

Markov logic can be used as a general framework for joining logical and statistical AI.

BY PEDRO DOMINGOS AND DANIEL LOWD

Unifying Logical and Statistical AI with Markov Logic

FOR MANY YEARS, the two dominant paradigms in artificial intelligence (AI) have been logical AI and statistical AI. Logical AI uses first-order logic and related representations to capture complex relationships and knowledge about the world. However, logic-based approaches are often too brittle to handle the uncertainty and noise present in many applications. Statistical AI uses probabilistic representations such as probabilistic graphical models to capture uncertainty. However, graphical models only represent distributions over propositional universes and must be customized to handle relational domains. As a result, expressing complex concepts and relationships in graphical models is often difficult and labor-intensive.

To handle the complexity and uncertainty present in most real-world problems, we need AI that is both logical and statistical, integrating first-order logic and graphical models. One or the other

» key insights

- Intelligent systems must be able to handle the complexity and uncertainty of the real world. Markov logic enables this by unifying first-order logic and probabilistic graphical models into a single representation. Many deep architectures are instances of Markov logic.
- A extensive suite of learning and inference algorithms for Markov logic has been developed, along with open source implementations like Alchemy.
- Markov logic has been applied to natural language understanding, information extraction and integration, robotics, social network analysis, computational biology, and many other areas.



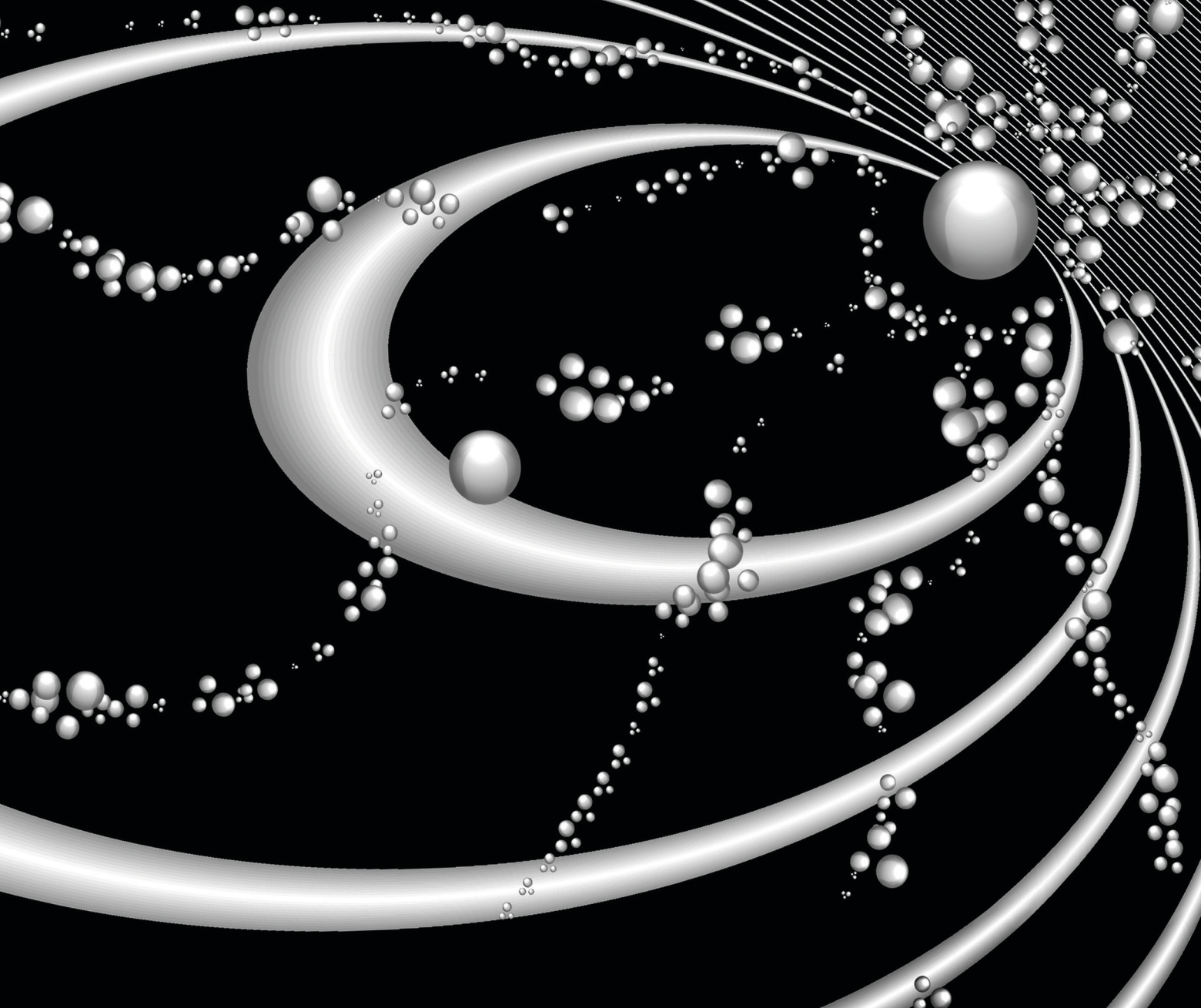


IMAGE BY GUIZEL KHUZHINA

by itself cannot provide the minimum functionality needed to support the full range of AI applications. Further, the two need to be fully integrated, and are not simply provided alongside each other. Most applications require simultaneously the expressiveness of first-order logic and the robustness of probability, not just one or the other. Unfortunately, the split between logical and statistical AI runs very deep. It dates to the earliest days of the field, and continues to be highly visible today. It takes a different form in each subfield of AI, but it is omnipresent. Table 1 shows examples of this. In each case, both the logical and the statistical approaches contribute something important. This justifies the abundant research on each of them, but also implies that ultimately a combination of the two is required.

