**Abstract**

Increasingly large Knowledge Bases are being created, by crawling the Web or other corpora of documents, and by extracting facts and relations using machine learning techniques. To manage the uncertainty in the data, these KBs rely on probabilistic engines based on Markov Logic Networks (MLN), for which probabilistic inference remains a major challenge. Today’s state of the art systems reduce the task of inference to weighted model counting and use an MCMC algorithm wrapped around SampleSAT to generate approximately uniform samples. This approach offers no theoretical error guarantees and, as we show, suffers from poor performance in practice.

In this paper we describe SlimShot (Scalable Lifted Inference and Monte Carlo Sampling Hybrid Optimization Technique), a probabilistic inference engine for Web-Scale knowledge bases. SlimShot converts the MLN to a tuple-independent probabilistic database, then uses a simple Monte Carlo-based inference, with three key enhancements: (1) it combines sampling with safe query evaluation, (2) it estimates a conditional probability by jointly computing the numerator and denominator, and (3) it adjusts the proposal distribution based on the sample cardinality. In combination, these three techniques allow us to give formal error guarantees, and we demonstrate empirically that SlimShot outperforms today’s state of the art probabilistic inference engines used in knowledge bases.

**Introduction**

Increasingly sophisticated information extraction and AI techniques have enabled the automatic construction of large knowledge bases, created by crawling the web and extracting facts and relations using machine learning techniques (Etzioni et al. 2011; Niu et al. 2012; Dong et al. 2014; Krompaß, Nickel, and Tresp 2014; Chang et al. 2014). While conventional fact bases such as YAGO (Hoffart et al. 2013) and Freebase¹ contain high-quality, human-curated facts that are guaranteed (with a very high degree of certainty) to be correct, the tuples extracted by automatic methods unavoidably contain noisy and uncertain data. Google’s KnowledgeVault (Dong et al. 2014) and Geo DeepDive (Zhang et al. 2013) are examples of large scale knowbases where automatic methods produce highly calibrated probabilistic annotations for its extracted facts. Such systems store probabilistic data, for example, in KnowledgeVault a fact like BornIn(Obama, Hawaii) may have probability 0.7.

To control and manage the quality of the data and query answers, these systems rely critically on background knowledge, which is captured in a formalism like Markov Logic Networks (Domingos and Lowd 2009) (reviewed in ). An MLN consists of a collection of soft and hard constraints, which are quite similar to constraints in databases, but are annotated with a weight, representing the degree to which they should hold in the KB. An MLN introduces complex correlations between the large collection of facts in the knowledge base. Query answering requires probabilistic inference, and this is a challenging task. Answering even the simplest query that returns a single tuple requires reasoning over a large set of correlated tuples. Scalable and reliable probabilistic inference remains a major unsolved challenge for large scale knowledge bases.

All MLN systems today perform inference using some variant of Monte Carlo Markov Chain (MCMC), to sample from the space of possible worlds. While MCMC provably converge, their theoretical convergence rate is too slow for practical purposes. Instead, systems usually run a fixed number of simulation steps, e.g. 10,000, and return whatever they find. However, in practice they perform much worse than one expects. The problem is that, at their core, they need to sample uniformly from the set of solutions of a Boolean formula, and precise uniformity is critical for convergence. But uniform sampling is a challenging task in itself. The state of the art is SampleSAT (Wei, Erenrich, and Selman 2004), which is heuristic-based and can only approximate the uniform distribution. Together, the weak theoretical guarantees coupled with the fact that their main assumption does not hold in practice means that MCMC-based systems have very poor quality in practice.

In this paper we propose a new approach to scalable probabilistic inference over large Knowledge Bases. The key new idea in our approach is to combine sampling with lifted inference (also called safe query evaluation), thus converting the standard 0/1 estimation problem, where each sample leads to either 0 or 1, to the problem of estimating the expected value of a function in [0, 1]. This allows us to deploy two powerful optimizations: Rao-Blackwellized (or collapsed) sampling

---

¹https://www.freebase.com/
and importance sampling.

