Scalable Probabilistic Databases with Factor Graphs and MCMC

Michael Wick
University of Massachusetts
Computer Science
140 Governor's Drive
Amherst, MA
mwick@cs.umass.edu

Andrew McCallum
University of Massachusetts
Computer Science
140 Governor's Drive
Amherst, MA
mccallum@cs.umass.edu

Gerome Miklau
University of Massachusetts
Computer Science
140 Governor's Drive
Amherst, MA
miklau@cs.umass.edu

ABSTRACT

Probabilistic databases play a crucial role in the management and understanding of uncertain data. However, incorporating probabilities into the semantics of incomplete databases has posed many challenges, forcing systems to sacrifice modeling power, scalability, or restrict the class of relational algebra formula under which they are closed. We propose an alternative approach where the underlying relational database always represents a single world, and an external factor graph encodes a distribution over possible worlds; Markov chain Monte Carlo (MCMC) inference is then used to recover this uncertainty to a desired level of fidelity. Our approach allows the efficient evaluation of arbitrary queries over probabilistic databases with arbitrary dependencies expressed by graphical models with structure that changes during inference. MCMC sampling provides efficiency by hypothesizing *modifications* to possible worlds rather than generating entire worlds from scratch. Queries are then run over the portions of the world that change, avoiding the onerous cost of running full queries over each sampled world. A significant innovation of this work is the connection between MCMC sampling and materialized view maintenance techniques: we find empirically that using view maintenance techniques is several orders of magnitude faster than naively querying each sampled world. We also demonstrate our system's ability to answer relational queries with aggregation, and demonstrate additional scalability through the use of parallelization.

1. INTRODUCTION

A growing number of applications output large quantities of uncertain data. For example, sensor networks produce imprecise readings and information extraction systems (IE) produce errorful relational records. Despite their inevitable inaccuracies, these types of automatic prediction systems are becoming increasingly important. This is evident by the sheer number of repositories culled from the web by IE systems: CiteSeer, REXA, DbLife, ArnetMiner, and Google Scholar. Probabilistic databases (PDBs) are a natural framework for storing this uncertain output, but unfortunately most current PDBs do not achieve the difficult balance of expressivity and efficiency necessary to support such a range of scalable real-world structured prediction systems.

Indeed, there is an inherent tension between the expressiveness of a representation system and the efficiency of query evaluation. Many recent approaches to probabilistic databases can be characterized as residing on either pole of this continuum. For example, some systems favor efficient query evaluation by restricting modeling power with strict independence assumptions [8, 9, 1]. Other systems allow rich representations that render query evaluation intractable for a large portion of their model family [15, 29, 24, 25]. In this paper we combine graphical models and MCMC sampling to provide a powerful combination of expressive freedom and efficient query evaluation over arbitrary relational queries.

Graphical models are a widely used framework for representing uncertainty and performing statistical inference in a myriad of applications, including those in computational biology [26], natural language processing [16], computer vision [30], information extraction [21], and data integration [33]. These models are becoming even more accessible with the proliferation of many general purpose probabilistic programming languages [23, 19, 17]. Factor graphs are a particular type of representation for graphical models that serve as an umbrella framework for both Bayesian networks and Markov random fields, and are capable of representing any exponential family probability distribution.

Unfortunately, graphical models have been largely overlooked as a choice for representing uncertainty in probabilistic databases. In rare cases, connections have been drawn between graphical models and PDBs [15, 7], and only recently have they been used explicitly in either the representation [29], or in the mechanism for query evaluation [24, 25]. However, these systems are in practice severely limited by the $\#\mathcal{P}$-hard problem of query evaluation and would not scale to the types of sophisticated models and large data crucial to many real-world problems [21, 6, 33].

We distinguish ourselves from these lines of work in several important ways. First, we directly address the problem of intractable query evaluation and propose an approximate any-time approach that scales both to dense factor graphs and large amounts of data. Second, we avoid the issue of closing factor graph semantics under relational algebra operators giving us the ability to evaluate any relational algebra query (including aggregation). Third, we evaluate our approach on a difficult real-world information extraction problem on which exact statistical inference is intractable (even in the graphical model framework). We are able to achieve this with a query evaluation technique based on MCMC sampling. This is in contrast to previous sampling approaches, which use traditional *generative* Monte Carlo methods [5, 13]. The Monte Carlo sampling method of MCDB [13] requires knowing the normalization

.

constant for each function; unfortunately, for general factor graphs this problem is as difficult as computing marginals (#$\mathcal{P}$-hard). On the other hand MCMC samplers hypothesize local changes to worlds, avoiding the need to know the normalizer. Additionally, MCMC enables us to track tuples affected by local changes and we exploit this information to efficiently re-evaluate the queries—avoiding the need to re-run the full query from scratch over each sampled world.

Indeed we demonstrate query evaluation on such a factor graph (where computing the normalization constant is intractable) and show that our MCMC sampler based on view maintenance techniques reduces running time by several orders of magnitude over the simple approach of running the full query over each hypothesized world. We also empirically demonstrate our ability to scale these intractable models to large datasets with tens of millions of tuples and show further scalability through parallelization. Finally, we demonstrate our evaluator's ability to handle aggregate queries.

After introducing related work, the rest of the paper is organized as follows: first we describe our representation, introduce factor graphs, and use information extraction as a running pedagogical example and application of our approach (although it more generally applies to other problems that can be modeled by factor graphs). We then introduce query evaluation techniques, including the materialized view maintenance approach advocated in this paper. Finally, we present experimental results demonstrating scalability to both large data and highly correlated PDBs.

2. RELATED WORK

Because early theoretical work on incomplete data focuses largely on algebras and representation systems (e.g., [3, 12]), it was only natural to extend this line of thinking to probabilities [2, 34, 15, 10, 8]. However, this extension is quite difficult since the probabilities in query results must include *expressions* derived from the confidence values originally embedded in the database. Systems meeting these theoretical conditions must overcome a set of challenges that are often satisfied at the expense of modeling-power or understandability.

Although there is a vast body of work on probabilistic databases, graphical models have largely been ignored until recently. The work of Sen et al. [24, 25] casts query evaluation as inference in a graphical model and BayesStore [29] makes explicit use of Bayesian networks to represent uncertainty in the database. While expressive, generative Bayesian networks have difficulty representing the types of dependencies handled automatically in discriminative models [16], motivating a database approach to linear chain conditional random fields [28]. We, however, present a more general representation based on factor graphs, an umbrella framework for both Bayesian networks and conditional random fields. Perhaps more importantly we directly address the problem of scalable query evaluation in these representations—with an MCMC sampler—whereas previous systems based on graphical models are severely restricted by this bottleneck. Furthermore our approach can easily evaluate any relational algebra query without the need to close the graphical model under the semantics of each operator.

There has also been recent interest in using sampling methods to estimate tuple marginals or rankings. For example, the MystiQ [5] system uses samplers to estimate top-k rankings [22]. Joshi and Jermaine apply variance reduction techniques to obtain better sample estimates [14]. MCDB [13] employs a generative sampling approach to hypothesize possible worlds. However, these approaches are based on feed-forward Monte Carlo techniques and therefore cannot take advantage of the Markovian nature of MCMC methods. The MCDB system does use the concept of "tuple bundles" to exploit overlap across possible worlds, but this approach is difficult to implement because it requires custom query optimization code and redefining operators over bundles of tuples (requiring over 20,000 lines of C++ code; in contrast our approach is able to treat the DBMS as a blackbox and still exploit overlap between samples). Furthermore, MCDB requires an additional pre-processing step to compute the overlap. In MCMC sampling, the overlap is determined automatically as a byproduct of the procedure. This allows our method to employ ideas from DBMS view materialization technology to take advantage of the overlap between possible worlds. To the best of our knowledge, we are the first Markov-chain Monte Carlo sampler for estimating probabilities in probabilistic databases [31]. We are also the first to combine graphical models and sampling techniques into a single cohesive probabilistic database representation system.