Markov logic⁷ is a simple yet powerful generalization of first-order logic and probabilistic graphical models, which allows it to build on and integrate the best approaches from both logical and statistical AI. A *Markov logic network* (MLN) is a set of weighted first-order formulas, viewed as templates for constructing Markov networks. This yields a well-defined probability distribution in which worlds are more likely when they satisfy a higher-weight set of ground formulas. Intuitively, the magnitude of the weight corresponds to the relative strength of its formula; in the infinite-weight limit, Markov logic reduces to first-order logic. Weights can be set by hand or learned automatically from data. Algorithms for learning or revising formulas from data have also been developed. Inference algorithms for

Markov logic combine ideas from probabilistic and logical inference, such as Markov chain Monte Carlo, belief propagation, satisfiability, and resolution.

Markov logic has already been used to efficiently develop state-of-the-art models for many AI problems, such as collective classification, link prediction, ontology mapping, knowledge base refinement, and semantic parsing in application areas such as the Web, social networks, molecular biology, information extraction, and others. Markov logic makes solving new problems easier by offering a simple framework for representing well-defined probability distributions over uncertain, relational data. Many existing approaches can be described by a few weighted formulas, and multiple approaches can be combined by

Table 1. Examples of logical and statistical AI.

Field	Logical approach	Statistical approach
Knowledge representation	First-order logic	Graphical models
Automated reasoning	Satisfiability testing	Markov chain Monte Carlo
Machine learning	Inductive logic programming	Neural networks
Planning	Classical planning	Markov decision processes
Natural language processing	Definite clause grammars	Probabilistic context-free grammars

Table 2. Example of a first-order knowledge base and MLN.

English	First-order logic	Weight
"Friends of friends are friends."	$\forall x \forall y \forall z \text{ Fr}(x, y) \wedge \text{Fr}(y, z) \Rightarrow \text{Fr}(x, z)$	0.7
"Friendless people smoke."	$\forall x (\neg(\exists y \text{ Fr}(x, y)) \Rightarrow \text{Sm}(x))$	2.3
"Smoking causes cancer."	$\forall x \text{ Sm}(x) \Rightarrow \text{Ca}(x)$	1.5
"If two people are friends, then either both smoke or neither does."	$\forall x \forall y \text{ Fr}(x, y) \Rightarrow (\text{Sm}(x) \Leftrightarrow \text{Sm}(y))$	1.1

Fr() is short for Friends(), Sm() for Smokes(), and Ca() for Cancer().

including all of the relevant formulas. Many algorithms, as well as sample datasets and applications, are available in the open source Alchemy system¹⁷ (alchemy.cs.washington.edu).

In this article, we describe Markov logic and its algorithms, and show how they can be used as a general framework for combining logical and statistical AI. Before presenting background and details on Markov logic, we first discuss how it relates to other methods in AI.

Markov logic is the most widely used approach to unifying logical and statistical AI, but this is an active research area, and there are many others (see Kimmig et al.¹⁴ for a recent survey with many examples). Most approaches can be roughly categorized as either extending logic programming languages (for example, Prolog) to handle uncertainty, or extending probabilistic graphical models to handle relational structure. Many of these model classes can also be represented efficiently as MLNs (see Richardson and Domingos³² for a discussion of early approaches to statistical relational AI and how they relate to Markov logic). In recent years, most work on statistical relational AI has assumed a parametric factor (par-factor)²⁹ representation which is similar to the weighted formulas in an MLN. Probabilistic soft logic (PSL)¹

uses weighted formulas like Markov logic, but with a continuous relaxation of the variables in order to reason efficiently. In some cases, PSL can be viewed as Markov logic with a particular choice of approximate inference algorithm. One limitation of PSL's degree-of-satisfaction semantics is that more evidence does not always make an event more likely; many weak sources of evidence do not combine to produce strong evidence, even when the sources are independent.

Probabilistic programming²⁸ is another paradigm for defining and reasoning with rich, probabilistic models. Probabilistic programming is a good fit for problems where the data is generated by a random process, and the process can be described as the execution of a procedural program with random choices. Not every domain is well-suited to this approach; for example, we may wish to describe or predict the behavior of people in a social network without modeling the complete evolution of that network. Inference methods for probabilistic programming languages work backwards from the data to reason about the processes that generate the data and compute conditional probabilities of other events. Probabilistic graphical models perform similar reasoning, but typically have more structure to exploit for reasoning at scale.

Deep learning methods⁹ have led to competitive or dominant approaches in a growing number of problems. Deep learning fits complex, nonlinear functions directly from data. For domains where we know something about the problem structure, MLNs and other graphical models make it easier to capture background knowledge about the domain, sometimes specifying competitive models without having any training data at all. Due to their complementary strengths, combining deep learning with graphical models is an ongoing area of research.

First-Order Logic

A *first-order knowledge base* (KB) is a set of sentences or formulas in first-order logic. Formulas are constructed using four types of symbols: constants, variables, functions, and predicates. Constant symbols represent objects in the domain of interest (for example, people: Anna, Bob, and Chris). Variable symbols range over the objects in the domain. Function symbols (MotherOf) represent mappings from tuples of objects to objects. Predicate symbols represent relations among objects in the domain (Friends) or attributes of objects (Smokes). An *interpretation* specifies which objects, functions, and relations in the domain are represented by which symbols.

