Some Effects of a Reduced Relational Vocabulary on the Whodunit Problem

Daniel T. Halstead and Kenneth D. Forbus
Qualitative Reasoning Group
Northwestern University
2145 Sheridan Rd., Evanston, IL 60208 USA

Abstract

A key issue in artificial intelligence lies in finding the amount of input detail needed to do successful learning. Too much detail causes overhead and makes learning prone to over-fitting. Too little detail and it may not be possible to learn anything at all. The issue is particularly relevant when the inputs are relational case descriptions, and a very expressive vocabulary may also lead to inconsistent representations. For example, in the Whodunit Problem, the task is to form hypotheses about the identity of the perpetrator of an event described using relational propositions. The training data consists of arbitrary relational descriptions of many other similar cases. In this paper, we examine the possibility of translating the case descriptions into an alternative vocabulary which has a reduced number of predicates and therefore produces more consistent case descriptions. We compare how the reduced vocabulary affects three different learning algorithms: exemplar-based analogy, prototype-based analogy, and association rule learning. We find that it has a positive effect on some algorithms and a negative effect on others, which gives us insight into all three algorithms and indicates when reduced vocabularies might be appropriate.

1 Introduction

One problem that is consistent across nearly every application of machine learning is identifying the appropriate amount of detail in the input data. 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.

This paper, which was motivated by just such an attempt to improve knowledge extraction from text, is an analysis of the contrasting effects between using a large, very detailed but often redundant vocabulary, and a small, consistent, but extremely simplified one. The two different vocabularies are applied to the *Whodunit Problem*, which tries to learn how to predict the perpetrator of a terrorist event, given many descriptions of similar events expressed by propositional assertions. The descriptions are currently entered by hand but will eventually be extracted from text, which makes the choice of vocabulary especially relevant. Each vocabulary is used by three different learning algorithms, in order to better understand the effects of the vocabulary size and to find the best overall combination.

Section 2 begins by introducing the Whodunit Problem. Following that, each of the three learning algorithms for solving the problem are explained in Section 3. Section 4 describes how the original, large vocabulary was reduced, and Section 5 presents our results. Finally, we conclude with a discussion of related work.

2 The Whodunit Problem

An important task for analysts is coming up with plausible hypotheses about who performed an event. 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.

Stated more formally, given some event E whose perpetrator is unknown, the Whodunit problem is to construct a small set of hypotheses {Hp} 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.

We define a more restricted class of Whodunit problems to begin 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 {D}. For some algorithms, this involves forming non-overlapping generalizations {G} 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.

Table 1. Example of a Whodunit case description

```
(ISA ATTACK-1 TERRORISTATTACK )

(EVENTOCCUREDAT ATTACK-1 FALLUJAH)

(CITYINCOUNTRY FALLUJAH IRAQ)

(THEREEXISTATLEAST 2 ?X
  (AND (CITIZENOF ?X IRAQ) (WASINJURED ATTACK-1 ?X)))
```

The corpus we use in our experiments is an independently-developed knowledge base of terrorist incidents, provided to us by *Cycorp*. The version they provided consists of 3,379 descriptions of different terrorist attacks, each hand-entered and checked by domain experts. These attacks were all expressed using the symbolic representation vocabulary of the *Cyc* KB, which (in our subset) consists of over 36,000 concepts, over 8,000 relationships and over 5,000 functions, all constrained by 1.2 million facts. The descriptions of terrorist attacks ranged in size from 6 to 158 propositions, with the average being 20 propositions.

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 (67) to choose from during testing.

3 Learning Algorithms

We used three different algorithms to try to solve the Whodunit problem. Since the contribution of this paper is on the effects of vocabulary reduction and not the learners themselves, we have only selected algorithms which have been previously published. Here then, we present only a terse description of each learner and the implementation details particular to the domain.

All three algorithms utilized structural analogy in some fashion, in order to make comparisons between and across cases.