Our approach reduces MLNs to Weighted Model Counting (WMC). Recently, there have been three parallel, very promising developments for both exact and approximate WMC. Sentential Decision Diagrams (Darwiche 2011) (SDDs) are an exact model counting approach that compile a Boolean formula into circuit representations, where WMC can be done in linear time. SDDs have state-of-the-art performance for many tasks in exact weighted model counting, but also have some fundamental theoretical limitations: (Beame and Liew 2015) prove exponential lower bounds even for simple UCQ’s whose probabilities are in PTIME. WeightMC (Chakraborty et al. 2014) is part of a recent and very promising line of work (Ermon et al. 2013; Chakraborty et al. 2014), which reduces approximate model counting to a polynomial number of oracle calls to a SAT solver. Adapting this technique to weighted model counting is non-trivial: Chakraborty (Chakraborty et al. 2014) proves that this is possible if the models of the formula have a small nilt (ratio between the largest and smallest weight of any model). The nilt is a more stringent measure than our output-tilt, which is the ratio of two aggregates and can be further reduced by using importance sampling. Finally, a third development consists of lifted inference (Poole 2003; Braz, Amir, and Roth 2005; Singla and Domingos 2008; den Broeck et al. 2011; Gribkoff, Van den Broeck, and Suciu 2014), which are PTIME, exact WMC methods, but only work for a certain class of formulas: in this paper we combine lifted inference with sampling, to apply to all formulas.

We start by translating the MLN into a tuple-independent probabilistic database: the query probability in the MLN becomes a conditional probability \( P(Q | \Gamma) \) in the tuple independent database, where \( \Gamma \) is a constraint. The (unconditional) probability of a formula \( P(\Phi) \) can be estimated using Monte Carlo simulation (sample \( N \) worlds, return the fraction that satisfy the formula), but this requires a number of simulation steps inversely proportional to the probability. This works well when \( \Phi \) is a Union of Conjunctive Queries (UCQ), which is an \( \exists^* \) formula and usually has a high probability. But it fails when applied to \( \Gamma \) (the denominator in \( P(Q | \Gamma) \)), which is a \( \forall^* \) sentence with a tiny probability (say, \( 10^{-9} \)). In other words, even though \( P(Q | \Gamma) \) is relatively large (say 0.1 to 0.9), we need a huge number of steps to compute it, because \( P(\Gamma) \) is tiny. This has prevented MLN systems from adopting a translation into probabilistic databases, relying on MCMC instead.

Our approach is to combine Monte Carlo sampling with lifted inference (also called safe query evaluation), in a technique we call SafeSample. Fix a subset of the relations such that, once these relations are made deterministic, the query can be evaluated in polynomial time. Then run the Monte Carlo simulation by sampling only over these relations, and computing the probability of each sample using a safe query plan; in other words, instead of estimating a \( O(1) \)-random variable, we estimate a random variable with values in \([0, 1]\) (also called discrete integration). Safe query evaluation has been studied both in probabilistic databases (Suciu et al. 2011) and in AI (den Broeck et al. 2011) where it is called lifted inference; we will use both terms interchangeably in this paper.

The real power of SafeSample comes from enabling two additional optimizations. CondSample evaluates the numerator and denominator of \( P(Q \land \Gamma) / P(\Gamma) \) together, by using each sample to increment estimates for both the numerator and the denominator. This correlated sampling technique works only in combination with SafeSample. Otherwise, if both numerator and denominator are \( O(1) \)-random variables, CondSample becomes equivalent to rejection sampling, which samples worlds, rejects those that do not satisfy \( \Gamma \), and returns the fraction of worlds that satisfy \( Q \land \Gamma \). Rejection sampling wastes many samples, leading to poor performance. By using SafeSample instead, we can compute the exact probability of both \( Q \land \Gamma \) and \( \Gamma \) at each sample, and add it to both sums. In other words, we no longer waste samples, and instead make every sample count. Our main theoretical result (included in the full version of this paper) shows that the convergence rate of CondSample is inverse proportional to the conditional probability of \( Q \) given \( \Gamma \), and a parameter called output-tilt (ratio between the largest and smallest probability over all samples). In other words, we no longer depend on the tiny probability of \( \Gamma \), but instead on the conditional probability \( P(Q | \Gamma) \), and the output-tilt. The second optimization, ImportanceSample, further decreases the output-tilt by weighting samples in inverse proportion to the probability of \( \Gamma \).