A *term* is any expression representing an object in the domain. It can be a constant, a variable, or a function applied to a tuple of terms. For example, Anna, x , and GreatestCommonDivisor(x, y) are terms. An *atomic formula* or *atom* is a predicate symbol applied to a tuple of terms (for example, Friends(x , MotherOf(Anna))). Formulas are recursively constructed from atomic formulas using logical connectives and quantifiers. If F_1 and F_2 are formulas, the following are also formulas: $\neg F_1$ (negation), which is true if F_1 is false; $F_1 \wedge F_2$ (conjunction), which is true if both F_1 and F_2 are true; $F_1 \vee F_2$ (disjunction), which is true if F_1 or F_2 is true; $F_1 \Rightarrow F_2$ (implication), which is true if F_1 is false or F_2 is true; $F_1 \Leftrightarrow F_2$ (equivalence), which is true if F_1 and F_2 have the same truth value; $\forall x F_1$ (universal quantification), which is true if F_1 is true for every object x in the domain; and $\exists x F_1$ (existential quantification), which is true if F_1 is true for at least one object x in the domain.

Parentheses may be used to enforce precedence. A *positive literal* is an atomic formula; a *negative literal* is a negated atomic formula. The formulas in a KB are implicitly conjoined, and thus a KB can be viewed as a single large formula. A *ground term* is a term containing no variables. A *ground atom* is an atomic formula all of whose arguments are ground terms. A *grounding* of a predicate or formula is a replacement of all of its arguments by constants (or functions all of whose arguments are constants or other functions, recursively, but we consider only the case of constants in this article).

A *possible world* (along with an interpretation) assigns a truth value to each possible ground atom.

A formula is *satisfiable* if and only if there is at least one world in which it is true.

Determining if a formula is satisfiable is only semidecidable. Because of this, knowledge bases are often constructed using a restricted subset of first-order logic with more desirable properties.

Table 2 shows a simple KB. Notice that although these formulas may be *typically* true in the real world, they are not *always* true. In most domains, it is very difficult to come up with non-trivial formulas that are always true, and such formulas capture only a fraction of the relevant knowledge. Thus, despite its expressiveness, pure first-order logic has limited applicability to practical AI problems.

Many ad hoc extensions to address this have been proposed. In the more limited case of propositional logic, the problem is well solved by probabilistic graphical models such as Markov networks, as we describe next. We will later show how to generalize these models to the first-order case.

Markov Networks

A *Markov network* (also known as *Markov random field*) represents a joint probability distribution over variables $X = \{X_1, X_2, \dots, X_n\}$ as a product of *factors* (also known as *potential functions*):

$$P(X = x) = \frac{1}{Z} \prod_c \phi_c(x_c) = \frac{1}{Z} \Phi(x) \quad (1)$$

where each ϕ_c is a nonnegative, real-valued function defined over variables $X_c \subset X$ and Z is a normalization constant known as the partition function.

For convenience, we also define $\Phi(x)$ as the unnormalized probability distribution, the product of all potential functions. A Markov network can be represented as a graph with one node per variable and an undirected edge between any two variables that appear together in the same factor.

Markov networks are often conveniently represented as *log-linear models*, with each potential function replaced by an exponentiated weighted sum of features of the state, leading to

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

A feature may be any real-valued function of the state. We will focus on binary features, $f_j(x) \in \{0, 1\}$, typically indicating if the variables are in some particular state or satisfy some logical expression.

Markov networks have been successfully applied to many problems in AI, such as stereo vision, natural language translation, information extraction, machine reading, social network analysis, and more. However, Markov networks only represent probability distributions over propositional domains with a fixed set of variables. There is no standard language for extending Markov networks to variable-sized domains, such as social networks over different numbers of people or documents with different numbers of words. As a result, applying Markov networks to these problems is a labor-intensive process requiring custom implementations.

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 (for example, see Table 2) 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 1.³² 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 $C = \{c_1, c_2, \dots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (Equations 1 and 2) as follows:

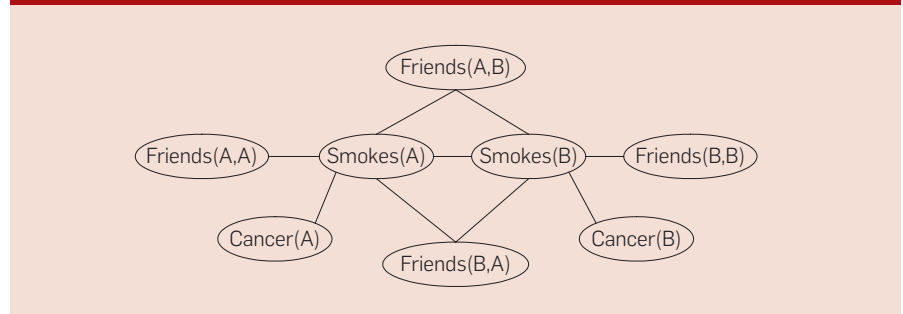
$M_{L,C}$ contains one random variable for each possible grounding of each atom appearing in L . The value of the variable is true if the ground atom is true and false otherwise.

$M_{L,C}$ 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 .

In the graph corresponding to $M_{L,C}$, there is a node for each grounding of each atom, and an edge appears between two nodes if the corresponding ground atoms appear together in at least one grounding of one formula in L .

For example, an MLN containing the formulas $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$ (smoking causes cancer) and $\forall x \forall y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$ (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

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




include $\text{Smokes}(\text{Anna}) \Rightarrow \text{Cancer}(\text{Anna})$. Notice that, although the two formulas 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. Notice also that nodes and links in the social networks are both represented as nodes in the Markov network; arcs in the Markov network represent probabilistic dependencies between nodes and links in the social network (for example, Anna's smoking habits depend on her friends' smoking habits).