Our approach to analogy is based on Gentner's [1983] structure-mapping theory of analogy and similarity. In structure-mapping, analogy and similarity are defined in terms of structural alignment processes operating over structured representations. The output of this comparison process is one or more mappings, constituting a construal of how the two entities, situations, or concepts (called *base* and *target*) can be aligned. For the purposes of this paper, a mapping consists of a set of *correspondences* and a *structural evaluation score*. A correspondence maps an item (e.g. an entity or expression) from the base to an item in the target. The structural evaluation score indicates the overall quality of the match.

We used the Structure-Mapping Engine (SME) to implement this theory of analogical mapping [Falkenhainer, Forbus, & Gentner, 1986]. SME uses a greedy algorithm to compute approximately optimal mappings in polynomial time [Forbus & Oblinger, 1990].

Formally, the task of each learning algorithm is, given descriptions to train on {D}, and input event E, to produce *n* hypotheses about the identity of the perpetrator of event E.

3.1 Exemplar-based Analogy: MAC/FAC

The first algorithm operates purely on exemplar retrieval. That is, it is designed to find a small number of input cases which are most similar to the probe case E. For each such case, it hypothesizes that the perpetrator of E is the same as the perpetrator of the similar case. The process can be iterated until n unique hypotheses are generated. MAC/FAC [Forbus, Gentner, & Law, 1994] is an algorithm which performs this similarity-based retrieval in two stages.

The first stage uses a special kind of feature vector, called a content vector, which is automatically computed from each structural description. A content vector consists simply of the counts of each predicate in the corresponding description. Their dot product then is an estimate of how many correspondences SME will generate when considering possible mappings between two descriptions, and therefore an estimate of the quality of the match. Content vector computations are used to rapidly select a few (typically three) candidates from a large memory.

In the second stage, SME is used to do an analogical comparison between the subset of {D} which was returned by the first stage and the probe description E. It returns the one (or more, if very close) most similar of the cases in memory as what the probe reminded it of.

As deployed performance systems, both SME and MAC/FAC have been used successfully in a variety of different domains, and as cognitive models, both have been used to account for a variety of psychological results [Forbus, 2001].

3.2 Prototype-based Analogy: SEQL

The second algorithm is designed to also use analogy. However, it first builds generalizations of the cases to serve as prototypes. Each generalization is constructed from the cases of only one perpetrator at a time. This way, each generalization serves as a prototypical description of the events associated with that single perpetrator. Then instead of comparing E to every case in {D} (as the first algorithm does), this algorithm compares E to every prototype.

The generalizations are built by using analogy to determine which concepts in one case best correspond to the concepts in another case. We use a probabilistic implementation of the SEQL algorithm to do this. SEQL [Kuehne, Forbus, et al., 2000], which stands for Sequential Learner, is designed to produce generalizations incrementally from a stream of examples. It uses SME to compare each new example to a pool of prior generalizations and exemplars. If the new example is sufficiently similar to an existing generalization, it is assimilated into that generalization. Otherwise, if it is sufficiently similar to one of the prior exemplars, it is combined with it to form a new generalization.

A generalization of two cases is done by taking the union of the expressions in the two descriptions, and adjusting the probability of each expression according to whether or not it was in both descriptions. Matching entities that are identical are kept in the generalization, and non-identical entities are replaced by new entities that are still constrained by all of the statements about them in the union.

SEQL provides an interesting tradeoff between traditional symbolic generalization algorithms like EBL [Dejong & Mooney, 1986] and statistical generalization algorithms, such as connectionist systems. 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.

One potential drawback with SEQL is that generalizations – the union of many case descriptions – can grow quickly in size, but SME has polynomial memory requirements. Therefore SEQL must often pare down the facts in a generalization (it culls those with lowest probability), throwing away information to preserve a higher level of abstraction within a reasonable space requirement.

