Structure Learning via Parameter Learning

William Yang Wang
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213, USA.
yww@cs.cmu.edu

Kathryn Mazaitis
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA.
krivard@cs.cmu.edu

William W. Cohen
Machine Learning Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA.
wcohen@cs.cmu.edu

ABSTRACT

A key challenge in information and knowledge management is to automatically discover the underlying structures and patterns from large collections of extracted information. This paper presents a novel structure-learning method for a new, scalable probabilistic logic called ProPPR. Our approach builds on the recent success of meta-interpretive learning methods in Inductive Logic Programming (ILP), and we further extends it to a framework that enables robust and efficient structure learning of logic programs on graphs: using an abductive second-order probabilistic logic, we show how first-order theories can be automatically generated via parameter learning. To learn better theories, we then propose an iterated structural gradient approach that incrementally refines the hypothesized space of learned first-order structures. In experiments, we show that the proposed method further improves the results, outperforming competitive baselines such as Markov Logic Networks (MLNs) and FOIL on multiple datasets with various settings; and that the proposed approach can learn structures in a large knowledge base in a tractable fashion.

Categories and Subject Descriptors

[**Information Systems Applications**]: Miscellaneous

Keywords

Probabilistic Prolog, structure learning, personalized PageRank

1. INTRODUCTION

Many information-management tasks (including classification [20], retrieval [10], information extraction [28], and information integration [29, 6]) can be formalized as learning and inference in an appropriate probabilistic first-order logic. To simplify the task, in many cases, first-order logic clauses in probabilistic logic systems are hand-written by developers, and the tasks consist of only parameter learning and inference. However, building such a hand-written first-order logic system can be very challenging: in the initial stage, domain experts or the developers themselves have to manually define the logic clauses that situates the domain-specific application scenarios. However, this approach might not generalize well to real-world problems, and the predefined clauses can be limited. Another issue is about the efficiency: constructing first-order logic programs manually can be very time-consuming, and have high financial costs. Finally, the maintenance cost is also non-trivial: when new data comes in, developers have to manually analyze the new dataset for updating and expanding the existing first-order clauses. Therefore, automated learning of logic program structures is of crucial significance for building robust first-order logic systems.

Unfortunately, many of the existing structure learning methods for learning probabilistic first-order logics are not efficient enough to be used for practical-sized datasets: when the total number of predicates and entities in a database become large, the costs of searching through all possible candidates to construct first-order clauses are also growing rapidly. For example, some existing techniques for learning Markov Logic Networks (MLNs) [31] may take days to run [14, 15], even though the input datasets include only a few dozen predicates and a few thousand grounded atoms.

This paper presents a structure-learning method for a new, scalable probabilistic logic called ProPPR [37]. ProPPR is efficient enough to support inference over large, noisy knowledge bases, and supports parameter learning using a parallelized version of stochastic gradient descent. In some cases, ProPPR is dramatically faster than prior approaches: for instance, ProPPR takes well under a minute on an ordinary desktop to learn numeric parameters for a theory with hundreds of clauses, over a knowledge base containing a million constants, while a state-of-the-art MLN implementation requires several hours for the same task, on a knowledge base only 1/1000 of the size [38]. However, there are no existing methods for learning ProPPR theories: previous experiments have used theories learned assuming radically different semantics for the clauses, or hand-written theories.

We present here a structure-learning method based on a recently-learned approach to learning the structure of conventional logic programs, in which logic programs are generated by using a second-order *abductive* program to "prove" that every observed positive example is covered. In abductive reasoning, assumptions are made, as necessary, to complete a proof: in this setting, the assumptions made by

CIKM'14, November 3–7, 2014, Shanghai, China.

ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2662022 .

Table 1: A simple program in ProPPR [37]. See text for explanation.

```
about(X,Z) :- handLabeled(X,Z)          # base.
about(X,Z) :- sim(X,Y),about(Y,Z)       # prop.
sim(X,Y) :- links(X,Y)                  # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)                     # sim,word.
linkedBy(X,Y,W) :- true                 # by(W).
```

the second-order program concern the existence of elements of the first-order program being generated. This approach, which is embodied in a system called Metagol [25], turns out to be both elegant and powerful, providing a conceptually clear framework for such important tasks as predicate invention, and learning recursive programs.

In this paper, we adapt a Metagol-like approach to learning ProPPR rules. In particular, we present an "abductive" stochastic second-order program for ProPPR, in which every assumption corresponds directly to a useful clause in a ProPPR program, and where further, *every learnable parameter corresponds directly to a first-order clause*. Structure learning is performed by computing the *gradient* of these features on training data, and constructing a small first-order stochastic program, consisting of those clauses that are potentially useful according to the gradient information. Parameters for this first-order program can then be learned in the usual way. We show that on small problems, this approach provides more accurate theories on the task of *knowledge base completion*, where the goal is to learn an interrelated set of rules to infer missing facts in an incomplete knowledge base. The method is also scalable enough to perform knowledge base completion for realistic knowledge bases containing tens of thousands of facts. We then propose an iterative variant of the gradient-guided structure learning approach that incrementally refines the hypothesized space of plausible clauses. Furthermore, to demonstrate the robustness of our approach, we also show promising results in two additional structure learning tasks in the biomedical and anthropological domains.

In the next section, we review the foundations and basic characteristics of ProPPR. In Section 3, we introduce the idea of using an abductive second-order theory for structure learning in ProPPR, focusing the problem of knowledge base (KB) completion, and learning inter-related relations. In Section 4, we demonstrate the robustness of our approach by showing the results for learning structures in the NELL KB, a biomedical ontology, as well as a complex kinship system. We then discuss related work in Section 5. Finally, we conclude in Section 6.

2. BACKGROUND: PROPPR

Table 1 and Figure 1 illustrate a ProPPR theory and a corresponding proof graph. We refer the reader to prior papers [37, 38] for a detailed explanation of ProPPR's semantics: briefly, however, a ProPPR theory T is a Horn clause (Prolog) theory, where each clause c is associated with a function ϕ_c which computes a set of features. Conceptually, a ProPPR program is executed by backward chaining, as in Prolog; however, rather that simply searching for a proof of a goal Q to determine if it is "true" according to the theory, ProPPR will use the the number of "probable" proofs to assign a degree of "truth" to a goal. In order to do this, ProPPR builds a graph rooted at the query goal, where each node corresponds to a conjunction of goals $A_1, \ldots, A_k$ and a substitution θ that imply the query goal (i.e. $T \wedge (A_1, \ldots, A_k\theta) \rightarrow Q\theta$), and each edge corresponds to the application of a clause c.

