Unifying Logical and Statistical AI

Pedro Domingos, Stanley Kok, Hoifung Poon, Matthew Richardson, Parag Singla
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, U.S.A.
{*pedrod, koks, hoifung, mattr, parag*}*@cs.washington.edu*

Abstract

Intelligent agents must be able to handle the complexity and uncertainty of the real world. Logical AI has focused mainly on the former, and statistical AI on the latter. Markov logic combines the two by attaching weights to first-order formulas and viewing them as templates for features of Markov networks. Inference algorithms for Markov logic draw on ideas from satisfiability, Markov chain Monte Carlo and knowledge-based model construction. Learning algorithms are based on the voted perceptron, pseudo-likelihood and inductive logic programming. Markov logic has been successfully applied to problems in entity resolution, link prediction, information extraction and others, and is the basis of the open-source Alchemy system.

Introduction

From the earliest days, AI research has tended to fall into two largely separate strands: one focused on logical representations, and one focused on statistical ones. The first strand includes approaches like logic programming, description logics, classical planning, symbolic parsing, rule induction, etc. The second includes approaches like Bayesian networks, hidden Markov models, Markov decision processes, statistical parsing, neural networks, etc. Logical approaches tend to emphasize handling complexity, and statistical ones uncertainty. Clearly, however, both of these are necessary to build intelligent agents and handle real-world applications. Our goal is thus to combine the two in a single coherent framework. In this paper, we describe Markov logic, a representation that has both finite first-order logic and probabilistic graphical models as special cases, and give an overview of current inference and learning algorithms for it.

Markov logic is part of an enterprise that has been gathering momentum for some time. Attempts to combine logic and probability in AI date back to at least Nilsson (1986). Bacchus (1990), Halpern (1990) and coworkers (e.g., Bacchus *et al.* (1996)) produced a substantial body of relevant theoretical work. Around the same time, several authors began using logic programs to compactly specify complex Bayesian networks, an approach known as knowledge-based model construction (Wellman, Breese, & Goldman 1992). More recently, many combinations of (subsets of) first-order logic and probability have been proposed in the burgeoning field of statistical relational learning (Dietterich, Getoor, & Murphy 2004), including probabilistic relational models (Friedman *et al.* 1999), stochastic logic programs (Muggleton 1996), Bayesian logic programs (Kersting & De Raedt 2001), relational dependency networks (Neville & Jensen 2004), and others. Markov logic is a step further in generality from these.

Preliminaries

A *Markov network* (also known as *Markov random field*) is a model for the joint distribution of a set of variables $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ (Della Pietra, Della Pietra, & Lafferty 1997). It is composed of an undirected graph G and a set of potential functions ϕ_k. The graph has a node for each variable, and the model has a potential function for each clique in the graph. A potential function is a non-negative real-valued function of the state of the corresponding clique. The joint distribution represented by a Markov network is given by

$$P(X=x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{1}$$

where $x_{\{k\}}$ is the state of the kth clique (i.e., the state of the variables that appear in that clique). Z, known as the *partition function*, is given by $Z = \sum_{x \in \mathcal{X}} \prod_k \phi_k(x_{\{k\}})$. Markov networks are often conveniently represented as *log-linear models*, with each clique potential replaced by an exponentiated weighted sum of features of the state, yielding

$$P(X=x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \tag{2}$$

A feature may be any real-valued function of the state. This paper will focus on binary features, $f_j(x) \in \{0, 1\}$. In the most direct translation from the potential-function form (Equation 1), there is one feature corresponding to each possible state $x_{\{k\}}$ of each clique, with its weight being $\log \phi_k(x_{\{k\}})$. This representation is exponential in the size of the cliques. However, we are free to specify a much smaller number of features (e.g., logical functions of the state of the clique), allowing for a more compact representation than the potential-function form, particularly when large cliques are present. Markov logic takes advantage of this.

Formulas in first-order logic are constructed from logical connectives, quantifiers, and symbols for predicates, constants, variables and functions. A *grounding* of a predicate is a replacement of all of its arguments by constants.[1] Likewise, a grounding of a formula is a replacement of all of its variables by constants. A *possible world* is an assignment of truth values to all possible groundings of predicates (ground atoms).

Markov Logic

A first-order KB can be seen as a set of hard constraints on the set of possible worlds: if a world violates even one formula, it has zero probability. The basic idea in Markov logic is to soften these constraints: when a world violates one formula in the KB it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects how strong a constraint it is: the higher the weight, the greater the difference in log probability between a world that satisfies the formula and one that does not, other things being equal.