3. REPRESENTATION

In our approach, the underlying relational database always stores a single possible world (a setting to all the random variables), enabling us to run any relational algebra query. Database objects such as fields, tuples, and attributes represent random variables, however, the factor graph expresses complex statistical relationships between them. As required, we can recover uncertainty to a desired level of fidelity through Markov chain Monte Carlo (MCMC), which hypothesizes changes to random variable values that represent samples of possible worlds. As this underlying database changes, we execute efficient queries on the modified portions of worlds and obtain an increasingly accurate approximation of the probabilistic answers. Another advantage of a graphical model approach is that it enables automatic learning over the database—avoiding the need to tune weights by hand.

We begin by describing factor graphs and the well known *possible worlds* semantics, where the uncertain database is a set of possible worlds W, and each $w \in W$ is a deterministic instance of the uncertain DB. Following tradition, we endow W with a probability distribution $\pi : W \rightarrow [0, 1]$ s.t. $\sum_{w \in W} \pi(w) = 1$, yielding a distribution over possible worlds.

3.1 Factor Graphs

In our approach π is encoded by a factor graph, a highly expressive representation that can encode any exponential family probability distribution (including Bayesian and Markov networks). Indeed their success in areas such as natural language processing, protein design, information extraction, physics, and machine vision attest to their general representational power. Factor graphs can succinctly capture relationships between random variables with complex dependencies, making them a natural choice for relational data.

Mathematically, a factor graph (parametrized by θ) is a bipartite graph whose nodes consist of the pair $\mathcal{G}_\theta = \langle V, \Psi \rangle$ where $V = X \cup Y$ is the set of random variables: X is the set of observed variables, and Y is the set hidden variables; $\Psi = \{\psi_k\}$ is the set of factors.

Random Variables

Intuitively, random variables represent the range of values that an uncertain object in the database may acquire. Each hidden variable $Y_i \in Y$ is associated with a domain DOM(Y_i) representing the range of possible values for Y_i. For example, the domain could be binary {yes,no}, enumerations {tall, grande, venti} or real-valued $\{r \in \Re | r \geq 4\}$. Observed variables are fixed to a particular value in the domain and can be considered a constant. For simplicity, and without loss of generality, we will assume that random variables are scalar-valued (vector and set valued variables can be re-written

as a combination of factors and variables).

In our notation, capital letters with a subscript (e.g., Y_i, X_i) represent a single random variable, and lowercase letters (e.g., y_i) represent a value from the corresponding variable's domain: $y_i \in \text{DOM}(Y_i)$. We use the notation $X_i = x_i$ to indicate that variable X_i is taking on the value x_i. Finally, we use superscripts to denote sets (of arity represented by the superscipt): the notation $X^r = x^r$ means the set of variables $\{X_i, X_{i+1}, \cdots, X_{i+r}\}$ take on the values $(X_i = x_i, X_{i+1} = x_{i+1}, \cdots, X_{i+r} = x_{r+1})$ where it is implicitly assumed that x_i is a value from X_i's domain. Capital letters without a subscript refer to the entire variable space (Y is all hidden variables and X is all observables).

Factors

Factors model dependencies between the random variables. In fact, multiple factors may govern the behavior of the same variable by expressing preferences for certain assignments to that variable over others. This flexible overlapping structure is powerful for modeling real world relational data.

Formally, each factor $\psi : x^m \times y^n \rightarrow \Re_+$ maps *assignments* to subsets of observed variables $x^m \subseteq \text{DOM}(X)$ and hidden variables $y^n \subseteq \text{DOM}(Y)$ to a non-negative real-valued scalar. Intuitively, factors measure the compatibility of settings to a group of variables, providing a measurement of the uncertainty that the particular assignment contributes to the world. For an example, see Figure 1.

Typically, factors are computed as a log-linear combination of a sufficient statistic (or feature function) ϕ_k and corresponding parameter θ_k as $\psi_k(x^m, y^n) = \exp(\phi_k(x^m, y^n) \cdot \theta_k)$. Where ϕ are user-specified features for representing the underlying data and θ are corresponding real-valued weights measuring each features impact. There are a number of available methods from machine learning and statistics for automatically determining these weights (avoiding the need for manual tuning).

Given the above definitions, the factor graph $\mathcal{G}_\theta$ expresses a probability distribution (parametrized by θ, and conditioned on X) $\pi_\mathcal{G} : X \times Y \rightarrow [0, 1]$ s.t. $\sum_{y \in \text{DOM}(Y)} \pi_\mathcal{G}(y|x) = 1$. More specifically, if the graph decomposes into a set of factors Ψ (where each $\psi \in \Psi$ has a factor-specific arity of $s + t$) then the probability distribution $\pi_\mathcal{G}$ is given as:

$$\pi_\mathcal{G}(Y = y|X = x; \theta) = \frac{1}{Z_X} \prod_{\psi \in \Psi} \psi(y^s, x^t) \tag{1}$$

where $Z_X = \sum_{y \in Y} \prod_{k=1}^{n} \psi_k(y^s, x^t)$ is an input-dependent normalizing constant ensuring that the distribution sums to 1. Note two special cases: if X is empty then $\mathcal{G}$ is a Markov random field, and when factors are locally normalized $\mathcal{G}$ is a Bayesian network.

3.2 Possible Worlds

An uncertain database $\mathcal{D}$ is a set of relations $R = \{R_i\}$ each with schema S_i^k (of arity k) containing attributes $R_i.a_1, \cdots, R_i.a_k$. Each attribute is equipped with a finite domain $\text{DOM}(R_i.a_1)$ (a field is certain if its value is known, otherwise it is uncertain). A deterministic tuple t for relation R_i is a realization of a value for each attribute $t = \langle v_1, \cdots, v_k \rangle$ for constants $v_1 \in \text{DOM}(a_1) \cdots v_k \in \text{DOM}(a_k)$. Let T be the set of all such tuples for all such relations in the database. Then the set of *all* (unrestricted) worlds realizable by this uncertain database is $W_\mathcal{D} = \{w \mid w \subseteq T\}$.

Let each field in the database be a random variable whose domain is the same as the field's attribute's domain. A deterministic field is an observed variable X and an uncertain field is a hidden variable Y. Because each field is interpreted as a random variable with a domain equivalent to its attribute's, the hypothesis space of the random variables (X and Y) contain the set of possible worlds. Deterministic factors can model constraints over arbitrary sets of variables by outputting 1 if the constraint is satisfied, and 0 if it is violated (rendering such a world world impossible). We then formally define W to be all possible worlds with respect to the factor graph's probability distribution π:

$$W = \{w \in W_\mathcal{D} \mid \pi_\mathcal{G}(w) > 0\} \tag{2}$$

3.3 Example

We show two information extraction problems in Figure 1 as represented in our approach. The top three panes show named entity recognition (NER), and the bottom three panes show entity resolution (disambiguation). NER is the problem of identifying mentions of real-world entities in a text document; e.g., we might identify that "Clinton" is a *person* entity and "IBM" is an *organization* entity. The problem is usually cast as sequence labeling, where each input sentence is divided into a token sequence, and each word (token) in the sequence is treated as an observed variable with a corresponding hidden variable representing the label (entity type) that we are trying to predict. To model this problem with a factor graph, we use factor templates to express relationships between different types of random variables; in our NER example, we express three such relationships (Pane B). The first is a relationship between observed strings and hidden labels at each position in the sequence (called the emission dependency: e.g., this models that the string "Clinton" is highly correlated with the label "person"). The second is a relationship between labels that neighbor in the sequence (known as transition or 1st order Markov dependency: for example, it is likely that a person label will follow a person label because people have first and last names), the final dependency is over each label, modeling the fact that some labels are more frequent than others. Given the template specifications, the graph can be unrolled onto a database. Pane C shows the random variables and factors instantiated over the possible world initially shown in Pane A. The probability of this world is simply a product of all the factors (black boxes) illustrated in Pane C.