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


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

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.

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



A first-order knowledge base can be seen as a set of hard constraints on the set of possible worlds. The basic idea in Markov logic is to soften these constraints.



$P(S(A) | R(A)) \rightarrow 1$, recovering the logical entailment.

Bayesian networks, Markov networks, and many other propositional models frequently used in machine learning and data mining can be stated quite concisely as MLNs, and combined and extended simply by adding the corresponding formulas.³² Most significantly, Markov logic facilitates the modeling of multi-relational data, where objects are not independent but are related in diverse and complex ways. Such representations are important for social networks, biological networks, natural language understanding, and more. Boltzmann machines and the deep models based on them are special cases of Markov networks.

MLN weights can be normalized globally, as in undirected graphical models, or locally, as in directed ones. The latter option enables MLNs to handle variable numbers of objects and irrelevant objects similar to directed first-order formalisms (counter to Russell's³⁴ claim; see also Domingos⁶). In practice, even globally normalized MLNs are quite robust to these variations, largely because the number of relations per object usually varies much less than the number of objects, and because factors from irrelevant objects cancel out in conditional probabilities.

When working with Markov logic, we typically make three assumptions about the logical representation: different constants refer to different objects (unique names), the only objects in the domain are those representable using the constant and function symbols (domain closure), and the value of each function for each tuple of arguments is always a known constant (known functions). These assumptions ensure the number of possible worlds is finite and that the Markov logic network will give a well-defined probability distribution. These assumptions are quite reasonable in most practical applications, and greatly simplify the use of MLNs. We will make these assumptions in most of the remainder of this article, but Markov logic can be generalized to domains where they do not hold, such as those with infinite objects or continuous random variables.^{36,37} Open-world reasoning methods discussed by Russell³⁴ can also be applied to

Markov logic, as Bayesian networks can be translated into MLNs.

Inference

Given an MLN model, the questions of interest are answered by performing inference on it. (For example, “What are the topics of these Web pages, given the words on them and the links between them?”) Because an MLN acts as a template for a Markov network, we can always apply standard Markov network inference methods on the instantiated network. However, methods that also exploit the logical structure in an MLN can yield tremendous savings in memory and time. We first provide an overview of inference in Markov networks, and then describe how these methods can be adapted to take advantage of the MLN structure.

Markov network inference. The main inference problem in Markov networks is computing the probability of a set of query variables Q given some evidence E :

$$\begin{aligned} P(Q=q|E=e) &= \frac{P(q,e)}{P(e)} \\ &= \frac{\sum_{h'} \Phi(q,e,h')}{\sum_{q',h'} \Phi(q',e,h')} = \frac{Z_{q,e}}{Z_e} \end{aligned} \quad (4)$$

where $H = X - Q - E$ denotes the remaining nonquery, nonevidence variables, Φ is the unnormalized product of potentials from Equation 1, and $Z_{q,e}$ and Z_e are the partition functions of reduced Markov networks, where the query and evidence variables have been fixed to constants. Thus, if we can compute partition functions, then we can answer arbitrary probabilistic queries.

In general, computing Z or answering other queries in a Markov network is #P-complete. When the Markov network has certain structural properties, such as a tree or tree-like structure, inference can be done in polynomial time. For network structures with many variables and loops, exact inference is usually intractable and approximate inference algorithms are used instead. The two most common approaches to approximation are to generate random samples through some random process that converges to the true distribution, or to solve a relaxed problem that captures as many constraints from the original as possible. Examples of the former approach include Markov chain

Monte Carlo (MCMC) and importance sampling, and examples of the latter include loopy belief propagation and variational methods.

Any of these methods could be used to perform inference in an MLN after instantiating the ground network, and many of them have. However, inference remains challenging in Markov networks and even more challenging in MLNs, which are often very large and have many loops. Next we will discuss on one of the most promising inference methods to date, which can take advantage of logical structure, perform exact inference when tractable, and be relaxed to perform approximate inference when necessary.

Weighted model counting. Computing the partition function in a Markov network can be reduced to a *weighted model counting* (WMC) problem. Weighted model counting finds the total weight of all satisfying assignments to a logical formula F . Following Chavira and Darwiche,² we focus on *literal-weighted* model counting, in which each literal is assigned a real-valued weight and the weight of an assignment is the product of the weights of its literals.

To represent Z as a WMC problem, we need each assignment x to receive weight $\Phi(x)$. Suppose each potential $\phi_i(x)$ evaluates to a constant Θ_i when a logical expression F_i is satisfied and 1 otherwise. (If the Markov network is not already in this form, we can convert it efficiently.) To define the WMC problem, for each potential ϕ_i , we introduce a literal A_i with weight Θ_i . We also introduce a logical formula, $A_i \Leftrightarrow F_i$, so that A_i is only true when F_i is satisfied. Thus, the product of the weights of the A_i literals is exactly the product of the original potential functions.

WMC algorithms can then be used to solve the problem and compute Z . One approach is recursive decomposition, in which we break the WMC problem into two subproblems, one where some variable x_i is fixed to true and one where x_i is fixed to false. This requires exponential time in the worst case, but WMC algorithms can often exploit problem structure to solve it much faster in practice. Another approach is to compile the model into a logical form where WMC is tractable, such as d-DNNF, and build an arithmetic circuit based on it.² Once

compiled, the arithmetic circuit can be reused for multiple queries.

