SlimShot: In-Database Probabilistic Inference for Knowledge Bases *

Eric Gribkoff
University of Washington
eagribko@cs.washington.edu

Dan Suciu
University of Washington
suciu@cs.washington.edu

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 use variants of MCMC, which have no theoretical error guarantees, and, as we show, suffer 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 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.

1. INTRODUCTION

Motivation Knowledge Base Construction (KBC) [36] is the process of populating a structured relational database from unstructured sources: the system reads a large number of documents (Web pages, journal articles, news stories) and populates a relational database with facts. KBC has been popularized by some highly visible knowledge bases constructed recently, such as DBPedia [3], Nell [6], Open IE [16], Freebase [17], Google knowledge graph [37], Probase [42], Yago [21]. There is now huge demand for automating KBC, and this has lead to both industrial efforts (e.g. high-profile start-up companies like Tamr and Trifacta) and to several research prototypes like DeepDive [36], Tuffy [28], ProbKB [8], SystemT [27], and DBLife [35].

As explained in [36], a KBC engine performs two major tasks. The first is *grounding*, which evaluates a large number of SQL queries to produce a large database called a factor graph. While this was the most expensive task in early systems [2], recently Tuffy [28] and ProbKB [8] have significantly improved its performance by first formulating it as a query evaluation problem then developing sophisticated optimizations; we do not discuss grounding in this paper. The second task is *inference*, which performs statistical inference on the factor graph: the output consists of a marginal probability for every tuple in the database. All systems today use Markov Chain Monte Carlo (MCMC) for probabilistic inference [2, 28, 44], more specifically they use a variant called MC-SAT [31]. They suffer from two major problems. First, while MCMC converges in theory, the theoretical convergence rate is too slow for any practical purposes: current systems simply run for a fixed number of iterations and do not provide any accuracy guarantees. Second, in order to guarantee even those weak convergence rates, current implementations need to sample uniformly from a set of satisfying assignments, and for that they rely on SampleSAT [40], which, unfortunately, is only a heuristic and does not guarantee uniform samples. As a consequence, MC-SAT may not converge at all. In one KBC engine, statistical inference is reported to take hours on a 1TB RAM/48-core machine [36] and, as we show in this paper, accuracy is unpredictable, and often very bad. In short, probabilistic inference remains the major open challenge in KBC.

Our Contribution This paper makes novel contributions that pushes the probabilistic inference task inside a relational database engine. Our system takes as input a probabilistic database, a Markov Logic Network (MLN) defining a set of soft constraints (reviewed in Section 2), and a query, and returns the query answers annotated with their marginal probabilities. Our approach combines three ideas: lifted inference on probabilistic databases (developed both in the database community under the name *safe query evaluation* [39] and in the AI community [13]; reviewed in Subsection 2.1), a translation of MLN into a tuple-independent database (introduced in [24]; reviewed in Subsection 2.2), and a novel approach of combining lifted inference with sampling, for which we provide provable and practical convergence guarantees. To the best of our knowledge, SlimShot is the first system to completely push complex probabilistic inference inside the relational engine.

*This work was partially supported by NSF AITF 1535565 and IIS-1247469.

Proceedings of the VLDB Endowment, Vol. 9, No. 7

After the translation of the MLN, query evaluation becomes the problem of computing a conditional probability, $\mathbf{P}(Q|\Gamma)$, in a tuple independent probabilistic database; here Q is the user query, and Γ a constraint. A naive way to compute this probability would be to compute separately the numerator and denominator in $\mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma)$. If both Q and Γ are very simple expressions then both probabilities can be computed very efficiently using lifted inference, but in general lifted inference does not apply. An alternative is to estimate each of them using Monte Carlo simulation: sample N worlds and return the fraction that satisfy the formula. However, for guaranteed accuracy, one needs to run a number of simulation steps that is inversely proportional to the probability. Since Γ is a $\forall^*$ sentence, its probability is tiny, e.g. $10^{-9} - 10^{-20}$ in our experiments, which makes MC prohibitive. In other words, even though $\mathbf{P}(Q|\Gamma)$ is relatively large (say 0.1 to 0.9), if we compute it as a ratio of two probabilities then we need a huge number of steps because $\mathbf{P}(\Gamma)$ is tiny.

Our new idea is to combine sampling with lifted inference, in a technique we call SafeSample; this is an instance of a general technique called in the literature collapsed-particle importance sampling, or Rao-Blackwellized sampling [26]. SafeSample selects a subset of the relations such that, once these relations are made deterministic, the query is liftable, then uses MC to sample only the selected relations, and computes the exact query probability (using lifted inference) for each sample. Instead of estimating a 0/1-random variable, SafeSample estimates a random variable with values in $[0, 1]$, also called discrete integration.

To make SafeSample work, we introduce two key optimizations. CondSample evaluates the numerator and denominator of $\mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma)$ *together*, by using each sample to increment estimates for both the numerator and the denominator. This technique is effective only for a $[0, 1]$-estimator, like SafeSample. If applied to a 0/1-estimator, it becomes equivalent to *rejection sampling*, which samples worlds and rejects those that do not satisfy Γ: rejection sampling wastes many samples and is very inefficient. In contrast, SafeSample computes the *exact* probability of both $Q \wedge \Gamma$ and Γ at each sample, and thus makes every sample count. We prove a key theoretical result, Theorem 3.6, showing that the convergence rate of CondSample is inverse proportional to $\mathbf{P}(Q|\Gamma)$ and a parameter called *the output-tilt of* Γ, defined as the ratio between the largest and smallest value of the $[0, 1]$-function being estimated. In other words, our technique reduces the number of steps from an impractically large $1/\mathbf{P}(\Gamma)$ in a naive MC simulation, to a realistic $1/\mathbf{P}(Q|\Gamma)$ times the output-tilt. The second optimization, ImportanceSample, further decreases the output-tilt by weighting samples in inverse proportion to the probability of Γ.

Thus, SlimShot enables the entire probabilistic inference task of a KBC to be pushed entirely inside a SQL engine. In doing so, SlimShot disposes of the grounding task in KBC, and, instead, performs probabilistic inference by repeatedly computing a SQL query (corresponding to a safe plan), once for every sample. As explained earlier, query optimization techniques have been used before for the grounding task of KBC, but never for the probabilistic inference task: ours is the first system to push probabilistic inference in the database engine, and thus enables database optimization techniques to be applied to the probabilistic inference task. We describe several such optimizations, then validate SlimShot experimentally by comparing it with other popular MLN systems, and show that it has dramatically better accuracy at similar or better performance, and that it is the only MLN system that offers relative error guarantees; our system can compute a query over a database with 1M tuples in under 2 minutes.

Related Work 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 [11] (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 prove exponential lower bounds even for simple UCQ's whose probabilities are in PTIME. WeightMC [7] is part of a recent and very promising line of work [15, 7], which reduces approximate model counting to a polynomial number of oracle calls to a SAT solver. Adapting this techniques to weighted model counting is non-trivial: Chakraborty [7] proves that this is possible if the models of the formula have a small *tilt* (ratio between the largest and smallest weight of any model). The tilt 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 [30, 5, 38, 13, 20], 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.

In summary, our paper makes the following contributions:

- We describe SafeSample, a novel approach to query evaluation over MLNs that combines sampling with lifted inference, and two optimizations: CondSample and ImportanceSample. We prove an upper bound on the relative error in terms of the output-tilt. Section 3.
- We describe several optimization techniques for evaluating safe plans in the database engine, including techniques for negation, for tables with sparse content or sparse complement, and for evaluating constraints (hence CNF formulas). Section 4.
- We conduct a series of experiments comparing SlimShot to other MLN systems, over several datasets from the MLN literature, proving significant improvements in precision at similar, or better, runtime. Section 5

2. BACKGROUND

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

$$\Phi(\mathbf{x}) = E_1 y_1 E_2 y_2 \ldots E_\ell y_\ell \varphi(\mathbf{x}, \mathbf{y})$$

where each E_i is either $\forall$ or $\exists$ and φ is a quantifier-free formula using the logical variables $x_1, \ldots, x_k, y_1, \ldots, y_\ell$. 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 $Q = C_1 \vee C_2 \vee \cdots$ where each C_i is a conjunctive query with negation, while a constraints can be written as $\Delta_1 \wedge \Delta_2 \wedge \cdots$ where each Δ_i is a clause with quantifier prefix $\forall^*$.

Equivalence of queries $Q \equiv Q'$ is NP-complete [33], and, by duality, equivalence of constraints $\Gamma \equiv \Gamma'$ is coNP-complete.

2.1 Probabilistic Databases

A *tuple-independent probabilistic database*, or probabilistic database for short, is a pair $(\mathtt{DB}, p)$ where $p : \mathtt{DB} \to [0,1]$ is a function that associates to each tuple $t \in \mathtt{DB}$ 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 $D \subseteq \mathtt{DB}$, called a *possible world*, its probability is $\mathbf{P}_{\mathtt{DB},p}(D) = \prod_{t \in D} p(t) \cdot \prod_{t \in \mathtt{DB}-D}(1-p(t))$. The probability of a sentence Φ is:

$$\mathbf{P}_{\mathtt{DB},p}(\Phi) = \sum_{D \subseteq \mathtt{DB}: D \models \Phi} \mathbf{P}_{\mathtt{DB},p}(D)$$

If $Q(\mathbf{x})$ is a query, then its output is defined as the set of pairs $(\mathbf{a}, p)$, where $\mathbf{a}$ is a tuple of constants of the same arity as $\mathbf{x}$, and $\mathbf{a}$, $p \stackrel{\text{def}}{=} \mathbf{P}_{\mathtt{DB},p}(Q[\mathbf{a}/\mathbf{x}])$. We drop the subscripts from $\mathbf{P}_{\mathtt{DB},p}$ when clear from the context.