The bottom row of Figure 1 shows the problem of entity resolution. Once mentions of named entities have been identified, entity resolution clusters them into real-world entities. The database in pane C shows a single possible world, the templated factor graph in Pane D models relationships between the mentions, allowing dependencies over entire clusters of mentions, dependencies between mentions in the same cluster (modeling that mentions in clusters should be cohesive), and dependencies between variables in different clusters (modeling that mentions in separate clusters should be distant). Finally, Pane E shows the graph unrolled on the database; once again, the score of this possible world is proportional to the product of factors in the unrolled graph. These examples simply serve as an illustration, in practice we will exploit the benefits of MCMC inference to *avoid instantiating the factor graphs over the entire database*.

3.4 Metropolis-Hastings

Metropolis-Hastings (MH) [18, 11] is an extremely general MCMC framework used for estimating intractable probability distributions over large state spaces. One advantage of MCMC is that it can produce samples from the probability distribution π without knowledge of the normalization constant Z_X (which is $\#\mathcal{P}$-hard to compute). We will see in this section that Metropolis-Hastings has many advantages, allowing us to avoid the need to instantiate the

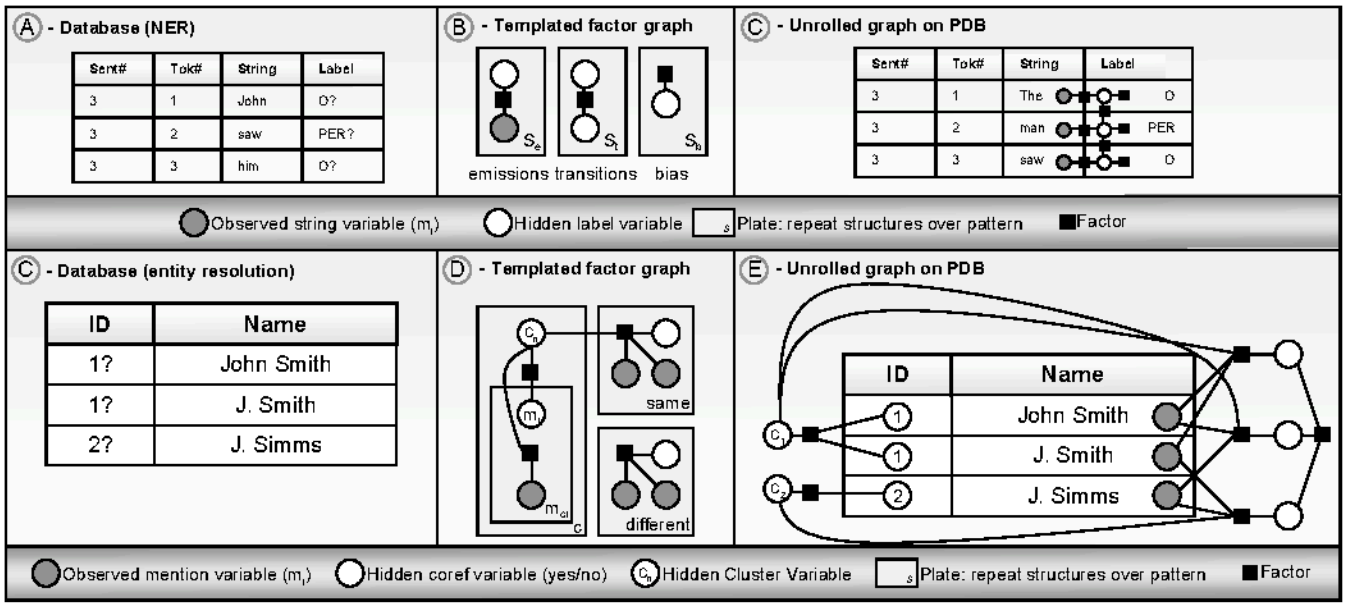


Figure 1: Two information extraction problems in our framework. The top row is the problem of NER and the bottom row is problem of entity resolution. The first column (Panes A and C) shows a deterministic possible world for the respective problems. The second column (Panes B and D) shows the template specification for a factor graph for modeling these problems. The third column (Panes C and E) shows the templated graph unrolled on the possible world.

graphical model over the entire database. The basic idea is that MCMC can hypothesize changes to the single underlying possible worlds by proposing modifications to previous worlds. We describe MH more generally below.

MH requires two components, a target distribution that we wish to sample (in our case $\pi(w)$) and a proposal distribution $q(\cdot|w)$ which conditioned on a state w probabilistically produces a new world w' with probability $q(w'|w)$. The idea is that $q(\cdot|w)$ is a distribution from which we can easily sample (in practice such distributions are easy to construct, and even allow us to inject domain-specific knowledge).

The algorithm is initialized to a possible world w_0 (for example, randomly). Next samples are drawn from the proposal distribution $w' \sim q(\cdot|w)$, and each sample can either be accepted or rejected according to a Bernoulli distribution given by parameter α:

$$\alpha(w', w) = \min\left(1, \frac{\pi(w')q(w|w')}{\pi(w)q(w'|w)}\right) \tag{3}$$

The acceptance probability is determined by the product of two ratios: the model probability ratio $\pi(w')/\pi(w)$ and the proposal distribution ratio $q(w|w')/q(w'|w)$. Intuitively, the model ratio captures the relative likelihood of the two worlds, and the proposal ratio eliminates the bias introduced by the proposal distribution. Given the requirement that the proposal distribution can transition between any two worlds with positive probability in a finite number of steps, the Metropolis-Hastings algorithm is guaranteed to converge to the true distribution encoded by our factor graph. Note that the normalization constant Z appears in both the numerator and denominator and cancels from the computation of α. Further, notice that only factors whose argument variables are changed by q need to be computed, and therefore only a small portion of the graph needs to be unrolled on the database to evaluate each proposal (for the two information extraction problems presented in the previous section, a proposal that modifies only a constant number of variables requires evaluating only a constant number of factors). We show pseudo-code for performing a random-walk with MH in the appendix: Algorithm 2, and demonstrate how factors cancel. We remark that another important advantage of MH is that it avoids the need to explicitly enforce deterministic constraints because the proposer q is designed to transition within the space of possible worlds only (in this sense q is constraint-preserving). An example of a constraint preserving proposal distribution is the split-merge proposer for entity resolution, where clusters of mentions are randomly split or merged (it is easy to check that these two operations preserve the transitivity constraint: avoiding the need to include the expensive cubic number of deterministic transitivity factors).