In the experiments of this paper, the feature-functions ϕ_c are simple: each feature corresponds to a Prolog goal, which may include bound variables from the head of the clause, the bindings of which are supplied when ϕ_c is invoked. A node corresponding to an the empty conjunction (denoted $\Box$) is a leaf of the graph, and corresponds to a completed proof of the query goal.

Additionally, every edge created by the clause c is labeled with the features produced by ϕ_c for that application of c. As in many deductive-database systems, we distinguish between the two types of clauses: the large number of unit clauses (aka facts) that comprise the "database", or extensionally-defined predicates (in the figure, the clauses defining *handLabeled*, *hasWord* and *linkedBy*); and the small number of clauses that comprise the "intensional" predicates (e.g., *about* and *sim*). Applications of any database clause are labeled with the special feature *db*.

Note that each different $\Box$ node corresponds to a different proof for Q, and different proofs may be associated with different substitutions θ and hence different solutions to the query Q. (In the figure, for instance, the leaf in the lower left corresponds to the solution $\theta = \{Z = fashion\}$ of the query $Q = about(a,Z)$, while the leaf in the lower right corresponds to $\{Z = sport\}$.) Following the related formalism of *stochastic logic programs (SLPs)*, we will assign a probabilistic score to every node in the proof graph, and then, to assign a probabilistic score to a particular solution Q', we simply marginalize over all leaf nodes $(\Box_i, \theta_i)$ such that $Q\theta_i = Q'$.

Finally, we associated a score with each node in the graph by performing a personalized PageRank (PPR) (aka "random walk with restart") process, where transition probabilities are based on edge weights, which are in turn determined by a weighted function of the features. In more detail [37, 38] the random process which defines the node weights are as follows. (a) Compute the total weight z of each outgoing edge as $g(\sum_i \lambda_i f_i)$, where the sum is taken of the features $f_1, \ldots, f_K$ which label that edge. Here, $g(\cdot)$ is an edge strength function, and in this study, we choose the well-known hyperbolic tangent function $tanh$ as g. The outgoing edges include an implicit *restart edge*, which goes back to to query node, and a self-loop edge which connects every empty goal list (i.e., solution node) to itself. (b) Normalize the edge weights to form a probability distribution, and pick an edge from that distribution. (c) Follow the edge to its destination and repeat.

These semantics are similar to those used SLPs [7], with two changes. One is the addition of the restart edges, which allow for a fast approximate proof procedure, in which only a small subset of the full proof graph is generated. In particular, if α upper-bounds the reset probability, and d upperbounds the degree of nodes in the graph, then one can efficiently find a subgraph with $O(\frac{1}{\alpha\epsilon})$ nodes which approximates the weight for every node within an error of $d\epsilon$ [37],

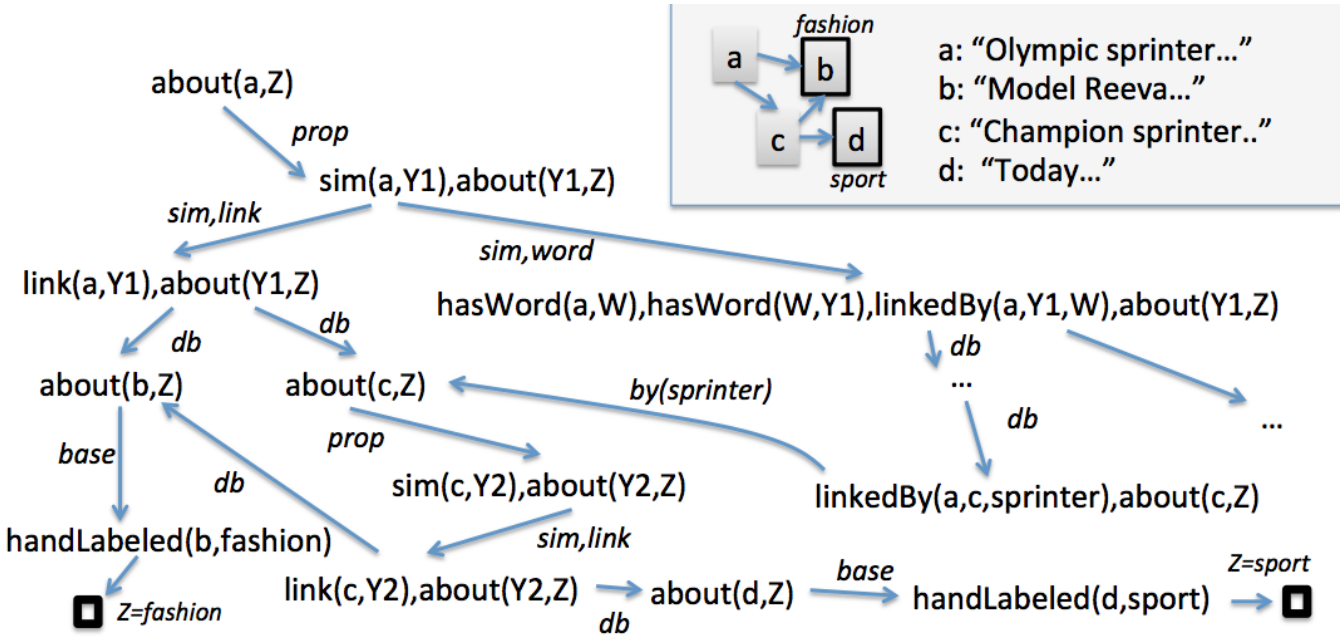

Figure 1: A partial proof graph for the query *about(a,Z)*. The upper right shows the link structure between documents a, b, c, and d, and some of the words in the documents. Restart links are not shown. [37]

using a variant of the PageRank-Nibble algorithm of Reid *et al* [1] . The second change is the addition of the feature functions, ϕ_c, which are used heavily in this work.

ProPPR's parameter learning framework is implemented using a parallel stochastic gradient descent variant to optimize the L_2-regularized logarithmic loss using the supervised personalized PageRank algorithm [2]: given the training queries and known solutions, we perform a random walk with restart process, and upweight the weights (edges) that are more likely to end up with a known positive solution.

