-7033)))	(thereExistExactly 5 ?X-7001 (and ((InstanceNamedFn "teacher" PersonTypeByOccupation) ?X-7001) (agentCaptured TerroristAttack-March-20-2000-Basilan-province-Philippines ?X-7001)))
(intendedAttackTargets TerroristAttack-March-1985-Rome-Italy (InstanceNamedFn "Alia, Royal Jordanian Airlines" HumanlyOccupiedSpatialObject))	(thereExistExactly 53 ?X-7000 (and (Person ?X-7000) (agentCaptured TerroristAttack-March-20-2000-Basilan-province-Philippines ?X-7000)))
((CityInCountryFn Italy) CityOfRomeItaly)	(thereExistExactly 1 ?X-6999 (and (Priest ?X-6999) (agentCaptured TerroristAttack-March-20-2000-Basilan-province-Philippines ?X-6999)))
(CapitalCityOfRegion CityOfRomeItaly)	(CalendarMonth March)
(CalendarMonth March)	(dateOfEvent TerroristAttack-March-20-2000-Basilan-province-Philippines (DayFn 20 (MonthFn March (YearFn 2000))))
(dateOfEvent TerroristAttack-March-1985-Rome-Italy (MonthFn March (YearFn 1985)))	(eventOccursAt TerroristAttack-March-20-2000-Basilan-province-Philippines (InstanceNamedFn "Basilan province, Philippines" GeographicalRegion))
(eventOccursAt TerroristAttack-March-1985-Rome-Italy CityOfRomeItaly)	(KidnappingSomebody TerroristAttack-March-20-2000-Basilan-province-Philippines)
(PhysicallyAttackingAnAgent TerroristAttack-March-1985-Rome-Italy)	(TerroristAttack TerroristAttack-March-20-2000-Basilan-province-Philippines)
(TerroristAttack TerroristAttack-March-1985-Rome-Italy)	

In the first section, I describe initial work on this domain. The second section presents results and analysis of the Whodunit experiments. Finally, in the third section we use this domain to examine the question of finding the appropriate level of representation.

5.1 Early Whodunit Experiments

Much of the SEQL algorithm itself was written before this thesis work began. Early experiments with this work already showed promise. For example, in Kuehne et al (2000), it was shown that SEQL performed at human-like levels when asked to do categorization on a large set of stories. Humans were first given a set of 60 stories by Ramscar and Pain (1996) in order to model human category learning. When SEQL was given the same sets of stories, its behavior mirrored that of the human subjects, with a few notable exceptions. However, these exceptions did cause conjecture about some theoretical limitations of this early version of the algorithm, which were later found in practice. Chief among these limitations was that the algorithm had no mechanism for preventing successive generalizations from becoming so abstract that they lost any inferential power.

This chief limitation was erased when probability was introduced into the SEQL algorithm for this research in counter-terrorism. Facts that didn't occur in every single case were no longer thrown away. This meant that as many cases could be introduced to the generalization as needed, without too much loss of detail through abstraction.

Table 5-2. In (a), two generalizations made by SEQL without any probabilities are indistinguishable. In (b), the probabilistic generalizations contain much more information.

(a)

Generalization of 37 attacks by the Popular Front for Liberation of Palestine.	Generalization of 43 attacks by Sendero Luminoso
(TerroristGroup genent1)	(TerroristGroup genent1)
(InternationalOrganization genent1)	(TerroristAttack genent2)
(eventOccursAt genent2 genent3)	(perpetrator genent2 genent1)

(b)

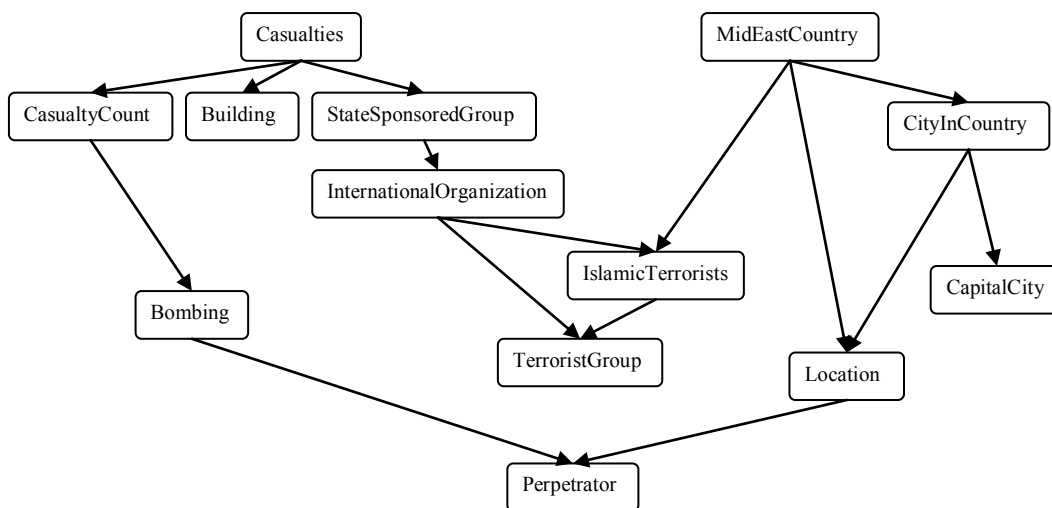
Probabilistic Generalization of 37 attacks by the Popular Front for Liberation of Palestine.		Probabilistic Generalization of 43 attacks by Sendero Luminoso	
100%	(TerroristGroup genent1)	100%	(TerroristGroup genent1)
100%	(InternationalOrganization genent1)	100%	(TerroristAttack genent2)
100%	(eventOccursAt genent2 genent3)	100%	(perpetrator genent2 genent1)
95%	(perpetrator genent2 genent1)	91%	(eventOccursAt genent2 genent3)
89%	(TerroristAttack genent2)	70%	(CasualtyDesignatingPredicate genent4)
84%	(CasualtyDesignatingPredicate genent4)	67%	(relationInstanceExists genent4 genent2 genent5)
81%	(relationInstanceExists genent4 genent2 genent5)	58%	(CasualtyDesignatingPredicate genent6)
78%	(relationInstanceExistsCount genent4 genent2 genent5 genent6)	56%	(relationInstanceExistsCount genent6 genent2 genent7 genent8)
73%	(OrganismClassificationType genent5)	47%	(relationInstanceExistsCount genent4 genent2 genent5 genent8)
51%	(KillingByOrganism genent2)	44%	(relationInstanceExistsCount genent4 genent2 genent7 genent9)
43%	(relationInstanceExists genent7 genent2 genent8)	44%	(CasualtyDesignatingPredicate genent10)
43%	(organismKilled genent2 genent9)	42%	(CapitalCityOfRegion genent3)
38%	(relationInstanceExists genent4 genent2 genent10)	42%	((CityInCountryFn Peru) genent3)
35%	(MiddleEasternCountry genent3)	42%	(OrganismClassificationType genent11)
35%	(IndependentCountry genent3)	30%	(CertainDistantCountriesWithInterestsInTheHormuzArea-HormuzArea-Topic genent12)
35%	(CountriesInTheSurroundingRegion-HormuzArea-Topic genent3)	30%	(LatinAmericanCountry genent13)
27%	(eventOccursAt genent2 genent11)	27%	((FoundingMemberFn NATO) genent12)
27%	(CasualtyDesignatingPredicate genent12)	27%	(NuclearWeaponStateUnderNNPT genent12)

Even the early, unpublished results of running SEQL (match threshold .7) on these cases showed a marked difference. This preliminary test used a series of 3,379 relational descriptions of the terrorist attacks provided to us by Cycorp, Inc. in the 2004 version of their ResearchCyc Knowledge Base. The vocabulary of the knowledge base consisted of over 36,000 concepts,

over 8,000 relationships and over 5,000 functions, all constrained by 1.2 million facts. The case descriptions varied widely in size from 6 to 158 propositions, with the average being 20. The results of running SQL demonstrated that generalizations of 30 cases or more, which all looked almost identical in the non-probabilistic version of the algorithm, now could be accurately distinguished (Table 5-2).

From this generalization, we were able to build a Bayes Net from all of the cases. This was done, as described in section 3.2.3.1, by using simulated annealing with random restarts to learn the optimal shape of the network according to a Bayes Information Criterion (Friedman and Yakhini, 1996) heuristic. The lowest-scoring nodes were culled, and we show the largest subnet in Figure 5-1.

Figure 5-1. Bayes Net generated by a Whodunit Experiment



This probabilistic abstraction was taken to the ultimate level when we tried building a Bayes Net of the domain. Since the procedure requires building a statistical model of a generalization, this meant making one large generalization of the entire domain. That is, we tried running SEQL with a match threshold of 0 so that every case would be incorporated into one very large generalization.

The resulting Bayes Net was published (Halstead and Forbus, 2005) as a proof of concept. It was encouraging that the features which describe location were all closely connected, as were the features which describe the terrorist group. All things considered, these preliminary results demonstrated a clear benefit from using the probabilistic version of SEQL and showed that the flattening process really did allow us to do successful statistical modeling of relational data.

5.2 Answering Whodunit

The results in the previous section were encouraging enough to explore further. An important task for analysts in counter-terrorism is to come up with plausible hypotheses about who performed an event. Details of these events often come in pieces at a time, and it is important to figure out possible suspects quickly and accurately, as well as to provide explanations for these suspicions.

For example, recall the pre-election bombing in Madrid, Spain. While the Spanish government originally claimed that the Basque Separatist group ETA was the most likely suspect, evidence

quickly mounted that Al Qaeda was very likely responsible. Multiple, highly coordinated attacks, for example, are more similar to Al Qaeda's modus operandi than previous ETA actions. This is an example of what we call the Whodunit problem.

5.2.1 Definition of the Whodunit Problem

Stated more formally, given some event E whose perpetrator is unknown, the Whodunit problem is to construct a small set of hypotheses $\{H_p\}$ about the identity of the perpetrator of E . These hypotheses should include explanations as to why these are the likely ones, and be able to explain on demand why others are less likely.

This is a difficult problem, but one which concisely expresses a key task that intelligence analysts perform. We therefore define a more restricted class of Whodunit problems to work with:

- *Formal inputs.* We assume that the input information is encoded in the form of structured descriptions, including relational information, expressed in a formal knowledge representation system. Note that we do not require uniform representations in each input; that is, we treat each case as simply a collection of arbitrary predicate calculus statements rather than as an object with predefined slots that may or may not be filled.
- *Accurate inputs.* We assume that the input information is completely accurate, i.e., that there is no noise.

- *One-shot operation.* Once the outputs are produced for a given E, the system can be queried for explanations, but it does not automatically update its hypotheses incrementally given new information about E.
- *Passive operation.* The hypotheses are not processed to generate differential diagnosis information, i.e. “tells” that could be sought in order to discriminate between the small set of likely hypotheses.
- *Supervised learning.* We allow the system to train on a set of pre-classified examples. For some algorithms, this involves forming non-overlapping generalizations over those examples.

The assumption of formal inputs is reasonable, given that producing such representations from news sources is the focus of considerable research in the natural language community these days. The assumptions of accurate inputs, of one-shot, passive operation, and of supervised learning are good starting points, because if we cannot solve this restricted version of the problem, it makes no sense to try to solve harder versions.

The Whodunit problem is an excellent domain for exploring relationships between similarity and probability. The input data consists entirely of arbitrarily high order symbolic relations with arbitrary structure between them. This means we will have to pay careful attention to structure in order to get probabilities over the correct statements (i.e. those which uniformly correspond to the same concept within each case). There is a very large number of records of terrorist attacks

on which to train, but there is also a large number of possible perpetrators to choose from during testing.

The Whodunit experiments were designed to analyze how different analogical learning algorithms performed on the Whodunit problem, and also to verify whether we could build probabilistic generalizations that were accurate enough in their descriptions to produce correct results in such a complex, real-world domain.

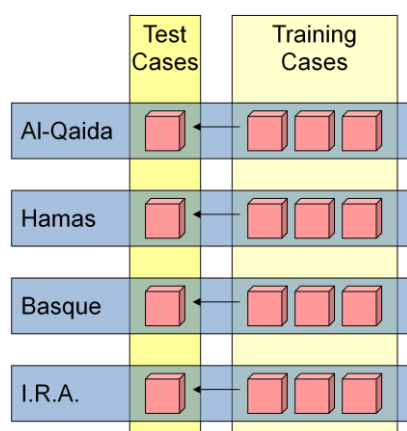
We used three criteria for bounding the size of the set of hypotheses n . The most restrictive is producing only a single perpetrator, i.e., guessing directly who did it. The least restrictive is a "top 10" list, rank ordered by estimated likelihood (for which we use the SME structural similarity score). The middle ground is the "top 3" list, which has the virtue of providing both the best and some hopefully mind-jogging alternatives. These criteria are motivated by discussions with members of the intelligence community. The best answer and top 3 criteria test for accuracy and near-misses. The top 10 list is closer to what is desired by analysts, whose subsequent analyses and information collection would be focused by this set.

5.2.2 The First Whodunit Experiment

For the initial experiment, we used the same 3,379 descriptions of terrorist attacks that were described in section 5.1 and generated from the ResearchCyc knowledge base for those tests. These attacks had 450 different perpetrators between them. Of these, 98 perpetrators were

chosen for the experiment, on the sole basis of having at least 3 attacks performed by them in the knowledge base. The other perpetrators and their corresponding cases were discarded, along with any cases for which the perpetrator was unknown. This left a total of 2,235 cases and 98 perpetrators to use in the initial experiment.

Figure 5-2. Inputs to Whodunit



From each set of attacks by a given perpetrator, we pulled one case at random to serve as a test case. The rest were used for training, as shown in Figure 5-2. Three different learning strategies were then employed on the training data. This was done to test how probabilistic generalization compared to previous structural learning algorithms.

The first strategy, MAC/FAC, was an existing algorithm for doing exemplar-based analogy to hypothesize the perpetrator. It is designed to quickly find the most analogically similar exemplar from a large library of possible cases. This is done by creating a content vector for each case describing the frequency of each predicate within it. A simple dot product with the test case (probe) can be used to estimate the most similar cases, and from these SME can be used to do a more careful similarity measurement.

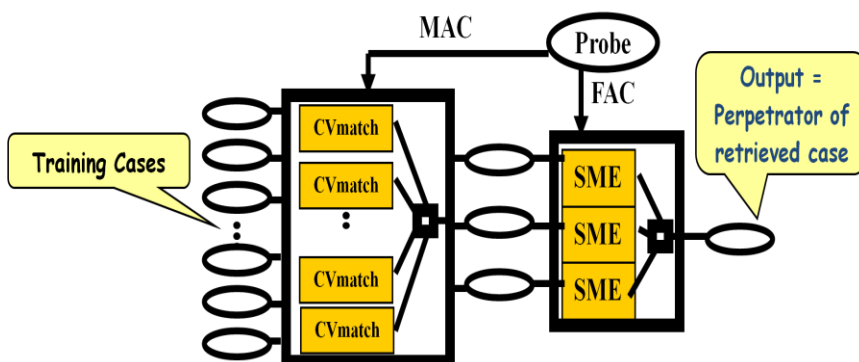
Figure 5-3. Whodunit Strategy #1. MAC/FAC returns the most similar exemplars.

Pre-processing: Generate a content vector based on predicate count for each case in the library.

Step 1: Retrieve the cases that have the highest dot product with the probe case.

Step 2: Make analogies between the probe and the chosen cases to find the most similar cases.