**Background**

We fix a relational vocabulary \( \sigma = (R_1, \ldots, R_m) \), and denote \( DB = (R^{DB}_1, \ldots, R^{DB}_m) \) a database instance over \( \sigma \). We identify \( DB \) with the set of all tuples, and write \( D \subseteq DB \) to mean \( R^D_i \subseteq R^{DB}_i \) for all \( i \). A First Order formula with free variables \( x = (x_1, \ldots, x_k) \) in prenex-normal form is an expression:

\[
\Phi(x) = E_1 y_1 E_2 y_2 \cdots E_t y_t \varphi(x, y)
\]

where each \( E_i \) is either \( \forall \) or \( \exists \) and \( \varphi \) is a quantifier-free formula using the logical variables \( x_1, \ldots, x_k, y_1, \ldots, y_t \). A sentence is a formula without free variables. In this paper we consider two kinds of formulas: a query is a formula with quantifier prefix \( \exists^* \), and a constraint is a formula with quantifier prefix \( \forall^* \); note that both queries and constraints may have free variables. A query can be rewritten as \( Q = C_1 \lor C_2 \lor \cdots \) where each \( C_i \) is a conjunctive query with negation, while a constraint can be written as \( \Delta_1 \land \Delta_2 \land \cdots \) where each \( \Delta_i \) is a clause with quantifier prefix \( \forall^* \).

**Probabilistic Databases**

A tuple-independent probabilistic database, or probabilistic database for short, is a pair \((DB, p)\) where \( p : DB \to [0, 1] \) is a function that associates to each tuple \( t \in DB \) a probability \( p(t) \). It defines a probability space on the set of possible worlds, where each tuple \( t \) is included independently, with probability \( p(t) \). Formally, for each subset \( D \subseteq DB \), called a possible world, its probability is \( P_{DB,p}(D) = \prod_{t \in D} p(t) \cdot \prod_{t \in DB - D} (1 - p(t)) \). The probability of a sentence \( \Phi \) is:

\[
P_{DB,p}(\Phi) = \sum_{D \subseteq DB} P_{DB,p}(D)
\]
If \( Q(x) \) is a query, then its output is defined as the set of pairs \((a, p)\), where \( a \) is a tuple of constants of the same arity as \( x \), and \( p \) is the probability of the Boolean query obtained by substituting the free variables \( x \) with the constants \( a \), 

\[
p = \text{def } P_{\text{DB},p}(Q[a/x]).
\]

We drop the subscripts from \( P_{\text{DB},p} \) when clear from the context.

A relation \( R \) is called deterministic, if for every tuple \( t \in R \), \( p(t) \in \{0, 1\} \), otherwise it is called probabilistic. We sometimes annotate with a subscript \( R_d \) the deterministic relations. We denote \( A \) the active domain of the database, and \( n = |A| \).

**Query Evaluation** In general, computing \( P(\Phi) \) is \#P-hard in the size of the active domain\(^2\). The standard approach is to first compute the *grounding* of \( \Phi \) on the database \( \text{DB} \), also called the *lineage* (Abiteboul, Benjelloun, and Milo 2008) which is a Boolean formula, then compute the probability of this Boolean formula; we do not use lineage in this paper and will not define it formally. If \( \Phi \) is an \( \exists \) sentence, then the lineage is a DNF formula, which admits an FPTRAS using Karp and Luby’s sampling-based algorithm (Karp and Luby 1983). But the lineage of an \( \forall \) sentence is a CNF formula, and even very restricted classes of CNF formulas do not admit an FPTRAS unless \( P = \text{NP} \) (Roth 1996).