A relation R is called *deterministic*, if for every tuple $t \in R^{\mathtt{DB}}$, $p(t) \in \{0,1\}$, 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 $\mathbf{P}(\Phi)$ is #P-hard in the size of the active domain. The standard approach for computing $\mathbf{P}(\Phi)$ is to first *ground* Φ on the database $\mathtt{DB}$, which results in a Boolean formula called the *lineage* [1], then compute its probability; we do not use lineage in this paper and will not define it formally. If Φ is an $\exists^*$ sentence, then the lineage is a DNF formula, which admits an FPTRAS using Karp and Luby's sampling-based algorithm [25]. But the lineage of a $\forall^*$ sentences is a CNF formula, and does not admit an FPTRAS unless P=NP [32].

An alternative approach to compute $\mathbf{P}(\Phi)$ is called *lifted inference* in the Statistical Relational Learning literature [13], or *safe query evaluation* in probabilistic databases [39]. It always runs in PTIME in n, but only works for some sentences Φ. Following [39], lifted inference proceeds recursively on Φ, by applying the rules in Table 1, until it reaches ground atoms, whose probabilities are looked up in the database. Each rule can only be applied after checking a certain syntactic condition on the formula Φ; if no rule can be applied, lifted inference fails. For example, the *independent* $\forall$ rule says that $\mathbf{P}(\forall x \Phi) = \prod_{a \in A} \mathbf{P}(\Phi[a/x])$, but only if x is a *separator variable*, which is defined as a variable that occurs in every probabilistic atom, and, if two atoms have the same relational symbol, then it occurs in the same position in both. When the rules succeed we call Φ *safe*, or *liftable*; otherwise we call it *unsafe*. For a simple illustration, if T_d is a deterministic relation, then $\Gamma_1 = \forall x \forall y (R(x) \vee S(x,y) \vee T_d(y))$ is safe, because x is a separator variable, and therefore $\mathbf{P}(\Gamma_1) = \prod_{a \in A} \mathbf{P}(R(a) \vee S(a,y) \vee T_d(y))$, where $\mathbf{P}(R(a) \vee S(a,y) \vee T_d(y)) = 1 - (1 - \mathbf{P}(R(a))) \cdot \prod_{b \in A}(1 - \mathbf{P}(S(a,b))) \cdot (1 - \mathbf{P}(T_d(b)))$. On the other hand, if all three relations are probabilistic, then $\forall x \forall y (R(x) \vee S(x,y) \vee T(y))$ is #P-hard: we call it unsafe, or non-liftable. We refer the reader to [39] for more details on lifted inference. We note that in the literature the term lifted inference sometimes refers to symmetric databases [12]: 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.

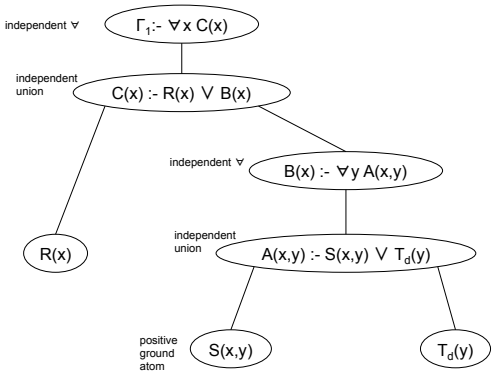


Figure 1: Safe plan for $\Gamma_1 = \forall x \forall y (R(x) \vee S(x,y) \vee T_d(y))$.

Safe plans. Following other systems [4, 29], SlimShot performs lifted inference by rewriting the query into a *safe query plan*, which is then evaluated inside a relational database engine. The leafs of the plan are relations with a special attribute called p, representing the probability of the tuple. There is one operator for each rule in Table 1, which computes the probabilities of the output in terms of the input. For example the independent join operator multiplies the probabilities of the left and the right operand, while the independent $\forall$ aggregates all probabilities in a group by multiplying them. We describe more details of the safe plans in Sec.4. For example, the query Γ_1 has the safe plan shown in Figure 1.

2.2 Markov Logic Networks

An MLN is a set of pairs $(w, \Delta(\mathbf{x}))$, where $\Delta(\mathbf{x})$ is a constraint with free variables $\mathbf{x}$, and $w \in [0, \infty]$ is a weight. If $w = \infty$ then we call Δ a *hard constraint*, otherwise a *soft constraint*. For example:

$$(3, \text{Smoker}(x) \wedge \text{Friend}(x,y) \Rightarrow \text{Smoker}(y)) \qquad (1)$$

is a soft constraint with weight $w = 3$. Notice that the constraint has free variables x, y, and the weight 3 is applied for every grounding of x, y where the constraint holds. Intuitively, it says that, typically, friends of smokers are smokers.

For a fixed domain A, a possible world D is a set of ground tuples over the domain A that satisfies all hard constraints. Its weight is computed as follows: for each soft constraint $(w, \Delta(\mathbf{x}))$ and for each tuple of constants $\mathbf{a}$ such that $\Delta(\mathbf{a})$

Φ	$\mathbf{P}(\Phi)$	Rule name	Conditions to check
$R(t)$	$p(R(t))$	Positive ground atom	–
$\neg R(t)$	$1 - p(R(t))$	Negated ground atom	–
$\Phi_1 \wedge \Phi_2$	$\mathbf{P}(\Phi_1) \cdot \mathbf{P}(\Phi_2)$	Independent join	no common probabilistic relation symbol in Φ_1, Φ_2
$\Phi_1 \vee \Phi_2$	$1 - (1 - \mathbf{P}(\Phi_1)) \cdot (1 - \mathbf{P}(\Phi_2))$	Independent union	no common probabilistic relation symbol in Φ_1, Φ_2
$\forall x\Phi$	$\prod_{a \in A} \Phi[a/x]$	Independent $\forall$	x is a separator variable (see text)
$\exists x\Phi$	$1 - \prod_{a \in A}(1 - \Phi[a/x])$	Independent $\exists$	x is a separator variable (see text)
$\Phi_1 \vee \Phi_2$	$\mathbf{P}(\Phi_1) + \mathbf{P}(\Phi_2) - \mathbf{P}(\Phi_1 \wedge \Phi_2)$	I/E for constraints	–
$\Phi_1 \wedge \Phi_2$	$\mathbf{P}(\Phi_1) + \mathbf{P}(\Phi_2) - \mathbf{P}(\Phi_1 \vee \Phi_2)$	I/E for queries	–

Table 1: Safe Query Evaluation Rules for $\mathbf{P}_{\mathtt{DB},p}(\Phi)$.

holds in D, multiply its weight by w:

$$W_{MLN}(D) = \prod_{(w,\Delta(\mathbf{x})) \in MLN, \mathbf{a} \in A^{|\mathbf{x}|} : w < \infty \wedge D \models \Delta[\mathbf{a}/\mathbf{x}]} w \quad (2)$$

For example, considering an MLN that consists only of the soft constraint (1), the weight of a possible world is 3^N, where N is the number of pairs a, b such that the implication $\text{Smoker}(a) \wedge \text{Friend}(a, b) \Rightarrow \text{Smoker}(b)$ holds in D. The probability of a possible world D is defined as the normalized weight: $\mathbf{P}_{MLN}(D) = W_{MLN}(D)/Z$, where $Z = \sum_D W_{MLN}(D)$. The probability of a query Q is $\mathbf{P}_{MLN}(Q) = \sum_{D:D \models Q} \mathbf{P}_{MLN}(D)$.

A tuple-independent probabilistic database is a special case of an MLN, where each tuple $R(\mathbf{a})$ becomes a soft constraint $(w, R(\mathbf{a}))$ with $w = p(R(\mathbf{a}))/(1 - p(R(\mathbf{a})))$.

State of the art. MLN's have been used in information extraction, record linkage, large scale text processing, and data collection from scientific journals [14, 28, 43]. MLN systems like Tuffy [28] and DeepDive [43] scale up by storing the *evidence* (hard constraints consisting of a single ground tuple) in a relational database system, and split query evaluation into two parts: grounding and probabilistic inference. Grounding is performed in the database engine [28], probabilistic inference is done entirely outside the database engine. Inference remains the major challenge to date: all MLN systems use MCMC, which, as we show in Section 5, can suffer from poor accuracy in practice.

Translation to Probabilistic Databases. Somewhat surprisingly, every MLN can be converted into a tuple-independent probabilistic database. One simple way to do this is to replace each soft rule $(w, \Delta(\mathbf{x}))$ with two new rules:

$$(w, R(\mathbf{x})) \qquad (\infty, \forall \mathbf{x} R(\mathbf{x}) \Leftrightarrow \Delta(\mathbf{x})) \quad (3)$$

where $R(\mathbf{x})$ is a new relational symbol (a new symbol for each rule), of the same arity as the free variables of $\Delta(\mathbf{x})$. After this transformation, the new MLN consists of the new tuple-independent relations $R(\mathbf{x})$, plus hard constraints of the form (3); denote by Γ the conjunction of all hard constraints. Let $\mathbf{P}_{MLN}$ and $\mathbf{P}$ be the probability space defined by the MLN, and by the tuple-independent probabilistic database. The following can be easily checked, for any query Q:

$$\mathbf{P}_{MLN}(Q) = \mathbf{P}(Q|\Gamma) = \mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma) \quad (4)$$

In other words, we have reduced the problem of computing probabilities in the MLN to the problem of computing a conditional probability over a tuple-independent probabilistic database. Notice that Γ is a $\forall^*$ sentence, hence $\mathbf{P}_{MLN}(Q)$ no longer admits an FPTRAS, because the grounding of a $\forall^*$ sentence is a CNF formula.