Probabilistic theorem proving. WMC is a natural approach to inference in MLNs, as MLNs already use a logical representation for their features. However, MLNs have additional structure to exploit: each formula is instantiated many times with different combinations of constants. For example, suppose we are modeling a social network in which each pair of people is either friends or not. Before introducing any information about the individuals, the probability that any two people are friends must be the same as any other pair. *Lifted inference* exploits these symmetries to reason efficiently, even on very large domains.²⁹

Probabilistic theorem proving (PTP)⁸ applies this idea to perform *lifted* weighted model counting, so that many equivalent groundings of the same formula can be counted at once without instantiating them. As in the propositional case, lifted WMC can also be performed by compiling the first-order knowledge base to a (lifted) arithmetic circuit for repeated querying.⁵

In some cases, lifted inference lets us reason efficiently independent of domain size, so that inferring probabilities over millions of constants and trillions of ground formulas takes no longer than reasoning over hundreds. More often, evidence about individual constants breaks symmetries, so that different groundings are no longer exchangeable. The efficiency gains from lifting depend on both the structure of the knowledge base and the structure of the evidence.

When there is not enough structure and symmetry to perform inference exactly, we can replace some of the recursive conditioning steps in PTP with sampling.⁸ This leads to an approximate lifted inference algorithm, where sampling is used to estimate the weighted count of some of the subformulas.

Learning

Here, we discuss methods for automatically learning weights, refining formulas, and constructing new formulas from data.

Weight learning. In generative learning, the goal is to learn a joint probability distribution over all atoms. A standard approach is to maximize the likelihood of the data

through gradient-based methods. Note that we can learn to generalize from even a single example because the formula weights are shared across their many respective groundings. This is essential when the training data is a single network, such as the Web. Given mild assumptions about the relational dependencies, maximizing the likelihood (or pseudo-likelihood) of a sufficiently large example will recover the parameters that generated the data.³⁸

For MLNs, the gradient of the log-likelihood is the difference between the true formula counts in the data and the expected counts according to the model. When learning a generative probability distribution over all atoms, even approximating these expectations tends to be prohibitively expensive or inaccurate due to the large state space. Instead, we can maximize pseudo-likelihood, which is the conditional probability of each atom in the database conditioned on all other atoms. Computing the pseudo-likelihood and its gradient does not require inference, and is therefore much faster. However, the pseudo-likelihood parameters may lead to poor results when long chains of inference are required. In order to combat overfitting, we penalize each weight with a Gaussian prior, but for simplicity, we ignore that in what follows.

In many applications, we know a priori which atoms will be evidence and which ones will be queried. For these cases, discriminative learning optimizes our ability to predict the query atoms Y given the evidence X . A common approach is to maximize the *conditional likelihood* of Y given X ,

$$P(y|x) = \frac{1}{Z_x} \exp\left(\sum_{i \in F_y} w_i n_i(x, y)\right) \quad (5)$$

where F_y is the set of all MLN formulas with at least one grounding involving a query atom, and $n_i(x, y)$ is the number of true groundings of the i th formula involving query atoms. The gradient of the conditional log-likelihood is

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

where the sum is over all possible databases y' , and $P_w(y'|x)$ is $P(y'|x)$ computed using the current weight vector $w = (\dots, w_i, \dots)$. In other words, the i th component of the gradient is simply the difference between the number of true groundings of the i th formula in the data and its expectation according to the current model.

When computing the expected counts $E_w[n_i(x, y)]$ is intractable, we can approximate them using either the MAP state (that is, the most probable state of y given x) or by averaging over several samples from MCMC. We obtain the best results by applying a version of the scaled conjugate gradient algorithm. We use a small number of samples from MCMC to approximate the gradient and Hessian matrix, and use the inverse diagonal Hessian as a preconditioner (see Lowd and Domingos¹⁸ for more details and results).

MLN weights can also be learned with a max-margin approach, similar to structural support vector machines.¹¹

Structure learning. The structure of a MLN is the set of formulas to which we attach weights. Although these formulas are often specified by one or more experts, such knowledge is not always accurate or complete. In addition to learning weights for the provided formulas, we can revise or extend the MLN structure with new formulas learned from data. We can also learn the entire structure from scratch. The inductive logic programming (ILP) community has developed many methods for this purpose.⁴ ILP algorithms typically search for rules that have high accuracy, or high coverage, among others. However, because an MLN represents a probability distribution, much better results are obtained by using an evaluation function based on pseudo-likelihood.¹⁵ Log-likelihood or conditional log-likelihood are potentially better evaluation functions, but are much more expensive to compute.

Most structure learning algorithms focus on clausal knowledge bases, in which each formula is a disjunction of literals (negated or nonnegated atoms). The classic approach is to begin with either an empty network or an existing KB and perform a combinatorial search for formulas that improve the pseudo-likelihood.¹⁵

Either way, we have found it useful to start by adding all atomic formulas (single atoms) to the MLN. The weights of these capture (roughly speaking) the marginal distributions of the atoms, allowing the longer formulas to focus on modeling atom dependencies. To extend this initial model, we either repeatedly find the best formula using beam search and add it to the MLN, or add all “good” formulas of length l before trying formulas of length $l + 1$. Candidate formulas are formed by adding each predicate (negated or otherwise) to each current formula, with all possible combinations of variables, subject to the constraint that at least one variable in the new predicate must appear in the current formula. Hand-coded formulas are also modified by removing predicates.