An alternative approach to compute \( P(\Phi) \) is called lifted inference in the Statistical Relational Learning literature (den Broeck et al. 2011), or safe query evaluation in probabilistic databases (Suciu et al. 2011). It always runs in \( \text{PTIME} \) in \( n \), but only works for some sentences \( \Phi \). We refer the reader to (Suciu et al. 2011) for more details on lifted inference in probabilistic databases. We note that in the literature the term lifted inference sometimes refers to symmetric databases (den Broeck, Meert, and Darwiche 2014): a relation \( R \) is called symmetric if all ground tuples \( R(t) \) over the active domain have the same probability, and a probabilistic database is called symmetric if all its relations are symmetric. In this paper we do not restrict databases to be symmetric, and will use lifted inference to mean the same thing as safe query evaluation.

**Safe plans** Following other systems (Boulos et al. 2005; Olteanu, Huang, and Koch 2009), SlimShot performs lifted inference by rewriting the query into a safe query plan, which is then evaluated inside a relational database engine.

**SlimShot**

SlimShot performs query evaluation on MLNs. The data (evidence) is stored in a relational database, and the probabilistic inference is pushed inside the relational database engine; there is no separate grounding step. The MLN is translated into a tuple-independent probabilistic database then \( P_{\text{MLN}}(Q) \) is computed by the conditional probability:

\[
P(Q|\Gamma) = \frac{P(Q \land \Gamma)}{P(\Gamma)}
\]  

\((1)\)

We denote \( x = P(Q|\Gamma) \) throughout this section.

A naive approach to compute eq. \((1)\) is to estimate the numerator and denominator separately, using Monte Carlo sampling. This is impractical because if \( \Gamma \) is a \( \forall^* \) sentence, hence \( y \) is a tiny quantity, say, \( 10^{-9} \), requiring \( N \approx 10^{11} \) simulation steps to guarantee a relative error \( \delta = 0.1 \). In contrast, \( x = P(Q|\Gamma) \) is relatively large, say \( 0.1 \) to \( 0.9 \), but to compute it we need to divide two tiny quantities.

SlimShot performs three extensions to the naive Monte Carlo simulation, each with a provable reduction in the required number of steps: (1) SafeSample: combines sampling with exact lifted inference, (2) CondSample: guarantees an error bound for the conditional directly, without relying on separate error bounds for the numerator and denominator, and (3) ImportanceSample: deploys importance sampling to further improve the convergence rate.

**SafeSample**

The main idea of our entire approach is to combine sampling with lifted inference; we call this SafeSample, and it is an example of Rao-Blackwellized (or collapsed) sampling. We start with a definition:

**Definition 1.** Let \( \Phi \) be a sentence, and \( T \) be a set of relation names. We say that \( \Phi \) is safe modulo \( T \) if it becomes safe after making all relation names in \( T \) deterministic.

SafeSample is the naive Monte Carlo algorithm applied to the \([0,1]\)-valued function \( g \), instead of the 0/1-function \( f \), to compute \( P(\Gamma) \). It samples \( N \) possible world \( T^D, i = 1, N \), then returns the estimate

\[
\hat{y} = \sum_{i=1,N} g(T^D_i)/N
\]

\((2)\)

Where \( g(T^D) = P(\Gamma|T = T^D) = E_R[f|T = T^D] \), where \( T^D \) is a possible world for the relations \( T \).

**CondSample and ImportanceSample**

The main advantage of SafeSample is that it leads to dramatic speedup when combined with two additional optimizations: CondSample, where we compute the numerator and denominator of the conditional probability \( \text{Pr}(Q \land \Gamma) \) together (correlated sampling), and ImportanceSample, where we adjust the sampling distribution to increase the convergence of our estimate.