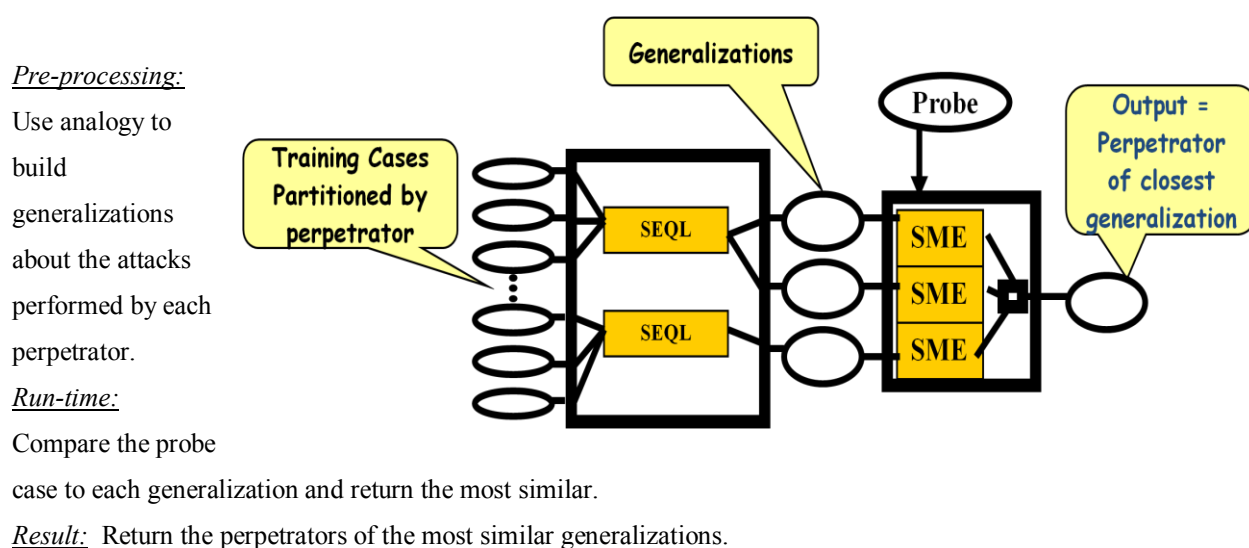
Result: Return the perpetrators of the most similar cases.



Intuitively, this method corresponds to taking what one is reminded of when hearing about E as the most likely suspects. People can be surprisingly biased about such decisions, e.g. the Spanish government stuck with its ETA hypothesis long enough to lose credibility. People also have their own lives, with many other kinds of things in their memories. A cognitive simulation need not have either of those limitations.

The second learning strategy focused on using the older, non-probabilistic version of SEQL, and the third strategy focused on using the new, probabilistic version proposed in this thesis. In both cases, a generalization is built of the cases for each perpetrator. Then analogy is used to find the generalization that is most similar to the test case.

Figure 5-4. Whodunit Strategies #2 and #3. SEQL returns the most similar generalizations.

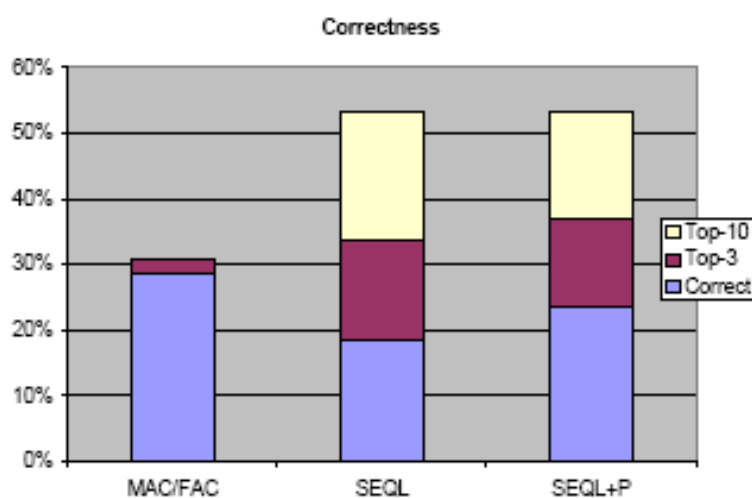


While examples are important, a powerful aspect of human cognition is the ability to make generalizations. These strategies are designed to reflect the use of that ability. Generalizations are important because they strip away what is accidental, and thus highlight what is essential about a class of similar examples.

The results of using all three learning strategies are shown as published (Forbus, et al. 2005) in Figure 4-5. In returning only the closest exemplar, MAC/FAC did surprisingly well. It identified the correct perpetrator 29% of the time and included it in its top 3 list 31% of the time. However, continuing to construct hypotheses beyond that point proved useless: no additional correct identifications were included. On the other hand, looking for the best generalization did not do as well as MAC/FAC at zeroing in on a single best hypothesis, getting it only 18% of the time for SEQL without probabilities, and 23% of the time for the probabilistic version. Both versions did slightly better than MAC/FAC on the top 3 list. However, where generalizations

really seem to be adding value is in the top 10 list, where both versions of SEQL included the correct perpetrator 53% of the time.

Figure 4-5. Results of initial Whodunit Experiment.



It is interesting that the higher-level abstractions generated by SEQL are simply not as good as finding the one most similar example overall in the data. However, those same abstractions save the day in the long run, allowing over-arching patterns in the behavior of a given perpetrator to come through. These patterns of behavior appear to be more similar from one perpetrator to another than the individual incidents are. Hence, the correct pattern is not always found on the first attempt. However, given enough choices, it makes more sense to observe the patterns, since paying attention to the most similar examples has a sharp drop-off in usefulness. Whereas given only one choice, it makes sense to find the most similar example as a whole and go with it.

We were surprised that the probabilistic version of SEQL did not perform much better than the original version. Part of this was because, as section 4-1 (esp. Table 4-2) demonstrated, it

required approximately 30 cases for the generalizations in the older version of SEQL to really become indistinguishable. In this domain, there was an average of only 22 cases per perpetrator, making the generalizations of many of the perpetrators still relatively coherent even without probabilities. However, we hope that probabilistic SEQL allows us to go one large step further: to go beyond reductive learning and build statistical models of the *values* of entities and attributes within a generalization. This was done and evaluated in a second experiment.

5.2.3 The Second Whodunit Experiment

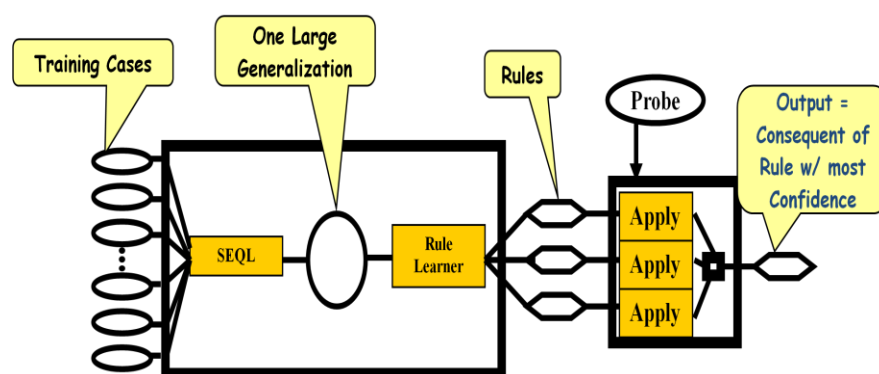
Once the code for doing statistical learning of association rules was in place (section 3.2.3.2), we ran the Whodunit experiment a second time. This time, non-probabilistic SEQL was left out of the evaluation and in its place we looked at rule learning as a fourth learning strategy. Again, this required building one large domain-wide generalization. Using the features extracted from this generalization, we learned a set of rules for predicting each possible perpetrator. All possible rules were applied to the probe case, and those which fired were sorted by confidence

Figure 4-6. Whodunit Strategy #4. Learn rules for the domain and apply them to the test case.

Pre-processing: Build a generalization and learn rules.

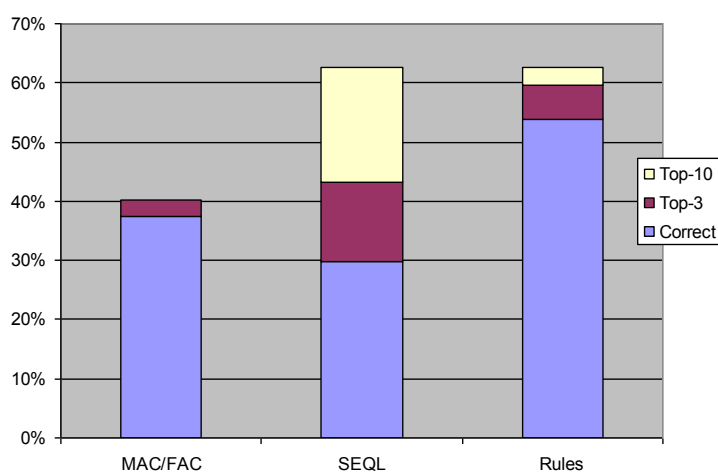
Run-time: Apply each rule to the probe case to generate hypotheses.

Result: Return the hypothesis with the highest confidence.



Another difference in running the second experiment was that more information had been entered about more cases. In fact, Cycorp, Inc. had split off a separate knowledge base, dubbed the Terrorism Knowledge Base (TKB), just to house all the information. The cases now varied even more widely in size, ranging anywhere from 4 to 762 facts, with the average being 24. Given the added information, we decided to increase the minimal number of cases per perpetrator from 3 to 6. Every other aspect of the experiment was set up the same way as before. This provided us with 67 perpetrators and 2,215 terrorist attacks on which to train.

Figure 4-7. Performance of three learning strategies



The results of this second experiment are shown in Figure 4-7. The rule learner does surprisingly well. It is able to return the correct answer on its first guess more than 50% of the time. SEQL finds as many correct answers in the long run, but is less certain in the beginning, providing the correct answer in its first guess only 30% of the time. Finally, MAC/FAC does a little better than SEQL on its first guess. Interestingly though, continuing to construct hypotheses from MAC/FAC beyond that point proved useless: very few additional correct identifications were included.

The rules generated by the rule learner look appropriate when examined. Examples of some of the rules it generates are in Table 4-3. As can be seen, many of the rules keyed in to the location of the attack. Nearly 40% of the rules generated refer to a specific location, and many more made a more general reference to the location, such as the term `(ISA ?LOCATION CITY)`. These location-based rules frequently had the highest confidence. Despite a maximum length of 5, the average rule length needed was only 1.9 terms, with 75% of the rules generated having 2 or less terms. This meant the risk of over-fitting the data with number of parameters was low. Similarly, the number of rules needed in a rule list was also low, averaging to 2.7 rules per perpetrator. The average lift⁸ for rules was very high (131 for first-generation rules), and lifts reached high into the hundreds in some cases. This is due again to a high output arity, causing a very high ratio of final to initial confidence.

Table 4-3. Sample of generated rules.

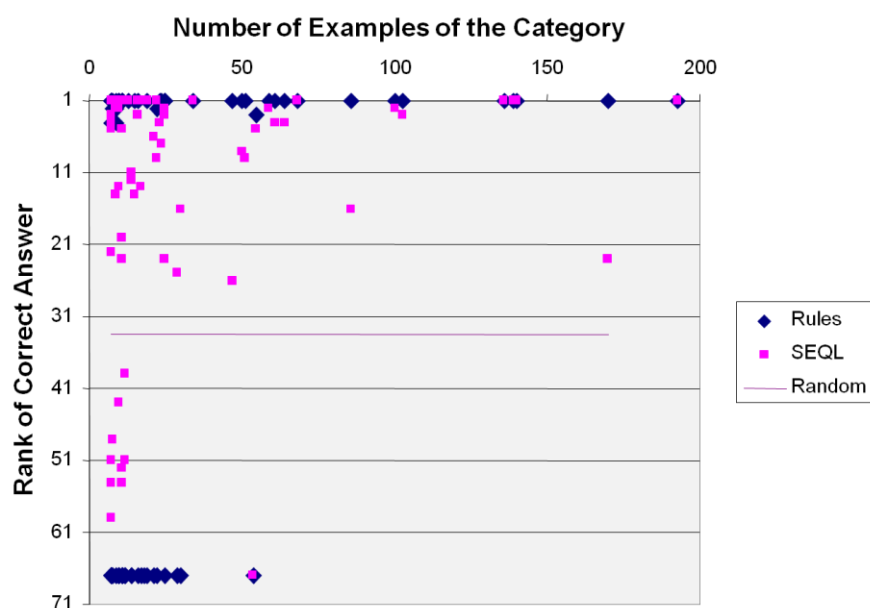
<p>A 45% correlation between the scores given by the rule learner and by SQL indicated some similarity between the cases on which they</p>	<p style="text-align: center;">Some Rules Learned</p> <hr/> <pre>(IMPLIES (LOCATION ?ATTACK CAMBODIA) (PERPETRATOR ?ATTACK DEMOCRATICPARTYOFKAMPUCHEA)) (IMPLIES (AND (LOCATION ?ATTACK PHILIPPINES) (AGENTCAPTURED ?ATTACK ?AGENT)) (PERPETRATOR ?ATTACK MOROISLAMICLIBERATIONFRONT)) (IMPLIES (AND (AGENTCAPTURED ?ATTACK ?AGENT) (ISA ?ATTACK AMBUSH)) (PERPETRATOR ?ATTACK REVOLUTIONARYUNITEDFRONT)) (IMPLIES (AND (THEREEXISTS ?X (AND (ISA ?X BOMB) (DEVICEUSEDIN ?X ?ATTACK))) (ISA ?LOCATION CITY)) (PERPETRATOR ?ATTACK BASQUEFATHERLANDANDLIBERTY))</pre> <hr/>
--	---

succeed or fail. Despite this, the number of input cases had a markedly greater effect on the rule learner than on SQL (40% vs. 17% correlation). All three algorithms improved with the

⁸ Ratio of confidence to expected/initial confidence, where the latter is confidence of the rule `nil => query`, which in this case is the proportion of cases that were committed by the queried perpetrator.

number of cases. However, the first two required only about 15 cases to dramatically improve their success rate, whereas the rule learner required 30 cases to really get going. Likely, this is because the rule learner tries to capture relationships between features, whereas the other two algorithms only do reductive learning. All three algorithms are robust to loss of information in the training set, degrading reasonably.

Figure 4-8. Performance with respect to the number of training examples



Overall, we were surprised by how well all three strategies performed, even the non-statistical ones, given the difficulty of the problem. Although each case contained an average of only 24 facts, there are over 100 features in the dataset. This meant that for any given record, over 75% of the features were missing. This made for a very sparse dataset. Fortunately, the closed world assumption that was made for existential features seems to have held up. Yet, when we consider that the arity of the output attribute was 67, it seems that those 100 features may not be enough. A random algorithm would select the correct perpetrator 1.5% of the time, and would get it right

with ten guesses only 15% of the time. Therefore, we believe that success rates of over 50% on the first guess are quite good.

In conclusion, the Whodunit experiments demonstrated marked success for all three strategies on a difficult problem. The rule learner in particular performed well at returning the correct answer as its first choice, although it generally required more examples to do so. More importantly, they demonstrated that the method proposed here for converting from relational, predicate calculus representations to feature-based representations and back again was viable and allowed us to apply statistically sound learning methods to complex, real world domains.

5.3 Reduced Vocabularies

Identifying the appropriate amount of detail in the input data is a problem for nearly every application of machine learning. As tempting as it is to learn from all of the available data, in a real-world application most of it will be irrelevant or redundant. Including such extraneous detail in an analysis will not only slow down the process, but may also lead to over-fitting and hence learning of an incorrect model. Clearly though, a balance must be found, since with too little data it becomes unlikely that anything can be learned at all.

The problem is perhaps even worse when dealing with relational data. Since most relational learning algorithms operate by comparing across instances of the relation itself, redundant relations become particularly dangerous. The more expressive a vocabulary, the more ways

there may be to express the same information. Unless all such language redundancies are detected ahead of time, relational learning algorithms will suffer.