In this paper we use a more effective translation from MLN's to probabilistic databases, adapted from [24]: replace each soft rule $(w, \Delta(\mathbf{x}))$ by the following two new rules,

$$(w - 1, R(\mathbf{x})) \qquad (\infty, \forall \mathbf{x} \neg R(\mathbf{x}) \vee \Delta(\mathbf{x})) \quad (5)$$

The advantage of this translation is that, if Δ is a single clause (the typical case in MLN), the translated formula is also a single clause. Eq.(4) still holds for this translation. To see this, consider a world D over the vocabulary of the old MLN, and a tuple of constants, $\mathbf{a}$. If $D \not\models \Delta(\mathbf{a})$, then $\mathbf{a}$ does not contribute to the weight of D in Eq.(2): in the new MLN, the hard constraint (5) requires $R(\mathbf{a})$ to be false, and $\mathbf{a}$ also does not contribute any factor to the weight. If $D \models \Delta(\mathbf{a})$, then in the old MLN the constants $\mathbf{a}$ contributed a factor of w, and in the new world there are two possibilities $R(\mathbf{a})$ is true or is false, and these two worlds contribute jointly a weight $(w - 1) + 1 = w$.

2.3 Chernoff Bound and Monte Carlo Simulation

If $X_1, \ldots, X_N \in [0, 1]$ are i.i.d. with mean x, then Chernoff's Bound is [22]:

$$\mathbf{P}(\sum_{i=1,N} X_i \geqslant (1 + \delta)Nx) \leqslant \mathtt{exp}\left(-N \cdot D((1 + \delta)x||x)\right) \quad (6)$$

where $D(z||x) \stackrel{\text{def}}{=} z \cdot ln(\frac{z}{x}) + (1 - z) \cdot ln(\frac{1-z}{1-x})$ is the binary relative entropy. By using the inequality $D((1 + \delta)x||x) \geqslant x \cdot h(\delta)$, where $h(x) = (1 + x)ln(1 + x) - x$, and further using $h(\delta) \geqslant \delta^2/3$ for $\delta \leqslant 1/2$, the probability on the right simplifies to $\mathtt{exp}(-Nx\delta^2/3)$. All variants of Chernoff bounds discussed in this paper have both upper bounds ($\mathbf{P}(\sum X_i \geqslant \cdots)$) and lower bounds ($\mathbf{P}(\sum X_i \leqslant \cdots)$), with slightly different constants, but to simplify the presentation we follow common practice and discuss only the upper bound.

If f is a real-valued random variable, the Monte Carlo estimator for $x = \mathbf{E}[f]$ consists of computing N independent samples of f, denoted $X_1, \ldots, X_N$, then returning:

$$\hat{x} = \sum_{i=1,N} X_i/N \quad (7)$$

If $f \in [0, 1]$, then Chernoff's bound applies, and it follows that we need $N \gtrsim 1/(x\delta^2)$ samples (ignoring a small constant factor) in order for the estimator $\hat{x}$ to have relative error δ with high probability. In practice, of course, x is unknown, however Dagum [9] describes a dynamic stopping condition that guarantees the error bound δ, without knowing x. In

summary, the Monte Carlo estimator guarantees an error bound, requiring $1/(x\delta^2)$ simulation steps.

3. SlimShot

SlimShot takes as input a probabilistic database, a Markov Logic Network (MLN), and a query, and computes the query answers annotated with their marginal probabilities. It translates the MLN using Eq.(5), then computes $P_{MLN}(Q)$ as:

$$\mathbf{P}(Q|\Gamma) = \frac{\mathbf{P}(Q \wedge \Gamma)}{\mathbf{P}(\Gamma)} \quad (8)$$

We denote $x = \mathbf{P}(Q|\Gamma)$ throughout this section. The main contribution in SlimShot is a novel approach to computing Eq.(8), which consists of combining sampling with lifted probabilistic inference.

If both numerator and denominator in Eq.(8) are liftable, then x can be computed efficiently inside the database engine, as outlined in Subsection 2.1, but in general Γ is too complex and none of the two expressions are liftable. In that case, a naive approach is to estimate the numerator and denominator separately, using Monte Carlo sampling. Throughout this section we define f as the 0/1-function $f(D) = 1$ when $D \models \Gamma$, and $f(D) = 0$ otherwise, where D is a possible world, and denote $y = \mathbf{E}[f] = \mathbf{P}(\Gamma)$. A Monte Carlo estimator for $\mathbf{E}[f]$ is impractical because it requires $O(1/y)$ steps, and y is a tiny quantity ($10^{-9} - 10^{-20}$ in our experiments) because Γ is a $\forall^*$ sentence. In contrast, $x = \mathbf{P}(Q|\Gamma)$ is relatively large, say 0.1 to 0.9, but to compute it we need to divide two tiny quantities.

SlimShot makes three contributions. (1) It combines sampling with exact lifted inference (a technique called SafeSample), (2) it computes together the numerator and denominator in Eq.(8) (a technique called CondSample) and provides a novel theoretical accuracy guarantee, reducing the number of steps from $1/y$ to $1/x$ times the output-tilt; (3) it uses importance sampling to reduce the output-tilt and thus further improve the convergence rate (a technique called ImportanceSample).

3.1 SafeSample

SafeSample combines sampling with lifted inference. It is an application of the general principle of collapsed-particle or Rao-Blackwellized sampling [26], which replaces the estimation of 0/1-function f to a $[0,1]$-function g. We start with a definition:

Definition 3.1. *Let Φ be a sentence, and $\mathbf{T}$ be a set of relation names. We say that Φ is* safe modulo $\mathbf{T}$ *if it becomes safe after making all relation names in $\mathbf{T}$ deterministic.*

Throughout this section we denote $g(\mathbf{T}^D) = \mathbf{P}(\Gamma|\mathbf{T} = \mathbf{T}^D) = \mathbf{E}_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]$, where $\mathbf{T}^D$ is a possible world for the relations $\mathbf{T}$. If Γ is safe modulo T, then the function $g(\mathbf{T}^D)$ can be computed in polynomial time. For example the constraint $\forall x \forall y(R(x) \vee S(x,y) \vee T(y))$ is unsafe, but it is safe modulo T, because once we make T deterministic, the constraint becomes safe; the function $g(T^D)$ is computed by the safe plan in Figure 1. We will usually denote T a relation in $\mathbf{T}$, and denote R any other relation[1]

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

$$\hat{y} = \frac{\sum_{i=1,N} g(\mathbf{T}^{D_i})}{N} \quad (9)$$

This is unbiased, because $\mathbf{E}_{\mathbf{T}}[g] = \mathbf{E}_{\mathbf{T}}[\mathbf{E}_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]] = \mathbf{E}[f] = y$.

The advantage of SafeSample over a naive MC is that it samples from the smaller space of relations $\mathbf{T}$ as opposed to the entire space of possible worlds. Concretely, this leads to a reduction in the variance: by Rao-Blackwell's theorem we know that the variance cannot increase, but in fact we can show exactly by how much it decreases, namely by the quantity $\sigma^2(f) - \sigma^2(g) = \mathbf{E}[f^2] - \mathbf{E}_{\mathbf{T}}[g^2] = \mathbf{E}_{\mathbf{T}}[\mathbf{E}_{\mathbf{R}}[f^2|\mathbf{T} = \mathbf{T}^D]] - \mathbf{E}_{\mathbf{T}}[\mathbf{E}^2_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]] = \mathbf{E}_{\mathbf{T}}[\sigma^2_{\mathbf{R}}[f|\mathbf{T} = \mathbf{T}^D]] \geqslant 0$. In other words, it decreases by the variance in $\mathbf{R}$ (since we no longer sample $\mathbf{R}$ but use lifted inference instead); the only variance that remains is due to $\mathbf{T}$. To see how the variance affects the number of simulation steps, we prove:

Proposition 3.2. *Let $g \geqslant 0$ be a random variable s.t. $g \leqslant c$ from some constant c, with mean $y = \mathbf{E}[g]$ and variance $\sigma^2 = \mathbf{E}[g^2] - \mathbf{E}^2[g]$. Let $\hat{y}$ be the estimator in Eq.(9). Then, for all $\delta \leqslant \sigma^2/(2cy)$:*

$$\mathbf{P}(\hat{y} \geqslant N(1+\delta)y) \leqslant 2\,\mathtt{exp}(-\frac{N\delta^2 y^2}{3\sigma^2})$$

Proof. Bennett's theorem states that, if $X_1, \ldots, X_N$ are iid's s.t. $|X_i| \leqslant c$ with mean 0 and variance σ^2, then $\mathbf{P}(\sum_i X_i \geqslant t) \leqslant \mathtt{exp}(-\frac{N\sigma^2}{c^2} h(\frac{ct}{N\sigma^2}))$. By setting $X_i = g(D_i) - y$, $t = N \cdot \delta \cdot y$ we obtain $\mathbf{P}(\hat{y} \geqslant N(1+\delta)y) \leqslant \mathtt{exp}(-N\frac{\sigma^2}{c^2} h(\frac{ct}{N\sigma^2})) = \mathtt{exp}(-N\frac{\sigma^2}{c^2} h(\frac{c\delta y}{\sigma^2}))$, and finally we use the fact that $h(x) \geqslant x^2/3$ for $0 \leqslant x \leqslant 1/2$ (Subsection 2.3). □

It follows that the number of steps required by (9) to estimate y with an error $\leqslant \delta$ is $N \gtrsim 3\sigma^2/(y^2\delta^2)$, and therefore SafeSample is faster than a 0/1-Monte Carlo estimator by a factor equal to the ratio of the variances, $\sigma^2(f)/\sigma^2(g)$, which is always $\geqslant 1$ (since $\sigma^2(f) \geqslant \sigma^2(g)$).