Definition 1 *(Richardson & Domingos 2006) A Markov logic network (MLN) L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (Equations 1 and 2) as follows:*

1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground predicate is true, and 0 otherwise.*
2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.*

Thus there is an edge between two nodes of $M_{L,C}$ iff the corresponding ground predicates appear together in at least one grounding of one formula in L. For example, an MLN containing the formulas $\forall \mathtt{x}\ \mathtt{Smokes}(\mathtt{x}) \Rightarrow \mathtt{Cancer}(\mathtt{x})$ (smoking causes cancer) and $\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Friends}(\mathtt{x}, \mathtt{y}) \Rightarrow (\mathtt{Smokes}(\mathtt{x}) \Leftrightarrow \mathtt{Smokes}(\mathtt{y}))$ (friends have similar smoking habits) applied to the constants Anna and Bob (or A and B for short) yields the ground Markov network in Figure 1. Its features include $\mathtt{Smokes}(\mathtt{Anna}) \Rightarrow \mathtt{Cancer}(\mathtt{Anna})$, etc. Notice that, although the two formulas above are false as universally quantified logical statements, as weighted features of an MLN they capture valid statistical regularities, and in fact represent a standard social network model (Wasserman & Faust 1994).

An MLN can be viewed as a *template* for constructing Markov networks. From Definition 1 and Equations 1 and 2, the probability distribution over possible worlds x specified by the ground Markov network $M_{L,C}$ is given by

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_{i=1}^{F} w_i n_i(x)\right) \qquad (3)$$

[1] Or, more generally, by ground terms, but for simplicity in this paper we consider only groundings with constants.

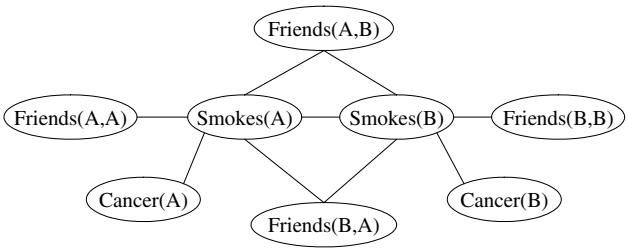

Figure 1: Ground Markov network obtained by applying an MLN containing the formulas $\forall \mathtt{x}\ \mathtt{Smokes}(\mathtt{x}) \Rightarrow \mathtt{Cancer}(\mathtt{x})$ and $\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Friends}(\mathtt{x}, \mathtt{y}) \Rightarrow (\mathtt{Smokes}(\mathtt{x}) \Leftrightarrow \mathtt{Smokes}(\mathtt{y}))$ to the constants $\mathtt{Anna}(\mathtt{A})$ and $\mathtt{Bob}(\mathtt{B})$.

where F is the number of formulas in the MLN and $n_i(x)$ is the number of true groundings of F_i in x. As formula weights increase, an MLN increasingly resembles a purely logical KB, becoming equivalent to one in the limit of all infinite weights. When the weights are positive and finite, and all formulas are simultaneously satisfiable, the satisfying solutions are the modes of the distribution represented by the ground Markov network. Most importantly, Markov logic allows contradictions between formulas, which it resolves simply by weighing the evidence on both sides. This makes it well suited for merging multiple KBs. Markov logic also provides a natural and powerful approach to the problem of merging knowledge and data in different representations that do not align perfectly, as the application section below illustrates.

It is interesting to see a simple example of how Markov logic generalizes first-order logic. Consider an MLN containing the single formula $\forall \mathtt{x}\ \mathtt{R}(\mathtt{x}) \Rightarrow \mathtt{S}(\mathtt{x})$ with weight w, and $C = \{\mathtt{A}\}$. This leads to four possible worlds: $\{\neg \mathtt{R}(\mathtt{A}), \neg \mathtt{S}(\mathtt{A})\}$, $\{\neg \mathtt{R}(\mathtt{A}), \mathtt{S}(\mathtt{A})\}$, $\{\mathtt{R}(\mathtt{A}), \neg \mathtt{S}(\mathtt{A})\}$, and $\{\mathtt{R}(\mathtt{A}), \mathtt{S}(\mathtt{A})\}$. From Equation 3 we obtain that $P(\{\mathtt{R}(\mathtt{A}), \neg \mathtt{S}(\mathtt{A})\}) = 1/(3e^w + 1)$ and the probability of each of the other three worlds is $e^w/(3e^w + 1)$. (The denominator is the partition function Z; see the previous section.) Thus, if $w > 0$, the effect of the MLN is to make the world that is inconsistent with $\forall \mathtt{x}\ \mathtt{R}(\mathtt{x}) \Rightarrow \mathtt{S}(\mathtt{x})$ less likely than the other three. From the probabilities above we obtain that $P(\mathtt{S}(\mathtt{A}) | \mathtt{R}(\mathtt{A})) = 1/(1 + e^{-w})$. When $w \to \infty$, $P(\mathtt{S}(\mathtt{A}) | \mathtt{R}(\mathtt{A})) \to 1$, recovering the logical entailment.