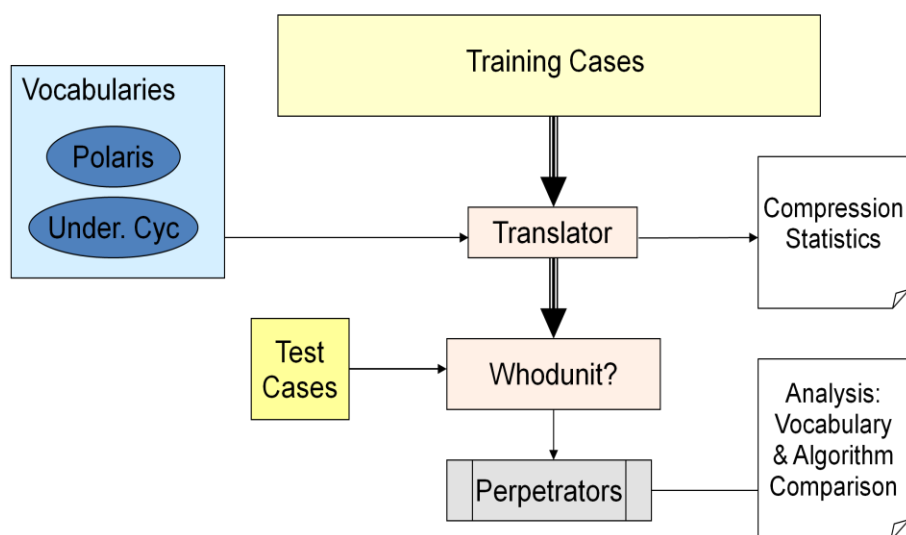
The issue is also particularly relevant whenever a learning system is to be deployed in any sort of real-world environment. Such environments tend to be brimming with unrelated observations. Robotic systems for example, will wisely choose to focus on only a few sensory inputs, which they can analyze carefully, rather than a cursory analysis of many inputs which would only confuse the picture. A similar application is the automatic extraction of knowledge from text. This is becoming more popular as the internet grows, and it is crucial to identify which relationships are useful to know, and which convey practically the same information.

In summary, we felt it was important for this research to explore the proper level of abstraction for the vocabulary itself. One reason is to help filter the less relevant information. A still more important reason is to eliminate redundancies. Since generalization over relations is a key component of the algorithm, and since everything relies on the ability to find analogical correspondences by comparing predicates, it is important to ensure that those predicates are expressed at the proper level of abstraction for the correct correspondences to be found. That is, predicates which are equivalent for the purpose of the domain should be written equivalently.

We therefore examine the contrasting effects between using a large, very detailed but often redundant vocabulary, and a small, consistent, but extremely simplified one on the Whodunit domain. To do this, we employ two externally designed reduced relational vocabularies (RRVs).

The cases from the previous Whodunit experiment were translated into each of these vocabularies, and the experiment was done again using the new representations. Then each vocabulary was evaluated based on both compression and performance. This research was motivated and assisted in large part by Michael Witbrock and Robert Kahlert of Cycorp, Inc., to whom we owe special acknowledgement.

Figure 4-9. Experimental Test Harness for evaluating RRVs



5.3.1 The Vocabularies

The two vocabularies were both chosen for their stated design goals of natural language understanding and improving the feasibility of automatic extraction of knowledge from text. This in turn was done for the long-term goal of building an intelligence analysis tool which would read and track stories such as those in the counter-terrorism domain, as they were

published, giving the analysts the ability to model and predict these events in real time while freeing them from the constraints on human memory and attention.

Additionally, one of the vocabularies – Polaris (Bixler, Moldovan and Fowler 2005) – had the further design goal of choosing predicates that would have the broadest semantic coverage with the least amount of overlap. As the designers of the vocabulary, Language Computer Corporation, put it: “*While no list [of predicates] will ever be perfect... this list strikes a good balance between being too specific (too many relations making reasoning difficult) and too general (not enough information to be useful).*” We shall see that this added goal makes a very large difference.

The Polaris vocabulary contains 40 predicates based on semantic relations: abstractions of the underlying relations between concepts. Semantic relations can occur in natural language within a word, between words, between phrases, and between sentences. For a better understanding, consider this example given by Bixler, Moldovan, and Fowler:

An example of semantic relations is the sentence “*He carefully disarmed the letter bomb.*” The compound nominal *letter bomb* alone contains at least 5 semantic relations: *letter bomb* IS-A *bomb*, *letter bomb* IS-A *letter*, *letter* is the LOCATION of the *bomb*, *bombing* is the PURPOSE of *letter bomb*, and *letter* is the MEANS of *bombing*. The sentence also includes several other relations: *He* is the AGENT of *disarm*; *carefully* is the MANNER of *disarmed*; and *the letter bomb* is the THEME (or object) of *disarmed*. Together, these semantic relations can give a structured picture of the event: who was involved, what was done, and to what; and what was the purpose, etc. of the object involved.

Of the 40 predicates found in Polaris, we only required 23 to describe the information present in the Whodunit Domain. Examples of predicates that were not used include kinship, synonym, companion, justification, and explanation. A full list of the predicates that were used is given in Table 4-4.

Table 4-4. Polaris Predicates needed for Whodunit

Agent	Goal	Result	Theme
Predicate	Purpose	Location	Time
Measure	Property	Part	Cause
Instrument	Topic	Belief	Associated
Reason	Source	Experiencer	Recipient
Possible	Entails	Isa	

The original representations of the data consisted of 2,581 predicates. Discarding facts which were not translated⁹ left a total of 2,113 original predicates. Therefore the translation into 22 predicates caused a 99% reduction in vocabulary size and a mean knowledge compression (the number of specific relations encompassed by an abstracted relation) of 105.7.

The other reduced vocabulary that we tried – Underspecified-Cyc – was a layer of abstraction built into the Cyc knowledge base itself. This vocabulary also had the goal of simplifying lexical semantics and natural language parsing. Each of the predicates in this vocabulary exists to free the NL parser from needing to specify exact semantic relationships when this responsibility

⁹ Either because they couldn't be translated or were irrelevant. See next section for details.

might be better handled post-parse by knowledge-based and contextual reasoning. Again, consider this example given in the Cycorp documentation:

For instance, (Contains-Underspecified love Siddhartha-Gautama) is an underspecified (and conventionally metaphoric) means of stating “*Siddhartha is in love.*” Inference rules which encode the common-sense relationship between abstract states and containment are responsible for producing the fully specified assertion (feeling-type-experienced Siddhartha-Gautama love) from this underspecified form.

Thus, the goal of this vocabulary was to align the predicates to natural human language use in order to simplify natural language processing. However, this vocabulary did not have the additional goal stated for Polaris of choosing predicates which had the broadest semantic coverage with the least amount of overlap.

The version of the Underspecified-Cyc vocabulary that we used contained 55 predicates. Again, not all of the predicates were needed to describe the information in the Whodunit domain. Predicates such as `orientation`, `without`, `under`, `off`, and `along` were not needed. This left us with a list of 20 predicates, which are shown in Table 4-5.

Note that unlike the Polaris vocabulary, many of the predicates in this vocabulary do seem to have some semantic overlap. For example, `at`, `on`, and `in` are all various ways of describing relative position or containment. The nuances between these predicates are not very relevant to this domain. Worse still, some of the translations (described in the next section) end up merging concepts that really are relevant distinctions. For example, `location` and `date` may both be

described by the predicate `in`, since an attack might occur *in* a given city/country as well as *in* a given month/year. One would expect that this might actually make the redundancy problem worse, and the results do bear this out.

Table 4-5. Underspecified-Cyc Predicates needed for Whodunit

Connects	Contains	AwareOf	About
Possessive	By	With	Measure
Affects	Disconnects	Expresses	Releases
Related	After	At	On
In	Location	During	Isa

Again, some of the original predicates were not translated, leaving a total of 2,198 original predicates that were compressed down to 20. Although this causes the same 99% reduction in vocabulary size, it caused a mean knowledge compression of 332.6, three times higher than that of Polaris. This reflects the fact that the more frequent predicates were often compressed more.

5.3.2 Translation

Two methods were used to translate from the original representations into the new vocabularies. Both methods take advantage of a hierarchy of predicates that was already built into the Cyc Knowledge Base, of which the Terrorist Knowledge Base is a subset. This hierarchy is instantiated by another predicate, `genlPred`. For example, the KB assertion (`genlPred`

`eventOccursAt situationLocation`) indicates that the predicate `eventOccursAt` is one of a set of predicates subsumed by the more generic predicate `situationLocation`.

For translating Underspecified-Cyc, the new vocabulary was already built directly into this hierarchy. Thus, the translation was mostly a matter of simple lookup. The only exception is that one new predicate, `Measure`, was introduced to describe the number/amount of something. Translation of this new predicate was done manually, following the same conventions that were used to translate `Polaris`.

For `Polaris`, translation was done by manually encoding a set of translation rules. Note that each new, reduced predicate directly corresponded to any of a set of original predicates. Furthermore, these original predicates usually had some common parentage in the `genlPred` hierarchy. Therefore it was not too difficult to write a handful of rules for each of the new predicates, describing under what circumstances it should occur, in terms of hierarchical collections of the original predicates.

For example, the following rule handles intentional actions:

```
(TRANSLATE (OR (DONEBY ?EVENT ?AGENT) (EVENTPLANNEDBY ?EVENT ?AGENT))
  (AND (AGENT ?EVENT ?AGENT) (GOAL ?AGENT ?EVENT)))
```

The above rule fires on any facts in the original Cyc representation whose predicates are either

`DONEBY` or `EVENTPLANNEDBY`, or are specializations of `DONEBY` or `EVENTPLANNEDBY`. Each such fact is

translated into two new facts: the first describing that the agent played some causal role in the event, and the second describing that the event was in fact a goal of the agent.

Some rules are designed to be chained together. Thus, some facts are actually translated by the successive application of more than one rule. An example of this can be seen in Table 4-6.

Some rules also introduce variables. For example, (NUMBEROFHOSTAGES TAKEN ATTACK 3) would translate as the first row of the table, introducing the variable ?AGENT. A stranger example because it introduces artificial structure is the statement (CLAIMS ALQAEDA THREAT23). This is actually translated into the constraints of the Polaris vocabulary as (SOURCE ?INFO ALQAEDA) and (TOPIC THREAT23 ?INFO).

Table 4-6. An example of the translation process

Cyc	(THEREEXISTEXACTLY 3 ?AGENT (AGENTCAPTURED ATTACK ?AGENT))
Rule 1	(TRANSLATE (THEREEXISTEXACTLY ?NUMBER ?VARIABLE ?FACT) (AND (MEASURE ?VARIABLE ?NUMBER) ?FACT))
Rule 2	(TRANSLATE (OBJECTACTEDON ?EVENT ?OBJECT) (AND (THEME ?EVENT ?OBJECT) (PREDICATE ?OBJECT ?PREDICATE) (RESULT ?EVENT ?PREDICATE)))
Polaris	(MEASURE ?AGENT 3) (THEME ATTACK ?AGENT) (PREDICATE ?AGENT AGENTCAPTURED) (RESULT ATTACK AGENTCAPTURED)

A total of 38 translation rules were encoded, which are listed in full in Appendix A.

Note that since the original Cyc vocabulary is much richer, many of the facts in the original data must be represented by more than one fact upon translation. Specifically, the average translation rule in Polaris turns one fact into 1.3 new facts. Overall, the number of facts in the domain went from 55,816 to 94,880, causing the average number of facts in a case to go up by 70%. This includes 1,967 facts that were not translated, 93% of which were irrelevant. The translation therefore did help to get rid of irrelevant information. The 135 facts that could not be translated but were relevant all used predicates that had not been formally identified and so could not be handled by Polaris, such as (PREDICATE NAMED FN "RELEASED UNHARMED" CASUALTY DESIGNATING PREDICATE).

For Underspecified-Cyc, the story was much the same. The number of facts increased by 86% to 103,852. 1,871 facts were not translated, only 42 of which were actually relevant. Thus it actually did a slightly better job of filtering out irrelevant information without discarding too many of the relevant facts. This is because even some of the informal predicates still fell into the Underspecified-Cyc hierarchy that was established in the KB itself. All of these statistics are summarized in Table 4-7 in the next section.

5.3.3 RRV Results

Once all of the original data was translated into the two new vocabularies, we simply re-ran the Whodunit experiment on each of the two new representations. We performed exactly the same

experiment: all of the same assumptions from the previous Whodunit experiment, and the same evaluation criteria, were applied again to the new data.

We use a few compression statistics to evaluate each vocabulary. Recall that *knowledge compression* measures the number of original, specific relations that are covered by a new, abstracted relation. We also look at the increase in the number of facts, which is equal to the number of abstracted relations produced by each specific relation present in the data. To get a sense of what the translation does to the data structurally to make learning easier or harder, we measure the change in average *information gain*, i.e. how much knowing one feature helps us to know the others. Finally, we look at how much better or worse the new vocabulary actually performed on the Whodunit problem.

The results of the experiment are summarized together with these compression statistics in Table 4-7. The table shows that using a reduced vocabulary creates a trade-off: one gives up small sizes in exchange for extra conciseness. The hope is to eliminate redundancy and irrelevant information and thus improve learning.

We see that while both vocabularies were good at filtering irrelevant information (especially Underspecified-Cyc, as described in section 4.3.2), only one of them actually improved learning performance. This is undoubtedly because the Underspecified-Cyc vocabulary actually made the redundancy problem worse, as described in section 4.3.1.

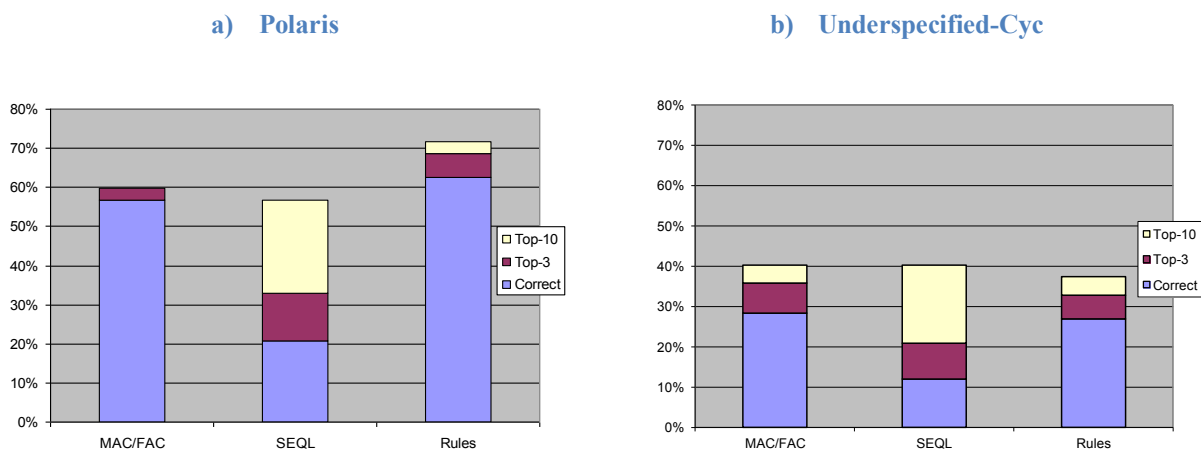
Table 4-7. Side-by-Side Comparison of Polaris and Underspecified-Cyc

Originally: 55,816 total facts	
Polaris	Underspecified Cyc
94,880 total facts (70% increase)	103,852 total facts (86% increase)
1,832 irrelevant facts were filtered	1,829 irrelevant facts were filtered
135 facts could not be translated	42 facts could not be translated
23 predicates abstracted from 2,113	20 predicates abstracted from 2,198
Mean knowledge compression: 105.7	Mean knowledge compression: 332.6
Mean information gain: +23%	Mean information gain: -17%
Mean performance gain: +6%	Mean performance gain: -29%

We should note here that neither of these two vocabularies were designed with this experiment in mind. This experiment has no relation to how the two might perform on their stated goals of doing natural language processing. However, it does seem clear that the additional goal of Polaris to choose predicates which would have the broadest semantic coverage with the least amount of overlap is very important to doing successful learning.

A more detailed investigation of the performance on each of the three learning strategies can be seen in Figure 4-10. Unfortunately, Underspecified-Cyc made the results worse for all three learners. However, the Polaris vocabulary improved two out of the three learners tested. MAC/FAC improved by 20% (p-value .045), and the Rule-Learner improved by 13% (p-value .015). Only SEQL suffered, worsening by 10% (p-value .004).

Figure 4-10. Results of running Whodunit on the reduced vocabularies



Closer examination reveals that the SEQL algorithm was hard-pressed. In the original vocabulary, SEQL generalized from case descriptions which contained an average of 24 facts each. However, 16% of those facts had to be discarded to preserve memory as dimensionality (case description size) increased with abstraction. Under the reduced vocabulary, which is already an abstraction of the original data, this information loss is compounded. When the average description size increases by 70%, so does the number of facts discarded by SEQL during generalization, which rises to 28%. Furthermore, the facts which remain carry less information than they did under the original vocabulary (the average reduced predicate corresponds to 106 different predicates from the original vocabulary).