The PPR-based scores used by ProPPR are quite different from the probability scores adopted by MLNs [31] and similar formalisms: intuitively, they measure the proportion of short proofs which support a belief, rather than counting the proportion of models in which the belief is true. Past experiments have used these scores in a retrieval context, to order potential answers to a query. In this context it has been shown that with parameter-learning, ProPPR performs well on tasks such as entity resolution, and inference over noisy knowledge bases [37]. Below we will consider the more difficult problem of learning the clauses of a ProPPR program.

3. STRUCTURE LEARNING FOR PROPPR

3.1 Structure learning is difficult for KB completion

To illustrate and motivate the problem of structure learning for ProPPR, we will use a classic problem introduced by Hinton in 1986 [11]. In this problem we have two families, each with twelve individuals, and twelve binary relations between these individuals: husband, wife, father, mother, son, daughter, brother, sister, uncle, aunt, nephew, and niece. From this data, we can also define 104 "queries", such as *uncle(charlotte, Y)*, each of which has some positive (correct) answers (e.g.*uncle(charlotte,james)*), and some incorrect answers. In our experiments the universe of potential answers (which we use ProPPR to rank) consists of all person pairs that are related by one of the twelve known predicates. We measure mean average precision (MAP)[1] over all the T_e test queries.

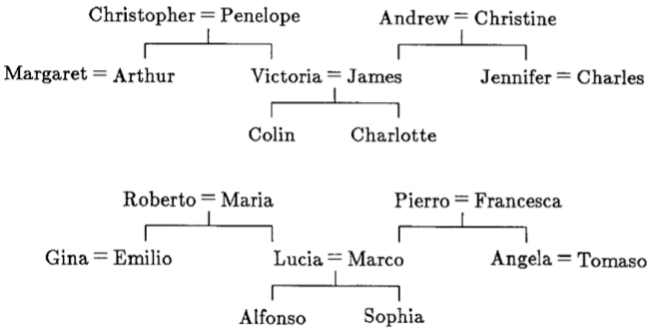

Figure 2: The families dataset, from [11].

In past experiments, high accuracies have been obtained by holding back a small number of test queries and training on the rest. We confirmed these results with two systems: Quinlan's FOIL [30] and Alchemy with structure-learning [14]. We designated one family as test and one as train, and we then performed 12 experiments where we

[1]Note that AP not only measures precision at rank k, but also measures recall: it uses the denominator to penalize the cases where positive solutions are missing. MAP has shown to be very robust and stable in many tasks [21], and it is widely used in relation learning tasks (e.g. [32] [16]).

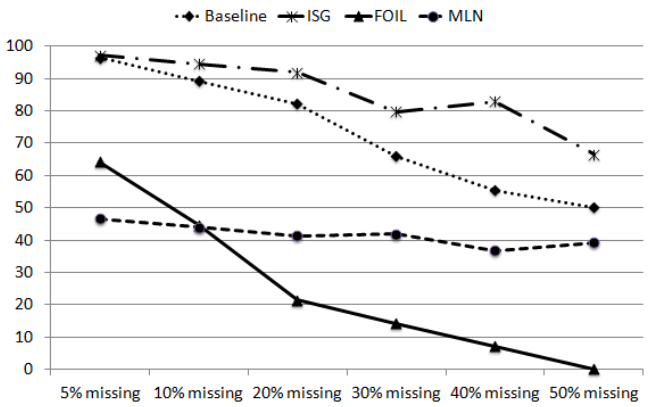

Figure 3: Completing an incomplete DB of family relations. X-axis: the percentage of background facts missing. Y-axis: the MAP result.

held out the queries from a single relation: in other words, for relation R, database consisted of facts defining the other 11 relations for both the train and test family; the training data consisted of the queries for R from the train family; and the test data was the queries for R from the test family. FOIL obtained precision of 100% for all 12 relations, and Alchemy obtained precision of 100% on 11 of the 12. This is not surprising, since all of the predicates have succinct definitions in terms of the others: for instance, *wife(X,Y)* $\Leftrightarrow$ *husband(Y,X)* (in this data).

However, the prior systems do not perform well if they need to learn interrelated concepts. We held out six pairs of relations (wife/husband, sister/brother, aunt/uncle, niece/nephew, and daughter/son) and repeated the same experiments. Alchemy's mean average precision (MAP) on the six problems drops to 27%, and FOIL's drops to zero. The problem for both systems stems from the use of pseudo-likelihood[2] to estimate the semantics of the partially-learned program from *examples*, rather than actual inference using the learned program. As a typical result, for the relation pair aunt/uncle, FOIL learns the rules *uncle(X,Y) :- husband(X,Z), aunt(Z,Y)* and *aunt(X,Y) :- wife(X,Z), uncle(Z,Y)*, which are circular.

Another illustration of the weaknesses of existing algorithms is provided by the following set of experiments. We trained with a DB containing all but k% of the facts for the training family, and all of the training-family queries as training data: thus, we are asking the system to learn rules which can *complete* an incomplete database. As test data, we used a database with all but k% of the facts for the test family, and again, all test-family queries as the test set. The results are shown in Figure 3: neither FOIL nor Alchemy's MLN method[3] outperform the simple baseline of predicting exactly the facts in the incomplete database.[4]

[2]Or in FOIL's case, an approach broadly similar to pseudo-likelihood.

[3]Alchemy's performance is quite sensitive to the precise set of missing facts, so we average over ten runs in the figure.

[4]Note that we have also experimented with a more recent "Learning with Structural Motifs (LSM)" variant [15] for learning MLN, but the results were much worse than Alchemy: we only observe a MAP of 10.7 on the missing 5% setting. This is because LSM is designed to learn long clauses (with more than 5 predicates) using recurring short patterns, whereas in our task, our goal is to learn short clauses with a maximum of 3 predicates in a clause. Another issue is that LSM has more than 20 hyperparameters to tune, which makes the structure learning process sensitive to the choice of the datasets and hyperparameters.

3.2 Iterated Structural Gradients for Structure Learning

Motivated by this, we introduce a new technique which we call the *iterated structural gradient* method for structure-learning in ProPPR. In particular, as noted in the introduction, we implement a Metagol-like method for introducing structure. We start with an "abductive" stochastic second-order program, in which every assumption corresponds directly to a useful clause in a first-order program, and where further, every learnable parameter corresponds directly to a first-order clause. Table 2 shows the theory that we use: this theory assumes that the first-order DB contains only binary facts, which are encoded for the second-order theory as triples of the form *rel(r,x,y)*: e.g., the fact *father(james,colin)* is represented as *rel(father,james,colin)*. Thus, rule (g) is a second-order version of the baseline algorithm of Figure 3: to interpret the predicate $P(X,Y)$, rule (g) simply checks for the fact *rel(P,X,Y)*.