It is easily seen that all discrete probabilistic models expressible as products of potentials, including Markov networks and Bayesian networks, are expressible in Markov logic. In particular, many of the models frequently used in AI can be stated quite concisely as MLNs, and combined and extended simply by adding the corresponding formulas. Most significantly, Markov logic facilitates the construction of non-i.i.d. models (i.e., models where objects are not independent and identically distributed).

See Richardson and Domingos (2006) for further details on the Markov logic representation.

Inference

MAP/MPE Inference

In the remainder of this paper we assume that the MLN is in function-free clausal form. A basic inference task is finding the most probable state of the world given some evidence. (This is known as MAP inference in the Markov random field literature, and MPE inference in the Bayesian network literature.) Because of the form of Equation 3, in Markov logic this reduces to finding the truth assignment that maximizes the sum of weights of satisfied clauses. This can be done using any weighted satisfiability solver, and (remarkably) need not be more expensive than standard logical inference by model checking. (In fact, it can be faster, if some hard constraints are softened.) We have successfully used MaxWalkSAT, a weighted variant of the WalkSAT local-search satisfiability solver, which can solve hard problems with hundreds of thousands of variables in minutes (Kautz, Selman, & Jiang 1997).

One problem with this approach is that it requires propositionalizing the domain (i.e., grounding all predicates and clauses in all possible ways), which consumes memory exponential in the arity of the clauses. We have overcome this by developing a lazy version of WalkSAT, which grounds atoms and clauses only as needed (Singla & Domingos 2006b). This takes advantage of the sparseness of relational domains—where most atoms are false and most clauses are trivially satisfied—and can reduce memory usage by orders of magnitude. Our LazySAT algorithm is useful not just for Markov logic, but as a highly scalable satisfiability tester for relational domains.

Marginal and Conditional Probabilities

Another key inference task is computing the probability that a formula holds, given an MLN and set of constants, and possibly other formulas as evidence. By definition, the probability of a formula is the sum of the probabilities of the worlds where it holds, and computing it by brute force requires time exponential in the number of possible ground atoms. A more efficient alternative is to use Markov chain Monte Carlo (MCMC) inference (Gilks, Richardson, & Spiegelhalter 1996), which samples a sequence of states according to their probabilities, and counting the fraction of sampled states where the formula holds. This can be extended to conditioning on other formulas by rejecting any state that violates one of them.

If the evidence is a conjunction of ground atoms, as is typically the case, further efficiency can be gained by applying a generalization of knowledge-based model construction (Wellman, Breese, & Goldman 1992). This constructs only the minimal subset of the ground network required to answer the query, and runs MCMC (or any other probabilistic inference method) on it. The network is constructed by checking if the atoms that the query formula directly depends on are in the evidence. If they are, the construction is complete. Those that are not are added to the network, and we in turn check the atoms they depend on. This process is repeated until all relevant atoms have been retrieved. While in the worst case it yields no savings, in practice it can vastly reduce the time and memory required for inference. See Richardson and Domingos (2006) for details.

One problem with applying MCMC to MLNs is that it breaks down in the presence of deterministic or near-deterministic dependencies (as do other probabilistic inference methods). Deterministic dependencies break up the space of possible worlds into regions that are not reachable from each other, violating a basic requirement of MCMC. Near-deterministic dependencies greatly slow down inference, by creating regions of low probability that are very difficult to traverse. We have successfully addressed this problem by combining MCMC with satisfiability testing. Our MC-SAT algorithm uses WalkSAT to jump between regions of non-zero probability, while guaranteeing that the probability estimation process remains unbiased (Poon & Domingos 2006). MC-SAT greatly outperforms standard MCMC methods like Gibbs sampling and simulated tempering, and is applicable to any model that can be expressed in Markov logic, including many standard models in statistical physics, vision, social network analysis, spatial statistics, etc.