However, it is interesting that the Rule Learner outperformed SEQL, despite relying on SEQL to create features from a generalization. This is because of the advantage that the rule-learner gets from higher-order learning. The added conciseness of the reduced vocabulary makes it easier for the rule-learner to select arguments that should serve as the values of a feature. A much higher proportion of the features become characteristic features, and the average arity increases by

250%. This plethora of feature values gives the rule learner more grist by having more relevant options to consider than before. Further analysis shows that when features are treated as existential and allowed only two values again (as 84% of them had under the original vocabulary), the rule-learner reverts to almost SEQL-like levels of performance.

We therefore conclude that reduced vocabularies may not be advisable to use in an algorithm which already relies heavily on abstraction (e.g. learning by generalization alone), but that a well-chosen reduced vocabulary is likely to improve those learning algorithms which are known to do well with large amounts of data (i.e. high dimensionality) and which can take advantage of the extra conciseness that it provides.

In summary, we find that rerepresentation can help to simplify and produce better results. It certainly makes it easier to filter irrelevant information. And a carefully chosen representation can go a long way towards improving learning by reducing redundancies. However, a reduced vocabulary also increases the amount of data and abstraction, which can hurt learners which do not handle these things well. All in all though, the correct choice of representation is a very big factor in determining the success in learning. Ideally, predicates should be simple and few, encompassing all of the relevant ideas with minimal amount of overlap.

Note that Jin Yan et al. (2003) proposed a theory of rerepresentation in analogical matching which could potentially be useful for this scenario. Their work might be extended into an

automated rerepresentation technique that would be useful for the generalizations produced by this research.

6 Comparison to other Algorithms

In addition to the Whodunit experiments, we also tested the performance of SEQL against several other popular techniques for doing statistical relational learning.

The first stage of the learning algorithm, generalization, was compared to other algorithms by several researchers other than myself. For instance, in (Lockwood et al, 2006), it was shown that SEQL could be used to learn human linguistic prepositions (e.g. “in”, “on”, “above”) from automatically analyzed sketches. Lovett (Lovett et al, 2006) and Deghani (Deghani and Lovett, 2006) also used SEQL to do category learning in the domains of sketching and music, respectively. Furthermore, they each demonstrated that SEQL could learn these categories from at least an order of magnitude (and often two) fewer training cases than other learning algorithms had required.

Some researchers have also tried alternative means of solving the Whodunit problem addressed in Chapter 5. It was described in that chapter how both the generalization and model-learning stages of our algorithm were tested on the Whodunit domain with impressive results. Others who have tried to implement Domingos’ relational Markov Logic Networks (Richardson and Domingos, 2005) -- a popular alternative discussed in the Background and Related Work

chapters -- to solve this domain found that the explosive search space of the MLN required too much memory to even finish the problem¹⁰.

However, in all of these scenarios, SEQL was used to do supervised learning only. In this chapter, we compare the unsupervised learning aspects of our approach to three state-of-the-art alternative algorithms for relational clustering. We compare our performance to a Bayesian modeling approach called Infinite Relational Models (IRMs) developed by Kemp and Tenenbaum (Kemp and Tenenbaum, 2006) on three domains, and also to Markov Logic Networks (MLNs) (Singla and Domingos, 2006) and Probabilistic Relational Models (PRMs) (Bhattacharya and Getoor, 2005) on four datasets from a popular citation matching domain.

6.1 Comparing Generalization to IRMs

In their paper on Infinite Relational Models (Kemp and Tenenbaum, 2006), Kemp and Tenenbaum introduce IRMs as a means of clustering relational data. It works by treating each cluster as a generative model of each relation; i.e., the probability that a relation will be true for a given sequence of arguments/entities is determined by which cluster the sequence falls into.

While this means that the IRM actually clusters *permutations* of entities rather than *individual* entities, clusters of the latter can still be obtained by analysis of the composition of the former.

¹⁰ Michael Witbrock, personal correspondence

A key feature of the IRM is that despite using the simple potency of Bayesian modeling, it is able to determine for itself the number of clusters by using a prior induced by a Chinese Restaurant Process (Pitman 2002). However, we believe IRMs to have serious drawbacks in relation to generalization via SEQL in that IRMs are forced to search over every permutation of entities, and also that the data must be transformed into a "flattened", functional space which, without SEQL, requires an explicit relational schema and limits the expressiveness of the representations.

6.1.1 The IRM Experiments

The comparison to Kemp's work was done across three different domains, all of which were presented in Kemp's original paper. The first and simplest domain involved clustering 50 animals based on 85 binary attributes used to describe them. The attributes were collected in a psychological experiment (Osherson et al., 1991). In the second domain, we looked at clustering members of the Alyawarra tribe of Central Australia, based on the highly complex relationships they use in their own kinship system (Denham, 1973). Thus, this domain examined the clustering of relations (26 of them) rather than attributes. In the final domain, we examined a medical ontology (McCray, 2003), containing 135 entities representing concepts in biomedicine, described by 46 attributes and 49 binary relationships. All three domains are described in more detail in the original paper by Kemp.

On all three domains, clustering was done using SEQL, wherein as described in Chapter 3, the distance metric for finding the clusters is the analogical similarity score returned by SME, with

modifications as given in 3.3 for calculating the similarity between two generalizations (i.e. clusters). However, note that since the cases do not occupy a continuous metric space, the clustering algorithm must rely on the distance between cases and/or clusters alone. This rules out algorithms such as K-means, which would require a Cartesian space in order to explicitly represent the points between cases. Beyond this constraint, any agglomerative clustering algorithm can be used.

For purpose of comparison in this experiment, we demonstrated three such algorithms. The first, the psychologically plausible GEL, was presented in section 3.2.2 and used in the Whodunit experiments of Chapter 5. It is a greedy algorithm which operates in $O(n \cdot \log(n))$ time, where n is the number of cases. It tries to merge each successive case with the clusters (i.e. generalizations) generated before it, from largest to smallest, until it finds one that exceeds the given similarity threshold; or if it doesn't find one, it simply becomes a cluster of one, and goes on to the next case.

The other two algorithms that we used were the standard and well-accepted Nearest-Neighbor and Quality Threshold (QT) clustering algorithms. In contrast with GEL, nearest-neighbor ignores the order of the cases and simply merges the two closest clusters and/or cases until there are no pairs left that match better than the given threshold. QT builds a hypothetical "ideal" generalization of each case by incorporating every case which is more similar than the given threshold. It then accepts the largest of these generalizations as truth, removes those cases from further consideration, and repeats until no more matches better than the threshold can be found.

6.1.2 Results of comparison to IRM

The first domain, describing binary attributes of animals, did not have a known ground truth.

Therefore, we have only the IRM results for a basis of comparison. However, it is not a difficult domain and so easy to inspect visually. Of the three clustering algorithms (GEL, nearest-neighbor, and QT), nearest neighbor performed the best here.

Table 6-1. Some clusters learned from animal feature data (a), and a comparison to the IRM model for clustering (b).

a)	siamese cat, persian cat, chihuahua, collie, german shepherd, dalmation	b)	Adj. Rand Index (to IRM results)	Number of clusters
	antelope, deer, giraffe, zebra, horse	GEL	0.66	17
	gorilla, monkey, chimpanzee	NN	0.81	12
	blue whale, humpback, walrus	QT	0.61	21
	killer whale, dolphin, seal			

Some of the clusters that nearest-neighbor returned are shown in Table 6-1(a). In comparison to the infinite relational model, it performed nearly identically, with a 0.81 Adjusted Rand Index (Hubert and Arabie, 1985) between them. An example of the kind of difference between them is that, whereas the IRM grouped all of the aquatic animals together, our analogy-based nearest-neighbor algorithm split them into two groups: one for blue whales, humpbacks, and walruses, and the other for killer whales (a type of dolphin), dolphins, and seals. The IRM and nearest neighbor algorithms also both settled on a count of 12 clusters. (In addition, Kemp reported that

two surveys of human subjects produced 10 and 13 clusters). Table 6-1(b) shows how closely each of the three algorithms matched the IRM.

In the Alyawarra kinship domain, just as Kemp did, we created a “ground truth” partition of 16 clusters based on demographic data, against which to compare the learned partitions. Each person was assigned to one of these clusters based on their gender, their age (older than 45), and their kinship section.

Table 6-2. Comparison of algorithms for learning from binary kinship relationships.

	Adj. Rand Index (to ground truth)	Number of clusters
GEL	0.28	32
NN	0.46	32
QT	0.44	17
IRM	0.59	15

The results for this domain can be seen in Table 6-2. Again, the nearest neighbor algorithm outperformed the other two, with a 0.46 Adjusted Rand Index to the ground partition. However, the IRM approach managed to do better than all three on this domain. Still, it is interesting that each cluster generated by the nearest neighbor algorithm was almost entirely exclusive to a single kinship section: the average kinship section consistency within a cluster was 92%.

For the final, medical ontology domain, we again compare the results to the same ground partition used by Kemp: a 15-cluster partition of the concepts, created by domain experts

(McCray et al. 2001). Again, we find that nearest neighbor did the best, but this time the QT algorithm managed to tie it. Also, both algorithms performed better than the IRM on this more complex dataset.

Table 6-3. Comparing the analogical algorithms to IRM on a more complex domain.

	Adj. Rand Index (to ground truth)	Number of clusters
GEL	0.49	23
NN	0.69	22
QT	0.69	30
IRM	0.53	14

In summary, on all three domains, we compared our results to those of Kemp, and found them to be very similar. Using an adjusted rand index to measure the difference from the “correct” clustering, we found that we did almost identically in the animal domain, slightly worse in the Alyawarra domain, and slightly better in the medical ontology domain. This third domain was the most complex of the three, providing the greatest relational structure, from which we feel our analogy-based algorithm was able to obtain an advantage.

Of the three clustering algorithms that we employed, Nearest-Neighbor performed the best. However, we should note that the GEL algorithm, in its reliance on the order of the inputs, is designed more to simulate human learning than to produce optimal results. Nonetheless, the SEQL system can utilize whichever algorithm is more appropriate for the task at hand.

Finally, by using SEQL, we were also able to go a step further and use analogical flattening to learn rules for the clusters -- something which the IRM model cannot do. For example, we learned that the animal cluster containing antelope, deer, giraffe, zebra, and horse could be best distinguished by the antecedent (and (hooved ?animal) (fast ?animal)).

6.2 Comparing Generalization to MLNs and PRMs

We also compared the results of our approach on the popular citation matching problem to those produced by MLNs and PRMs. More structure, and especially more complex, higher-order relational structure, tends to improve SME's performance rather than degrading it (although the worst-case is still $O(N^3 \log(N))$). However, SME has rarely been studied in domains like citation matching, where the amount of structure is minimal, which makes this problem particularly interesting.

We experimented on four databases, two of which were tried by Singla and Domingos (2006) using MLNs and two of which were tried by Bhattacharya and Getoor (2005) using PRMs. Each record of each database represents a document from technical research literature. Each document also has a title and an author and, in some of the databases, a venue, all represented by other entities in the form of strings. The goal then is simply to determine which entities actually refer to the same document. In our implementation, this means ending up with one

generalization per document, after clustering. We also perform the same entity resolution task on authors.

6.2.1 The entity resolution procedure

The strings representing the document titles are all normalized during pre-processing using the Porter stemmer (1980) together with a stop-list. For example, the words "computer," "computing," and "computation" all become "comput" in order to facilitate comparison. Singla uses the same pre-processing steps as well in his experiments.

However, the real task is to match strings despite missing or rearranged letters or words. To do this, we add a little more structure to the domain by adding relations to further describe the author, title, and venue strings. In order to precisely replicate the pre-processing of Singla's work, we would like to use the exact same relations as they did. That is, there is a relation `hasWord` tying word entities to the strings, and also a relation `hasTrigram` tying every 3-letter substring within a word to that word.

For example, if the word "least" is in the title of a citation, then Singla would include the following facts in its case description:

```
(hasTitle document7 title7)
(hasWord title7 "least")
(hasTrigram "least" "lea")
(hasTrigram "least" "eas")
(hasTrigram "least" "ast")
```

The only change that we make is to transpose the second argument into the predicate. We do this because the SME matcher only pays attention to the relational structure of facts that refer to an argument, not to the argument's name or spelling. Since we would like the trigram "lea" in one case to only match to the trigram "lea" in other cases, we need to make sure that SME notices the spelling.

We do this by rerepresenting the facts as shown in (a) below. Since these facts concern collection membership, and SME treats collections as predicates, it in turn sees the facts as shown in (b). Since SME only matches predicates which are identical, and since the syntax of the arguments that we care about now appear in those predicates, we expect SME to only match those facts which contain identical trigrams to each other.

Table 6-4. Examples of facts used for citation-matching

a) Standard Representation

b) Transposed for better structure-matching

<code>(hasTitle document7 title7)</code>	<code>(hasTitle document7 title7)</code>
<code>(isa title7 (PhraseWithWordFn "least"))</code>	<code>((PhraseWithWordFn "least") title7)</code>
<code>(isa "least" (WordWithTrigramFn "lea"))</code>	<code>((WordWithTrigramFn "lea") "least")</code>
<code>(isa "least" (WordWithTrigramFn "eas"))</code>	<code>((WordWithTrigramFn "eas") "least")</code>
<code>(isa "least" (WordWithTrigramFn "ast"))</code>	<code>((WordWithTrigramFn "ast") "least")</code>

For the primary goal of the experiment, determining which document citations refer to the same document, we construct a single case for each citation. This case consists of the facts describing the title, venue, and primary author, like those shown above. Note that in other scenarios, where

we would also like to test how well it does at determining which strings refer to the same author, title, or venue, we simply include only the facts referring to that field.

Once all of this pre-processing is done, then it is simple enough to run SQL on the cases we have created. It will merge those citations that it believes to be most similar into generalizations. At the end, we can label each generalization to be a hypothetical document entity, which all of the members of the generalization cite.

6.2.2 The citation resolution experiments

For each of the four databases, we ran the clustering algorithm on up to seven different tasks. The primary task was to correctly identify which document citations were actually referring to the same document. We also tried the algorithm on the secondary tasks of doing the same kind of cluster identification for the author, title, and venue fields. That is, determining which different spellings of an author actually referred to the same person, etc. Finally, we also tried each of these last three tasks with the added information gained from the hypotheses it had already formed about the document matches. This was to test whether the algorithm was making an accurate hypothesis set right away, or whether bootstrapping it with supplemental case facts gleaned from earlier hypotheses would provide any additional benefit.

As shown in the previous section, case construction for the primary task entails generating facts which describe the words and trigrams present in each of the author, title, and venue fields for

each document, as well as a fact that links the field to the document via the `hasAuthor`, `hasTitle`, or `hasVenue` predicate. For the secondary task of resolving these individual fields, we include only the facts for the words and trigrams of the given field. For the final task, we add a single fact to each of these latter cases that describes to which document generalization the citation was hypothesized to belong. For example, we would use `(isa title23 (ElementOfFn generalization4))` to mean that SEQL had put the 23rd document citation into the 4th generalization. This gives SEQL an incentive to match `title23` with the title of other documents put into the same generalization.

Not every database contained fields for all three of author, title, and venue. On these databases, we ran only those tasks for which the corresponding field was present.

We compare SEQL to PRMs on the Cora and Bibserv databases. The Cora database is a collection of citations to computer science research papers from the Cora Computer Science Research Paper Engine. We used the version of the database that had been processed and segmented by Bilenko and Mooney (2003) and then cleaned up by Singla and Domingos (2006), since we compare to their results. This version contains 1,295 citations to 132 unique documents, and includes author, title, and venue fields. The Bibserv database is a random subset of 10,000 records from the public repository of about half a million pre-segmented citations available at Bibserv.org. This subset contains 21,805 citations, and also includes all three subfields.

We compare SEQL to MLNs on the Citeseer and arXiv databases. The Citeseer database was originally created by Giles et al. (Giles, Bollacker, and Lawrence 1998) and contains citations to papers from four different areas of machine learning. It has 2,892 citations to 1,504 documents and 1,165 authors. The arXiv database was the largest of the four. It contains references to papers on high-energy physics used in KDD Cup 2003. This last database had 58,515 citations to 29,555 papers written by 9,200 authors.