Rules (a-c) can be viewed as a more powerful interpreter for the binary predicate P, which also makes assumptions about the presence of rules in the first-order theory. For instance, in an application of rule (b), the goal of "interpreting" (with the predicate *interp*) the a predicate *uncle(arthur,Y)* is reduced to the goal of interpreting (with the lower-level predicate *interp0*) some predicate *nephew(Y,arthur)*, assuming that the first-order theory contains a clause *uncle(X,Y) :- nephew(Y,X)*.

The way the assumption mechanism is implemented is quite simple. Associated with every interpretive action—i.e., clauses (a-c)—is an extra goal, such as *abduce_ifInv* for clause (b). These abductive goals are defined to always succeed, but whenever the proof step which lets them succeed is applied, the corresponding edge in the proof graph is labeled with an appropriate feature (e.g., *f_ifInv(uncle,nephew)*) which records that this assumption was made. Table 3 gives an example proof in the theory, showing how the abductive feature *f_ifInv(uncle,nephew)* might be used. ProPPR's natural bias towards short proofs (in the second-order theory) guides it toward near-minimal sets of assumptions regarding the first-order theory. Furthermore, *the gradient of the abductive features indicates the utility of the corresponding first-order clauses.*

Structure learning is performed by computing the *gradient* of these features on training data, and then producing a small first-order stochastic program, consisting of those clauses that are potentially useful according to the gradient information. Parameters for this first-order program can then be learned in the usual way.

We thus adopt the learning algorithm of Table 4, which we call the *iterated structural gradient* (ISG) method. As an extended example, we consider the operation of ISG on the problem of learning aunt/uncle together. In the first iteration, the following rules are proposed (not in order, and abbreviating *interp0* as *in0*):

in0(aunt,X,Y) :- in0(sister,X,Z), in0(father,Z,Y).
in0(uncle,X,Y) :- in0(brother,X,Z), in0(mother,Z,Y).
in0(aunt,X,Y) :- in0(nephew,Y,X).

Table 2: The abductive ProPPR program, in the right-most column, along with labels for each rule, and the second-order rules to which they correspond.

	Assumption	ProPPR clause
(a)	P(X,Y) :- R(X,Y)	interp(P,X,Y) :- interp0(R,X,Y),abduce_if(P,R).
(b)	P(X,Y) :- R(Y,X)	interp(P,X,Y) :- interp0(R,Y,X),abduce_ifInv(P,R).
(c)	P(X,Y) :- R1(X,Z),R2(Z,Y)	interp(P,X,Y) :- interp0(R1,X,Z),interp0(R2,Z,Y),abduce_chain(P,R1,R2).
(d)		abduce_if(P,R) :- true # f_if(P,R).
(e)		abduce_ifInv(P,R) :- true # f_ifInv(P,R).
(f)		abduce_chain(P,R1,R2) :- true # f_chain(P,R1,R2).
(g)		interp0(P,X,Y) :- rel(R,X,Y).

Table 3: A slightly-abbreviated sample proof using the second-order theory. The second column is the rule used at that point in the derivation, along with the features generated by that clause application, if any. (For clarity, we list the process of binding *rel(R,Y,arthur)* to the head of the unit clause *rel(nephew,colin,arthur) :-* , and then removing it, as two steps, DB_1 and DB_2.)

Goal List	Rule + Features
in(uncle,arthur,Y)	
↓	(b)
in0(R,Y,arthur),ab_ifInv(uncle,R)	
↓	(g)
rel(R,Y,arthur),ab_ifInv(uncle,R)	
↓	DB_1
rel(nephew,colin,arthur), ab_ifInv(uncle,nephew)	
↓	DB_2
ab_ifInv(uncle,nephew)	
↓	(e) + *f_ifInv(uncle,nephew)*
□	

in0(aunt,X,Y) :- in0(niece,Y,X).
in0(uncle,X,Y) :- in0(nephew,Y,X).
in0(uncle,X,Y) :- in0(niece,Y,X).

The first two of these are correct rules, and the remaining ones are over-general, as they confuse aunts and uncles. In the second iteration, ISG proposes the rules:

in0(aunt,X,Y) :- in0(wife,X,Z), in0(uncle,Z,Y).
in0(uncle,X,Y) :- in0(husband,X,Z), in0(aunt,Z,Y).
in0(aunt,X,Y) :- in0(wife,X,Z), in0(aunt,Z,Y).
in0(uncle,X,Y) :- in0(husband,X,Z), in0(uncle,Z,Y).
in0(aunt,X,Y) :- in0(uncle,X,Y).
in0(uncle,X,Y) :- in0(aunt,X,Y).
in0(aunt,X,Y) :- in0(aunt,X,Y).
in0(uncle,X,Y) :- in0(uncle,X,Y)

The first two of these are correct, while the remaining rules are, to various degrees, overgeneral and/or redundant. However, after parameter-learning, the learned theory performs perfectly on the test set.

Figure 3 shows the performance of ISG on the database-completion task, and compares it to FOIL, MLNs with structure learning, and the KB-only baseline. Table 5 shows results for the leave-two-relation out experiments discussed above. We also introduce two additional baselines for comparison: one is to perform the main loop only once, which we call the *structural gradient* (SG) method, and a final

Table 4: The Iterated Structural Gradient (ISG) Algorithm

1. For $t = 1, 2, \ldots$:
   (a) Perform $t-1$ epochs of parameter-learning on the theory of Table 2.
   (b) Compute the gradient of the loss, and for each feature with a negative gradient, add the corresponding clause to the theory:
   - for *f_if(p,q)*, add *interp0(p,X,Y) :- interp0(q,X,Y).*
   - for *f_ifInv(p,q)*, add *interp0(p,X,Y) :- interp0(q,Y,X).*
   - for *f_chain(p,q,s)*, add *interp0(p,X,Y) :- interp0(q,X,Z), interp0(s,Z,Y).*

   (c) Stop when no new rules are added.
2. Discard all rules but the added ones, and retrain the parameters for N epochs.

Table 5: Average precision performance for learning two mutually-related relations at once.