We illustrate with two examples: one where SafeSample has an exponentially large speedup, the other where it plateaus at a small constant factor.

Example 3.3. *Consider $\Gamma = \forall x(R(x) \vee T(x))$, and a symmetric probabilistic database over a domain of size n, where, for all $i \in [n]$, the tuple $R(i)$ has probability $p(R(i)) = r$ and $T(i)$ has probability t. We show that the speedup $\sigma^2(f)/\sigma^2(g)$ grows exponentially with the domain size. We have $\mathbf{P}(\Gamma) = \mathbf{E}[f] = \mathbf{E}[f^2] = (r + t - rt)^n$, hence:*

$$\sigma^2(f) = (r + t - rt)^n - (r + t - rt)^{2n}$$

If T^D has size $|T^D| = n - k$, then $g(T^D) = \mathbf{E}[f|T = T^D] = r^k$, which implies $\mathbf{E}_T[g] = \sum_k \binom{n}{k} t^{n-k}(1-t)^k r^k = (t + (1-t)r)^n$, and similarly $\mathbf{E}_T[g^2] = (t + (1-t)r^2)^n$, or

$$\sigma^2(g) = (t + (1-t)r^2)^n - (t + (1-t)r)^{2n}$$

When $r = t = 1/2$, then the variance decreases from $\sigma^2(f) = (3/4)^n - (9/16)^n = (12^n - 9^n)/16^n$ to $\sigma^2(g) = (5/8)^n - (3/4)^{2n} = (10^n - 9^n)/16^n$. Their ratio $\sigma^2(f)/\sigma^2(g) = (12^n - 9^n)/(10^n - 9^n) \approx (6/5)^n$.

Example 3.4. *Consider now $\Gamma = \forall x \forall y(R(x) \vee S(x,y) \vee T(y))$. As before, we consider for illustration a symmetric*

[1]Suggesting de**T**erministic and **R**andom, although the relations $\mathbf{T}$ are *not* deterministic.

database, where $R = T = [n]$, $S = [n] \times [n]$, and the tuples in R, S, T have probabilities r, s, t respectively. We show that here the speedup is only a small constant factor. We sample T, and let R, S be the random relations. If $|T^D| = n-k$, then $g(T^D) = \mathbf{P}(\Gamma|T^D) = (r+s^k(1-r))^n$, because for every value of the variable $x = i$, the sentence $\forall y \in [k](R(i) \vee S(i,y))$ must be true. Thus, we have

$$\mathbf{P}(\Gamma) = \mathbf{E}[f^2] = \sum_{k=0,N} \binom{n}{k} t^{n-k}(1-t)^k (r + s^k(1-r))^n$$

$$\mathbf{E}_T[g^2] = \sum_{k=0,N} \binom{n}{k} t^{n-k}(1-t)^k (r + s^k(1-r))^{2n}$$

Here the decrease in variance is only by a constant factor because, if we group the terms in $\mathbf{E}[f^2] - \mathbf{E}_T[g^2]$ by k, then for each $k > 0$, the difference $(r+s^k(1-r))^n - (r+s^k(1-r))^{2n}$ is $\leqslant (r + s^k(1-r))^n(1-r)(1-s)$. That, is except for the first term $k = 0$ (whose contribution to the sum is negligible), all others decrease by a factor of at most $(1-r)(1-s)$.

As the last example suggest, SafeSample alone is insufficient, which justifies our second technique.

3.2 CondSample

CondSample computes the numerator and denominator in Eq.(8) together. We prove that CondSample requires a number of steps proportional to $1/x$ times the *output-tilt* of g, which is the ratio of the largest and smallest values of g. Given a set of relations $\mathbf{T}$ such that both $Q \wedge \Gamma$ and Γ are safe modulo $\mathbf{T}$, CondSample estimates Eq.(8) by the following quantity:

$$\hat{x} = \frac{\sum_{i=1,N} \mathbf{P}(Q \wedge \Gamma|\mathbf{T}^{D_i})}{\sum_{i=1,N} \mathbf{P}(\Gamma|\mathbf{T}^{D_i})} \tag{10}$$

For any fixed N, $\hat{x}$ is a biased estimator of x; however, $\hat{x}$ converges to x when $N \to \infty$ (it is called a *consistent* estimator in the literature).

Chakraborty [7] define the *tilt* of a Boolean formula as the ratio between the largest and smallest weight of any of its models. We adapt their terminology to a random variable:

Definition 3.5. *The* output-tilt *of a random variable $Y \geqslant 0$ is $T = \max Y / \min Y$.*

We prove:

Theorem 3.6. *Let $(X_1, Y_1), \ldots, (X_N, Y_N)$ be i.i.d. such that for all i, $X_i \in [0,1]$ and has mean $x = \mathbf{E}[X_i]$, and $Y_i \geqslant 0$. Let T be the output-tilt of Y_i. Then:*

$$\mathbf{P}\left(\frac{\sum_{i=1,N} X_i Y_i}{\sum_{i=1,N} Y_i} \geqslant (1+\delta)x\right) \leqslant \mathtt{exp}(-ND/T) \tag{11}$$

where $D = D((1+\delta)x||x)$.

Proof. We use the following lemma:

Lemma 3.7. *Let $y_1, \ldots, y_N \geqslant 0$ be N real numbers, and let $M = (\sum_i y_i)/(\max_i y_i)$; notice that $M \leqslant N$. Let $X_1, \ldots, X_N$ be i.i.d.'s, where each $X_i \in [0,1]$ and has mean $x = \mathbf{E}[X_i]$. Then, for any $\delta > 0$:*

$$\mathbf{P}(\sum_i X_i y_i > (1+\delta)x(\sum_i y_i)) \leqslant \mathtt{exp}(-M \cdot D)$$

(where $D = D((1+\delta)x||x)$).

Intuitively, the lemma generalizes two extreme cases: when all weights y_i are equal, then $M = N$ and the bound becomes the Chernoff bound for N items; and when the weights are as unequal as possible, $y_1 = y_2 = \ldots = y_M$, $y_{M+1} = \ldots = y_N = 0$, then the bound becomes the Chernoff bound for M items. The proof is included in the tech report for this paper.

To prove the theorem, we condition on the outcomes of the variables Y_i, then apply the lemma:

$$\begin{aligned}\mathbf{P}(\cdots) =& \mathbf{E}_{Y_1,\ldots,Y_N}\left[\mathbf{P}_{X_1,\ldots,X_N}\left(\frac{\sum_{i=1,N} X_i Y_i}{\sum_{i=1,N} Y_i} \geqslant (1+\delta)x\right)\right] \\ \leqslant& \mathbf{E}_{Y_1,\ldots,Y_N}[\mathtt{exp}(-(\sum_{i=1,N} Y_i)/(\max_{j=1,N} Y_j) \cdot D)] \\ \leqslant& \mathbf{E}_{Y_1,\ldots,Y_N}[\mathtt{exp}(-N/T \cdot D)] = \mathtt{exp}(-N/T \cdot D)\end{aligned}$$

proving the claim. □

By setting $X_i = \mathbf{P}(Q|\Gamma, T^{D_i})$ and $Y_i = \mathbf{P}(\Gamma|T^{D_i})$, Eq.(10) becomes $\sum_i X_i Y_i / \sum_i Y_i$, which gives us the error bound (11) for the estimator $\hat{x}$. It suffices to run $N \gtrsim T/D((1+\delta)x||x) \approx 3T/(x\delta^2)$ simulation steps ($D((1+\delta)x||x) \gtrsim x\delta^2/3$, Subsection 2.3), in other words the number of steps depends on the mean x of X_i and the output-tilt T of Y_i; it does not depend on the mean y of Y_i. In Theorem 3.6 X_i, Y_i do not have to be independent[2], which justifies using the same sample $\mathbf{T}^{D_i}$ both in the numerator and the denominator.

The reader may wonder why we don't estimate $x = \mathbf{P}(Q|\Gamma)$ directly, as $\sum_i \mathbf{P}(Q|\Gamma, T^{D_i})/N$, which would only require $N \gtrsim 3/(x\delta^2)$ steps. The problem is that we need to sample possible worlds $T^{D_1}, T^{D_2}, \ldots$ from the *conditional* distribution $\mathbf{P}(T^D|\Gamma)$, a task as difficult as estimating $\mathbf{P}(\Gamma)$ [23].

The speedup given by Theorem 3.6 is only possible for a $[0,1]$-random variable; a 0/1-variable has output-tilt ∞, and the theorem becomes vacuous. In that case CondSample becomes *rejection sampling* for computing $\mathbf{P}(Q|\Gamma)$: repeatedly sample a world D_i, ignore worlds that do not satisfy Γ, and return the fraction of worlds that satisfy Q, which is known to require $N \gtrsim 1/\mathbf{P}(\Gamma)$ steps, because we need as many steps in expectation to hit Γ once. Rejection sampling wastes simulation steps. In contrast, SafeSample *makes every sample count*, by using lifted inference to compute $\mathbf{P}(\Gamma|\mathbf{T}^D)$ exactly, no matter how small, and requires $N \gtrsim T/\mathbf{P}(Q|\Gamma)$ steps.