A wide variety of other methods for MLN structure learning have been developed, such as generative learning with lifted inference,¹⁰ discriminative structure learning,¹¹ gradient boosting,¹³ and generating formulas using a Markov network²¹ or random walks.¹⁶ For the special case where MLN formulas define a relational Bayesian network, consistent Bayesian network structure learning methods can be extended to consistent structure learning in the relational setting.³⁵

Applications

MLNs have been used in a wide variety of applications, often achieving state-of-the-art performance. Their greatest strength is their flexibility for defining rich models in varied domains.


Collective classification. One of the most common uses of MLNs is for predicting the labels of interrelated entities, as in the friends and smoking example. Applications include labeling Web pages and predicting protein function.³ MLNs can also model collective classification tasks on sequential data, such as segmenting text for information extraction.³⁰

Link prediction. A second common task is to predict unknown or future relationships based on known relationships and attributes. Examples include predicting protein interaction,³ predicting advising relationships in a computer science department,³² and predicting work relationships among directors and actors.²⁰


Knowledge base mapping, integration, and refinement. Reasoning about the world requires combining diverse sources of uncertain information, such as noisy KBs. MLNs can easily represent KBs and soft constraints on the knowledge they represent: facts and rules in the knowledge base can be represented directly as atoms and formulas in the MLN. Ontology alignment can then be formulated as a link prediction problem, predicting which concepts in one ontology map to which concepts in the other. MLN formulas enforce structural similarity, so that related concepts in one ontology map to similarly related concepts in the other.²³ Similar rules can be used for knowledge base refinement, automatically detecting and correcting errors in uncertain knowledge by enforcing consistency among classes and relations.¹²

Semantic network extraction (SNE). A semantic network is a type of KB consisting of a collection of concepts and relationships among them. The SNE³⁹ system uses Markov logic to define a probability distribution over semantic networks. The MLN entities are the relation and object symbols from extracted tuples and the cluster assignments that group them into concepts and relationships. The MLN rules state that the truth of an extracted relationship depends on the clusters of the objects and relation involved. SNE uses a specialized bottom-up clustering algorithm to find the semantic clusters for objects and relations. This lets SNE scale to discover thousands of clusters over millions of tuples in just a few hours.

Semantic parsing. The goal of semantic parsing is to map sentences to logical forms representing the same information. The resulting information can be used to build a medical KB from PubMed abstracts, infer the meaning of a news article, or answer questions from an encyclopedia entry. Unsupervised semantic parsing³¹ learns to map dependency trees from sentences to their logical forms without any explicitly annotated data. The USP system does this by recursively clustering expressions (lambda forms) with similar subexpressions. The MLN for this includes four rules: one to cluster expressions into “core forms,” and three to cluster their



The goal of semantic parsing is to map sentences to logical forms representing the same information.



arguments into “argument forms” of some type and number. A clustering is more probable if expressions in the same cluster tend to have the same number of subexpressions as each other and those subexpressions are in the same clusters. As with SNE, USP uses a clustering algorithm to learn and reason more efficiently.

Extensions

Beyond the capabilities described here, Markov logic has been extended in a variety of ways to satisfy additional properties or accommodate different domains.

For decision theoretic problems, we can extend MLNs to Markov logic decision networks (MLDNs) by attaching a utility to each formula as well as a weight.²² The utility of a world is the sum of the utilities of its satisfied formulas. The optimal decision is the setting of the action predicates that jointly maximizes expected utility.

Domains with continuous as well as discrete variables can be handled by hybrid Markov logic networks (HMLNs).³⁷ HMLNs allow numeric properties of objects as nodes, in addition to Boolean ones, and numeric terms as features, in addition to logical formulas. For example, to reason about distances, we can introduce the numeric property $\text{Distance}(x, y)$. To state that a car should be centered in a lane, we can add terms such as:

$$\begin{aligned} \text{Car}(c) \wedge \text{LeftLine}(l) \wedge \text{RightLine}(r) \\ \Rightarrow -(\text{Dist}(c, l) - \text{Dist}(c, r))^2 \end{aligned}$$

When c is a car, l is the left lane boundary, and r is the right lane boundary, this term penalizes differences between the distance to the left and right boundaries. Inference algorithms for HMLNs combine ideas from satisfiability testing, slice-sampling MCMC, and numerical optimization. Weight learning algorithms are straightforward extensions of ones for MLNs.

Markov logic can be extended to infinite domains using Gibbs measures, the infinite-dimensional extension of Markov networks.³⁶ An MLN in an infinite domain is *locally finite* if each ground atom appears in a finite number of ground formulas. Local finiteness guarantees the existence of a probability measure; when the


interactions are not too strong, the measure is unique as well. Nonunique MLNs may still be useful for modeling large systems with strong interactions, such as social networks with strong word-of-mouth effects. In such cases, we can analyze the different “phases” of a nonunique MLN and define a satisfying measure to reason about entailment.

Recursive Markov logic networks or recursive random fields (RRFs)¹⁹ extend MLNs to multiple layers by replacing the logical formulas with MLNs, which can themselves have nested MLNs as features, for as many levels or layers as necessary. RRFs can compactly represent distributions such as noisy DNF, rules with exceptions, and *m*-of-all quantifiers. RRFs also allow more flexibility in revising or learning first-order representations through weight learning. An RRF can be seen as a type of deep neural network, in which the node activation function is exponential and the network is trained to maximize the joint likelihood of its input. In other ways, an RRF resembles a deep Boltzmann machine, but with no hidden variables to sum out.