	FOIL	PL	MLN	SG	ISG
father+mother	0.0	23.32	42.53	70.05	100.0
husband+wife	0.0	4.73	3.20	39.63	79.4
daughter+son	0.0	11.49	22.74	70.05	100.0
sister+brother	0.0	3.29	10.37	62.18	78.85
uncle+aunt	0.0	10.41	53.35	79.41	100.0
niece+nephew	0.0	6.49	28.54	72.25	80.09
average	0.0	9.96	26.79	65.60	89.70

baseline is parameter-learning for the second-order theory, which we label PL in the table. Note that ISG performs quite well on the task of learning two interrelated predicates, even though it is quite difficult for the other systems (e.g. FOIL and MLNs). In addition to this, the ISG method usually converges quickly: empirically it typically converges within only 5 iterations.

Our method is superior because instead of using pseudo-likelihood, we take a holistic point of view: we use a second-order abductive logic to construct the hypothesis space, and relax the structure learning problem to first-order param-

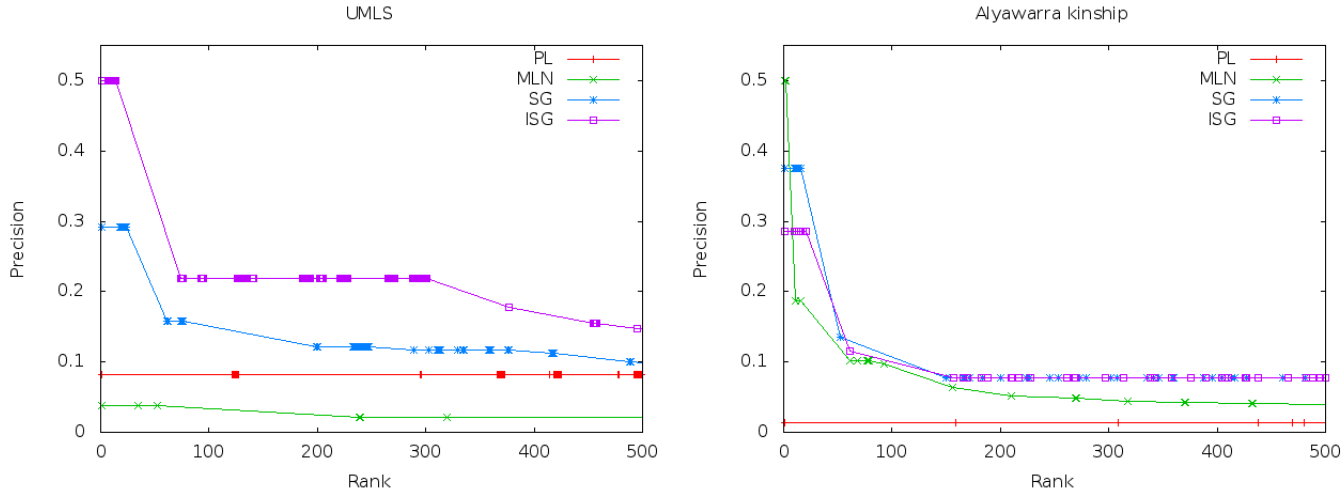

Figure 4: Performance on the UMLS and Alyawarra kinship datasets.

eter learning using supervised personalized PageRank with log likelihood. It is not surprising that the proposed ISG method has a good performance: the ISG method incrementally adds newly-learned gradient-guided plausible inference rules to the structure learning rule set, which helps to refine the overall multi-epoch structure learning process.

4. ADDITIONAL STRUCTURE LEARNING TASKS

4.1 The UMLS dataset

As tests of generality, we applied the proposed ISG approach on two larger-scale, more widely-used inference tasks. One is from the domain of biomedicine: in particular, we consider the Unified Medical Language System (UMLS) dataset [23, 22], which has been used in many structure learning tasks [13, 19]. The dataset[5] contains 46 predicates and 6,529 beliefs[6] : most predicates are verbs such as *"measures", "occurs in",* and *"treats"*, which indicate relations among entities. The entities are concepts such as *"enzyme", "mammal",* and *"virus"*. We used the following experimental procedure:

- We randomly select 90% of the data as training data, and the rest as testing data.
- We hold out the most frequent relation *affects* as the target query relation, and use other predicates as background knowledge in ProPPR[7].
- The above experiment is repeated 10 times.

Note that we choose this particular difficult setting because when removing the major predicate, it shows whether or not our approach is able to model the long tail of the Zipfian distribution of facts and relations, which previous systems may not be capable of.

[5]http://alchemy.cs.washington.edu/data/umls/

[6]Note that LSM [15] was too slow to run on this dataset, so we use Alchemy's structure learning algorithm to learn MLNs.

[7]Similarly, this means that we use "affects" as the non-evidence predicate for pseudo-likelihood computation in learning MLNs.

Table 6: Average precision performances for learning structures in UMLS, a biomedical ontology.

#run	PL	MLN	SG	ISG
1	2.8	2.3	9.6	12.1
2	8.6	2.0	9.2	10.7
3	6.0	2.2	10.8	12.7
4	5.5	2.7	8.2	10.4
5	9.1	2.6	10.7	13.8
6	8.1	1.9	9.2	11.6
7	7.5	2.1	9.1	11.3
8	4.8	1.8	7.8	11.8
9	4.0	3.2	7.8	11.2
10	5.3	1.9	8.6	11.6
average	6.2	2.3	9.1	11.7

Table 6 shows the experimental results for ISG, comparing them to Alchemy, PL, and SG. Although we learn a single predicate, rather than several inter-related predicates, this problem is still difficult, perhaps because the most common and well-connected predicate was not present in the database of facts. We see that even ProPPR's parameter-learning baseline PL obtains an averaged result of 6.2, which almost triple the result obtained by MLN's pseudo-likelihood structure learning approach. We also see that the gradient-based structure learning approaches outperform both the baseline and MLN, with ISG achieving a mean average precision of 11.7. The left figure in Fig. 4 shows the precision of predictions for *affects* predictions as a function of rank for each system, for a representative run. The rules proposed by ISG are qualitatively plausible: for example, in the second run, ISG proposes the following rules (in their first-order format):

affects(X,Y) :- causes(X,Z), isa(Z,Y).

Table 7: Average precision performances for Alyawarra kinship systems.