We show two examples, one where the output-tilt is large ($T \gtrsim 1/\mathbf{P}(\Gamma)$), and CondSample is not much better than SafeSample, the second where the output-tilt is $T \ll 1/\mathbf{P}(\Gamma)$.

Example 3.8. *Consider first $\Gamma = \forall x \forall y(R(x) \vee S(x,y) \vee T(y))$ in Example 3.4. As we have seen, if $|T^D| = n-k$, then $Y = \mathbf{P}(\Gamma|T^D) = (r + s^k(1-r))^n$, (because for every value of the variable $x = i$, the sentence $\forall y \in [k](R(i) \vee S(i,y))$ must be true). The maximum value of Y is 1 (for $k = 0$) and the minimum is $(r + s^n(1-r))^n \approx r^n$ respectively, thus the output-tilt is $T = 1/r^n \gtrsim 1/\mathbf{P}(\Gamma) = 1/\sum_k (r+s^k(1-r))^n$. In general, when $\max(Y) = 1$, then the output-tilt is $1/\min(Y)$ and this is bigger than $1/\mathbf{E}[Y]$ because the $\min(Y) \leqslant \mathbf{E}[Y] \leqslant \max(Y)$.*

Example 3.9. *Consider next $\Gamma = \forall x \forall y(R_1(x) \vee S_1(x,y) \vee T(y)) \wedge (R_2(x) \vee S_2(x,y) \vee \neg T(y))$. This constraint is safe modulo T, but now we no longer have $\mathbf{P}(\Gamma|T^D) = 1$, for any value T^D, and we show that the output-tilt is much*

[2]But (X_i, Y_i) has to be independent of (X_j, Y_j), for $i \neq j$.

smaller than $1/\mathbf{E}[Y]$*. Notice that, one can show (by repeating the argument in Example 3.4) that SafeSample alone is insufficient to compute this query much faster than a naive Monte Carlo simulation, so CondSample is necessary for a significant speedup. To compute the output-tilt, note that* $Y = \mathbf{P}(\Gamma|T^D) = (r_1 + s_1^k(1-r_1))^n(r_2 + s_2^{n-k}(1-r_2))^n$ *and, assuming* $r_1 = r_2$ *and* $s_1 = s_2$*, the maximum/minimum values are* $(r + s^n(1-r))^n \approx r^n$ *(for* $k = 0$ *or* $k = n$*) and* $(r + s^{n/2}(1-r))^{2n} \approx r^{2n}$ *(for* $k = n/2$*) respectively. The output-tilt is* $T = 1/r^n \ll 1/\mathbf{E}[Y]$*. To see this, assume for illustration that* $t = 1/2$*, then we claim that* $\mathbf{E}[Y] \leqslant 3r^{2n}$*. Expand* $\mathbf{E}[Y] = \sum_k \binom{n}{k}\frac{1}{2^n}(r+s^k(1-r))^n(r+s^{n-k}(1-r))^n$*, and split the sum into two regions: for* $k \in [n(1-\delta)/2, n(1+\delta)/2]$ *the contribution of the sum is* $\leqslant (r+s^{n(1-\delta)/2}(1-r))^{2n} \approx r^{2n}$*, while for* $k \notin [n(1-\delta)/2, n(1+\delta)/2]$ *the contribution of the sum is*[3] *is* $\leqslant 2r^n \exp(-n\delta^2)$*. It suffices to choose* δ *such that* $e^{-\delta^2} \leqslant r$ *(which is possible when* $r > 1/e \approx 0.36$*) to prove our claim.*

The examples suggest that CondSample works best for complex MLNs, where no setting of the relations $\mathbf{T}$ can make Γ true (and, thus, $\max Y \ll 1$); still, it is insufficient for speeding up all queries. Our third technique further improves the convergence rate by adding importance sampling.

3.3 ImportanceSample

Importance sampling [10] chooses a proposal distribution for the random variable $\mathbf{T}$, $\mathbf{P}'(\mathbf{T})$, then computes the expected value $\mathbf{E}'[g']$ of a corrected function, $g'(\mathbf{T}^D) = g(\mathbf{T}^D) \cdot \mathbf{P}(\mathbf{T}^D)/\mathbf{P}'(\mathbf{T}^D)$'. This is an unbiased estimator: to see this, we apply directly the definition of $\mathbf{E}[g]$ as a sum over all possible worlds:

$$\mathbf{E}[g] = \sum_{T^D} g(\mathbf{T}^D)\mathbf{P}(\mathbf{T}^D) = \sum_{T^D} g'(\mathbf{T}^D)\mathbf{P}'(\mathbf{T}^D)$$

Ideally we would like to set $\mathbf{P}'(\mathbf{T}^D) \sim g(\mathbf{T}^D) \cdot \mathbf{P}(\mathbf{T}^D)$, because in that case g' is a constant function with output-tilt 1, but computing this $\mathbf{P}'$ is infeasible, because the normalization factor is precisely the quantity we want to compute, $\sum_{\mathbf{T}^D} g(\mathbf{T}^D) \cdot \mathbf{P}(\mathbf{T}^D) = \mathbf{E}[g]$.

Instead, we define $\mathbf{P}'(\mathbf{T}^D)$ as a function of the cardinalities of the relations T^D in $\mathbf{T}^D$. We first describe a naive ImportanceSample, assuming $\mathbf{T} = \{T\}$ consists of a single relation. For every $k = 1, n^{\mathtt{arity}(T)}$ (recall: n is the size of the active domain), sample one relation T^{D_k} of size k, and compute $p_k = \mathbf{P}(\Gamma|T^{D_k})$. Let $q = \sum_k \binom{n^{\mathtt{arity}(T)}}{k} p_k$ be the normalization factor, and define proposal distribution $\mathbf{P}'(T^D) \stackrel{\text{def}}{=} p_k/q$, where $k = |T^D|$. Intuitively, this decreases the output-tilt of g', because the spread of probabilities $\mathbf{P}(\Gamma|T^D)$ decreases if we fix the cardinality of T^D. In fact, we prove that in the special case of *symmetric* databases, the output-tilt becomes 1. Symmetric structures have been studied in the AI literature motivated by their applications to MLNs [13, 12].

Proposition 3.10. *If the probabilistic database is symmetric, and* Γ *is safe modulo a set of unary relations, then the output-tilt of* g' *is 1.*

[3]Let Z_i be i.i.d. in $\{0,1\}$ s.t. $\mathbf{P}(Z_i = 0) = \mathbf{P}(Z_i = 1) = 1/2$. Then the following stronger version of Chernoff's bound holds: $\mathbf{P}(\sum Z_i \geqslant (1+\delta)n/2) \leqslant e^{-n\delta^2}$. Thus, $\sum_{k=n(1+\delta)/2,n} \binom{n}{k}\frac{1}{2^n} \leqslant e^{-n\delta^2}$.

The proof follows from the observation that, if T is a unary relation, then in a symmetric database $\mathbf{P}(\Gamma|T^D)$ depends only on the cardinality of T^D.

To reduce the output-tilt on asymmetric databases, we optimize ImportanceSample as follows. First, we transform all non-sampled relations $\mathbf{R}$ into symmetric relations, by setting $\mathbf{P}(R(\mathbf{a}))$ to the average probability of all tuples in R. Then we compute $p_k = \mathbf{P}(\Gamma||T^D| = k)$: we compute the latter probability exactly, even if the relations $\mathbf{T}$ are not symmetric, by adapting techniques from lifted inference over symmetric databases [12]. Notice that, when all relations are symmetric, then the optimized ImportanceSample coincides with the naive ImportanceSample.

Example 3.11. *Continuing Example 3.8, ImportanceSample computes* $p_k = \mathbf{P}(\Gamma||T^D| = k) = (r + s^{n-k}(1-r))^n$*, for each* $k = 0, n$*. Define* $q = \sum_k \binom{n}{k} p_k$*. The proposal distribution is* $\mathbf{P}(T^D) = p_{|T^D|}/q$*, and the corrected function is* $g'(T^D) = (r + s^{n-k}(1-r))^n t^k (1-t)^{n-k}/p_k$*. It each iteration step* $i = 1, N$*, SlimShot samples a value* $k = 0, n$ *with probability* $\binom{n}{k}p_k/q$*, then uses reservoir sampling to sample a set* T^{D_i} *of cardinality* k*, and computes* $X_iY_i = \mathbf{P}(Q \wedge \Gamma|T^{D_i})$ *and* $Y_i = \mathbf{P}(\Gamma|T^{D_i})$ *using lifted inference, adding the quantities to Eq.(11). The value of* Y_i *is constant (it is always* q*), because the relations* R, S, T *are symmetric; if the query* Q *contains any non-symmetric relations, then* X_iY_i *is a random variable, and Eq.(11) converges to* x *after* $N \gtrsim 3/(x\delta^2)$ *steps; if* Q *uses only symmetric relations, then* X_iY_i *is also a constant, and Eq.(11) converges after 1 step.*

We note that ImportanceSample is, again, only possible in combination with SafeSample. If g were a 0/1-random variable, then the corrected function g' is also a 0/1-random variable, and its output-tilt remains infinity.