For each of the seven tasks, we ran four different variations of the algorithm. We tried it both with and without probability information, and using each of two different normalization techniques. Variations that did not use probability were therefore equivalent to the original version of SEQL, in which any facts that did not appear in all of the member cases were dropped entirely from a generalization. For normalization techniques, we tried normalizing by the size of the base as well as by the mean of the target and base sizes together. For more information on this difference in normalization, please see Section 3.3

Finally, for every database, task, and variation, we ran the algorithm 20 different times under increasing similarity thresholds to discover which threshold was ideal. A higher threshold would mean greater precision and lower recall, whereas a lower threshold would mean greater recall but lower precision. For each threshold value, we computed the mean precision and mean recall, generating an overall precision-recall curve. We report on both the area under this curve (*AUC*), and also the best and mean *F1* score for the curve.

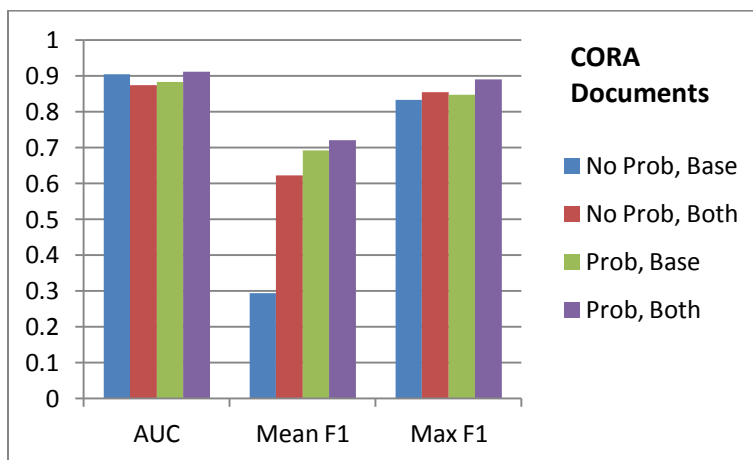
It must be noted that our experiment design is completely unsupervised. Unlike the MLNs and PRMs of Singla and Bhattacharya, which required training on 1/3 of the cases prior to testing, SEQL uses no training period in this experiment. Training on classes that did not appear in testing would provide no benefit to our algorithm. We would therefore expect SEQL to be less successful, but wanted to be able to quantize the difference.

Furthermore, it is important to remember that SME and SEQL are designed to take advantage of relational structure in the domain. Since this is a domain with little to no relational structure, something which SME had never been tested upon, we were curious to see how well it performed.

6.2.3 The citation experiment results

We found that of the four variations, the algorithm performed best when incorporating probability, as expected, and when normalizing to the mean size of both target and base. However, the results across all four variations were remarkably similar. Figure 6-1 shows the area under the precision-recall curve (AUC), and the mean and max F1 scores along that curve for each variation. This chart shows the results for the primary task on the CORA database. Results on the other three databases were similar and can be seen in Appendix B.

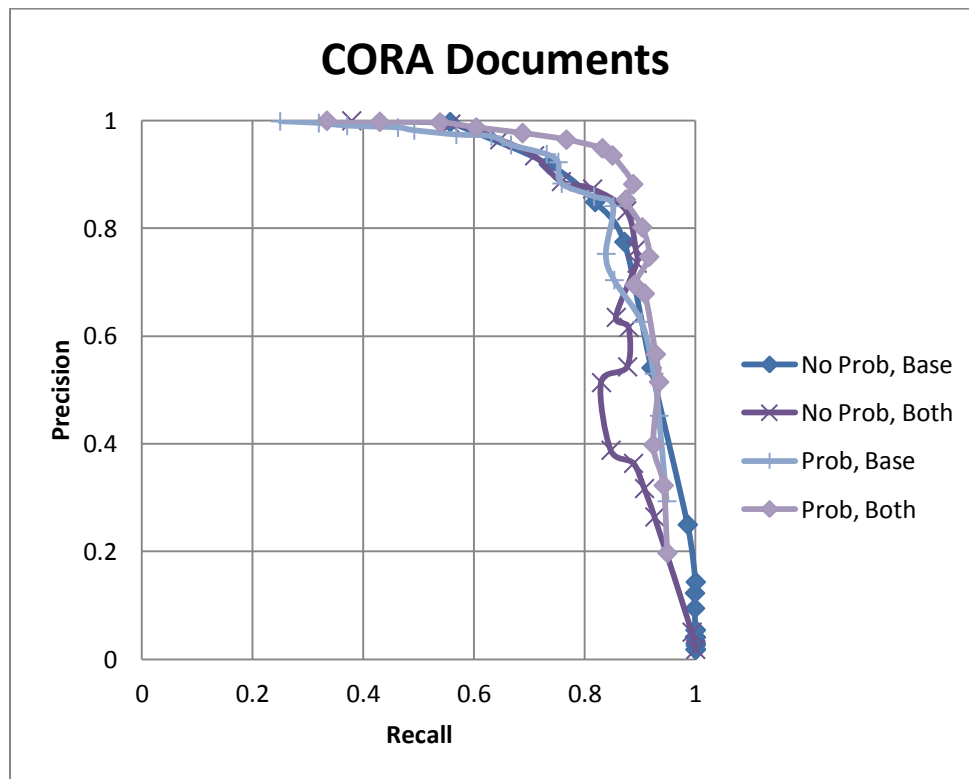
Figure 6-1. The relative performance of four SEQL variations on the Cora database.



It does in fact make sense that the difference between variations with probability and those without would be greater in domains with more structure than in a domain like this one. As the amount of structure increases, so too does the amount of information lost increase when a fact is dropped. In domains with high relational structure, dropping a fact that does not occur in all the cases means also dropping all of the structural information that was tied to that fact. This structural information in turn describes how the dropped fact related to other facts, so that we are losing information about those facts too. The more relational structure in the domain, the more information we lose about more facts when we do not use probabilities.

It is also interesting that the first variation had such a low Mean F1 score. A look at the precision-recall curves themselves in Figure 6-2 demonstrates why this is so.

Figure 6-2. Plotting the precision-recall curve for document identification on the Cora database.



The precision-recall curve for the first variation contains many points in the region of high recall, low precision -- enough to significantly lower the mean F1 score. This occurs when SQL produces one large generalization, which matches (i.e. recalls) all of the cases. When probability is not used, so that facts not occurring in all of the generalizations are dropped, then the generalizations are left with very few facts. They therefore become quite vague and able to match with anything.

In the second variation, the use of a more complex normalization metric ameliorates the problem of over-vague generalizations. But it seems to come at the cost of a precision-recall curve that is

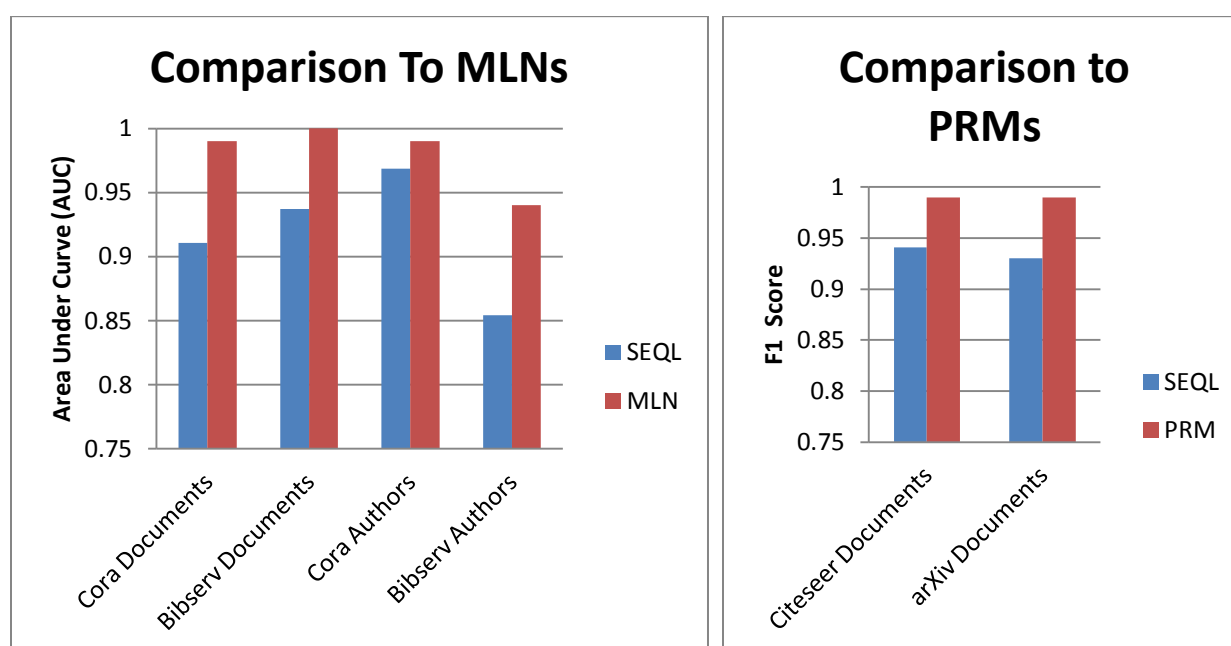
non-convex -- an irregularity which makes it difficult to predict optimal values and to work with in general. As described in Section 3.3, the normalization metric used here is intended to deal with generalizations that increase in size (number of facts) with the number of cases. When not using probability, we in fact have the opposite effect: generalizations shrink in size with the number of cases. This will sometimes make the match score higher than it should be, making the generalization merge with cases that it should not and sending it down the wrong direction. Proceeding counter-clockwise along the curve, as the match threshold increases, the threshold soon becomes high enough that these false positives disappear and things are restored to normal. And in the other direction of lower match thresholds, it is also not a problem since the generalization is expected to match to everything at those levels anyway. For this reason, when selecting a match threshold, it is probably advisable to err on the side of too high a threshold than too low.

From the points at the top-right of the chart, it is quite apparent where the final variation significantly surpasses the others in maximum F1 score. This occurred at a match threshold of 0.65. Although this was the best-performing variation on all four databases, the optimal threshold value varied between 0.65 and 0.85.

Figure 6-3 demonstrates how the SEQL algorithm compared to both MLNs and PRMs. We find that in this case, the lack of structure does seem to hurt SEQL. It does not outperform either of the state-of-the-art algorithms on any of the four datasets. We ran Monte Carlo simulations on each database to characterize the degree of the difference between our results and the state-of-the

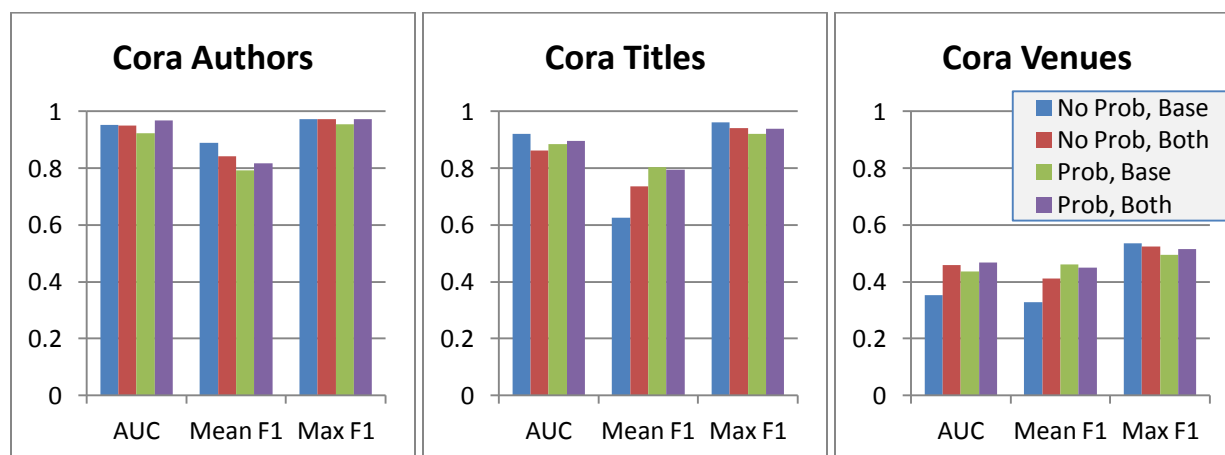
art. For each database, we did 100 simulations of the clustering algorithm at each of the 20 different similarity threshold levels. The similarity score was sampled randomly from its real distribution. This provides a distribution of results to expect under the null hypothesis. From this, we found that our results are indeed statistically significant, with a p-value of essentially 0.

Figure 6-3. A comparison to the state-of-the-art algorithms in citation matching.



Finally, we also tested the algorithm on its ability to match identical authors, titles, and venues. This was done both with and without the help of the results from the primary task of matching identical documents. The results of matching without any help are shown in Figure 6-4 below.

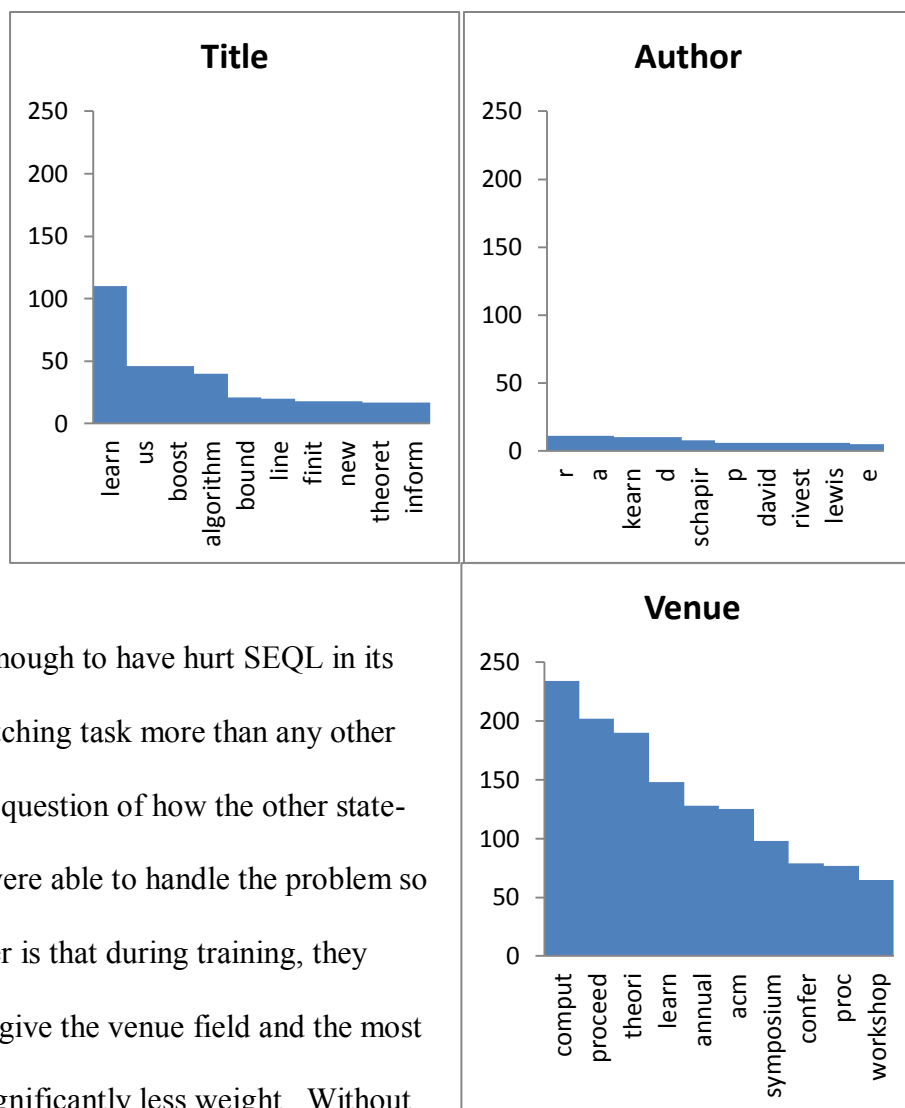
Figure 6-4. Matching the subfields in the citation-matching problem.



From the above, we can see that the author and title fields behaved much like the primary document-matching task did. A look at their precision-recall curves in the Appendix also shows the same behavior for these fields, even down to the irregular non-convexity in the curve for the second variation around a threshold of 0.4.