#run	PL	MLN	SG	ISG
1	0.9	2.8	4.6	5.3
2	0.8	3.3	6.8	7.2
3	1.0	3.8	7.1	7.3
4	0.7	2.9	5.6	6.0
5	1.0	3.1	5.6	5.9
6	0.7	3.8	5.6	6.2
7	0.6	3.2	7.2	7.2
8	0.7	3.4	5.0	5.4
9	0.9	3.6	6.4	6.8
10	0.9	2.1	6.2	6.6
average	0.8	3.2	6.0	6.4

affects(X,Y) :- causes(X,Z), associated_with(Z,Y).
affects(X,Y) :- complicates(X,Z), complicates(Z,Y).
affects(X,Y) :- complicates(X,Z), result_of(Z,Y).
affects(X,Y) :- interacts_with(X,Z), causes(Z,Y).
affects(X,Y) :- interacts_with(X,Z), diagnoses(Z,Y).
affects(X,Y) :- produces(X,Z), associated_with(Z,Y).
affects(X,Y) :- produces(X,Z), disrupts(Z,Y).

We see that all the above clauses obtained by the ISG method are plausible inference rules in the biomedical domains: for example, the first clause *"affects(X,Y) :- causes(X,Z), isa(Z,Y)."* is a formula of direct causes, where as the second clause *"affects(X,Y) :- causes(X,Z), associated_with(Z,Y)."* concerns the indirect causes. Interestingly, the third clause the ISG method has learned is a case of transitive complication.

4.2 The Alyawarra kinship dataset

Again as further test of robustness, we analyzed the Alyawarra kinship dataset [9], which is a more complex dataset from Hinton's domain of family kinship [11]. The Alyawarra are an aboriginal tribe from central Australia: the tribe has four kinship sections, and the author of the corpus asked 104 tribe members to provide kinship terms for each other. The original author of the dataset, an anthropologist named Denham, has since recorded the demographic features for each of his subjects, and created the ground truth partition by assigning each tribe members to one of the clusters. The version of the Alyawarra dataset[8] we use has 25 predicates, and 10,686 beliefs[9]. The task is to infer the latent paths that associate these predicates, and use them to make binary link predictions.

Similar to the setup in the UMLS dataset, we use the most frequent predicate "Term 16" as the hold-out target query relation in both training and testing, and use other predicates as background facts for ProPPR and evidence in MLN. We randomly select 90% of the data for training, and the rest for testing. Experiments are repeated 10 times. Again, this is a challenging setting, as a key predicate is not present in the background database of facts.

[8]http://alchemy.cs.washington.edu/data/kinships/

[9]Again, we are unable to run LSM [15] on this dataset due to the number of grounded assertions and relations, and Alchemy was used to learn the MLNs.

Table 8: Summary of the KBs used in experiments on completing subsets of NELL's KB.

	KB seed	
	Google	*baseball*
top 1k entities		
#train/test queries	100	100
#DB facts	853	890
#rules learned	20	15
train time (sec)	29	19
top 10k entities		
#rain/test queries	1000	1000
#DB facts	10630	11972
#rules learned	401	392
train time (sec)	62	65
top 100k entities		
#train/test queries	5000	5000
#DB facts	12902	9746
#rules learned	1094	939
train time (sec)	354	276

The detailed experimental results are shown in the Table 7. The right figure in Fig. 4 shows rank-based results from various systems of a sample run in this dataset. We see that the overall performances are consistent with the results from previous subsections: the proposed iterative structural gradient approach over ProPPR outperforms both the baseline methods and the MLN structure-learning approach.

4.3 Learning inference rules for the NELL knowledge base

Finally, as a larger-scale and more realistic task, we explore learning inference rules for the NELL knowledge base. The NELL (Never Ending Language Learning) research project is an effort to develop a never-ending learning system that operates 24 hours per day, for years, to continuously improve its ability to read (extract structured facts from) the web [5]. NELL is given as input an ontology that defines hundreds of categories (e.g., person, beverage, athlete, sport) and two-place typed relations among these categories (e.g., *athletePlaysSport(Athlete, Sport)*), which it must learn to extract from the web. NELL is also provided a set of 10 to 20 positive seed examples of each such category and relation, along with a downloaded collection of 500 million web pages from the ClueWeb2009 corpus (Callan and Hoy, 2009) as unlabeled data, and access to 100,000 queries each day to Google's search engine. NELL uses a multi-strategy semi-supervised multi-view learning method to iteratively grow the set of extracted "beliefs".

Inference on NELL's learned KB is challenging for two reasons. First, the learned KB is not only incomplete, but also noisy, since it is extracted imperfectly from the web. For example, a football team might be wrongly recognized as two separate entities, one with connections to its team members, and the other with a connection to its home stadium. Second, the inference problems are large.

Following prior work [37, 38], we used a number of varying-sized versions of the NELL knowledge base (KB): specifically we took KBs containing 1,000, 10,000 and 100,000 entities, centered around two NELL concepts, "Google" and "baseball". We took M NELL queries from these KBs to use for training queries, and a disjoint M to

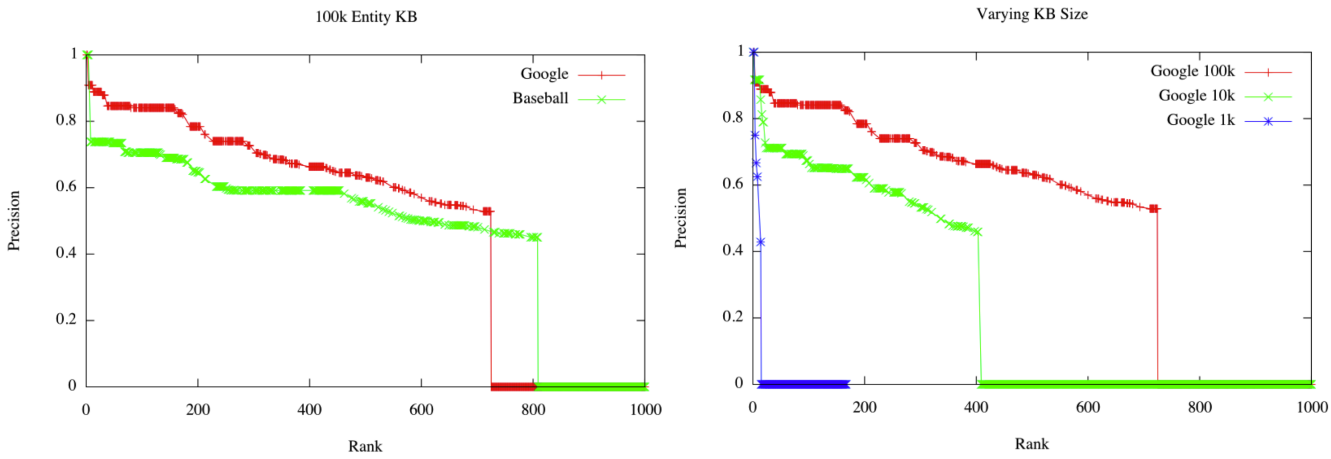

Figure 5: Performance on completing subsets of the NELL KB. Left, interpolated precision vs rank for two KBs with 100k entities; right; comparison on three KBs of different sizes based on the seed "Google".

use for testing queries. All facts not associated with these queries were used as the database. We used $M = 5000$ for the 100k-entity KBs, $M = 1000$ for the 10k-entity KBs, and $M = 100$ for the 1k-entity KBs[10]. Note that the baseline method of predicting using the database would, by construction, achieve an average precision of 0% on these test sets.