Besides empirically demonstrating that these techniques lead to efficient probabilistic inference in MLNs, in the full version of this paper we prove a novel convergence analysis of SlimShot’s sampling algorithms, leading to the bounds in Table 1. Our bounds rely on the output-tilt of a random variable. Chakraborty (Chakraborty et al. 2014) define the tilt of a Boolean formula as the ratio between the largest and smallest weight of any of its models. We define our output-tilt as follows:

**Definition 2.** The output-tilt of a random variable \( Y \ge 0 \) is \( T = \text{max } Y/ \text{min } Y \).
Monte Carlo | Safe sample | Cond sample | Importance sample |
---|---|---|---|
| \( \frac{1}{X_0^2} \) | \( \frac{\sigma^2(Y)}{y^2} \) | \( \frac{T}{x^2} \) | \( \frac{T'}{x^2} \) |

\[ X = P(Q | \Gamma, T^D), \quad x = E[X] \quad Y = P(\Gamma | T^D), \quad y = E[Y] \]

\[ T = \text{output-tilt of } Y \quad T' = \text{output-tilt of } Y' \]

Table 1: Number of simulation steps \( N \) to compute \( x = P(Q | \Gamma) \) with relative error \( \delta \).

Experiments

Datasets We used two datasets from the published literature. Smokers MLN (Singla and Domingos 2008) models a social network and the influence of friendship on smoking habits and lung cancer, while the Drinkers MLN (den Broeck et al. 2011) adds a new Drinks relation. SlimShot converts the MLNs to tuple-independent probabilistic databases by introducing a new relation name for each MLN rule with two or more literals. The Smokers MLN is safe modulo Smokes, while the Drinkers MLN is safe modulo Smokes and Drinks. Queries 1 and 2 ask for the probability of the Smokes and Cancer relations respectively. Experiments on the Drinkers MLN are included in the full paper.

We considered three variations on these datasets: symmetric, asymmetric unary, and asymmetric. In the first, all probabilities are given by the weights in (Singla and Domingos 2008). In the second, the binary relation Friends is symmetric while all unary relations have distinct, randomly-generated probabilities. Finally, in the asymmetric dataset the Friends relation is a randomly-generated graph with fixed fan-out 3, and edge probabilities randomly generated. The database applications of interest to us are captured by the third scenario (fully asymmetric), but we needed the first two in order to compute the exact probabilities (ground truth) for most experiments. No system to date can compute the exact probabilities for the asymmetric data.

MLN Systems We ran SlimShot using either CondSample only or using ImportanceSample, and report both results; we use “SlimShot” to refer to ImportanceSample. We compared to two popular MLN systems: Alchemy version 2.0 (Alchemy) and Tuffy version 0.4 (Niu et al. 2011). Both use MC-SAT for probabilistic inference (Poon and Domingos 2006), but they differ in how they perform grounding and their internal implementations of SampleSAT (Wei, Erenrich, and Selman 2004). In earlier work, the first author found several flaws in Tuffy’s implementation of MC-SAT and developed a modified version of Tuffy, currently available at the Allen Institute for Artificial Intelligence (AI2): it incorporates a new implementation of MC-SAT along with a number of other performance improvements such as elimination of redundant clauses. We refer to the two versions as Tuffy-Original and Tuffy-AI2.

All our experiments were conducted on a RHEL 7.1 system with 2xIntel Xeon E5-2407v2 (2.4GHz) processors and 48GB of RAM.

Accuracy We compared the accuracy of SlimShot to the other MLN systems on queries 1 and 2 over the Smokers MLN. We used unary asymmetric data, because we needed to compute the ground truth; we used a domain of size \( n = 100 \), resulting in 10200 random variables. Figure 1 shows the maximum relative error for all answers returned by the query, as a function of the number of iterations \( N \). The probability of the constraint, \( y = P(\Gamma) \) was around \( 10^{-10} \), while the query probability \( x = P(Q(a) | \Gamma) \) ranged between 0.04 and 0.9. In all experiments SlimShot (ImportanceSample) outperformed all others. For SlimShot we also measured the empirical tilt and report the number of iterations where the theoretical formula predicts that the probability of exceeding the relative error \( \delta = 0.1 \) is < 0.1: this is the empirical stopping condition used in SlimShot. In all cases, the stopping condition for ImportanceSample was around \( N = 1000 \) iterations.