However, SEQL struggled greatly with the task of matching venues. A look at the distribution of the data in Figure 6-5 quickly demonstrates why. The venue field contained many more repeat occurrences of the same word than the other fields. Furthermore, the venue field had 67% as many unique words as the title field, and the average venue string was 1.5 words shorter than the average title string. All of this taken together means that the venue field had less variety and so was much harder to distinguish between than the other two fields.

Figure 6-5. Frequency distribution of the top ten words of each field.



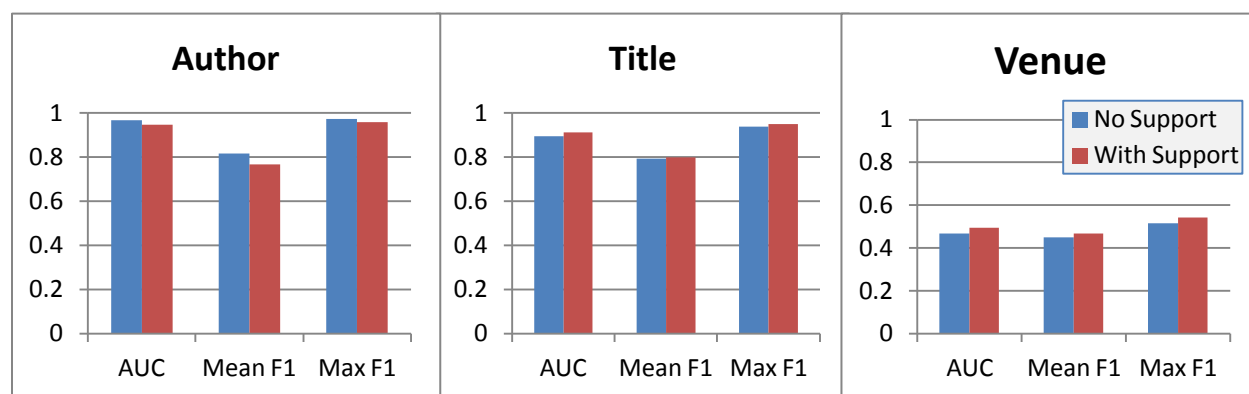
The problem is great enough to have hurt SEQL in its primary document-matching task more than any other factor. It does beg the question of how the other state-of-the-art algorithms were able to handle the problem so gracefully. One answer is that during training, they would have learned to give the venue field and the most common words in it significantly less weight. Without

such a training period, SEQL had no option but to weigh every field and every word the same.

Finally, we also tested whether taking into account SEQL's hypotheses about document matches would help it in matching the other fields. As shown below, the difference was negligible. In fact, sometimes the added information helped and sometimes it hurt, but always by an insignificant amount. This is a good indication that the naive assumption that the fields are

conditionally independent is valid here, and that performance is not expected to improve with successive iterations.

Figure 6-6. Whether performance can be improved with successive iterations.



6.2.4 Summary

Although SEQL did not perform as well as the state-of-the-art algorithms on this domain, it is important to remember that this is a domain without any of the relational structure of which SEQL was built to take advantage and with which other algorithms typically struggle.

Furthermore, SEQL did comparably, averaging only 6% less success while going completely unsupervised.

Not only is our approach still very statistically significant and within range of the performance of state-of-the-art algorithms, it is cognitively motivated and highly scalable. Furthermore, our algorithm handles any kind of relation, does not require any kind of relational schema to be laid

out beforehand, does not require supervision, and there is evidence and good reason to believe that it performs better rather than worse as the amount of relational structure increases. Perhaps most importantly, by relying on a simple polynomial-time distance metric, we can tackle much larger and more complicated domains such as Whodunit, which other algorithms fail to even complete on. We believe that this makes SEQL a good choice as a general statistical relational learning mechanism.

7 Literature Review

7.1 Analogy and Similarity

We use structural analogy as a core component of our systems: it provides a means of aligning concepts from the data into unique features, and also provides a distance metric. A number of alternative cognitive simulations of analogical mapping have been developed. Some are domain-specific (Mitchell, 1993). Others are based on connectionist architectures (Hummel & Holyoak, 1997; Eliasmith & Thagard, 2001), and are known to not be able to scale up to the size of examples used in these experiments. While CBR systems have been used to tackle similar structured representation problems (Leake, 1996), they also tend to be built as special-purpose systems for each domain and task. By contrast, the simulations used here are applicable to a broad variety of representations and tasks.

There are also a few other known similarity metrics. Most are graphical in nature and derived from ideas in collaborative filtering (Jeh and Widom 2002; Yin et al. 2006). Bohnebeck (1998) proposed a similarity metric for ILPs based on recursive comparisons of first-order terms. In contrast, our metric uses SME (Falkenheiner et al. 1986) to compute the analogical similarity, which takes into account not only the type of each object and relation but also the role that they play in the context of the case.

7.1.1 Similarity and Probability

We use similarity to derive the probability that a relation is true within the context of a generalization. Some others have looked at the problem of assigning probability to structural relations. They are essentially descendants of evidential reasoning (Pearl, 1987). They require a causal model to start with, and then infer probabilities based on observations and simulation from the model. For example, Koller and Pfeiffer (1997) use a maximum-likelihood approach of fitting probabilities to rules, given the observed data. They require “rule skeletons” to be given first. Only then can they do Knowledge-Based Model Construction (KBMC) to build a Bayes net of the causality of the domain, and then use that causality to find the set of probabilities which give the greatest likelihood to the observations. In contrast, our approach learns the prior probabilities of the relations themselves, and can then learn the appropriate models for those probabilities. That is, we try to both infer the probabilities and induce the model from the evidence alone. We also differ in that we set no preconditions about either the uniformity of input representations, or the order of the expressions in those representations.

There is almost no other literature on deriving probability from similarity assessments. The literature which does look at the problem either looks at it purely theoretically (Wellman, 1994), or uses similarity to affect rough estimates of probability only. For example, Blok et al (2003) present a heuristic for estimating the probability that a fox has trichromatic vision, based on the probabilities that a dog and a goldfish might have it, and their respective similarities to the fox. We would instead build a probabilistic model of our generalization of animals, and use it to predict the probability for foxes. Thus, our probability calculations never use a heuristic, and are carefully grounded in the reality of observed commonalities.

7.2 Relational Clustering

We use SEQL to construct generalizations based on our analogy-driven distance metric. SEQL can be used in either an unsupervised scenario, in which case it clusters the observed cases into generalizations, or a supervised one, in which case it constructs generalizations and/or statistical models of each class to make predictions. Although data clustering is an old and well-studied field, clustering of relational data is relatively new. Most of the literature on relational clustering involves clustering objects of only one or two types.

Homogeneous clustering involves the clustering of objects of the same type. One common way to do this is to use statistically generative models. A currently popular approach is the generative Bayesian mixture model, such as that described by Kemp and Tenenbaum (2006), to which we have made a direct comparison. The stochastic block model is a typical older choice

(Feinberg et al. 1985; Hoff et al. 2002). Besides generative models, another common approach is to do graph partitioning, usually either with a spectral algorithm (Chan et al. 1993) or a multilevel approach (Hendrickson and Leland 1995).

By contrast, co-clustering algorithms are able to cluster two types at once. This is often done to cluster both a set of objects and their properties simultaneously. Equivalently, it can also be used to cluster two types of objects which all share a single binary relation. Again, there are model-based approaches (Hofmann 1999), and bipartite graph-based approaches (Zha et al. 2001; Ganter and Wille 1998; Dayanik and Nevill-Manning, 2004). More recently, there are also approaches driven by information theory. For example, Dhillon et al. (2003) tries to maximize the mutual information subject to constraints on the number of clusters. Another recent technique (Long et al. 2005) uses an EM algorithm to do matrix factorization based on multiplicative updating rules.

A lot of literature has also been published on link analysis. For example, Kubica et al. (2002) look at the problem of clustering airplane passengers based on demographics and the flights that they take together. Similarly, Cohn and Hofmann (2001) actually create joint probability tables of terms and citations in document analysis through a probabilistic decomposition process related to LSA, and then use it to perform classification.

However, none of these algorithms can claim to handle any number of different relations. There are recent attempts to do more generic relational clustering though. For example, Long et al.

(2006) formulate the relational data as K-partite graphs and then present a set of algorithms for distorting it into a simplified network to identify the hidden structures. Others use mutual reinforcement clustering (Zeng, Chen and Ma 2002), although to our knowledge, there is no sound objective function for it nor a proof of convergence.

7.3 Model-Based Relational Learning

There is also a great deal of recent literature on taking model-based approaches to learn, supervised or otherwise, from more generic relational data; that is, over any number of objects and relations. Of these, perhaps the approach most similar to our own is that of Richardson and Domingos (2005). They introduce relational data into a Markov Logic Network by treating each possible grounding of each possible relation as a node in the network. However, the combinatorics of exploring every possible combination of groundings explodes very quickly. In fact, other researchers who have tried applying this MLN-based approach to the Whodunit problem have, so far, found it to be unable to scale to the size of the problem.

Another popular model-based approach is Probabilistic Relational Models, or PRMs (Getoor et al., 2001). A PRM is a Bayesian dependence model of the uncertainty across properties of objects of certain classes and the relations between those objects. It extends traditional Bayesian networks to incorporate a much richer relational structure. It can also handle information aggregated across several members of a class within the same case (for example, a student's highest grade or the lowest age among the victims of an attack). However, the PRM approach is

limited in two ways: first, it can only model first-order relations; and second, it has trouble when there is no prior knowledge of a *relational schema* or uniform representation of each case.

Although several papers have been published to try to overcome this second limitation (the modeling of *existence variables* (Getoor, et al., 2002) is particularly similar to our approach), none seems to present a uniform syntax for overcoming all forms of structural uncertainty, and none includes a method for modeling higher order relations. By contrast, our approach uses independently validated cognitive models of analogical matching to build such a unifying relational schema, from arbitrary predicate calculus descriptions of arbitrarily high order. A hybrid approach might be promising, using a PRM built upon the probabilistic generalizations we construct to provide the necessary schema.

Other approaches to doing relational learning via model induction all suffer from the same problems. For example, Blockeel and Uwents (2004) present a method for building a neural network from relational data. Long, et al (2007) use EM to optimize a generative mixed membership model called MMRC, and Kemp (2006) uses hill-climbing to optimize a Gaussian mixture model derived from the earlier stochastic block models. However, all of these models require prior knowledge of the relational schema, i.e. which relations operate over which types of entities. This is problematic if an argument of a relation can hold for multiple types, or if incremental learning is desired, whereby new types could be introduced at any time. Finally, our approach is also novel in that it can handle higher-order relations between relations, and flexible enough to allow other, simpler learning models to be applied (§5).

7.4 Induction-Based Learning

Finally, there are also approaches to learning based purely on induction. One example of this which we have provided a direct comparison to is FOIL (First Order Induction Learner) (Quinlan, 1990). FOIL learns rules by adding the grounded clauses one at a time which provide the greatest information gain. ILP's (Inductive Logic Programs) probably present the most generic of the existing solutions to relational induction. It can actually handle any number of different relations. (Raedt and Blockeel 2007; Kirsten and Wrobel 1998) extend the use of ILPs in various ways to do clustering. Further variants on ILPs such as Bayesian logic programs (BLPs) (Kersting, de Raedt, & Kramer, 2000) have also been suggested to do probabilistic induction. However, to our knowledge, there is not yet any approach which can fully satisfy the two conditions we have stated: learning relations of arbitrarily high order, and learning without any knowledge of prior relational schema (§5).

7.5 Other Works of Interest

Other works of interest include Keppens and Shen (2004), who demonstrate a way to build a Bayesian network from process knowledge such as that expressed by qualitative process theory (Forbus, 1984). And Tenenbaum & Griffiths (2001) provide a very well conceived and conveyed description of alternative models for generalization, all derived from Bayesian hypothesis formulation.

8 Summary of Questions and Findings

8.1 Support for Claims

This research supports five claims:

§ 1. We can achieve the benefits of both feature-based and relational representations of data by constructing a mapping for transforming logical expressions into features and back again

§ 2. It is more sensible in real-world scenarios to derive probability based on similarity rather than the other way around.

§ 3. With our techniques, we are able to apply any feature-based learning algorithm, despite the relational nature of the original data, without loss of information.

§ 4. Our classification algorithms are comparable in performance to existing algorithms, but they often are more efficient and require fewer examples.

§ 5. Our approach is the only one we know of that can handle arbitrary relations and that requires no relational schema.

Regarding the first claim, the creation of a mapping from relations to features and back again is key to this thesis. This mapping allows us to gain a little of the benefits of both feature-based and relational representations of the data. To demonstrate this, we first list some of these benefits. The inherent uniformity of feature-based representations opens us up to a wide variety of probabilistic learning algorithms. It also provides us with useful statistical metrics of

everything from the probability of an event and its significance to the relative amount of information gained from knowing any particular facet of a case. On the other hand, relational representations are more useful when dealing with the variety of structured information that people use and manipulate every day. Indeed, there is evidence that the use of structured language to accumulate relational knowledge is why humans are so smart compared to animals (Gentner, 2003). Relational representations also make it easier to perform certain human-like tasks such as inference, planning, and exploration of background knowledge.

Our system demonstrates many of these properties of both forms of representation. We showed the ability to handle a wide variety of structured inputs in the Whodunit domain, whose inputs incorporated many facts with a high order of structural complexity as well as complications like variables, logical operators, and logical quantifiers. It is also easy to imagine extending the reach of the architecture given here to perform inference or planning or to test against background knowledge on these cases. On the other hand, we also showed how feature-based probabilistic models such as association rules and Bayes nets could be generated, and how useful they could be on domains that are originally expressed in relational terms. It is clear (see §3) that the same principle can be extended to anything from decision trees to Gaussian mixture models or neural networks. Although many of these algorithms use feature-based metrics like information gain inherently, we showed in section 5.3 how they could be used to choose a more relevant vocabulary, enabling a re-representation of the cases and producing better results. The use of statistical significance to do anomaly detection or experiment design is easy to hypothesize as well.

The second claim concerns the very nature of how similarity and probability are related, and what contributions each can make to the determination of the other. Analogical generalization via similarity assessment guarantees that each generalized fact represents some analogically unique structure in the data. This in turn allows us to treat each generalized fact as a meaningfully unique random variable, creating a mapping between facts and features. In other words, it provides a meaningful, uniform framework across which we can compute probabilities. At the most abstract level, uniformity is the connection between similarity and probability.

We have demonstrated initial evidence that the two are intertwined, since similarity estimation allows us to compute meaningful probabilities (section 3.3.1), leading to better results overall in our experiments. Vice-versa, the probabilities of different relations within the cases affect similarity judgments in interesting ways (again, section 3.3.1). More importantly, we have shown rhetorically that similarity estimation (and by extension, analogy) is not just sufficient but necessary for the calculation of probability, and for learning itself. Similarity is required in order to find which items can reasonably be compared and their probability computed. It makes no sense to talk about the probability of X unless all the instances of X are of the same nature, or at least similar enough for the context at hand. This is true for both finding which cases can be compared (generalized from) in a domain, and for finding which *roles* within those cases are the same (structural alignment). If we weren't able to find these role correspondences, then different roles which shared the same predicate (such as the two drivers from Table 2-1) would be artificially conflated, and so the probability of any random variable related to it would no longer

be meaningful. This sets us apart from other approaches such as MLNs (Richardson and Domingos, 2005) which calculate the probabilities over predicates alone. Feature-based learning is only able to bypass this similarity requirement because the structural alignment is already assumed to be inherent in the representation itself. Only once this similarity estimation and structural alignment is complete can these probabilities then be taken into account in turn for making other similarity assessments.

The third claim stems from the fact that we are able to create a fully specified joint probability table. This is all that is needed to apply any feature-based learning algorithm there is, whether simple or complex. We have given examples of how useful rule-learners and Bayes nets can be in this relational environment, and we expect other learning algorithms to prove equally useful.

The fourth claim is supported by our experimental results from chapters 4 and 5. Both stages (probabilistic generalization and statistical learning) of the system are fully implemented, with accompanying statistical models. The first stage has undergone testing on a wide variety of domains by multiple persons, and has been shown to be successful in reputable publications, requiring orders of magnitude fewer examples than other algorithms. The second stage, completed more recently, has been tested only by myself. Nonetheless, it has also proven successful in several domains, three of which provided a comparison to competing algorithms.