The performance of ISG on these tasks is shown in Figure 5 and Table 8. (Runtime is using 20 threads for learning on a conventional desktop machine.) Even though the data is noisy, ISG learns large and useful theories—theories such that the high-confidence predictions do indeed correspond, in most cases, with facts actually in the NELL knowledge base.

5. RELATED WORK

Our overall approach builds on Metagol [25], but differs in many respects: most notably, unlike Metagol, our system learns probabilistic programs in ProPPR, rather than "hard" Prolog programs. ProPPR's use of the (approximate) PageRank-Nibble-based proof method leads to many other differences: for instance, to learn recursive programs, Metagol requires a well-founded ordering of the Herbrand base to prevent infinite loops, while ProPPR does not; also, Metagol uses iterative deepening search to find a minimal set of abductions for each positive example, an NP-hard problem in the worst case, while ProPPR's proof methods instead find a large set of approximately minimal abductions. Metagol has been recently extended to learn a certain class of probabilistic programs [26], but has not been used to learn recursive theories of the size considered in this paper.

Experimentally, our experimental comparisons focus mainly on the widely-adopted MLN formalism—in particular the approach described by Kok and Domingos [14] which uses a beam search, coupled with pseudo-likelihood based parameter estimation, to learn MLNs. As noted above, use of pseudo-likelihood, while efficient, causes problems in learning multiple related predicates, the specific task addressed here, and other more recently-proposed MLN structure-learning schemes (e.g., [24, 12, 15]) do not appear to address this issue. In our preliminary experiments, we have also investigated the performance of LSM [15] on our datasets, but it fails to outperform the existing pseudo-likelihood based MLN structure learning algorithm in Alchemy. We hypothesize that the reason may be that LSM is a variant for learning long clauses for MLNs by examining short recurring patterns (aka Structural Motifs), thus when motifs are too short, the benefits of LSM may not retain. Another issue we encounter with LSM is the complexity of the setup: it includes more than 25 hyperparameters, making the process of adapting LSM to new problems very difficult.

The Alchemy implementation of structure-learning for MLNs that we used in our experiments is well-documented and stable, but is based on an arguably suboptimal inference substrate. Faster inference schemes (e.g., [33, 27, 35, 34] could possibly support structure-learning approaches that do not rely on the pseudo-likelihood approximation. Although these faster MLN inference methods have not yet been incorporated into structure-learning systems, integration of these lines of work is a plausible alternative to the approach we have described and experimentally tested here. We note, however, there is no theoretical analysis of the complexity of these methods, and experiments with both FROG [33] and LazySAT [35] suggest that unlike ProPPR they still lead to a groundings that grow with DB size, albeit more slowly; we also caution that heuristic inference speedups based on hand-coded MLNs need not necessarily transfer well to MLNs generated automatically.

Our work also builds heavily on Lao et al's Path Ranking Algorithm (PRA) [17, 16], which supports structure learning; however, PRA can learn only a very limited type of program (roughly, disjunctions of non-recursive chains of binary predicates). The underlying ProPPR logic used here can be viewed as combining ideas from PRA with stochastic logic programs (SLPs) [7]. Relative to SLPs, ProPPR adds a restart to the random-walk process, and the addition of a more flexible scheme for featurizing the logic. The featurization scheme was heavily used in the structure-learning proposed in this paper: in fact, the ability to attach arbi-

[10]On these problems, we were not able to successfully run MLN's structure-learning, even with only 1,000 entities.

trary feature sets, computed on-the-fly at proof time, appears to be unique to ProPPR, among first-order logics. To the best of our knowledge, SLPs have not been coupled with structure-learning methods.

ProbLog [8] is an alternative probabilistic logic programming formalism, which grounds probabilistic program by converting the space of possible proofs to a binary decision diagram (BDD), which can be very large, in the worst case, for recursive programs. There has been some prior work on learning BDD-based probabilistic programs, however; in particular, they have been used as the substrate for structure learning in Bellodi and Riguzzi's systems SLIPCASE [3] and SLIPCOVER [4]. In these systems, beam search is used to explore a space of probabilistic logic programs, and candidate programs are scored by running a small number of iterations of EM. In past experiments, SLIPCASE has been run with either a small depth bound (e.g., three), or else limited to non-recursive theories, so it is not clear that it will perform well on the larger mutually-recursive programs considered here. Some limited comparison can be seen in prior work [37], which evaluates ProPPR on the WebKB dataset: ProPPR obtained an AUC of 0.80 here with a simple fixed theory, compared to 0.61 for ProbLog with the same fixed theory, or 0.76 for SLIPCOVER's learned theory. However, experimental comparisons with SLIPCASE and similar systems remains a topic for future work. The BDD-based approach seems especially promising in conjunction with approximate reasoning methods [36], but to our knowledge, these have not been integrated with structure-learning approaches.

Our motivating task of knowledge-base completion has also been addressed with radically different methods, notably information extraction from text coupled with generalization and/or inference methods (e.g., [18, 39]). We focus here exclusively on the inference task, ignoring lexical clues, similar to the early work in this area with PRA [17].

6. CONCLUSIONS

We propose an abductive, meta-interpretive, second-order probabilistic logic based structure learning approach for ProPPR, a recently-developed scalable probabilistic language [37]. ProPPR is efficient enough to support inference over large, noisy, knowledge bases, and supports parameter learning using a parallelized version of stochastic gradient descent.

We implement structure-learning for ProPPR using a scheme suggested by the Metagol [25] system. We define a second-order abductive logic program where each assumption, and each learnable parameter, corresponds to a hypothesized ProPPR clause. Structure learning exploits this correspondence, finding clauses using the efficient, parallelizable, stochastic gradient descent based parameter-learning framework that exists in ProPPR. In our implementation of iterative structural gradients (ISG), steps in parameter-learning space are interleaved with structural changes to the second-order theory. We mainly experiment with theories of short rules, and our method can be naturally extended to learn longer chains and high-arity predicates.