3.4 Summary

To summarize, Algorithm 1 first converts DB and the MLN to a tuple-independent database (Subsection 2.2), chooses the relations $\mathbf{T}$ s.t. $Q \wedge \Gamma$ and Γ are $\mathbf{T}$-safe, precomputes safe plans for $\mathbf{P}(Q \wedge \Gamma|\mathbf{T}^D)$ and $\mathbf{P}(\Gamma|\mathbf{T}^D)$ (Subsection 2.1), and precomputes the proposal distribution p_k (Subsection 3.3). Next, it computes Q using Eq.(10), by repeatedly sampling deterministic relations $\mathbf{T}^D$ and adding $\mathbf{P}(Q \wedge \Gamma|\mathbf{T}^D)$ and $\mathbf{P}(\Gamma|\mathbf{T}^D)$ to the numerator and denominator (it uses the safe plans and computes them in the SQL engine). It stops either after a fixed number of iterations steps N, or after $N = 3\tilde{T}/(x\delta^2)$ steps, where $\tilde{T}$ is the empirical output tilt (ratio of largest to smallest value of $\mathbf{P}(\Gamma|\mathbf{T}^D)$). SlimShot currently supports proposal distributions only for unary relations $\mathbf{T}$: otherwise, it samples $\mathbf{T}$ directly from the distribution $\mathbf{P}$.

4. SYSTEM ARCHITECTURE

SlimShot is written in Python and relies on Postgres 9.3 as its underlying query processing engine. Any other database engine could be used, as long as it supports standard query processing capabilities: inner and outer join, group by, random number generation, and mathematical operators such as sum and logarithm. The MLN is given in text file containing first-order $\forall^*$ sentences with associated weights. SlimShot converts these rules offline into hard constraints Γ and a set of new probabilistic tables (Subsection 2.2, Eq.(4)). The new relations are materialized in Postgres, with tuple weights converted to probabilities ($p = w/(1+w)$), stored in an additional attribute.

Algorithm 1 SlimShot

Input: DB, MLN, Q
Output: $\mathbf{P}_{MLN}(Q)$
1: Convert DB, MLN to DB', Γ (Eq. 4)
2: Select $\mathbf{T}$ s.t. both $Q \wedge \Gamma$ and Γ are $\mathbf{T}$-safe
3: Compute safe plans for $\mathbf{P}(Q \wedge \Gamma | \mathbf{T}^D)$, $\mathbf{P}(\Gamma | \mathbf{T}^D)$
4: Compute $p_{\mathbf{k}}$ (Subsection 3.3)
5: `num` = `denom` = 0
6: **For** $i = 1$ **to** N **do** (see text for N)
7: Sample $\mathbf{T}^D$ with $\mathbf{P}'(\mathbf{T}^D) \stackrel{\text{def}}{=} p_{\mathbf{k}}$, where $\mathbf{k} = |\mathbf{T}^D|$
8: `num`+ = $\mathbf{P}(Q \wedge \Gamma) \cdot \mathbf{P}(\mathbf{T}^D)/\mathbf{P}'(\mathbf{T}^D)$ (use safe plan)
9: `denom`+ = $\mathbf{P}(\Gamma) \cdot \mathbf{P}(\mathbf{T}^D)/\mathbf{P}'(\mathbf{T}^D)$ (use safe plan)
10: **Return** x = `num`/`denom`

SlimShot supports unions of conjunctive queries, but in typical MLN applications the query Q consists of a single relation name, e.g. $Q(x) =$ `Smokes`(x), and the inference engine returns the per-tuple probabilities of all tuples in the `Smokes` relation.

4.1 Choosing the relations T

The choice of relations $\mathbf{T}$ s.t. both $Q \wedge \Gamma$ and Γ are $\mathbf{T}$-safe is done by brute force enumeration: in all our experiments, the cost of this step was negligible.

4.2 Review of Safe Plan Evaluation

The operators of a safe plan correspond to the eight rules in Table 1. The first two, positive/negative atoms, are the leafs of the plan, all others are the unary or binary operators. The entire plan is the converted into a SQL query that manipulates probabilities explicitly, and then evaluated in postgres. For example, a join multiplies probabilities $p_1 p_2$, a union computes $1 - (1 - p_1)(1 - p_2)$, while the independent $\forall$ and $\exists$ are group-by queries returning probabilities $\prod_i p_i$ and $1 - \prod_i (1 - p_i)$ respectively. For example, referring to Figure 1, if we assumed that the relations R and B have the same sets of tuples, then *independent-union* $R(x) \vee B(x)$ is:

```
select R.x, 1-(1-R.p)*(1-B.p) as p
from R join B on R.x = B.x
```

where B is a sub-query. But since R, B may have different sets of tuples, SlimShot uses an outer-join instead.

4.3 Enhanced Safe Plan Evaluation

We discuss here several enhancements/optimizations.

Product aggregate Unfortunately, most popular database engines do not have a product-aggregate operator $\prod_i p_i$. [19, 18]. We considered two options. First is to express it using logarithm and sum, as `exp`$(\text{sum}_i(\log p_i))$. This requires a slightly more complex logic to correctly account for tuples with probability zero, or close to zero, or for missing tuples. The second is to define a user-defined aggregate function, UDA.

Independent Union If implemented as suggested above, independent union requires a query with outer joins and two `case` statement to account for missing/present tuples. Instead, we simulate it using a group-by and aggregate, for example the SQL expression above becomes:

```
select T.x, 1-prod(1-T.P)
from (R union B) as T group by T.x
```

Missing Tuples The MLN semantics is based on the standard active domain semantics, where the answer to an expression like $S(x, y) \vee T_d(y)$ in Figure 1 includes all tuples (a, b) where a is any constant in the domain, and $b \in T_d$. MLN implementations support this semantics naively, by simply representing explicitly the entire active domain, such that every relation of arity k contains all n^k tuples. Missing tuples have probability 0, and after a negation their probability becomes 1. Since our goal is to deploy SlimShot in database applications, representing explicitly all tuples in the active domain is prohibitive: instead we allow tuples to be missing, and treat them specially depending on the context. For example, in a join like $R(x) \wedge B(x)$, a missing B-tuple is considered to have probability 0 and simply not included: the SQL query is a standard join. However, in the query $R(x) \wedge \neg B(x)$, a missing B-tuple must be treated like a tuple with probability 1: the SQL query is now a left outer-join, with a case statement to compute the output probability as either $p_1(1 - p_2)$ or p_1, depending on whether the value x is present in B or not.

Batch processing To reduce the number of calls to the database system, we grouped multiple simulation steps into one. More precisely, we generate b samples $\mathbf{T}^{D_i}, i = 1, b$ and compute all probabilities $\mathbf{P}(Q \wedge \Gamma | \mathbf{T}^{D_i})$ for $i = 1, b$ using a single SQL query, and similarly for $\mathbf{P}(\Gamma | \mathbf{T}^{D_i}), i = 1, b$. For that, we added a new column to all relations in $\mathbf{T}$ representing the sample number, and modified the safe plan to return $\bigcup_{i=1}^{b} \{i\} \times \Gamma(\mathbf{T}^{D_i})$.

4.4 Further Optimizations

We briefly mention here other optimizations in SlimShot. *Generic Constants:* this refers to computing the probability for all query answers using a single query plan. That is, we have a single query plan to compute $\mathbf{P}(Q(x)|\Gamma)$, returning all pairs (a, p), rather than the naive way of iterating over all constants a in the domain and computing $\mathbf{P}(Q[a/x]|\Gamma)$. We note that MC-SAT algorithm used by existing MLN systems already obtains the probabilities of all outputs x at the same time. *QRel* refers to an optimization that is possible when the query relation Q is a member of the sampled relation set $\mathbf{T}$. In the case we can avoid computing $\mathbf{P}(Q \wedge \Gamma | \mathbf{T}^{D_i})$ (since it is either 0 or 1): instead we only need to compute $\mathbf{P}(\Gamma | \mathbf{T}^{D_i})$ and check if $\mathbf{T}^{D_i} \models Q$.

5. EXPERIMENTS

We validated SlimShot through a series of experiments comparing its performance to other MLN systems on several datasets reported in the literature. We addressed the following questions. How accurate are the probabilities computed by SlimShot compared to existing systems? How does its runtime performance compare to that of existing systems? How does SlimShot handle more complex sets of MLN rules? How effective are the optimizations in SlimShot? And how does SlimShot compare to other, general-purpose weighted model counters?

Datasets We used two datasets from the published literature, Table 2, and three queries, Table 3. Smokers MLN [38] models a social network and the influence of friendship on smoking habits and lung cancer, while the Drinkers MLN [13] adds a new *Drinks* relation. SlimShot converts the MLNs to tuple-independent probabilistic databases by introducing a new relation name for each rule in Table 2 with two or

MLN	Constraint	w
Smokers	$\neg Smokes(x)$	1.4
	$\neg Cancer(x)$	2.3
	$\neg Friends(x,y)$	4.6
	$Smokes(x) \Rightarrow Cancer(x)$	1.5
	$Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y)$	1.1
Drinkers	$Drinks(x) \wedge Friends(x,y) \Rightarrow Drinks(y)$	1.1

Table 2: The Smokers MLN and the Drinkers MLN.

MLN	Query
Smokers and Drinkers	$Q1(x) :- Smokes(x)$
	$Q2(x) :- Cancer(x)$
Drinkers only	$Q3(x) :- Drinks(x)$

Table 3: Experiment Queries

more literals. The Smokers MLN is safe modulo *Smokes*, while the Drinker MLN is safe modulo *Smokes* and *Drinks*.