3.3 Rule Learning

The third method that we applied was more statistical in nature. It learns a list of association rules for predicting each possible perpetrator. In order to do this, it must convert the structured relational data of {D} into a feature-value representation. We used a case flattening approach introduced by Halstead & Forbus [2005] to accomplish this. Namely, SEQL was applied to all of the input cases at once in order to build one very large generalization. This generalization was then used as the framework for building a feature set, with each assertion in the generalization corresponding to one feature. The generalization and feature set, taken together, form an invertible mapping from relational case descriptions to features and back again.

For interpretation of the results, it is important to note here that features come in two types. The first, an *existential* feature, is the default type. It simply takes the value *true* or *false* depending on whether the assertion it represents is present in a given case or not. A *characteristic* feature on the other hand, can be used when more is known about the structure of the assertion, and so more information can be conveyed. For example, the English sentences "the attack killed someone" and "it was an ambush" are existential features – they are either true or false. The sentences "the attack killed three people" and "it happened in Baghdad" are characteristic features, which have the values 3 and Baghdad, respectively. As we will show, the conciseness of a reduced relational vocabulary makes it easier to extract characteristic feature values from the data.

For rule learning, we use the definition of association rule introduced by Agrawal, et al. [1996]. Hence, each association rule is a conjunction of feature-value pairs (a.k.a. literals, such as <location . Iraq>) which implies another such conjunction.

The actual rule learning is done using an ad-tree to cache the joint probabilities of the input data [Anderson & Moore, 1998]. Adopting Anderson & Moore's algorithm, we start with a list of candidate hypotheses (antecedents) H, initially empty, and perform 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. One example of a rule learned looks like:

```
(IF (AND (LOCATION ?ATTACK PHILIPPINES)
         (AGENTCAPTURED ?ATTACK ?AGENT))
    (PERPETRATOR ?ATTACK MOROISLAMICLIBERATIONFRONT))
```

Finally, for every possibly perpetrator, the algorithm actually learns a rule-list by recursively calling the rule-learner. For any perpetrator P, it begins by learning a single rule H1 ⇒ P. On the next iteration, it tries to learn a rule for H2 ⇒ P^¬H1. This process continues until it reaches a maximum of 5 rules or no more rules can be found.

Once a list of rules R_P is learned for every perpetrator P, then the input case E is applied to every R_P, to see which rules fire. The *n* rules which fired with the highest confidence have their consequents returned as the hypothesis to the identity of the perpetrator.

Whereas the other two learners are limited to reductive learning, the rule learner learns simple, higher-level patterns.

4 Reduced Vocabulary

The original human-encoded input data consisted of 581 predicates, drawn from a vocabulary of over 8,000. This was translated into only 22 predicates, a 96% reduction in vocabulary size and knowledge compression ratio of 26.4.

The reduced vocabulary that we used was a subset of the vocabulary designed by *Language Computer Corporation* for their *Polaris* system (Bixler, Moldovan, & Fowler, 2005). Their vocabulary consisted of 40 predicates, chosen for their usefulness in natural language processing, the feasibility of their automatic extraction from text, and of particular importance to this research, the broadest semantic coverage with the least amount of overlap. LCC explains:

> 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).

The subset of 22 predicates which we selected from this vocabulary was based simply on those *Polaris* predicates which were needed to represent the information already present in the *Whodunit* training cases. The final list of predicates is provided in Table 2.

Table 2. All relations in the reduced vocabulary

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

Each new predicate corresponds to any of a set of old predicates. These relationships were easily hand-coded into a list of 38 translation rules, which were used for the actual data reduction. 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 facts are actually translated by the application of more than one rule. An example of this translation process can be seen in Table 3.