4. QUERY EVALUATION

The main query evaluation problem we are concerned with is to return the set of tuples in the answer of a query Q over the uncertain database $\langle W, \pi \rangle$, along with their corresponding probabilities (of being in that answer). We say that a tuple t is in the answer of a query Q if and only if $\exists w \in W$ s.t. $t \in Q(w)$. Then, the probability of this event is:

$$Pr[t \in Q(W)] = \sum_{w \in W} \mathbb{1}_{t \in Q(w)} \pi(w) \tag{4}$$

We can see that if a tuple occurs in the answer for all possible worlds, it is deterministic because Equation 4 sums to one. Similarly, a tuple occurring in none of the deterministic answer sets has zero probability and would be omitted from the answer.

Unfortunately, Equation 4 cannot be computed tractably because it requires summing over the set of possible worlds. Alternatively we can write the marginal probabilities as the infinite-sample limit over a set of samples S drawn from $\pi(\cdot|X)$:

$$Pr[t \in Q(W)] = \lim_{n\to\infty} \frac{1}{n} \sum_{i}^{n} \mathbb{1}_{t\in Q(w_i \sim \pi(\cdot))} \quad (5)$$

and estimate $Pr[t \in Q(W_{\mathcal{G}})]$ by using a finite n. Given equation 5, one approach is to draw independent samples $w \sim_{\text{iid}} \pi$, requiring a generative process that must completely instantiate each possible world (for example, as done in MCDB [13]). However, generating a possible world may be expensive in practice, motivating our approach of using Markov-chain Monte Carlo to generate samples by equivalently hypothesizing *modifications* to possible worlds.

There are two primary advantages of using a sampling approach for estimating marginals. The first is that as n goes to infinity, the approximation becomes correct, allowing a trade-off between time and fidelity: intuitively some applications are time sensitive and require only course estimates of query marginals, while in others high fidelity is extremely important. The second important property of sampling methods is that they are query agnostic. That is, we need not concern ourselves with closing the factor-graph representation over every hypothetical query operator. For example, sampling methods trivially handle aggregate extensions to relational algebra because sampling from a graph returned as a query answer would be equivalent to sampling from the original graph.

Up to this point, we have formally described our representation for the possible worlds, the probability distribution, and have posed a query evaluation problem of interest. We now focus our attention to solving this query evaluation problem in our framework. We first overview background material, then describe a basic sampling method. Finally, at the end of this section, we describe the main algorithm of this article: Metropolis Hastings sampling with materialized view maintenance.

4.1 Basic MH Query Evaluation

We now precisely define how to use Metropolis-Hastings to obtain marginal probabilities for tuples in query answers. In particular, we use Algorithm 2 to hypothesize a series of modifications to worlds. Queries are then executed over hypothesized worlds, and the marginal probabilities are computed according to Equation 5 . We should note that consecutive samples in MH are highly dependent; in situations such as ours, where collecting counts is expensive (requires executing the query), it is prudent to increase independence by collecting tuple counts only every k samples (a technique known as thinning). Choosing k is an open and interesting domain-specific problem. We present our basic MCMC sampling method in Algorithm 3 (Appendix).

Another interesting scientific question is how to inject query specific knowledge directly into the proposal distribution. For example, a query might target an isolated subset of the database, then the proposal distribution only has to sample this subset; this can be (1) provided by an expert with domain-specific knowledge, (2) generated by analyzing the structure of the graph and query, or even (3) learned automatically through exploration. However, thoroughly exploring this idea is beyond the scope of this paper.

Finally, there is an interesting balance between the traditional ergodic theorems of MCMC and DBMS-sensitive cost issues arising from disk-locality, caching, and indexing etc. For example, the ergodic theorems imply that every MCMC sample be used to compute an estimate. However, faced with the fact that each sample is non-trivial to compute (requires executing a query), we must balance the dependency of the samples with the expected costs of the queries. Adaptively adjusting k to respond to these various issues is one type of optimization that may be applied to this problem.

Algorithm 1 Query Evaluation with Maintenance Techniques

Input:
initial world w_0,
number of samples per query: k
Initialization:
//run full query to get initial results
$s \leftarrow \text{Q}(w_0)$
//initial counts for marginals
$\mathbf{m} \leftarrow m_i = \begin{cases} 1 & \text{if } m_i \in s \\ 0 & \text{o.w.} \end{cases}$
//initial normalizing constant for marginals
$z \leftarrow 1$
$w \leftarrow w_0$
for $i = 1, \ldots,$ number of steps **do**
  $(w', \Delta^-, \Delta^+) \leftarrow$ MetropolisHastings(w,k)
  $s \leftarrow s - Q'(w, \Delta^-) \cup Q'(w, \Delta^+)$
  $\mathbf{m} \leftarrow m_i + \begin{cases} 1 & \text{if } m_i \in s \\ 0 & \text{o.w.} \end{cases}$
  $z \leftarrow z + 1$
end for
return $\frac{1}{z}\mathbf{m}$

4.2 MH Sampling with View Maintenance

Often, executing queries over a relational database is an expensive resource consuming task. One way of obtaining tuple counts is to run the query over each sampled world; however MCMC enables us to do much better. Recall that consecutive samples in MCMC are actually dependent; in fact, as illustrated in Figure 2, a world w' is the result of a small modification to the original world w.

We use this figure to directly motivate the relevance of materialized view maintenance [4]. Rather than run the original (expensive) query over each consecutive sample, the query is run only once on the initial world, then for each subsequent sample, a modified query is run over the difference Δ and previous world w. That is, we can exploit the semantics of set operations to obtain an equivalent expression for the same answer set. Following the work of Blakeley et al. [4], we recursively express the answer set as:

$$Q(w') = Q(w) - Q'(w, \Delta^-) \cup Q'(w, \Delta^+) \quad (6)$$

where $Q'(w, \Delta^{\pm})$ is inexpensive because $|\Delta^{\pm}| \ll |w|$ and $Q(w)$ is inexpensive because it can be recursively expressed as repeated applications of Equation 6 (bottoming out at the base case of the initial world which is the only world that must be exhaustively queried).

We discuss briefly some view materialization techniques. First, observe that a selection $\sigma(w')$ can be re-written:

$$\sigma(w') \equiv \sigma(w) - \sigma(\Delta^-) \cup \sigma(\Delta^+)$$

and Cartesian products can similarly be re-written as:

$$\begin{aligned} w'.R_1 \times w'.R_2 \equiv\ & w.R_1 \times w.R_2 \\ & - w.R_1 \times \Delta^-.R_2 \\ & \cup\, w.R_1 \times \Delta^+.R_2 \end{aligned}$$

where Δ^- is the original setting of the tuples, Δ^+ is the new setting, and the notation $w.R_1$ is read as: relation R_1 from world w. Traditional results from relational algebra allow joins to be rewritten as a Cartesian product and a selection. Further, it is not difficult to conceive how additional relational operators such as various aggregators can be re-written in terms of the sets Δ^- and Δ^+.

In both the selection and join, the asymptotic savings can be as high as a full degree of a polynomial (for example if Δ is constant in size (as is often the case) and we lack indices over the fields involved in the predicates). The high-level code for our implementation based on view materialization techniques is exhibited in Algorithm 1. In practice, the implementation also requires the use of auxiliary tables for storing the sets Δ^- and Δ^+, which are necessary for running the modified query Q' from Equation 6. These tables must be updated during the course of Metropolis-Hastings, and additional cleaning and refreshing of the tables and multi-set maps are required in between deterministic query executions.

Remark: please note that in the presence of projections (as seen in all of our evaluation queries), that the set-difference and set-union operators (from Equation 6) actually requires multiset semantics, because counters need to be maintained [4]. We apply the necessary modifications to Algorithm 1 providing additional book keeping to track the number of occurrences of each tuple in the set s, so that the operators can be properly applied.