Tractable Markov logic (TML)²⁴ is a probabilistic description logic where inference time is linear for all marginal, conditional, and MAP queries. TML defines objects in terms of class and part hierarchies, and allows objects to have probabilistic attributes, probabilistic relations between their subparts, and probabilistic existence. Tractability is ensured by having a direct mapping between the structure of the KB and the computation of its partition function: each split of a class into subclasses corresponds to a sum, and each split of a part into subparts corresponds to a product.

Getting Started with Markov Logic

If you would like to try out Markov logic for yourself, there are several open source software packages for learning or reasoning with MLNs. In some cases, software for learning and reasoning with Markov networks or conditional random fields can also be used; however, the task of translating from an MLN to a ground Markov network is left to you, and standard algorithms do not exploit the structure and symmetries present in MLNs.



When choosing the MLN structure, domain knowledge about the relevant relationships is a good place to start.



The oldest MLN toolkit is *Alchemy*,¹⁷ currently in version 2. Compared to other toolkits, *Alchemy* offers the widest variety of algorithms, such as multiple methods for generative and discriminative weight learning, structure learning, and marginal and MAP inference. *Tuffy*²⁵ offers a subset of *Alchemy*'s features but obtains greater scalability by using a database to keep track of groundings. *Tuffy* is the basis of *DeepDive*,²⁶ a system for information extraction, integration, and prediction built on Markov logic. Other implementations of Markov logic include *Markov the-beast*³³ and *RockIt*.²⁷

When applying Markov logic to new problems, it is usually best to start with a simple model on a small amount of data. High-arity predicates and formulas may have a large number of groundings, resulting in large models, high memory use, and slow inference. It is important to determine which modeling choices are most important for making accurate predictions and how expensive they are. Lifted inference techniques or customized grounding or inference methods can help good models scale to larger data.

When choosing the MLN structure, domain knowledge about the relevant relationships is a good place to start. When such knowledge is available, it is usually better to use it than to learn the structure from scratch. As with other knowledge engineering problems, there are often several ways to represent the same knowledge, and some representations may work better than others. For example, a relationship between smoking and cancer could be represented as equivalence ($\text{Smokes}(A) \Leftrightarrow \text{Cancer}(A)$), implication ($\text{Smokes}(A) \Rightarrow \text{Cancer}(A)$ and $\text{Cancer}(A) \Rightarrow \text{Smokes}(A)$), or conjunction ($\text{Smokes}(A) \wedge \text{Cancer}(A)$ and $\neg \text{Smokes}(A) \wedge \neg \text{Cancer}(A)$).

Conclusion and Directions for Future Research

Markov logic offers a simple yet powerful representation for AI problems in many domains. As it generalizes first-order logic, Markov logic can easily model the full relational structure present in many problems, such as multiple relations and attributes of different types and arities, relational concepts such as transitivity, and background knowledge in first-order

logic. And as it generalizes probabilistic graphical models, Markov logic can efficiently represent uncertainty in the concepts, attributes, relationships, among others required by most AI applications.

For future research, one of the most important directions is improving the efficiency of inference. Lifted inference algorithms obtain exponential speed-ups by exploiting relational symmetries, but can fail when these symmetries are broken by evidence or more complex structures. Tractable Markov logic guarantees efficient inference but constrains model structure. More research is needed to make reasoning work well in a wider range of models. Because most learning methods rely on inference, this will lead to more reliable learning methods as well.

A second key direction is enriching the representation itself. Markov logic is built on first-order logic, which is not always the best way to compactly encode knowledge, even in logical domains. For example, concepts such as “every person has at least five friends” are difficult to express with standard first-order connectives and quantifiers. Markov logic has been extended to handle decision theory, continuous variables, and more. Some new applications may require new extensions.

We hope that Markov logic 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. We also hope that Markov logic will inspire development of even richer representations and more powerful algorithms to further integrate and unify diverse AI approaches and applications. More details on Markov logic and its applications can be found in Domingos and Lowd.⁷

Acknowledgments

This research was partly supported by ARO grants W911NF-08-1-0242 and W911NF-15-1-0265, DARPA contracts FA8750-05-2-0283, FA8750-07-D-0185, FA8750-16-C-0158, HR0011-06-C-0025, HR0011-07-C-0060, NBCH-D030010 and AFRL contract FA8750-13-2-0019, NSF grants IIS-0534881 and IIS-0803481, ONR grants N-00014-05-1-0313,

N00014-08-1-0670, N00014-12-1-0312 and N00014-16-1-2697, an NSF CAREER Award (first author), a Sloan Research Fellowship (first author), an NSF Graduate Fellowship (second author) and a Microsoft Research Graduate Fellowship (second 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 ARO, DARPA, AFRL, NSF, ONR, or the United States Government. **C**