We considered three variations on these datasets: symmetric, asymmetric unary, and asymmetric. In the first, all probabilities are given by the weights in Table 2. 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. We used datasets up to 1M tuples.

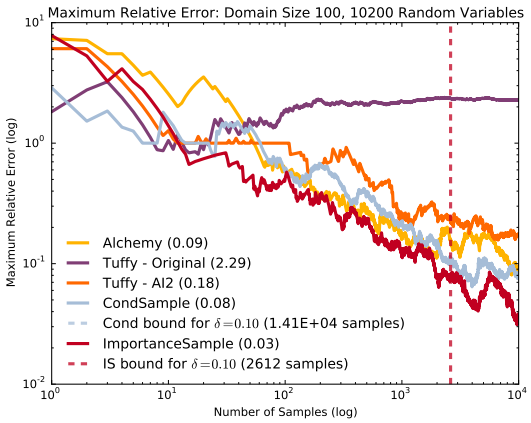
MLN Systems We ran SlimShot using either CondSample only or using ImportanceSample, and report both results; we use "SlimShot" to refer to ImportanceSample. We compared to two popular MLN systems: Alchemy version 2.0 [2] and Tuffy version 0.4 [28]. Both use MC-SAT for probabilistic inference [31], but they differ in how they perform grounding and their internal implementations of SampleSAT [40]. In earlier work, the first author found several flaws in Tuffy's implementation of MC-SAT, the foremost being a failure to perform simulated annealing steps to explore the solution space before returning a sample within the SampleSAT code, 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 eliminaton 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.

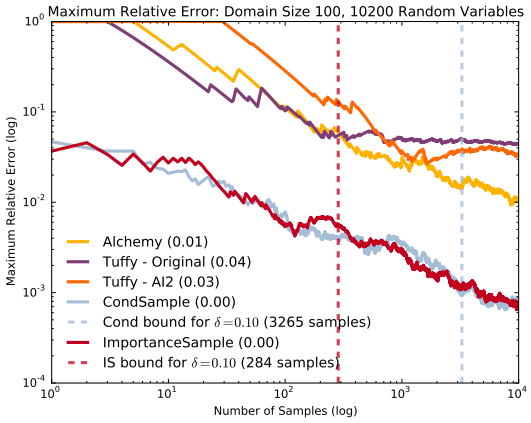
5.1 Accuracy

We compared the accuracy of SlimShot to the other MLN systems on queries 1 and 2 over the Smokers MLN. We used only symmetric and unary asymmetric data, because we needed to compute the ground truth; we used a domain of size $n = 100$, resulting in 10200 random variables[4]. Figure 2

[4]SlimShot's translation to a probabilistic database introduced $10000 + 100$ additional tuples.



(a) Query 1, Asymmetric Unary



(b) Query 2, Asymmetric Unary

Figure 2: Maximum Relative Error on the Smokers MLN

shows the maximum relative error[5] for all answers returned by the query, as a function of the number of iterations N on unary asymmetric data (symmetric results are similar and omitted due to space constraints). The probability of the constraint, $y = \mathbf{P}(\Gamma)$ was around 10^{-10} while the query probability $x = \mathbf{P}(Q(a)|\Gamma)$ 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 (11) 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. On symmetric data CondSample had a larger tilt, leading to a much worse stopping condition; $\mathbf{P}(\Gamma)$ is less evenly distributed over possible samples for the lower average tuple probabilities in the symmetric data, and CondSample ends up in regions of very small probability for most of its samples.

[5]We also measured the mean Kullback-Leibler (KL) divergence. The overall conclusions remain the same, but we found KL to be too forgiving by hiding grossly inaccurate probabilities for some tuples.

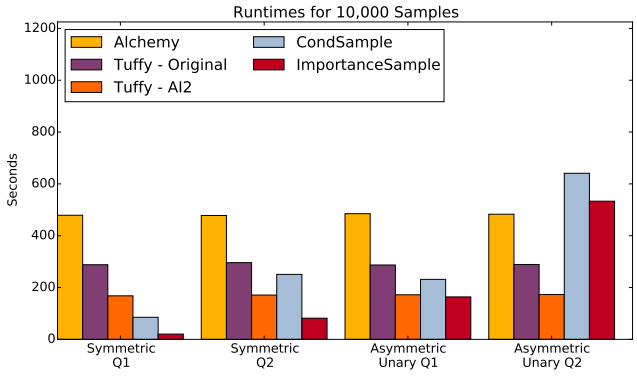


Figure 3: Absolute runtimes on 10K random variables.

Figure 4: Runtimes (log scale) as a function of accuracy with 10K random variables. Systems which fail to achieve a max rel error of 0.1 are annotated with their final max rel error.

5.2 Performance and Scalability

Next, we conducted three rounds of experiments to compare SlimShot's runtime performance to the other systems. Figure 3 shows the runtime for a fixed number of samples ($N = 10,000$), on queries 1 and 2 over the symmetric and unary asymmetric Smokers MLN (same as in the previous section) with 10,000 random variables; here and in the next experiment we stopped at $10,000$ random variables because Alchemy and Tuffy did not scale well beyond that. The fact that all binary relations are complete puts SlimShot at a disadvantage: like any relational plan, the safe plans in SlimShot benefit from sparse relations. In contrast, one simulation step in MC-SAT is rather cheap, favoring Alchemy and Tuffy. Nevertheless, in our experiments the runtime per iteration in SlimShot was within the same order of magnitude as the most efficient system, sometimes better.

Second, Figure 4 compares the (much more meaningful) runtime to achieve a fixed relative error. While for SlimShot we can derive a stopping condition from Eq.(11), no stopping condition exists for MC-SAT. Instead, we allowed all systems to run until they achieve for all tuples a maximum relative error $\leqslant 0.1$ 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.

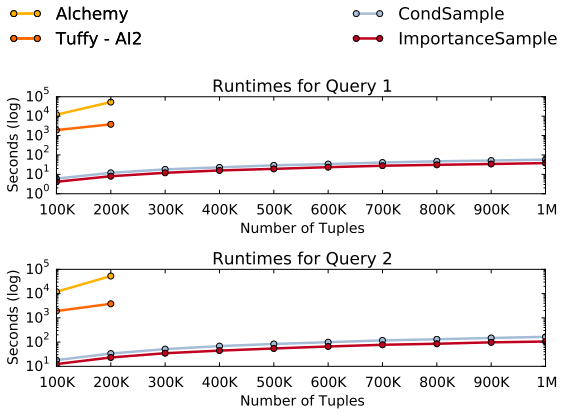


Figure 5: Runtimes for 100 iterations as a function of size in an asymmetric Smokers MLN.

Third, we studied performance of SlimShot on much larger datasets, up to 1,000,000 random variables. Unlike previous datasets, where the data was symmetric or unary asymmetric only, here we used fully asymmetric data, which is the typical scenario targeted by SlimShot. Since we do not have the ground truth for such data, we reverted to reporting the runtime for a fixed number of iterations. Figure 5 shows that, even for the largest dataset with 1M random variables over a domain of size 1000, SlimShot computed the output probabilities of all 1000 answers in about 100 seconds. Here ImportanceSample is more efficient than CondSample because it favors samples of small size, resulting in slightly better runtime for the SQL queries. We note that SlimShot was orders of magnitude faster than Alchemy and Tuffy (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. While Tuffy-AI2 is optimized for deterministic variables, there is still significant overhead compare to SlimShot.

5.3 Richer MLNs

Next, we evaluated SlimShot on a richer MLN: the Drinkers MLN [13], on up to 100,000 random variables. SlimShot must now simultaneously sample two unary relations, *Smokes* and *Drinks*, which slows down the computation of the proposal distribution. The results for a fixed number of iterations on asymmetric data are shown in Figure 6. While we do not have ground truth for asymmetric data, the experiments of the previous two sections strongly suggest that ImportanceSample is the only system that returns accurate results, so the runtime performance numbers should be interpreted in that light. Here the runtime is dominated by the time to compute the proposal distribution, which takes $O(n^3)$ steps, because it needs to compute a probability for each combination of three cardinalities, $|Smokes \cap Drinks|, |Smokes - Drinks|, and |Drinks - Smokes|$. While the proposal distribution is independent of the query and could be computed and stored offline, but in all our experiments we report it as part of the total runtime. As a consequence, ImportanceSample was slower than CondSample. We note

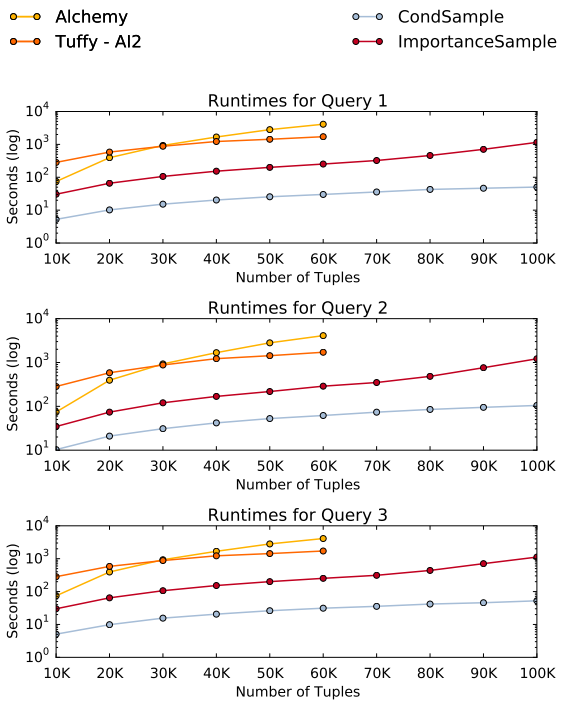


Figure 6: Runtimes for 100 iterations as function of size in the Smokers-Drinkers MLN using asymmetric data.