In practice, we handle projections in Algorithm 1 by maintaining the multi-set maps from tuples to counts. In line 5, set difference and set union are replaced with addition and subtraction operators to maintain map counts, and in line 6, the condition is changed to: count$(m_i) > 0$.

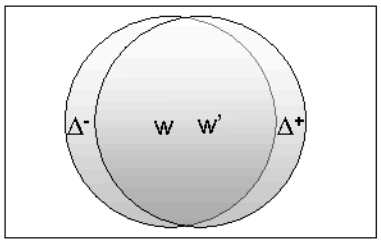


Figure 2: w is the original world, w' is the new world after k MCMC steps and $\Delta^- \subseteq w$ is the set of tuples that were removed from w and $\Delta^+ \subseteq w'$ is the set added to w'.

5. EXPERIMENTS

In this section we demonstrate the viability of our approach by representing the uncertain output of a real-world information extraction problem: named entity recognition (NER). However, we go beyond the simple linear-chain model and use a more sophisticated skip chain conditional random field [27] to represent the uncertainty in both the database and NER. Skip chains and similar complex models achieve state-of-the-art performance in many IE tasks; however no current PDB is capable of modeling them because exact marginal inference is intractable. However, we demonstrate that our MCMC approach effortlessly recovers the desired probabilities for query evaluation.

We implement a prototype system in Scala [20], a functional extension to Java. The system is built in coordination with our graphical model library for imperatively defined factor graphs, Factorie, [17], along with Apache Derby database drivers interfaced through Java's JDBC API. We implement functionality for (1) retrieving tuples from disk and then instantiating the corresponding random variables in memory, and (2) propagating changes to random variables back to the tuples on disk. Statistical inference (MCMC) is performed on variables in main memory while query execution is performed on disk by the DBMS. Additionally, we implement infrastructure for both the naive and materialized-view maintenance query evaluators. As random variables are modified in main memory, their initial and new values are tracked and written to auxiliary tables representing the "added" and "deleted" tuples required for applying the efficient modified queries.

5.1 Application: named entity recognition

We evaluate our probabilistic database on the real-world task of named entity recognition. In particular we obtain ten-million tokens from 1788 New York Times articles from the year 2004. Recall that the problem of named entity recognition is to label each token in the text document with an entity type. We label the corpus using CoNLL entities: "PER" (person entity such as Bill), "ORG" (organization such as IBM), "LOC" (location such as New York City), "MISC" (miscellaneous entity—none of the above), and "O" (not a named entity). We use BIO notation (see the appendix) to encode named entities more than one token in length making the total number of labels nine. We store the output of the ten million NYT tokens in a database relation called TOKEN, with attributes (*TOK_ID, DOC_ID, STRING, LABEL, TRUTH*) where *TOK_ID* is underlined to indicate that it is the primary key, *DOC_ID* is the document for which a token belongs, *STRING* represents the text of a token, *LABEL* is unknown for all tuples and is initialized to "O" , and *TRUTH* is a "ground truth" that we can use to train our model[1].

Next, we define the relational factor graph (Figure 3) over the TOKEN relation to create our probabilistic database. In particular, we first include the three factor templates described in Section 3.3: (1) factors between observed strings and corresponding labels, (2) transition factors between consecutive labels, and (3) bias factors over labels. Up to this point we have defined the traditional linear chain model for NER (see [16]). However, skip-chain models achieve much better results [27], so we include skip-edges or factors between labels whose strings are identical. Intuitively, this factor captures the dependency that if two tokens have the same string, then they have an increased likelihood of having the same label. To see why inference in this graph is intractable, note that the resulting factor graph (Figure 3) is not tree-structured.

Now that we have defined our database and factor graph, we now define our proposal distribution for query evaluation. Given a set of hidden label variables L, our proposal distribution q works as follows: first a label variable is selected uniformly at random from L, then the label for L is randomly changed to one of the nine CoNLL labels {*B-PER, I-PER, B-ORG, I-ORG, B-MISC, I-MISC, B-LOC, I-LOSC, O*}. This processes is repeated for 2000 proposals before L is changed by loading a new batch of variables from the database: up to five documents worth of variables may be selected (documents are selected uniformly at random from the database).

5.2 Methodology

We use the model and proposal distribution described in the previous section in all experiments; we train the model using one-million steps of SampleRank [32], a training method based on MH. The method is extremely quick, learning all parameters in a matter of minutes. The query evaluation problems we investigate are all instances of the general evaluation problem described in Section 4: the goal is to return each tuple along with its probability of being in the answer set. We evaluate the accuracy of our samplers by measuring the squared-error loss to the ground truth query answer (that is, the usual element-wise squared loss). Sometimes we report the normalized squared loss, which simply scales the loss so that the maximum data point has a loss of 1 (this allows us to compare multiple queries on the same graph). Unless otherwise stated,

[1]to estimate ground truth we used the Stanford NER system (nlp.stanford.edu/ner/index.shtml)

we estimate the ground-truth in each problem by running our sampler for one-hundred-million proposals and collect a sample every ten-thousand proposals. In all experiments we evaluate the query every ten-thousand proposals (that is $k = 10,000$ in Algorithm 3).

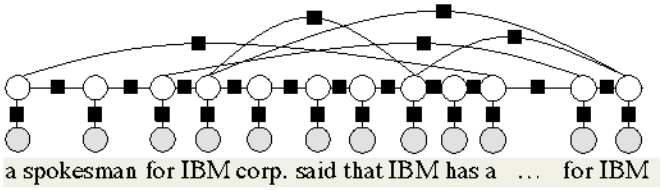


Figure 3: A skip chain conditional random field that includes "skip" edges, or factors between tokens with the same string. Bias factors over labels are omitted for clarity.

5.3 Scalability

In this section we demonstrate that we are able to scale query evaluation to a large number of tuples even though exact inference in the underlying graphical model (skip-chain CRF) is intractable and approximate methods such as loopy belief propagation fail to converge for these types of graphs [27]. We additionally compare the materialized MCMC sampler with the basic (naive) MCMC sampler, demonstrating dramatic efficiency gains. We use the following simple, but non-selective query that scales linearly with the number of tuples (note that the DBMS lacks an index over the STRING field):

Query 1

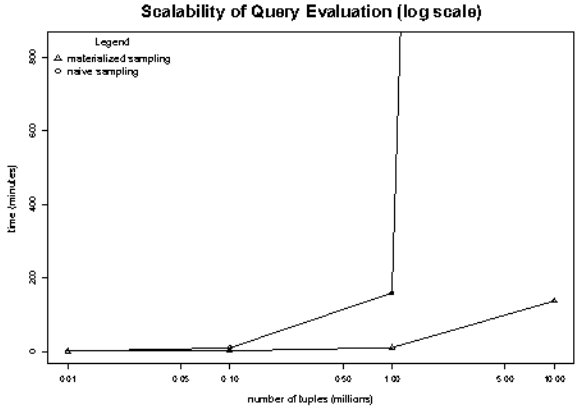
```
SELECT STRING
FROM TOKEN
WHERE LABEL='B-PER'
```

In Figure 4(a) we plot query evaluation time versus the number of tuples in the database (log scale) for both the naive and materialized approach (over several base ten orders of magnitude). As stated earlier, we have no way of obtaining the true probabilities, so we estimate the ground truth by sampling and then define the *query evaluation time* as the time taken to half the quad loss (squared error) from the initial "single-sample" deterministic approximation to the query.