References

- Bach, S., Broecheler, M., Huang, B., Getoor, L. Hinge-loss Markov random fields and probabilistic soft logic. *J. Mach. Learn. Res.* 18, 109 (2017), 1–67.
- Chavira, M., Darwiche, A. On probabilistic inference by weighted model counting. *Artif. Intell.* 6–7, 172 (2008), 772–799.
- Davis, J., Domingos, P. Deep transfer via second-order Markov logic. In *Proceedings of the 26th International Conference on Machine Learning*. ACM Press, Montréal, Canada.
- De Raedt, L. *Logical and Relational Learning*. Springer, Berlin, Germany, 2008.
- Van den Broeck, G., Taghipour, N., Meert, W., Davis, J., De Raedt, L. Lifted probabilistic inference by first-order knowledge compilation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)* (2011), Barcelona, Spain.
- Domingos, P., Kersting, K., Mooney, R., Shavlik, J. What about statistical relational learning? *Commun. ACM* 58, 12 (2015), 8.
- Domingos, P., Lowd, D. *Markov Logic: An Interface Layer for AI*. Morgan & Claypool, San Rafael, CA, 2009.
- Gogate, V., Domingos, P. Probabilistic theorem proving. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI-11)*. AUAI Press, Barcelona, Spain, 2011.
- Goodfellow, I., Bengio, Y., Courville, A. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Van Haaren, J., Van den Broeck, G., Meert, W., Davis, J. Lifted generative learning of Markov logic networks. *Mach. Learn.* 103 (2015), 27–55.
- Huynh, T., Mooney, R. Discriminative structure and parameter learning for Markov logic networks. In *Proceedings of the 25th International Conference on Machine Learning* (2008). ACM Press, Helsinki, Finland, 416–423.
- Jiang, S., Lowd, D., Dou, D. Ontology matching with knowledge rules. In *Proceedings of the 26th International Conference on Database and Expert Systems Applications (DEXA 2015)* (2015). Springer, Valencia, Spain.
- Khot, T., Natarajan, S., Kersting, K., Shavlik, J. Gradient-based boosting for statistical relational learning: the Markov logic network and missing data cases. *Mach. Learn.* 100 (2015), 75–100.
- Kimig, A., Mihalkova, L., Getoor, L. Lifted graphical models: a survey. *Mach. Learn.* (1), 99 (2015), 1–45.
- Kok, S., Domingos, P. Learning the structure of Markov logic networks. In *Proceedings of the 22nd International Conference on Machine Learning* (2005). ACM Press, Bonn, Germany, 441–448.
- Kok, S., Domingos, P. Learning Markov logic networks using structural motifs. In *Proceedings of the 27th International Conference on Machine Learning* (2010). ACM Press, Haifa, Israel.
- Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., Lowd, D., Domingos, P. The Alchemy system for statistical relational AI. Technical Report. Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2000. <http://alchemy.cs.washington.edu>.
- Lowd, D., Domingos, P. Efficient weight learning for Markov logic networks. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases* (2007). Springer, Warsaw, Poland, 200–211.
- Lowd, D., Domingos, P. *Recursive random fields*. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (2007). AAAI Press, Hyderabad, India, 950–955.
- Mihalkova, L., Huynh, T., Mooney, R.J. Mapping and revising Markov logic networks for transfer learning. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence* (2007). AAAI Press, Vancouver, Canada, 608–614.
- Mihalkova, L., Mooney, R. Bottom-up learning of Markov logic network structure. In *Proceedings of the 24th International Conference on Machine Learning* (2007). ACM Press, Corvallis, OR, 625–632.
- Nath, A., Domingos, P. A language for relational decision theory. In *Proceedings of the International Workshop on Statistical Relational Learning* (2009). Leuven, Belgium.
- Niepert, M., Meilicke, C., Stuckenschmidt, H. A probabilistic-logical framework for ontology matching. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence* (2010). AAAI Press.
- Niepert, M., Domingos, P. Learning and inference in tractable probabilistic knowledge bases. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence* (2015). AUAI Press, Brussels, Belgium.
- Niu, F., Ré, C., Doan, A., Shavlik, J., Tuffy, J. Scaling up statistical inference in Markov logic networks using an RDBMS. *PVLDB* 4 (2011), 373–384.
- Niu, F., Zhang, C., Ré, C., Shavlik, J. DeepDive: web-scale knowledge-base construction using statistical learning and inference. In VLDS, 2012.
- Noessner, J., Niepert, M., Stuckenschmidt, H. RockIT: exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the 27th AAAI Conference on Artificial Intelligence* (2013). AAAI Press, Bellevue, WA.
- Pfeffer, A. *Practical Probabilistic Programming*. Manning Publications, 2016.
- Poole, D. First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (2003). Morgan Kaufmann, Acapulco, Mexico, 985–991.
- Poon, H., Domingos, P. Joint inference in information extraction. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence* (2007). AAAI Press, Vancouver, Canada, 913–918.
- Poon, H., Domingos, P. Unsupervised semantic parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing* (2009). ACL, Singapore.
- Richardson, M., Domingos, P. Markov logic networks. *Mach. Learn.* 62 (2006), 107–136.
- Riedel, S. Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence* (2008). AUAI Press, Helsinki, Finland, 468–475.
- Russell, S. Unifying logic and probability. *Commun. ACM* 58, 7 (2015), 88–97. <https://doi.org/10.1145/2699411>
- Schulte, O., Gholami, S. Locally consistent Bayesian network scores for multi-relational data. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17)* (2017). Melbourne, Australia, 2693–2700.
- Singla, P., Domingos, P. Markov logic in infinite domains. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence* (2007). AUAI Press, Vancouver, Canada, 368–375.
- Wang, J., Domingos, P. Hybrid Markov logic networks. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence* (2008). AAAI Press, Chicago, IL, 1106–1111.
- Xiang, R., Neville, J. Relational learning with one network: An asymptotic analysis. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics* (2011), 779–788.
- Kok, S. and Domingos, P. Extracting semantic networks from text via relational clustering. In *Proceedings of ECML/PKDD-08* (Antwerp, Belgium, Sept. 2008). Springer, 624–639.

Pedro Domingos (pedrod@cs.washington.edu) is a professor in the Allen School of Computer Science and Engineering at the University of Washington, Seattle, WA, USA.

Daniel Lowd (lowd@cs.uoregon.edu) is an associate professor in the Department of Computer and Information Science at the University of Oregon, Eugene, OR, USA.

Copyright held by authors/owners.
Publication rights licenced to ACM.