To be specific about when our approach becomes more efficient than others, we refer to section 4.2.2. There, it is explained that we run SEQL, with a complexity of $O(N^2 M \log(M))$ where N is

the number of facts per case and M the number of cases, in order to reduce the number of possible hypotheses by $O(r^T V^{aT} (aT)^k)$ where r is the number of unique relations (indirectly related to N), T the maximum number of terms in a hypothesis, V the number of values that could be assigned to any one slot, and a the maximum number of slots, i.e. arity. Therefore, we will always use less hypothesis space, and for a learner that is linear with respect to the number of hypotheses, we will require less time when $N^2 M \log(M) \ll r^T V^{aT} (aT)^k$.

The fifth claim is supported directly by our survey of related work from chapter 7.

8.2 Other Conceptual Questions Raised

This research has also raised several interesting questions that we did not anticipate. Some of these questions were resolved in the process of the research, but there are a couple others which were not.

8.2.1 Generalization Questions

It is still unclear exactly which concepts a generalization should cover. It could include only the most similar cases without regard to type, or all of the cases in a collection, or the cases sharing some other significant property. This question has an impact not only on artificial intelligence, but also on cognitive science, and its effect trickles down to touch everything else in this thesis. For example, it affects whether learning will be done under supervision, which statistical model

to choose, and how the generalizations should be organized. Some of the tradeoffs involved in this decision were addressed in detail in section 3.1.1 on the goal of generalization.

Other questions are more technical than conceptual. In the generalization stage, one such technical question concerns how to implement a cutoff based on significance rather than probability. Since the number of facts in a generalization can grow quickly with the number of cases, it is important to have a means for culling out the less significant information. However, it is not possible to determine how significant some concept is to the rest of the generalization without first completing the generalization and the corresponding statistical model. Therefore we currently use probability as a poor substitute for significance as a threshold for culling facts. Although this helps reduce the chance of over-fitting, it certainly may be true that some fact which doesn't occur very frequently is very meaningful when it does occur. We lose the ability to use this in learning when we cull facts based on probability.

Finally, a key issue also required us to re-examine how analogy itself was handled by SME. The problem was that there are some "entities", such as numbers, other quantitative terms, colors, and some functions, which aren't really entities in the sense that they describe one individual thing in the world. In the example below, the pairs of statements won't line up properly in the match, because NegativeChange has additional structure in the second case that it doesn't have in the first. The problem is that a fact mentioning NegativeChange isn't describing something about NegativeChange, the entity. Instead, they are properties themselves, and every mention of

NegativeChange is a unique invocation. It shouldn't be inheriting structure from other invocations of it.

Table 8-1. Dealing with attribute-values

Case 1	Case 2
<code>(directionOfChange (DistanceFn mummy explorer) NegativeChange)</code>	<code>(directionOfChange (DistanceFn mummy explorer) NegativeChange)</code>
<code>(directionOfChange (HealthFn mummy) NoChange)</code>	<code>(directionOfChange (HealthFn mummy) NegativeChange)</code>

8.2.2 Feature Value Questions

The novel flattening process that we introduce raised many interesting questions. A number of these questions concern which values a feature should take. For instance, the vital benefit of a propositional representation is its flexibility. When cases are flattened under generalization, some of this flexibility is lost. It becomes difficult to represent certain scenarios under a feature-and-value scheme. Dealing with hierarchical values and multiple values are noteworthy examples.

An example of the hierarchical values problem is when one instance of an attack is expressed as occurring in Baghdad, while another instance is stated to have occurred in Iraq. Of course, the first instance also occurred in Iraq, but it is up to the system to determine that. Furthermore, when comparing nations, these two instances should match, but when comparing cities they should not. The solution to this problem has been to treat this as two features, one for the city

and one for the country. The system must use background knowledge to detect when this hierarchy of entities is occurring and split the features accordingly. It does this by pulling in attribute (collection membership) information on each entity and expressing the largest observed collection that it is a member of. For instance, in the Baghdad case, it would make sense to express that the attack also occurred in Iraq but not that it occurred on Earth, presuming that Iraq was seen elsewhere in the cases but Earth was not.

Table 8-2. Handling hierarchical values

Case 1	Case 2
(eventOccursAt-1 . Baghdad)	(eventOccursAt-1 . :missing)
(eventOccursAt-2 . Iraq)	(eventOccursAt-2 . Iraq)

The multiple values problem occurs when a single concept such as location literally has more than one legitimate and equal (one is not contained by another) value. An example of this problem is the September 11th attacks, which occurred in New York City, Pennsylvania, and Washington D.C. Although we have several ideas on how to address this problem, the correct way to do it is still unclear. One possible solution is to represent each value in a separate attack, perhaps even as separate cases. However, this answer loses the significant information that they were really all part of the same event. Less crucially, it also would lead to more overhead. Another possibility is to represent each as a separate sub-event, constructing some main event that they are a part of for other incidents to match to. This would mean that it retained the knowledge that the events were related. However, the main event will be missing lots of information that is specific to the sub-events, and so concepts such as “target” from other

incidents will not find any matches here. There might be a way to allow it to match to any of the targets in the sub-events, but this could also cause various other problems elsewhere and has not yet been explored. A third possibility is to find a way to allow a feature to have multiple values at once, but this would require a significant amount of work that would probably warrant a thesis of its own if it were even plausible to do.

Another question that is raised during flattening concerns closed world assumptions. That is, what should be done when the fact that corresponds to a particular feature is missing from the case, what value should be assigned? We handle this problem differently for characteristic features than for existential ones. When a characteristic feature value is missing, we simply assign it the value `:no-value`. In existential features though, the `:no-value` is represented by the value `False`. Yet there is a nuance. We label an existential feature as false only if some of the entities that it mentions are also missing from the case. When all of the entities that the feature mentions do appear elsewhere, then we label it as a `:missing-value` instead. Missing values do not count for or against anything during learning – they are treated during counting as if the case didn't exist at all. This reduces non-causal dependencies in the data, since otherwise the values of all features containing the same entity would be identical whenever that entity was not present.

Perhaps the first question that would need to be addressed for a truly generic learning agent though is how to determine which argument (if any) to treat as a feature value at all. For example, in the expression `(numCasualtiesOfType 4 Engineer attack)`, the best value to

extract is the number 4 from the second argument. We currently do this with a mixture of meta-knowledge and heuristic. We call a particular slot the value-slot of a feature if the predicate is known to be `FunctionalInArg`¹¹ for that slot, if the only other slot contains the case-entity (such as in `(eventOccursAt attack Baghdad)`), or if the value of the slot is always numeric or always non-atomic. This solution works on the domains we tried but is probably not a satisfactory solution for a long-running generic agent.

8.3 Future Work

We can now build probabilistic models from arbitrary relational data. However, a great deal can still be done.

Regarding the questions raised in the previous section, probably the most crucial to address first are how to implement a significance cutoff and how to best deal with multiple values. These are issues for which we have no real answers yet, though they appear in many of the experiments we do.

Outside of improving the process itself, there are two directions that would be nice to explore in the future. The first of these concerns putting this all together into a standalone generic learning

¹¹ Unfortunately, this is Cyc-specific meta-knowledge. It is intended to convey when an argument slot never has more than one value for any combination of the non-`FunctionalInArg` slots. Unfortunately, it is not used very often in the KB.

system. In other words, we would like to do incremental, arbitrary concept learning in real time. Among other things, this would require autonomously determining what and when to generalize, and when to revise a learned model. For example, learned conjectures which are consistent with the KB would provide validation. Those that are inconsistent would indicate either an error in the inputs or a poor model. And those which are neither would represent new ideas which might be beneficially explored in order to learn more about the domain.

Also, it would be highly interesting to further explore how the tools of probability can help in symbolic reasoning, and vice-versa. For example, there are many roles that statistical measures such as information gain and significance might play in symbolic inference. In the other direction, propositional background knowledge might help us to determine when assumptions about independence are being violated, so that learning efforts can be adjusted accordingly.

8.4 Conclusion

I have tested this approach to learning on 5 different domains: Terrorism research (The Whodunit problem), animal classification, tribal kinship terms, medical ontologies, and citation matching. It was compared to other algorithms, and used with success by researchers other than myself. We evaluated our success based on the results of clustering and/or prediction in these domains.

Similarity and analogy allow us to do this prediction despite the complex relational nature of the inputs by defining which items should be compared with which, through analogical equivalence. It is only this structural alignment of similar cases that allows us to describe the probabilities of their various aspects. We have therefore shown that the key to combining the power for learning provided by feature-based representation, with the flexibility that comes from structured representations, lies in recognizing that once the structures have been aligned using measures of similarity, only then can probabilities be computed and learning be done.

We hope that this research lays the groundwork for more efforts to combine the approaches used for each of these very different representations, and that ultimately, it may lead to the development of a truly generic machine learning agent.

REFERENCES

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., & Varkamo, A. I. (1996). Fast discovery of association rules. In Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., & Uthurusamy, R. *Advances in Knowledge Discovery and Data Mining*. AAAI Press.
2. Anderson, B., & Moore, A. (1998). ADTrees for Fast Counting and for Fast Learning of Association Rules. Knowledge Discovery from Databases, New York, 1998.
3. Bhattacharya, I. and Getoor, L. "Relational clustering for multi-type entity resolution." In KDD Workshop on Multi-Relational Data Mining (MRDM), 2005.
4. Bilenko, M. and Mooney, R. "Adaptive duplicate detection using learnable string similarity measures." In Proc. KDD-03, pages 39-48, 2003.
5. Bixler, D., Moldovan, D., and Fowler, A. (2005). Using Knowledge Extraction and Maintenance Techniques to Enhance Analytical Performance. *Proceedings of the 2005 International Conference on Intelligence Analysis*, Washington, DC, 2005
6. Blockeel, H., & Uwents, W. (2004). Using neural networks for relational learning. ICML-2004 Workshop on Statistical Relational Learning and its Connection to Other Fields, pp.23-28.
7. Blok, S., Medin, D., and Osherson, D. Probability from similarity. *AAAI conference on commonsense reasoning* (Stanford University, 2003).
8. Bohnebeck, U., T. Horvath, and S. Wrobel. "Term comparisons in first-order similarity measures." *Proc. 8th Int. Workshop on Inductive Logic Programming (ILP '98)*. Madison, WI, 1998.
9. Chan, P. K., M. D. F. Schlag, and Y. J. Zien. "Spectral k-way ratio cut partitioning and clustering." DAC '93. 1993.
10. Cohn, D., & Hofmann, T. (2001). The missing link – a probabilistic model of document content and hypertext connectivity. *Advances in Neural Information Processing Systems* 13:430-436, MIT Press.
11. Dayanik, A. and Nevill-Manning, C.G. (2004). Clustering in Relational Biological Data. ICML-2004 Workshop on Statistical Relational Learning and Connections to Other Fields, pp. 42-47
12. De Raedt, L. (1998). Attribute-Value Learning versus Inductive Logic Programming: the Missing Links (Extended Abstract). *Proceedings of the 8th International Conference on Inductive Logic Programming, Lecture Notes in Artificial Intelligence*, Springer-Verlag 1446.

13. Deghani and Lovett, 2006. Efficient genre classifications using qualitative representations. *Proceedings of the 7th Int. Conference on Music Retrieval*.
14. Dejong, G. and Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning, 1(2)*. 145-176.
15. Denham, W. *The detection of patterns in Alyawarra nonverbal behavior*. PhD Thesis, University of Washington, 1973.
16. Dhillon, I. S., S. Mallela, and D. S. Modha. "Information theoretic co-clustering." *KDD '03*. 2003. 89-98.
17. Eliasmith, C. and Thagard, P. 2001. Integrating structure and meaning: A distributed model of connectionist mapping. *Cognitive Science*.
18. Elio, R. and Anderson, J.R. (1984). The effects of information order and learning mode on schema abstraction. *Memory and Cognition 12(1)*, 20-30.
19. Falkenhainer, B., Forbus, K. and Gentner, D. 1989. The Structure-Mapping Engine: Algorithms and Examples. *Artificial Intelligence, 41*: 1-63.
20. Falkenhainer, B., Forbus, K. and Gentner, D. The Structure-Mapping Engine. *Proceedings of the Fifth National Conference on Artificial Intelligence*. 1986.
21. Feinberg, S. E., M. M. Meyer, and S. Wasserman. "Statistical analysis of multiple sociometric relations." *Journal of American Statistical Association 80*, 1985: 51-87.
22. Forbus, K., and Oblinger, D. 1990. Making SME Greedy and Pragmatic. In *Proceedings of the 12th Annual Meeting of the Cognitive Science Society*.
23. Forbus, K. Exploring analogy in the large. In Gentner, D., Holyoak, K., and Kokinov, B. *Analogy: Perspectives from Cognitive Science*. MIT Press. 2001.
24. Forbus, K., Gentner, D., and Law, K. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science, 19*, 141-205. 1994.
25. Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence, 24*, 85-168
26. Friedman, N. and Yakhini, Z. (1996) On the sample complexity of learning Bayesian networks. *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence (UAI 96)*, 274-282, Morgan Kaufmann.
27. Ganter, Bernhard, and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Berlin: Springer-Verlag, 1998.

28. Gentner, D. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7: 155-170 (2).
29. Gentner, D. "Why we're so smart." In *Language in mind: Advances in the study of language and thought*, by D. Gentner and S. Goldin-Meadow, 195-235. Cambridge, MA: MIT Press, 2003.
30. Gentner, D. and Forbus, K. (1991). MAC/FAC: A model of similarity-based retrieval. *Proceedings of the Cognitive Science Society*.
31. Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. Learning probabilistic relational models. In Dzeroski, S. and Lavrac, N. (Eds.), *Relational Data Mining* (pp. 307-335). Kluwer, 2001.
32. Getoor, L., Friedman, N., Koller, D., & Taskar, B. Learning probabilistic models of link structure. *JMLR* 3, 679-707. 2002.
33. Giles, C. L., Bollacker, K., and Lawrence, S. (1998). Citeseer: An automatic citation indexing system. *ACM Conference on Digital Laboratories*.
34. Golbreich, C., Dameron, O., Gibaud, B., & Burgun, A. (2003). Web ontology language requirements wrt expressiveness of taxonomy and axioms in medicine. 2nd International Semantic Web Conference, ISWC, 2003.
35. Halstead, D., and Forbus, K. (2005). Transforming between Propositions and Features: Bridging the Gap. *Proceedings of AAAI-2005*. Pittsburgh, PA.
36. Halstead, D. and Forbus, K. (2007). Some Effects of a Reduced Relational Vocabulary on the Whodunit Problem. In *Proceedings of the International Joint Conference on Artificial Intelligence*.
37. Hendrickson, B., and R. Leland. "A multilevel algorithm for partitioning graphs." *Supercomputing '95*. 1995. 28.
38. Hoff, P., A. Raftery, and M. Handcock. "Latent space approaches to social network analysis." *Journal of American Statistical Association* 97, 2002: 1090-1098.
39. Hofmann, T. "Probabilistic latent semantic analysis." *Proc. of Uncertainty in Artificial Intelligence (UAI) '99*. Stockholm, 1999.
40. Hubert, L., and P. Arabie. "Comparing partitions." *Journal of Classification* 2, 1985: 193-218.
41. Hummel, J.E., & Holyoak, K.J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104.