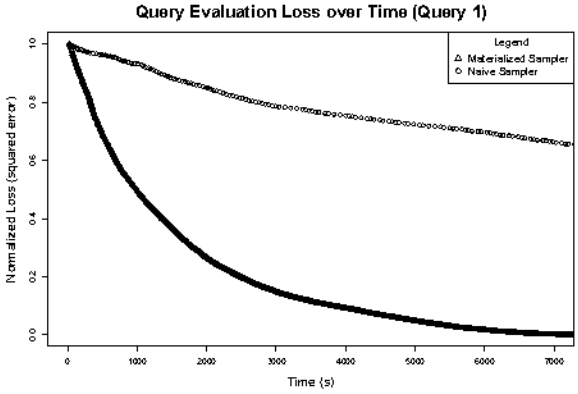
For small databases, the sampler based on view-maintenance techniques does not provide efficiency gains over the naive approach. Indeed, when the database contains just 10,000 tuples, the two approaches perform comparably: the naive sampler is slightly quicker (19 seconds compared to 21 seconds) possibly due to the overhead involved in maintaining the auxiliary diff tables (recall that the size of the diff tables is roughly 10,000 tuples because there are that many steps between query executions). For 100,000 tuples, the view-based approach begins to outperform the naive approach (162 seconds versus 178 for naive) and quickly yields dramatic improvements as the number of tuples increases. In fact, we were unable to obtain the final data-point (ten million tuples) for the naive sampling approach because we project it to take 227 hours to complete. In stark contrast, the sampler based on view-maintenance techniques takes under two-and-a-half hours on the same ten million tuples. We are impressed with the speed of the evaluation because inference in skip chains CRFs is extremely difficult and normally takes hours to complete—even in the non-database setting.

It is worth noting that for the skip-chain CRF (and the sophisticated entity-wise coreference model presented in Figure 1), the time to perform an MCMC walk-step is constant with respect to the size of the database. That is, if the proposal distribution only modifies a constant number of variables, then only a constant number of tuples in the database are involved in computing the acceptance computation (see the Appendix, Section 9.2). Therefore, because the time to perform a single walk-step is constant with respect to the size of the repository, only two primary factors affect scalability: (1) the DBMS's deterministic query execution time and (2) the number of samples required to change the database in a meaningful way. This suggests two avenues of future work for improving scalability even further. In particular investigating jump functions that better explore the space of possible worlds appears to be an extremely fruitful venture with high dividends.

Next, in Figure 4(b), we plot query-evaluation error versus time for both query evaluators on the 1-million tuple database. Recall that the two approaches generate the same set of samples, but the naive approach is slower because it must execute the query on each possible world (rather than exploiting the set of modified tuples). Impressively, the efficient evaluator nearly zeroes the error before the naive approach can even half the error. Also, notice how loss tends to decrease monotonically over time. This allows our approach to be used as an any-time algorithm: applications that require fine probability estimates can spend more evaluation time, while those that are time sensitive can settle for courser estimates.



(a) Scalability over several orders of magnitude (Query 1); x axis is millions of tuples in log scale and y axis is time taken to half squared error.



(b) Loss versus time comparison for the two query evaluation approaches. Query 1 is evaluated over one-million tuples.

Figure 4: The benefits view maintenance query evaluation.

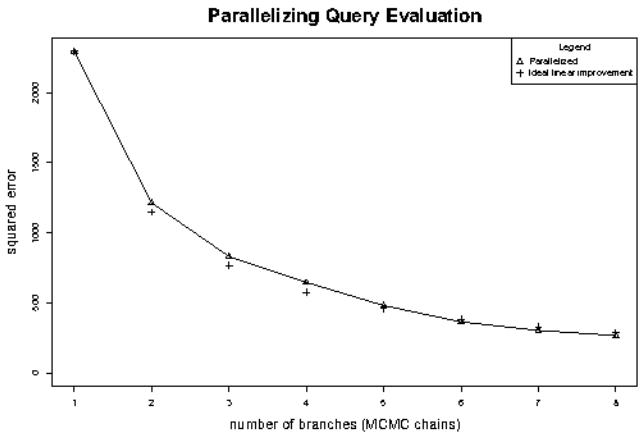


Figure 5: Multiple evaluators in parallel.

5.4 Parallelization

In general, sampling approaches to query evaluation can be easily parallelized to yield impressive performance improvements. These performance improvements are potentially even greater in the context of our MCMC approach because parallelization provides the additional benefit of generating samples with higher independence, leading to faster mixing rates. In this section we show that running multiple query evaluators in parallel dramatically improves the accuracy given a fixed time-span, demonstrating our system's potential to satisfy high-fidelity requirements in time-sensitive situations.

We evaluate the effects of parallelization as follows. First we produce eight identical copies (initial worlds) of the probabilistic database, each with ten million tuples. We evaluate Query 1 using the usual set-up except we obtain the ground-truth by averaging eight parallel chains for ten-thousand samples each. To evaluate the query, we run up to eight parallel query evaluators for one-hundred samples (with the usual ten-thousand MCMC steps in between each sample), the results are plotted in Figure 5 and compared against the ideal linear improvement. For example, by using two chains we almost half the loss of the one chain evaluator. Impressively, eight chains reduces the error by slightly more than a factor of eight, demonstrating that MCMC sampling evaluation can be further improved by parallelization.

As we can see, this simple form of parallelization is actually quite powerful because samples taken across multiple chains are much more independent than those taken within a single chain. This is one reason why we actually observe super-linear improvements in fidelity through parallelization. These benefits come at a relatively small cost of (1) additional hard-drive space for storing multiple worlds simultaneously, and (2) additional processors for parallelization.

5.5 Aggregates

Another benefit of sampling-based approaches is their ability to answer arbitrary relational algebra queries without the need to close a representation system under the necessary operators. In this section we empirically demonstrate that our system is capable of answering arbitrary extensions to relational algebra such as aggregates. We begin with a simple aggregate query that counts the number of person mentions in one-million tuples worth of New York Times tokens.

Query 2

```
SELECT COUNT(*)
FROM TOKEN
WHERE LABEL='B-PER'
```

The second aggregate query retrieves documents in which the number of *person* mentions is equivalent to the number of *organization* mentions within that document (again applied to one-million tuples).

Query 3

```
SELECT T.doc_id
FROM Token T
WHERE (SELECT COUNT(*)
  FROM Token T1
  WHERE T1.label='B-PER' AND T.doc_id=T1.doc_id)
=(SELECT COUNT(*)
  FROM Token T1
  WHERE T1.label='B-ORG' AND T.doc_id=T1.doc_id)
```

We plot squared error loss as a function of time for these two queries. The ground-truth was obtained by running each query for five-thousand samples with the usual ten-thousand MCMC walk-steps between each sample. We see that Query 2 rapidly converges to zero loss, and Query 3 converges at a respectable rate. In fact, the rapid convergence of Query 2 can be explained by examining its answer set, which we provide in Figure 9 of the Appendix. Notice how the distribution is highly peaked about the center and appears to be normally distributed. This is not unusual for real-world data, and MCMC sampling is celebrated for its ability to exploit this concentration of measure, leading to rapid convergence.

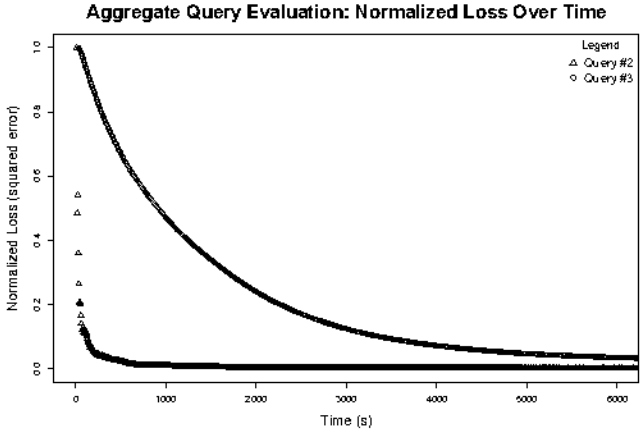


Figure 6: Squared loss over time time for two aggregate queries (Query 2 and Query 3), results reported on one-million tuples.