Table 3. Translation Example

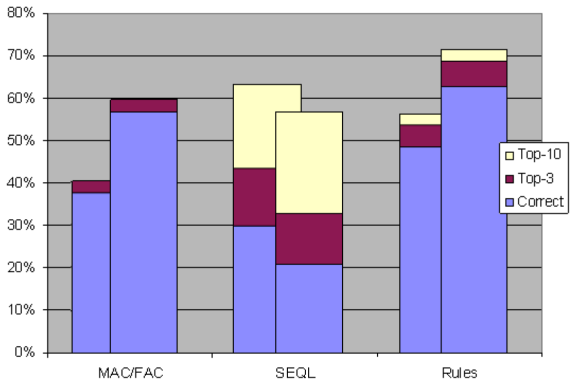
Cyc	(THEREEXISTEXACTLY 3 ?AGENT (AGENTCAPTURED ATTACK ?AGENT))
Rule 1	(TRANSLATE (THEREEXISTEXACTLY ?NUMBER ?VARIABLE ?FACT) (AND (MEASURE ?VARIABLE ?NUMBER) ?FACT))
Rule 2	(TRANSLATE (OBJECTACTEDON ?EVENT ?OBJECT) (AND (THEME ?EVENT ?OBJECT) (PREDICATE ?OBJECT ?PREDICATE) (RESULT ?EVENT ?PREDICATE)))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

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 turns one fact into 1.3 new facts. The average number of facts in a case increased by 70%.

5 Results

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. The middle ground is the "top 3" list, which has the virtue of providing both the best and some (hopefully mind-jogging) alternatives.

Chart 1. Results under old and new vocabularies, resp.



The results turn out to be very different for each algorithm. Chart 1 shows that under the conditions of the original vocabulary, the rule learner performs the best. 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.

Under the new vocabulary, the exemplar-based algorithm improved (p-value .045). SEQL though, performed worse (p-value .004). Perhaps stranger still, the rule learner, which depends on the generalization algorithm provided by SEQL, performed even better than it had before (p-value .015). It gets the correct answer on its first guess 60% of the time.

Closer examination reveals that the SEQL algorithm was hard-pressed. In the original vocabulary, SEQL generalized from case descriptions which contained an average of 20 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).

So, how did the rule-learner improve in performance? One possible explanation may be because of the leap it takes from reductive to higher-order learning. However, this seems to be only part of the story. More important is that the rule learner is able to take advantage of the conciseness in the reduced vocabulary. This conciseness allows the flattening process to generate many more characteristic values for the features than before: 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 were surprised by how well all three strategies performed, even the non-statistical ones, given the difficulty of the problem. Consider that although each case contains an average of only 24 facts, there are over 100 features in the dataset. This means that for any given record, over 75% of the features will be missing. This makes for a very sparse dataset. Fortunately, the closed world assumption seems to have held up. Yet, when we consider that the arity of the output attribute is 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 60% are quite good.

In conclusion, given that researchers tend to use larger relational vocabularies, it is extremely interesting that a reduced relational vocabulary can improve two out of three learners tested. Certainly, since these experiments only test one reduced vocabulary in one domain, some caution in interpreting the results is warranted. It also seems that reduced vocabularies are dangerous to use in a learning algorithm that already relies heavily on abstraction (e.g. SEQL), since it may lead to too much data and/or too much loss of information. However, vocabulary reduction does trade away smaller case descriptions and some information for added conciseness. Learning algorithms which are known to do well with large amounts of data (high dimensionality) and which can take advantage of this extra conciseness (such as the rule learner, which pays attention to the greater number of extractable feature values) appear likely to do better under a well-chosen reduced relational vocabulary.

6 Related Work

At first glance, this work appears to tie in strongly to forms of dimension reduction, such as feature selection. However, the task in feature selection is to automatically select a subset of concepts, a form of filtering. Reduced relational vocabularies abstract and transform more than they filter. Other forms of dimension reduction do perform a transformation on the data, such as Principal Component Analysis. However, the result of this transformation has often lost any real-world meaning. Furthermore, all of these techniques are typically automated within a given domain. This particular reduced vocabulary, in contrast, was manually constructed (by other researchers), based on needs of natural language processing, to work under any domain. Finally, although the number of relations in the data decreases under vocabulary reduction, the dimensionality of the data actually increases, since the simpler relations must be used in many different sentences to convey the same information.