It is also possible to carry out lifted first-order probabilistic inference (akin to resolution) in Markov logic (Braz, Amir, & Roth 2005).

Learning

Generative Weight Learning

MLN weights can be learned generatively by maximizing the likelihood of a relational database (Equation 3). (We make a closed-world assumption: all ground atoms not in the database are false. This assumption can be removed by using an EM algorithm to learn from the resulting incomplete data.) The gradient of the log-likelihood with respect to the weights is

$$\frac{\partial}{\partial w_i} \log P_w(X=x) = n_i(x) - \sum_{x'} P_w(X=x')\, n_i(x') \quad (4)$$

where the sum is over all possible databases x', and $P_w(X=x')$ is $P(X=x')$ computed using the current weight vector $w = (w_1, \ldots, w_i, \ldots)$. In other words, the ith component of the gradient is simply the difference between the number of true groundings of the ith formula in the data and its expectation according to the current model. Unfortunately, computing these expectations requires inference over the model, which can be very expensive. Most fast numeric optimization methods (e.g., conjugate gradient with line search, L-BFGS) also require computing the likelihood itself and hence the partition function Z, which is also intractable. Although inference can be done approximately using MCMC, we have found this to be too slow. Instead, we maximize the pseudo-likelihood of the data, a widely-used alternative (Besag 1975). If x is a possible world (relational database) and x_l is the lth ground atom's truth value, the pseudo-log-likelihood of x given weights w is

$$\log P_w^*(X=x) = \sum_{l=1}^{n} \log P_w(X_l = x_l | MB_x(X_l)) \quad (5)$$

where $MB_x(X_l)$ is the state of X_l's Markov blanket in the data (i.e., the truth values of the ground atoms it appears in some ground formula with). Computing the pseudo-likelihood and its gradient does not require inference, and is therefore much faster. Combined with the L-BFGS optimizer, pseudo-likelihood yields efficient learning of MLN weights even in domains with millions of ground atoms (Richardson & Domingos 2006). However, the pseudo-likelihood parameters may lead to poor results when long chains of inference are required.

Discriminative Weight Learning

Discriminative learning is an attractive alternative to pseudo-likelihood. In many applications, we know *a priori* which predicates will be evidence and which ones will be queried, and the goal is to correctly predict the latter given the former. If we partition the ground atoms in the domain into a set of evidence atoms X and a set of query atoms Y, the *conditional likelihood (CLL)* of Y given X is $P(y|x) = (1/Z_x) \exp\left(\sum_{i \in F_Y} w_i n_i(x,y)\right) = (1/Z_x) \exp\left(\sum_{j \in G_Y} w_j g_j(x,y)\right)$, where F_Y is the set of all MLN clauses with at least one grounding involving a query atom, $n_i(x,y)$ is the number of true groundings of the ith clause involving query atoms, G_Y is the set of ground clauses in $M_{L,C}$ involving query atoms, and $g_j(x,y) = 1$ if the jth ground clause is true in the data and 0 otherwise. The gradient of the CLL is

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w(y|x) &= n_i(x,y) - \sum_{y'} P_w(y'|x) n_i(x,y') \\ &= n_i(x,y) - E_w[n_i(x,y)] \end{aligned} \quad (6)$$

As before, computing the expected counts $E_w[n_i(x,y)]$ is intractable. However, they can be approximated by the counts $n_i(x, y^*_w)$ in the MAP state $y^*_w(x)$ (i.e., the most probable state of y given x). This will be a good approximation if most of the probability mass of $P_w(y|x)$ is concentrated around $y^*_w(x)$. Computing the gradient of the CLL now requires only MAP inference to find $y^*_w(x)$, which is much faster than the full conditional inference for $E_w[n_i(x,y)]$. This is the essence of the voted perceptron algorithm, initially proposed by Collins (2002) for discriminatively learning hidden Markov models. Because HMMs have a very simple linear structure, their MAP states can be found in polynomial time using the Viterbi algorithm, a form of dynamic programming (Rabiner 1989). The voted perceptron initializes all weights to zero, performs T iterations of gradient ascent using the approximation above, and returns the parameters averaged over all iterations, $w_i = \sum_{t=1}^{T} w_{i,t}/T$. The parameter averaging helps to combat overfitting. T is chosen using a validation subset of the training data. We have extended the voted perceptron to Markov logic simply by replacing Viterbi with MaxWalkSAT (Singla & Domingos 2005).