42. Jeh, G., and J. Widom. "Simrank: A measure of structural-context similarity." *KDD '02*. 2002.
43. Kahneman, D., & Tversky, A. (1979). Prospect theory: An analysis of decision under risk. *Econometrica* 47, 263–292.
44. Kemp, C., J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. "Learning systems of concepts with an infinite relational model." *AAAI '06*. 2006.
45. Keppens, J., & Shen, Q. (2004). Causality Enabled Modeling of Bayesian Networks. *Proceedings of the 18th International Workshop on Qualitative Reasoning*, 33-40.
46. Kersting, K., de Raedt, L., & Kramer, S. (2000). Interpreting Bayesian logic programs. *Proc. AAI-2000 Workshop on Learning Statistical Models from Relational Data*, pp. 29-35.
47. Kirsten, M., and Wrobel, S. "Relational distance-based clustering." *Proc. Fachgruppentreffen Maschinelles Lernen (FGML-98)*. 1998. 119-124.
48. Koller, D., and Pfeffer, A. (1997). Learning probabilities for noisy first-order rules. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 1316-1321).
49. Kubika, J., Moore, A., Schneider, J., and Yang, Y. (2002). Stochastic link and group detection. *AAAI*, 798-804. ACM Press, July 2002.
50. Kuehne, S. E., Gentner, D. & Forbus, K. D. (2000). Modeling infant learning via symbolic structural alignment. *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, 286-291.
51. Kuehne, S., Forbus, K., Gentner, D., and Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. *Proceedings of CogSci 2000*.
52. Lavrac, N., Dzeroski, S., and Grobelnik, M. (1991). Learning Nonrecursive Definitions of Relations with LINUS. *Proceedings of the European Working Session on Machine Learning*. Springer-Verlag, London, UK.
53. Law, K., Forbus, K., and Gentner, D. (August, 1994). Simulating similarity-based retrieval: A comparison of ARCS and MAC/FAC. *Proceedings of the Cognitive Science Society*.
54. Leake, D. (Ed.) 1996. *Case-based Reasoning: Experiences, Lessons and Future Directions*, MIT Press.

55. Lockwood, K., Forbus, K., Halstead, D. & Usher, J. (2006). Automatic Categorization of Spatial Prepositions. *Proceedings of the 28th Annual Conference of the Cognitive Science Society*. Vancouver, Canada.
56. Long, B., X. Wu, and Z. M. Zhang. "Unsupervised learning on k-partite graphs." *KDD '06*. 2006.
57. Long, B., Z. Zhang, and P. S. Yu. "A probabilistic framework for relational clustering." *KDD '07*. 2007. 470-479.
58. Long, B., Z. Zhang, and P. Yu. "Co-clustering by block value decomposition." *KDD '05*. 2005.
59. Lovett, A., Dehghani, M., and Forbus, K. 2006. Efficient Learning of Qualitative Descriptions for Sketch Recognition. In Proceedings of the 20th International Qualitative Reasoning Workshop.
60. McCray, A. T. "An upper level ontology for the biomedical domain." *Comparative and Functional Geonomics 4*, 2003: 80-84.
61. McCray, A. T., A. Burgun, and O. Bodenreider. "Aggregating UMLS semantic types for reducing conceptual complexity." *Medinfo 10* (2001): 216-20.
62. Medin, D.L. and Bettger, J.G. (1994). Presentation order and recognition of categorically related examples. *Psychonomic Bulletin and Review*, 1(2), 250-254.
63. Mitchell, M. (1993) *Analogy-making as perception: A computer model*. MIT Press.
64. Moore, A. and Lee, M. (1997). Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *Journal of Artificial Intelligence Research 8*, 67-91.
65. Mostek, T., Forbus, K. and Meverden, C. 2000. Dynamic case creation and expansion for analogical reasoning. Proceedings of AAAI-2000. Austin, Texas.
66. Osherson, D. N., J. Stern, O. Wilkie, M. Stob, and E. E. Smith. "Default Probability." *Cognitive Science*, no. 15 (1991): 251-169.
67. Paritosh, P.K. (2004). Symbolizing Quantity. *Proceedings of the 26th Cognitive Science Conference*, Chicago.
68. Pazzani M. and Brunk C. (1991). Detecting and correcting errors in rule-based expert systems: an integration of empirical and explanation-based learning. *Knowledge Acquisition*, 3, 157-173.
69. Pearl, J. Evidential Reasoning Using Stochastic Simulation of Causal Models. *Artificial Intelligence*, Vol. 32:2, 245-258, 1987.

70. Pitman, J. 2002. Combinatorial stochastic processes. *Lecture notes for St. Flour Summer School*. Springer-Verlag, New York, NY.
71. Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3) pp 130–137.
72. Quinlan, J.R.. Learning Logical Definitions from Relations. *Machine Learning, Volume 5, Number 3*, 1990.
73. Raedt, L. D., and H. Blockeel. "Using logical decision trees for clustering." *Proceedings of the 7th International Workshop on Inductive Logic Programming*. 2007.
74. Ramscar, M. and Pain, H. (1996). Can a real distinction be made between cognitive theories of analogy and categorization? *Proceedings of the 18th Annual Conference of the Cognitive Science Society*, 346-351, Erlbaum.
75. Richardson, M. and Domingos, P. "Markov Logic Networks." *Machine Learning*, 62:107-136, 2006.
76. Singla, P. and Domingos, P. "Entity Resolution with Markov Logic." *Proceedings of the Sixth IEEE International Conference on Data Mining* (pp. 572-582), Hong Kong, 2006.
77. Skorstad, J., Gentner, D., & Medin, D. (1988). Abstraction processes during concept learning: A structural view. *Proceedings of the Tenth Annual Conference of the Cognitive Science Society*, 419-425.
78. Taskar, B., E. Segal, and D. Koller. "Probabilistic classification and clustering in relational data." *Proceedings of IJCAI-01*. 2001.
79. Tenenbaum, J., & Griffiths, T. (2001). Generalization, similarity, and Bayesian inference. *Behavioral and Brain Sciences* 24 (629-640).
80. Tengli, A., Dubrawski, A., and Chen, L. (2005). Learning Predictive Models from Small Sets of Dirty Data. In *International Conference on Information and Automation*.
81. Tomai, E., Lovett, A., Forbus, K., & Usher, J. (2005). A Structure Mapping Model for Solving Geometric Analogy Problems. *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, Stressa, Italy, 2190-2195.
82. Wattenmaker, W. D. (1993). Incidental concept learning, feature frequency and correlated properties. *Journal of Experimental Psychology: Human Learning & Memory* 19, 203-222.
83. Wellman, M. P. Some varieties of qualitative probability. *Fifth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 437-442, July 1994.

84. Yin, X., J. Han, and P. Yu. "Efficient clustering via heterogeneous semantic links." *VLDB '06*. 2006.
85. Zeng, H. J., Z. Chen, and W. Y. Ma. "A unified framework for clustering heterogeneous web objects." *WISE '02*. 2002. 161-172.
86. Zha, M. X. H., C. Ding, and H. Simon. "Bi-partite graph partitioning and data clustering." *ACM CIKM '01*. 2001.

APPENDICES

Appendix A. Translation Rules for Reduced Relational Vocabularies

The tables below summarize the rules used to translate cases from their original representation, using a very large vocabulary provided by the ResearchCyc knowledge base, to a reduced vocabulary so that learning might be made simpler. These experiments and results are discussed in section 5.3.

The first table presents the rules for the Polaris vocabulary. Each rule is fired when both of two conditions are met. These conditions are shown in the first two fields of each row, after the rule number.

The first field is a list of possible predicates, any one of which must match the predicate of the candidate fact. In order to match, the candidate fact's predicate must either be identical to one of these predicates or a more specific version of one of them. For example, (`performedBy` `event1` `Bob`) would be translated by Rule 1, since `performedBy` is a more specialized version of the predicate `doneBy`, according to the predicate hierarchy of the Cyc knowledge base that we used.

The second field contains a list of bindings for the candidate fact. This field performs two functions. First, it provides a second necessary condition for the rule to be fired, since candidate

facts must match it exactly (more specific forms will not do). Second, it binds those items that may vary in a candidate fact to a variable (marked by a preceding ‘?’) for the translation step.

When both conditions are met, the rule is fired, invoking a translation. A candidate fact may be translated into multiple new facts, listed in the final field of each row. For example, Rule 2 would translate (eventPlannedBy ChessMatch BobbyFischer) into (agent BobbyFischer ChessMatch) and (goal ChessMatch BobbyFischer), based on the translated forms from the third field and the bindings from the second.

Table A-1. Translation rules for Polaris vocabulary

1	doneBy (agent ?actor ?action)	(?pred ?action ?actor)
2	deliberateActors eventPlannedBy plannerOfEvent (agent ?actor ?action), (goal ?action ?actor)	(?pred ?action ?actor)
3	<i>Nil</i> (possible (agent ?actor ?action))	(likelySuspects ?action ?actor)
4	objectActedOn (cause ?event ??state), (result ??state ?action), (theme ?obj ?action), (predicate ??state ?obj)	(?action ?event ?obj)
5	<i>Nil</i> (cause ?attack ??objectContaminated), (result ??objectContaminated contamination), (theme ?obj contamination) (predicate ??objectContaminated ?obj)	(objectContaminatedInAttack ?attack ?obj)
6	intendedAttackTargets (purpose ??state ?event), (possible (result ??state attack)), (possible (theme ?obj attack)), (possible (predicate ??state ?obj))	(?pred ?event ?obj)
7	intendedMaleficiary (purpose ??state ?event), (possible (result ??state thingHarmed)), (possible (theme ?obj thingHarmed)), (possible (predicate ??state ?obj))	(?pred ?event ?obj)
8	possibleIntendedAttackTargets (possible (purpose ??state ?event)), (possible (result ??state attack)) (possible (theme ?obj attack)), (possible (predicate ??state ?obj))	(?pred ?event ?obj)

9	situationLocation eventPartiallyOccursAt	(?location ?event ?place)
	(location ?place ?event)	
10	startingDate endingDate duration	(?when ?event ?time)
	(time ?time ?event)	
11	thereExistExactly thereExistRange	(?pred ?num ?var ?fact)
	(measure ?num ?var), ?fact	
12	Nil	(deathToll ?event ?type ?num)
	(measure ?num ??var), (?type ??var), (result ??death organismKilled), (cause ?event ??death), (theme ??var organismKilled), (predicate ??death ??var))	
13	Nil	(injuryCount ?event ?type ?num)
	(measure ?num ??var), (?type ??var), (result ??wound animalInjuredIn), (cause ?event ??wound), (theme ??var animalInjuredIn), (predicate ??wound ??var))	
14	Nil	(numberOfHostagesTaken ?event ?type ?num)
	(measure ?num ??var), (?type ??var), (result ??taken agentCaptured), (cause ?event ??taken), (theme ??var agentCaptured), (predicate ??taken ??var))	
15	Nil	(casualtyCount ?event ?type ?num)
	(measure ?num ??var), (?type ??var), (predicate ??state ??var), (cause ?event ??state), (or (and (result ??state agentCaptured) (theme ??var agentCaptured)) (and (result ??state animalInjuredIn) (theme ??var animalInjuredIn)) (and (result ??state organismKilled) (theme ??var organismKilled))))	
16	topicOfInfoTransfer infoTransferred	(?pred ?x ?y)
	(topic ?y ?x)	
17	claims	(?pred ?agent ?claim)
	(source ?agent ??info), (topic ?claim ??info)	
18	accusedOf	(?pred ?info ?agent)
	(topic (agent ?agent ??event) ?info)	
19	actAttributed	(?pred ?info ?event)
	(topic (agent ??agent ?event) ?info)	
20	Nil	(mostNotableIsa ?x ?y)
	(isa ?x ?y)	
21	causes-SitSit causes-SitProp	(?pred ?x ?y)
	eventResults eventOutcomes	(cause ?x ?y)
22	inReactionTo	(?pred ?x ?y)
	(cause ?x ?y)	

23	parts (part ?y ?x)	(?pred ?x ?y)
24	instrumentalRole (instrument ?y ?x)	(?pred ?x ?y)
25	positiveInterest-Prop positiveVestedInterest (experiencer ?agent ??emotion), (PositiveFeeling ??emotion) (topic ?thing ??emotion)	(?pred ?agent ?thing)
26	goals (goal ?y ?x)	(?pred ?x ?y)
27	thinksProbable (belief ?y ?x)	(?pred ?x ?y)
28	linked conceptuallyRelated (associated ?y ?x)	(?pred ?x ?y)
29	reasonsForAction (reason (?thinks ?agent ?why) ?act), (agent ?agent ?act)	(?pred ?act ?thinks ?agent ?why)
30	undamagedActors (not (and (theme ?actor damages) (predicate ??damaged ?actor) (result ??damaged damages) (cause ?event ??damaged)))	(?pred ?event ?actor)
31	unharmedActors (not (and (theme ?actor thingHarmed) (predicate ??harmed ?actor) (result ??harmed thingHarmed) (cause ?event ??harmed)))	(?pred ?event ?actor)
32	actionExpressesFeelingToward (agent ??agent ?act) (experiencer ??agent ?feeling) (recipient ?toward ?feeling)	(?pred ?act ?feeling ?toward)
33	actionExpressesFeeling (agent ??agent ?act) (experiencer ??agent ?feeling)	(?pred ?act ?feeling)
34	sponsorsAgentInAction (goal (agent ?agent ?act) ?sponsor)	(?pred ?sponsor ?agent ?act)
35	obligationsViolated (result ??violation obligationsViolated), (predicate ??violation ?obligation), (theme ?obligation obligationsViolated), (cause ?event ??violation)	(?pred ?event ?obligation)
36	extraditionFor (isa ?event CriminalAct), (cause ?event ?extradition), (topic (agent ??agent ?event) ?extradition)	(?pred ?extradition ?event)

The second reduced vocabulary consisted of simply taking advantage of the predicate hierarchy that was already built in to the Cyc knowledge base. For this reason, the vast majority of complicated rules that were needed for the first vocabulary were not needed for the second. Instead, each predicate was simply made as generic as possible. Although it is possible to write out every translation rule needed to do this for every generalized predicate, they would all be virtually identical. Hence, it is much simpler to simply list the predicates. These are given below in Table A-2, followed by a very small number of supplemental rules that are still needed in Table A-3.

Table A-2. Translated predicates for Underspecified-Cyc vocabulary

<p>awareOf, affects-Underspecified, conceptuallyRelated, during-Underspecified, instantiationOf-Underspecified, underspecifiedLocation, without-Underspecified, ahead-underspecifiedRelation, around-UnderspecifiedRegion, before-Underspecified, outOf-UnderspecifiedContainer, underspecifiedTypeExpressionOf, by-Underspecified, inside-UnderspecifiedRegion, at-UnderspecifiedLandmark, from-UnderspecifiedLocation, under-UnderspecifiedLocation, outside-UnderspecifiedRegion, is-Underspecified, about-UnderspecifiedRegion, causes-Underspecified, in-UnderspecifiedContainer, along-UnderspecifiedPath, orientation-Underspecified, with-UnderspecifiedAgent, up-UnderspecifiedPath, contains-Underspecified, holds-Underspecified, down-UnderspecifiedPath, for-UnderspecifiedLocation, agentSupportsAgent-Generic, sizeOfObject-Underspecified, through-UnderspecifiedPortal, possessiveRelation, on-UnderspecifiedSurface, dependsOn-Underspecified, releases-Underspecified, expresses-Underspecified, over-UnderspecifiedLocation, across-UnderspecifiedRegion, into-UnderspecifiedContainer, determination-UnderspecifiedRelation, supports-Underspecified, underspecifiedExpressionOf, connects-Underspecified, relativeOrientation-Underspecified, speedOfObject-Underspecified, generalizations, disconnects-Underspecified, after-Underspecified, to-UnderspecifiedLocation, off-UnderspecifiedSurface, unknownReIn-BBNReInType</p>
--

Table A-3. Supplemental translation rules for Underspecified-Cyc vocabulary

1	injuryCount	(?pred ?atk ?coln ?count)
	(measure ?count ??var), (isa ??var ?coln), (animalWoundedIn ?atk ??var), (affects-Underspecified ?atk ??var), (during-Underspecified ?atk ??var), (after-Underspecified ?atk ??var)	
2	deathToll casualtyCount	(?pred ?atk ?coln ?count)
	(measure ?count ??var), (isa ??var ?coln), (organismKilled ?atk ??var), (affects-Underspecified ?atk ??var), (during-Underspecified ?atk ??var), (after-Underspecified ?atk ??var)	
3	numberOfHostagesTaken	(?pred ?atk ?count)
	(measure ?count ??var), (isa ??var Person), (agentCaptured ?atk ??var), (affects-Underspecified ?atk ??var), (during-Underspecified ?atk ??var), (after-Underspecified ?atk ??var)	
4	thereExistExactly thereExistRange	(?pred ?num ?var ?fact)
	thereExistAtLeast	(measure ?num ?var), ?fact

Appendix B. Results of Citation-Matching on other Databases