Performance and Scalability

Figure 2 compares the runtime to achieve a fixed relative error. While for SlimShot we can derive a stopping condition, no stopping condition exists for MC-SAT. Instead, we allowed all systems to run until they achieve for all tuples a maximum

\(^3\)SlimShot’s translation to a probabilistic database introduced 10000 + 100 additional tuples.
relative error \( \leq 0.1 \) compared to the ground truth, and to maintain this for at least ten iterations: as before, we had to restrict to symmetric, and unary asymmetric, data. For both queries and both datasets, we can conclude that SlimShot converges faster than the other systems.

Third, we studied performance on asymmetric data, which is the main application scenario targeted by SlimShot: since we do not have the ground truth we reverted to reporting the runtime for a fixed number of iterations. Figure 3 shows that SlimShot is orders of magnitude faster than Alchemy and Tuffy over this type of sparse data: note that the scale is logarithmic. The reason for this is that SlimShot scales linearly with the number of probabilistic tuples present in the database. In contrast, Alchemy and Tuffy must include a unique MLN rule for each tuple missing from the Friends relation, expressing that it’s probability is zero: the runtime per sample increases quadratically with the domain size.

**Other Weighted Model Counters**

Since our approach reduces the query evaluation problem on MLNs to weighted model counting, as a ratio of two probabilities \( \frac{P(Q \land \Gamma)}{P(\Gamma)} \), we also attempted to compare SlimShot with state of the art general purpose Weighted Model Counting (WMC) systems.

A state of the art system for exact weighted model counting is WeightMC (Chakraborty et al. 2014), which is part of a recent and very promising line of work (Ermon et al. 2013; Chakraborty et al. 2014). We downloaded WeightMC, but unfortunately, we were only able to run it on a domain size of 3 before experiencing time-out errors.

Technical difficulties aside, general-purpose WMC tools do not appear well-suited for MLN inference: to approximate the ratio \( \frac{P(Q[a/x] \land \Gamma)}{P(\Gamma)} \) accurately requires extremely accurate approximations of each quantity individually, and one has to repeat this for every possible query answer \( a \).

**Discussion**

SlimShot is the only MLN system that can provide guaranteed accuracy: we have validated its accuracy on several symmetric and unary-asymmetric datasets (several omitted for lack of space). The theoretical stopping condition is sometimes overly conservative. SlimShot’s runtime performance per sample is comparable to other systems, however SlimShot converges much faster than the other systems. The main limitation of SlimShot is its dependency on the structure of logical formula of the MLN. The runtime suffers if two relations need to be sampled instead of one (while still being competitive). At an extreme, one can imagine an MLN where all relations need to be sampled, in which case SlimShot’s performance would degenerate. Full details on SlimShot may be found in the accompanying tech report, available at the authors’ homepages.

**References**


Boulos, J.; Dalvi, N. N.; Mandhani, B.; Mathur, S.; Ré, C.;
and Suciu, D. 2005. MYSTIQ: a system for finding more
answers by using probabilities. In SIGMOD, 891–893.

Braz, R.; Amir, E.; and Roth, D. 2005. Lifted first-order

Chakraborty, S.; Fremont, D.; Meel, K.; Seshia, S.; and Vardi,
M. 2014. Distribution-aware sampling and weighted model
counting for SAT. In AAAI, 1722–1730.

Chang, K.; Yih, W.; Yang, B.; and Meek, C. 2014. Typed tensor
decomposition of knowledge bases for relation extraction.
In EMNLP, 1568–1579.


den Broeck, G. V.; Taghipour, N.; Meert, W.; Davis, J.; and
Raedt, L. D. 2011. Lifted probabilistic inference by first-
order knowledge compilation. In IJCAI, 2178–2185.

den Broeck, G. V.; Meert, W.; and Darwiche, A. 2014.
Skolemization for weighted first-order model counting. In KR.


Zhang, C.; Govindaraju, V.; Borchardt, J.; Foltz, T.; Ré, C.;