6. CONCLUSION AND FUTURE WORK

In this paper we proposed a framework for probabilistic databases that uses factor graphs to model distributions over possible worlds. We further advocated MCMC sampling techniques and demonstrated how the Markovian nature can be exploited to efficiently evaluate arbitrary relational queries in an any-time fashion.

In future work we would like to investigate methods for *automatically* constructing jump functions to target specific queries.

7. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by Lockheed Martin prime contract #FA8650-06-C-7605 through a subcontract with BBNT Solutions LLC, in part by SRI International subcontract #27-001338 and ARFL prime contract #FA8750-09-C-0181, in part by The Central Intelligence Agency, the National Security Agency and National Science Foundation under NSF grant #IIS-0326249, and in part by UPenn NSF medium IIS-0803847. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] P. Andritsos, A. Fuxman, and R. J. Miller. Clean answers over dirty databases: A probabilistic approach. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 30, Washington, DC, USA, 2006. IEEE Computer Society.

[2] D. Barbará, H. Garcia-Molina, and D. Porter. The management of probabilistic data. *IEEE Trans. on Knowl. and Data Eng.*, 4(5):487–502, 1992.

[3] J. Biskup. A formal approach to null values in database relations. *Advances in Data Base Theory*, pages 299–341, 1981.

[4] J. A. Blakeley, P.-A. Larson, and F. W. Tompa. Efficiently updating materialized views. *SIGMOD Rec.*, 15(2):61–71, 1986.

[5] J. Boulos, N. Dalvi, B. Mandhani, S. Mathur, C. Re, and D. Suciu. Mystiq: A system for finding more answers by using probabilities. In *SIGMOD (demo)*, 2005.

[6] A. Culotta, M. Wick, R. Hall, and A. McCallum. First-order probabilistic models for coreference resolution. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL)*, pages 81–88, 2007.

[7] N. Dalvi, C. Re, and D. Suciu. Probabilistic databases: Diamonds in the dirt (extended version). In *CACM*, 2008.

[8] N. Dalvi and D. Suciui. Efficient query evaluation on probabilistic databases. In *VLDB*, 2004.

[9] N. Fuhr and T. Rölleke. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst.*, 15(1):32–66, 1997.

[10] T. J. Green and V. Tannen. Models for incomplete and probabilistic information. In *Current Trends in Database Technology*. Springer, 2006.

[11] W. Hastings. Monte carlo sampling methods using markov chains and their applications. In *Biometrika*, 1970.

[12] T. Imielinski and W. Lipski, Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[13] R. Jampani, F. Xu, M. Wu, L. L. Perez, C. Jermaine, and P. J. Haas. Mcdb: a monte carlo approach to managing uncertain data. In *SIGMOD '08: Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 687–700, New York, NY, USA, 2008. ACM.

[14] S. Joshi and C. Jermaine. Sampling-based estimators for subset-based queries. *VLDB-J*, 2009.

[15] C. Koch. Maybms: A system for managing large uncertain and probabilistic databases. In *To appear in Chapter 6 of Charu Aggarwal ed., Managing and Mining Uncertain Data*. Springer-Verlag, 2008.

[16] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[17] A. McCallum, K. Rohanimanesh, M. Wick, K. Schultz, and S. Singh. Factorie: Efficient probabilistic programming via imperative declarations of structure, inference and learning. In *NIPS workshop on Probabilistic Programming*, 2008.

[18] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. In *Journal of Chemical Physics*, 1953.

[19] B. Milch, B. Marthi, and S. Russell. Blog: Relational modeling with unknown objects. In *ICML 2004 Workshop on Statistical Relational Learning and Its Connections to Other Fields*, 2004.

[20] M. Odersky and al. An overview of the scala programming language. Technical Report IC/2004/64, EPFL Lausanne, Switzerland, 2004.

[21] H. Poon and P. Domingos. Joint inference in information extraction. In *AAAI*, pages 913–918, Vancouver, Canada, 2007. AAAI Press.

[22] C. Re, N. Dalvi, and D. Suciu. Efficient top-k query evaluation on probabilistic data. In *ICDE*, 2007.

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

[24] P. Sen and A. Deshpande. Representing and querying correlated tuples in probabilistic databases. In *ICDE*, 2007.

[25] P. Sen, A. Deshpande, and L. Getoor. Exploiting shared correlations in probabilistic databases. In *VLDB*, 2008.

[26] D. Sontag, T. Meltzer, A. Globerson, T. Jaakkola, and Y. Weiss. Tightening lp relaxations for map using message passing. In *UAI*, 2008.

[27] C. Sutton and A. McCallum. Collective segmentation and labeling of distant entities in information extraction. Technical Report TR # 04-49, University of Massachusetts, July 2004.

[28] D. Wang, E. Michelakis, M. Garofalakis, M. J. Franklin, and J. M. Hellerstein. Declarative information extraction in a probabilistic database system. Technical report, UC Berkeley, 2010.

[29] D. Z. Wang, E. Michelakis, M. Garofalakis, and J. M. Hellerstein. Bayesstore: Managing large, uncertain data repositories with probabilistic graphical models. In *VLDB*. VLDB Endowment, 2008.

[30] J. Weinman, E. Learned-Miller, and A. Hanson. Scene text recognition using similarity and a lexicon with sparse belief propagation. *PAMI*, 2009.

[31] M. Wick, A. McCallum, and G. Miklau. Representing uncertainty in probabilistic databases with scalable factor graphs. Master's thesis, University of Massachusetts, proposed September 2008 and submitted April 2009.

[32] M. Wick, K. Rohanimanesh, A. Culotta, and A. McCallum. Samplerank: Learning preference from atomic gradients. In *NIPS WS on Advances in Ranking*, 2009.

[33] M. Wick, K. Rohanimanesh, A. McCallum, and A. Doan. A discriminative approach to ontology alignment. In *In proceedings of the 14th NTII WS at the conference for Very Large Databases (VLDB)*, 2008.

[34] J. Widom. Trio: A system for data, accuracy, and lineage. In *Proc Second Biennial Conference on Innovative Data Systems Research (CIDR 05)*. Springer-Verlag, 2005.

9. APPENDIX

9.1 Examples Probabilistic Query Answers

Here we provide a few examples of query answers. Recall that answers contain tuples along with their probabilities. In each of these plots the x axis ranges over actual tuples and the height of the bar show the probability of that tuple being in the answer. Figure 9 shows the answer to Query 2, an aggregate query asking the number of person mentions ("B-PER"), over ten million tokens from NYT articles from the year 2004. Notice that the mass appears to be normally distributed, where the important observation is that most of the mass is clustered around a small subset of the answer set. This important property is exhibited by many real-world datasets, and enables MCMC to rapidly converge to the true stationary distribution.

In Figure 8 we show a subset of the answer to Query 4, which seeks all person mentions ("B-PER") that co-occur (in the same document) as a token with string "Boston" having label "B-ORG". Intuitively, "Boston" can ambiguously be a location or an organization (because organizations are often named after the city in which they are based).