Structure Learning

In principle, the structure of an MLN can be learned or revised using any inductive logic programming (ILP) technique. Since an MLN represents a probability distribution, the best results are obtained by using pseudo-likelihood as the evaluation function, rather than typical ILP ones like accuracy and coverage (Kok & Domingos 2005). Although this requires learning weights for each candidate structure, we have found that this is not a bottleneck, particularly if we initialize a candidate MLN's weights to those of its parent. The most expensive operation is in fact counting the number of true groundings of a candidate clause, but its cost can be greatly reduced by the use of sampling and caching. We have studied a number of clause construction operators and search strategies, bearing in mind that the goal is to learn arbitrary clauses, not just Horn ones like in most of ILP. In Kok and Domingos (2005), we start with either a set of unit clauses or an expert-supplied MLN, and either repeatedly find the best clause using beam search and add it to the MLN, or add all "good" clauses of length l before trying clauses of length $l + 1$. Candidate clauses are formed by adding each predicate (negated or otherwise) to each current clause, with all possible combinations of variables, subject to the constraint that at least one variable in the new predicate must appear in the current clause. Hand-coded clauses are also modified by removing predicates. We are currently investigating further approaches to learning MLNs, including automatically inventing new predicates (or, in statistical terms, discovering hidden variables).

Applications

Markov logic has been successfully applied in a variety of areas. A system based on it recently won a competition on information extraction for biology (Riedel & Klein 2005). Cycorp has used it to make parts of the Cyc knowledge base probabilistic (Matuszek 2006). We have applied it to link prediction, collective classification, entity resolution, social network analysis and other problems (Richardson & Domingos 2006; Singla & Domingos 2005; Kok & Domingos 2005; Singla & Domingos 2006b; Poon & Domingos 2006). Applications to Web mining, activity recognition, natural language processing, computational biology, game playing and others are under way.

The application to entity resolution illustrates well the power of Markov logic (Singla & Domingos 2006a). Entity resolution is the problem of determining which observations (e.g., database records, noun phrases, video regions, etc.) correspond to the same real-world objects, and is of crucial importance in many areas. Typically, it is solved by forming a vector of properties for each pair of observations, using a learned classifier (such as logistic regression) to predict whether they match, and applying transitive closure. Markov logic yields an improved solution simply by applying the standard logical approach of removing the unique names assumption and introducing the equality predicate and its axioms: equality is reflexive, symmetric and transitive; groundings of a predicate with equal constants have the same truth values; and constants appearing in a ground

predicate with equal constants are equal. This last axiom is not valid in logic, but captures a useful statistical tendency. For example, if two papers are the same, their authors are the same; and if two authors are the same, papers by them are more likely to be the same. Weights for different instances of these axioms can be learned from data. Inference over the resulting MLN, with entity properties and relations as the evidence and equality atoms as the query, naturally combines logistic regression and transitive closure. Most importantly, it performs *collective* entity resolution, where resolving one pair of entities helps to resolve pairs of related entities. Experiments on citation databases like Cora and BibServ.org show that this can greatly improve accuracy, particularly for entity types that are difficult to resolve in isolation (e.g., publication venues, where superficially very different names can refer to the same entity, like *AAAI-06* and *21st Natl. Conf. on AI*). See Singla and Domingos (2006a) for details.

The Alchemy System

The inference and learning algorithms described in the previous sections are publicly available in the open-source Alchemy system (Kok *et al.* 2005). Alchemy makes it possible to define sophisticated probabilistic models with a few formulas, and to add probability to a first-order knowledge base by learning weights from a relevant database. It can also be used for purely logical or purely statistical applications, and for teaching AI. From the user's point of view, Alchemy provides a full spectrum of AI tools in an easy-to-use, coherent form. From the researcher's point of view, Alchemy makes it possible to easily integrate a new inference or learning algorithm, logical or statistical, with a full complement of other algorithms that support it or make use of it.

Alchemy can be viewed as a declarative programming language akin to Prolog, but with a number of key differences: the underlying inference mechanism is model checking instead of theorem proving; the full syntax of first-order logic is allowed, rather than just Horn clauses; and, most importantly, the ability to handle uncertainty and learn from data is already built in.

Conclusion

Markov logic is a simple yet powerful approach to combining logic and probability in a single representation. We have developed a series of learning and inference algorithms for it, and successfully applied them in a number of domains. These algorithms are available in the open-source Alchemy system. We hope that Markov logic and its implementation in Alchemy will be of use to AI researchers and practitioners who wish to have the full spectrum of logical and statistical inference and learning techniques at their disposal, without having to develop every piece themselves.