Many different ontologies have also been developed in an effort to balance language expressiveness with computability. OWL is a common example targeted for use with the world wide web. Work has also been done on computing the expressiveness of a given ontology. For example, Corcho and Gomez-Perez [2000] establish a common framework for comparing the expressiveness and reasoning capbility between languages, and apply this methodology to compare and contrast between ontologies. Also, Golbreich, Dameron, et al. [2003] attempt to establish some minimal requirements that a web-based ontology should meet.

A number of alternative cognitive simulations of analogical mapping and retrieval have been developed. Some are domain-specific [Mitchell, 1993], and thus inapplicable to this problem. 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.

Also, there are many other approaches for doing learning from high-order relational data which do not use analogy at all. One of the most notable in recent years may be the building of probabilistic relational models, or PRM's [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. Other approaches include Blockeel and Uwents [2004], who present a method for building a neural network from relational data, and Dayanik and Nevill-Manning [2004], who discuss clustering relational data through a graph-partitioning approach. Recent work in link analysis also provides a means for tying probability in with relational data to do learning. Cohn and Hofmann [2001] actually create joint probability tables of both properties and links (in this case, terms and citations in document analysis) through a probabilistic decomposition process related to LSA, and then use it to perform classification. Variants on ILPs such as Bayesian logic programs (BLPs) [Kersting, de Raedt, & Kramer, 2000] have also been suggested for this sort of application.

References

[Agrawal et al., 1996] 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.

[Anderson & Moore, 1998] Anderson, B., & Moore, A. (1998). ADTrees for Fast Counting and for Fast Learning of Association Rules. *Knowledge Discovery from Databases*, New York, 1998.

[Blockeel & Uwents, 2004] 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.

[Cohn & Hofmann, 2001] 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.

[Corcho & Gomez-Perez, 2000] Corcho, O. and Gomez-Perez, A. (2000). A Roadmap to Ontology Specification Languages, *Lecture Notes in Computer Science*, Volume 1937, Jan 2000, p. 80

[Dayanik & Nevill-Manning, 2004] 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

[Eliasmith & Thagard, 2001] Eliasmith, C. and Thagard, P. 2001. Integrating structure and meaning: A distributed model of connectionist mapping. *Cognitive Science*.

[Falkenhainer et al., 1986] Falkenhainer, B., Forbus, K. and Gentner, D. The Structure-Mapping Engine. *Proceedings of the Fifth National Conference on Artificial Intelligence.* 1986.

[Forbus, 2001] Forbus, K. Exploring analogy in the large. In Gentner, D., Holyoak, K., and Kokinov, B. *Analogy: Perspectives from Cognitive Science.* MIT Press. 2001.

[Forbus et al., 1994] Forbus, K., Gentner, D., and Law, K. MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19, 141-205. 1994.

[Gentner, 1983] Gentner, D. Structure-mapping: A theoretical framework for analogy, *Cognitive Science* 7(2), 1983

[Getoor et al., 2001] Getoor, L., Friedman, N., Koller, D., & Pfeffer, A. Learning probabilistic relational models. In Dzeoski, S. and Lavrac, N. (Eds.), *Relational Data Mining* (pp. 307-335). Kluwer, 2001.

[Golbreich et al., 2003] 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.

[Halstead & Forbus, 2005] Halstead, D., & Forbus, K. (2005). Transforming between Propositions and Features: Bridging the Gap. *Proceedings of AAAI-2005. Pittsburgh, PA.*

[Hummel & Holyoak, 1997] Hummel, J.E., & Holyoak, K.J. (1997). Distributed representations of structure: A theory of analogical access and mapping. Psychological Review, 104.

[Kersting et al., 2000] Kersting, K., de Raedt, L., & Kramer, S. (2000). Interpreting Bayesian logic programs. *Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, pp. 29-35.

[Kuehne et al., 2000] 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.

[Leake, 1996] Leake, D. (Ed.) 1996. *Case-based Reasoning: Experiences, Lessons and Future Directions*, MIT Press.

[Mitchell, 1993] Mitchell, M. (1993) *Analogy-making as perception: A computer model*. MIT Press.