Query 4

```
SELECT T2.STRING
FROM TOKEN T1, TOKEN T2
WHERE T1.STRING='Boston' AND T1.LABEL='B-ORG'
  AND T1.DOC_ID=T2.DOC_ID AND T2.LABEL='B-PER'
```

We find that many of the people returned in our query are affiliated with baseball likely because the Boston Red Sox are a prominent example of an organization named after a city.

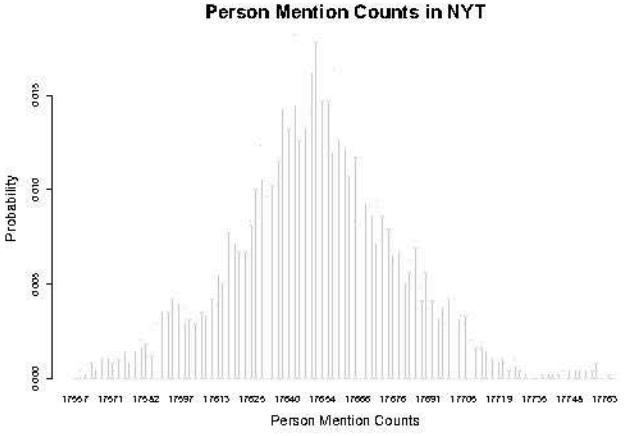


Figure 7: Aggregate query (Query 2) distribution as a histogram. Shows the distribution of person mention counts over 10 million NYT tuples.

9.2 MCMC Efficiencies

The following shows how the acceptance ratio in Metropolis Hastings can be computed efficiently. We begin with the MH acceptance ratio, which depends on the probabilities expressed by the factor graph (Equation 1). Simple algebraic manipulation allows the ratio to be expressed in terms factors neighboring only those variables that change:

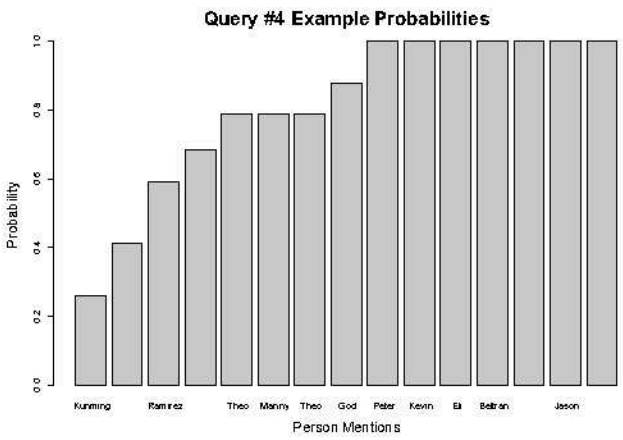


Figure 8: Selected tuples from Query 4 over NYT tuples

$$
\begin{aligned}
\frac{p(y')}{p(y)} &= \frac{p(Y = y'|X;\theta)}{p(Y = y|X;\theta)} \\
&= \frac{\frac{1}{Z_X}\prod_{y^i \in y'} \psi(X, y^i)}{\frac{1}{Z_X}\prod_{y^i \in y} \psi(X, y^i)} \\
&= \frac{\left(\prod_{y^i \in \delta_{y'}} \psi(X, y^i)\right)\left(\prod_{y^i \in y' - \delta_{y'}} \psi(X, y^i)\right)}{\left(\prod_{y^i \in \delta_y} \psi(X, y^i)\right)\left(\prod_{y^i \in y - \delta_y} \psi(X, y^i)\right)} \\
&= \frac{\prod_{y^i \in \delta} \psi(X, y^i)}{\prod_{y^i \in \delta} \psi(X, y^i)}
\end{aligned}
$$

For example, take the skip chain conditional random field presented in Figure 3 of Section 5. Suppose an initialization where the middle "IBM" token is assigned the label "LOC" and our jump function proposes to change the label to "ORG", then we only need to compute twelve factors to evaluate the MH acceptance ratio and decide whether to accept this jump. For this model and proposal distribution, the number of factors we ever need to evaluate is constant with respect to the number of tokens in the database.

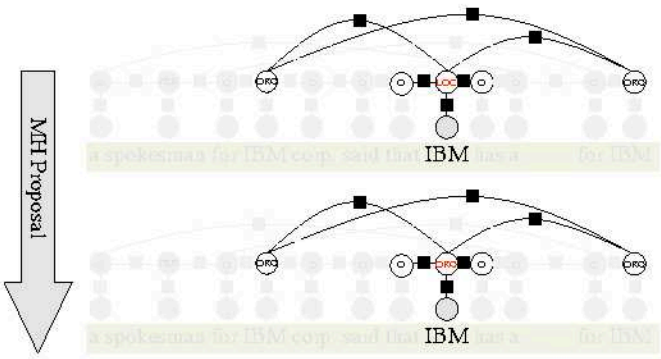


Figure 9: Efficient Metropolis-Hastings evaluation for the skip chain presented in Section 5. Greyed out factors cancel in the MH proposal ratio and their corresponding greyed-out arguments may be ignored. Only black factors need to be evaluated.

Algorithm 2 Random Walk with Metropolis Hastings (n steps)

Input:
Initial world w,
number of steps n
for $i = 1, \ldots, n$ **do**
  $w' \sim q(w)$
  if $true \sim \alpha(w', w)$ **then**
    $w \leftarrow w'$
  end if
  return w
end for

9.3 BIO Labels for Named Entity Recognition

BIO labels allow textual mentions to be composed of more than one token by prefixing the labels with a *B-<T>* indicating that the token is beginning a mention of type *<T>*, and *I-<T>* indicating the token is continuing a mention of type *<T>*; and an *O* indicating the word is not any type of mention.

As an example, if we annotate the sentence *he saw Hillary Clinton speak* as:

he **(B-PER)**, saw **(O)**, Hillary **(B-PER)**, Clinton **(I-PER)**, speaks **O**

then the sentence is interpreted as having two mentions: *he* and *Hillary Clinton*. Note that I-<T> can follow B-<U> if and only if $T = U$, otherwise, the interpretation is meaningless. This suggests we could devise a more intelligent jump function that takes this constraint into account.

9.4 Query Evaluation

Here we show the basic components of query evaluation. First a Metropolis-Hastings random walk is presented in Algorithm 2. The algorithm takes an initial world w_0, then executes n proposals, resulting in a random walk, ending in some final world w'.

Next we show the basic query evaluation method (Algorithm 3). This method evaluates a query Q on the probabilistic database. Recall that the database always stores a single possible world and is initialized to some world w. To collect a sample, k MH walksteps are taken to transition the database to some new world w'. Then the query Q is executed over this deterministic world and tuple-counts are collected. This process is repeated n times. Note that this is the basic MH query evaluator that does not exploit the overlap between consecutive MCMC samples; the more sophisticated view-maintenance evaluator is described in the body of this manuscript.

Algorithm 3 Basic Query Evaluation Method

Input:
initial world w_0,
number of steps n
number of samples per query: k
Initialization:
//initial state
$w \leftarrow w_0$
//initial marginal counts
$\mathbf{m} \leftarrow \mathbf{0}$
//initial normalizing constant for marginals
$z \leftarrow 0$
for $i = 1, \ldots, n$ **do**
  //run MH for k steps beginning on world w
  $w \leftarrow$ MetropolisHastings(w,k)
  //run query on sampled world
  $s \leftarrow Q(w)$
  //increase counts
  $\mathbf{m} \leftarrow m_i + \begin{cases} 1 & \text{if } m_i \in s \\ 0 & \text{o.w.} \end{cases}$
  $z \leftarrow z + 1$
end for
return $\frac{1}{z}\mathbf{m}$