Current and future directions of research include: extending the representation to handle continuous variables, infinite domains, and uncertainty at all levels; developing further efficient algorithms for inference and learning; learning from incomplete data; statistical predicate invention; first-order decision theory; various applications; and others.

Acknowledgements

This research was partly supported by DARPA grant FA8750-05-2-0283 (managed by AFRL), DARPA contract NBCH-D030010, NSF grant IIS-0534881, ONR grants N00014-02-1-0408 and N00014-05-1-0313, and a Sloan Fellowship and NSF CAREER Award to the first author. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, NSF, ONR, or the United States Government.

References

Bacchus, F.; Grove, A. J.; Halpern, J. Y.; and Koller, D. 1996. From statistical knowledge bases to degrees of belief. *Artificial Intelligence* 87:75–143.

Bacchus, F. 1990. *Representing and Reasoning with Probabilistic Knowledge*. Cambridge, MA: MIT Press.

Besag, J. 1975. Statistical analysis of non-lattice data. *The Statistician* 24:179–195.

Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order probabilistic inference. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 1319–1325. Edinburgh, UK: Morgan Kaufmann.

Collins, M. 2002. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, 1–8. Philadelphia, PA: ACL.

Della Pietra, S.; Della Pietra, V.; and Lafferty, J. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19:380–392.

Dietterich, T.; Getoor, L.; and Murphy, K., eds. 2004. *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*. Banff, Canada: IMLS.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1300–1307. Stockholm, Sweden: Morgan Kaufmann.

Gilks, W. R.; Richardson, S.; and Spiegelhalter, D. J., eds. 1996. *Markov Chain Monte Carlo in Practice*. London, UK: Chapman and Hall.

Halpern, J. 1990. An analysis of first-order logics of probability. *Artificial Intelligence* 46:311–350.

Kautz, H.; Selman, B.; and Jiang, Y. 1997. A general stochastic approach to solving problems with hard and soft constraints. In Gu, D.; Du, J.; and Pardalos, P., eds., *The Satisfiability Problem: Theory and Applications*. New York, NY: American Mathematical Society. 573–586.

Kersting, K., and De Raedt, L. 2001. Towards combining inductive logic programming with Bayesian networks. In *Proceedings of the Eleventh International Conference on Inductive Logic Programming*, 118–131. Strasbourg, France: Springer.

Kok, S., and Domingos, P. 2005. Learning the structure of Markov logic networks. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 441–448. Bonn, Germany: ACM Press.

Kok, S.; Singla, P.; Richardson, M.; and Domingos, P. 2005. The Alchemy system for statistical relational AI. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://www.cs.washington.edu/ai/alchemy.

Matuszek, C. 2006. Personal communication.

Muggleton, S. 1996. Stochastic logic programs. In De Raedt, L., ed., *Advances in Inductive Logic Programming*. Amsterdam, Netherlands: IOS Press. 254–264.

Neville, J., and Jensen, D. 2004. Dependency networks for relational data. In *Proceedings of the Fourth IEEE International Conference on Data Mining*, 170–177. Brighton, UK: IEEE Computer Society Press.

Nilsson, N. 1986. Probabilistic logic. *Artificial Intelligence* 28:71–87.

Poon, H., and Domingos, P. 2006. Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA: AAAI Press. This volume.

Rabiner, L. R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE* 77:257–286.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Machine Learning* 62:107–136.

Riedel, S., and Klein, E. 2005. Genic interaction extraction with semantic and syntactic chains. In *Proceedings of the Fourth Workshop on Learning Language in Logic*, 69–74. Bonn, Germany: IMLS.

Singla, P., and Domingos, P. 2005. Discriminative training of Markov logic networks. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 868–873. Pittsburgh, PA: AAAI Press.

Singla, P., and Domingos, P. 2006a. Entity resolution with Markov logic. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA.

Singla, P., and Domingos, P. 2006b. Memory-efficient inference in relational domains. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*. Boston, MA: AAAI Press. This volume.

Wasserman, S., and Faust, K. 1994. *Social Network Analysis: Methods and Applications*. Cambridge, UK: Cambridge University Press.

Wellman, M.; Breese, J. S.; and Goldman, R. P. 1992. From knowledge bases to decision models. *Knowledge Engineering Review* 7.