In empirical evaluations, we show that this structure learning approach obtains promising results on data from several domains and tasks, including reasoning about kinship, biomedical reasoning, and a large-scale KB completion task. Additional experiments show that the approach scales well, and can effectively learn theories with hundreds of rules, from thousands of noisy examples, against a database with tens of thousands of facts noisy facts, in a few minutes of time on a conventional desktop. Hence, compared to popular ILP methods such as FOIL, or pseudo-likelihood based structure learning methods for MLNs, the approach has advantages in both the average precision, and runtime efficiency.

Acknowledgements

We thank Stephen Muggleton and Dianhuan Lin for interesting discussions of an early version of this paper. We are also grateful to anonymous reviewers for useful comments. This research was supported in part by DARPA grant FA8750-12-2-0342 funded under the DEFT program, and a Google Research Award. The authors are solely responsible for the contents of the paper, and the opinions expressed in this publication do not reflect those of the funding agencies.

7. REFERENCES

[1] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local partitioning for directed graphs using pagerank. *Internet Mathematics*, 5(1):3–22, 2008.

[2] Lars Backstrom and Jure Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, 2011.

[3] Elena Bellodi and Fabrizio Riguzzi. Learning the structure of probabilistic logic programs. In *Inductive Logic Programming*, pages 61–75. Springer, 2012.

[4] Elena Bellodi and Fabrizio Riguzzi. Structure learning of probabilistic logic programs by searching the clause space. *CoRR*, abs/1309.2080, 2013.

[5] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[6] William W. Cohen. Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems*, 18(3):288–321, July 2000.

[7] James Cussens. Parameter estimation in stochastic logic programs. *Machine Learning*, 44(3):245–271, 2001.

[8] Luc De Raedt, Angelika Kimmig, and Hannu Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th international joint conference on Artifical intelligence*, 2007.

[9] Woodrow Denham. The detection of patterns in alyawarra nonverbal behavior. *PhD thesis, University of Washington*, 1973.

[10] Norbert Fuhr. Probabilistic datalog—a logic for powerful retrieval methods. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 282–290. ACM, 1995.

[11] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the*

eighth annual conference of the cognitive science society, pages 1–12. Amherst, MA, 1986.

[12] Tuyen N Huynh and Raymond J Mooney. Discriminative structure and parameter learning for markov logic networks. In *Proceedings of the 25th international conference on Machine learning*, pages 416–423. ACM, 2008.

[13] Charles Kemp, Joshua B Tenenbaum, Thomas L Griffiths, Takeshi Yamada, and Naonori Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, page 5, 2006.

[14] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448. ACM, 2005.

[15] Stanley Kok and Pedro Domingos. Learning markov logic networks using structural motifs. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 551–558, 2010.

[16] Ni Lao and William W. Cohen. Relational retrieval using a combination of path-constrained random walks. *Machine Learning*, 81(1):53–67, 2010.

[17] Ni Lao, Tom M. Mitchell, and William W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, pages 529–539. ACL, 2011.

[18] Ni Lao, Amarnag Subramanya, Fernando C. N. Pereira, and William W. Cohen. Reading the web with learned syntactic-semantic inference rules. In *EMNLP-CoNLL*, pages 1017–1026. ACL, 2012.

[19] Ni Lao, Jun Zhu, Xinwang Liu, Yandong Liu, and William W Cohen. Efficient relational learning with hidden variable detection. In *NIPS*, pages 1234–1242, 2010.

[20] Daniel Lowd and Pedro Domingos. Efficient weight learning for markov logic networks. In *Knowledge Discovery in Databases: PKDD 2007*, pages 200–211. Springer, 2007.

[21] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[22] Alexa T McCray. An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics*, 4(1):80–84, 2003.

[23] Alexa T McCray, Anita Burgun, Olivier Bodenreider, et al. Aggregating umls semantic types for reducing conceptual complexity. *Studies in health technology and informatics*, (1):216–220, 2001.

[24] Lilyana Mihalkova and Raymond J Mooney. Bottom-up learning of markov logic network structure. In *Proceedings of the 24th international conference on Machine learning*, pages 625–632. ACM, 2007.

[25] Stephen Muggleton and Dianhuan Lin. Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 1551–1557. AAAI Press, 2013.

[26] Stephen H Muggleton, Dianhuan Lin, Jianzhong Chen, and Alireza Tamaddoni-Nezhad. Metabayes: Bayesian meta-interpretative learning using higher-order stochastic refinement. In preparation, available from http://www.doc.ic.ac.uk/ shm/mypubs.html.

[27] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS. *Proceedings of the VLDB Endowment*, 4(6):373–384, 2011.

[28] Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *Proceedings of the National Conference on Artificial Intelligence*, 2007.

[29] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with markov logic. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 650–659. Association for Computational Linguistics, 2008.

[30] J. Ross Quinlan. Learning logical definitions from relations. *Machine Learning*, 5(3):239–266, 1990.

[31] Matthew Richardson and Pedro Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006.

[32] Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. Relation extraction with matrix factorization and universal schemas. In *Proceedings of NAACL-HLT*, pages 74–84, 2013.

[33] Jude Shavlik and Sriraam Natarajan. Speeding up inference in markov logic networks by preprocessing to reduce the size of the resulting grounded network. In *Proceedings of the Twenty-first International Joint Conference on Artificial Intelligence (IJCAI-09)*, 2009.

[34] Parag Singla and Pedro Domingos. Memory-efficient inference in relational domains. In *Proceedings of the national conference on Artificial intelligence*, 2006.

[35] Parag Singla and Pedro Domingos. Lifted first-order belief propagation. In *Proceedings of the 23rd national conference on Artificial intelligence*, 2008.

[36] Guy Van den Broeck, Ingo Thon, Martijn van Otterlo, and Luc De Raedt. Dtproblog: A decision-theoretic probabilistic prolog. In *AAAI*, 2010.

[37] William Yang Wang, Kathryn Mazaitis, and William W Cohen. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2129–2138. ACM, 2013.

[38] William Yang Wang, Kathryn Mazaitis, Ni Lao, Tom Mitchell, and William W Cohen. Efficient inference and learning in a large knowledge base: Reasoning with extracted information using a locally groundable first-order probabilistic logic. *arXiv:1404.3301*, 2014.

[39] Limin Yao, Sebastian Riedel, and Andrew McCallum. Probabilistic databases of universal schema. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction*, pages 116–121. Association for Computational Linguistics, 2012.