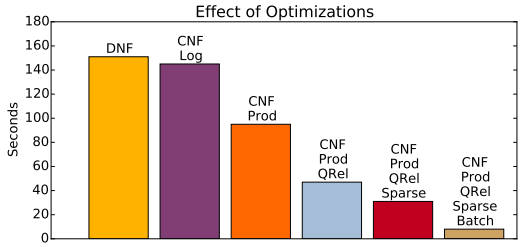


Figure 7: The runtime for 1,000 iterations of SlimShot with progressively more optimizations enabled.

that Tuffy-AI2 implements certain logical simplifications, keeping the size of the Smokers-Drinkers network equivalent to that of the Smokers network, improving its performance.

5.4 Impact of Optimizations

As we developed our system we progressively added optimizations, sometimes replacing rather naive first implementations. We report their effect in Figure 7 on Query 1, over a domain size 100 and asymmetric, but dense data. *Generic constants:* We actually started by computing a non-Boolean query $Q(x)$ as in theory textbooks: for each constant a in the domain, compute $\mathbf{P}(Q[a/x]|\Gamma)$: switching to a safe plan that computes all output probabilities in one query improved the runtime by more than two orders of magnitude. All runtimes in the figure use generic constants. *DNF:* Our first implementation used standard safe plans for UCQ [39], by expressing $\mathbf{P}(\Gamma) = 1 - \mathbf{P}(\neg\Gamma)$. Since $\mathbf{P}(\neg\Gamma)$ is very close to 1.0, it required Postgres's numeric data type to achieve sufficient precision. This first column shows this runtime.

Domain Size (Number of Variables)	Runtime in Seconds
3 (15)	1.3
5 (35)	1.8
8 (80)	11.0
10 (120)	205.5
15 (255)	Did not finish

Table 4: Runtimes for SDDs on Q1 over the Smokers MLN with symmetric data

CNF: Implementing specific operators for CNF reduced the runtime to the second column. Here we used logarithm to express the product aggregate in terms of sum. *Product:* Replacing the log-sum-exp with a UDA for product reduced the runtime by 35% (third column). *QRel:* if the query happens to be the sampled relation, then we can avoid computing the second query $\mathbf{P}(Q \wedge \Gamma|\mathbf{T}^{D_i})$, but instead simply check whether $\mathbf{T}^{D_i} \models Q$. This reduces the runtime by half. *Sparse:* Next, we show the benefit of adding extra logic to the SQL query to omit tuples with probability 0. Note that the dataset used here is dense: the savings comes entirely from the sampled *Smokes* relation. Significant additional savings occur on sparse data. *Batch:* Finally, the incorporation of batched sampling decreases runtimes by a factor of 2x-10x.

5.5 Other Weighted Model Counters

Since our approach reduces the query evaluation problem on MLNs to weighted model counting, as a ratio of two probabilities $\mathbf{P}(Q \wedge \Gamma)/\mathbf{P}(\Gamma)$, 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 uses Sentential Decision Diagrams [11] (SDDs). They arose from the field of knowledge compilation, and compile a Boolean formula into circuit representations s.t. WMC can be done in linear time in the size of the circuit. SDDs have state-of-the-art performance for many tasks in exact weighted model counting. We use SDD v1.1 [34] and report runtime results in Table 4. While SDDs have been reported in the literature to scale to much larger instances, they fared worse on the formulas resulting from grounding MLNs.

A state of the art system for approximate weighted model counting is WeightMC [7], which is part of a recent and very promising line of work [15, 7]. We downloaded WeightMC from [41], 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 $\mathbf{P}(Q[a/x] \wedge \Gamma)/\mathbf{P}(\Gamma)$ accurately requires extremely accurate approximations of each quantity individually, and one has to repeat this for every possible query answer a.

5.6 Discussion

SlimShot is the only MLN system that can provide guaranteed accuracy: we have validated its accuracy on several symmetric and unary-symmetric 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.

6. CONCLUSION

We have described SlimShot, a system that computes queries over large Markov Logic Networks. The main innovation in SlimShot is to combine sampling with lifted inference. This reduces the sample space, and thus reduces the variance, and also enables two additional techniques: estimation of a conditional probability and importance sampling. The lifted inference is performed entirely in the database engine, by evaluating safe plans. We have described several optimizations that improve the performance of safe plans. Our experiments have shown that SlimShot returns significantly better results than other MLN engines, at comparable or better speed.

One limitation of SlimShot is that it only works if the query and constraint can be made safe by determinizing a small number of relation names. In extreme cases that use a single relational predicate name, like the transitivity constraint $E(x,y) \wedge E(y,z) \Rightarrow E(x,z)$, SlimShot degenerates to a naive Monte Carlo evalution. Future work includes studying how SlimShot can be extended to such cases, for example by partitioning the database.

7. REFERENCES

[1] S. Abiteboul, O. Benjelloun, and T. Milo. The active XML project. *VLDB J.*, 2008.

[2] `http://alchemy.cs.washington.edu/`.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. Dbpedia: A nucleus for a web of open data. In *ISWC 2007 + ASWC 2007*, pages 722–735, 2007.

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

[5] R. Braz, E. Amir, and D. Roth. Lifted first-order probabilistic inference. In *IJCAI*. Citeseer, 2005.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

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

[8] Y. Chen and D. Z. Wang. Knowledge expansion over probabilistic knowledge bases. In *SIGMOD*, 2014.

[9] P. Dagum, R. Karp, M. Luby, and S. Ross. An optimal algorithm for monte carlo estimation (extended abstract). In *FOCS*, 1995.

[10] A. Darwiche. *Modeling and Reasoning with Bayesian Networks*. Cambridge University Press, 2009.

[11] A. Darwiche. SDD: A new canonical representation of propositional knowledge bases. In *IJCAI*, 2011.

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

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

[14] P. M. Domingos and D. Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool Publishers, 2009.

[15] S. Ermon, C. Gomes, A. Sabharwal, and B. Selman. Taming the curse of dimensionality: Discrete integration by hashing and optimization. In *ICML*, pages 334–342, 2013.

[16] A. Fader, S. Soderland, and O. Etzioni. Identifying relations for open information extraction. In *EMNLP*, pages 1535–1545, 2011.

[17] https://www.freebase.com/.

[18] W. Gatterbauer and D. Suciu. Oblivious bounds on the probability of boolean functions. *ACM Trans. Database Syst.*, 39(1):5, 2014.

[19] W. Gatterbauer and D. Suciu. Approximate lifted inference with probabilistic databases. *PVLDB*, 8(5):629–640, 2015.

[20] E. Gribkoff, G. Van den Broeck, and D. Suciu. Understanding the complexity of lifted inference and asymmetric weighted model counting. In *UAI*, 2014.

[21] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artif. Intell.*, 194:28–61, 2013.

[22] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. *ECCC*, 17:72, 2010.

[23] M. Jerrum, L. G. Valiant, and V. V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[24] A. K. Jha and D. Suciu. Probabilistic databases with markoviews. *PVLDB*, 5(11):1160–1171, 2012.

[25] R. Karp and M. Luby. Monte-carlo algorithms for enumeration and reliability problems. In *FOCS*, pages 56–64, 1983.

[26] D. Koller and N. Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[27] Y. Li, F. Reiss, and L. Chiticariu. Systemt: A declarative information extraction system. In *ACL*, pages 109–114, 2011.

[28] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.

[29] D. Olteanu, J. Huang, and C. Koch. SPROUT: lazy vs. eager query plans for tuple-independent probabilistic databases. In *ICDE*, pages 640–651, 2009.

[30] D. Poole. First-order probabilistic inference. In *IJCAI*, volume 3, pages 985–991. Citeseer, 2003.

[31] H. Poon and P. Domingos. Sound and efficient inference with probabilistic and deterministic dependencies. In *AAAI*, pages 458–463, 2006.

[32] D. Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2):273–302, 1996.

[33] Y. Sagiv and M. Yannakakis. Equivalences among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.

[34] `http://reasoning.cs.ucla.edu/sdd/`.

[35] W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *VLDB*, 2007.

[36] J. Shin, S. Wu, F. Wang, C. D. Sa, C. Zhang, and C. Ré. Incremental knowledge base construction using deepdive. *PVLDB*, 8(11):1310–1321, 2015.

[37] A. Singhal. Introducing the knowledge graph: things, not strings. *Official Google Blog, May*, 2012.

[38] P. Singla and P. Domingos. Lifted first-order belief propagation. In *AAAI*, pages 1094–1099, 2008.

[39] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[40] W. Wei, J. Erenrich, and B. Selman. Towards efficient sampling: Exploiting random walk strategies. In *IAAI*, 2004.

[41] `http://www.cs.rice.edu/CS/Verification/Projects/WeightGen/`.

[42] W. Wu, H. Li, H. Wang, and K. Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *SIGMOD*, 2012.

[43] C. Zhang, V. Govindaraju, J. Borchardt, T. Foltz, C. Ré, and S. Peters. Geodeepdive: statistical inference using familiar data-processing languages. In *SIGMOD*, 2013.

[44] C. Zhang and C. Ré. Towards high-throughput gibbs sampling at scale: a study across storage managers. In *SIGMOD*, pages 397–408, 2013.