
Learning and Inference in Tractable Probabilistic Knowledge Bases

Mathias Niepert and **Pedro Domingos**
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195-2350, USA

Abstract

Building efficient large-scale knowledge bases (KBs) is a longstanding goal of AI. KBs need to be first-order to be sufficiently expressive, and probabilistic to handle uncertainty, but these lead to intractable inference. Recently, tractable Markov logic (TML) was proposed as a non-trivial tractable first-order probabilistic representation. This paper describes the first inference and learning algorithms for TML, and its application to real-world problems. Inference takes time per query sublinear in the size of the KB, and supports very large KBs via parallelization and a disk-based implementation using a relational database engine. Query answering is fast enough for interactive and real-time use. We show that, despite the data being non-i.i.d. in general, maximum likelihood parameters for TML knowledge bases can be computed in closed form. We use our algorithms to build a very large tractable probabilistic KB from numerous heterogeneous data sets. The KB includes millions of objects and billions of parameters. Our experiments show that the learned KB outperforms existing specialized approaches on challenging tasks in information extraction and integration.

INTRODUCTION

A knowledge base that continuously acquires knowledge and answers complex queries is a vision as old as the field of AI itself. The formal representation employed by such a KB needs to be at the level of first-order logic, in which inference is intractable. Many tractable subsets of first-order logic have been proposed (e.g., description logics, Horn KBs), but building large KBs further requires that the representation be probabilistic, for at least two reasons: most of the knowledge will necessarily be acquired from text, the Web, etc., which are inherently noisy and ambiguous, and

maintaining the consistency of large KBs is extremely difficult. First-order probabilistic representations like Markov logic have been developed, but they are again intractable, as are subsets like probabilistic Horn KBs and description logics. Recently, Domingos and Webb [2011] developed tractable Markov logic (TML), a subset of Markov logic that has expressiveness comparable to probabilistic Horn KBs and description logics but remains tractable. However, to date this was only a theoretical proposal, with no inference and learning algorithms beyond the basic inference required to prove tractability, and no real-world applications.

This paper develops highly scalable inference and learning algorithms for tractable probabilistic knowledge bases (TPKBs), using TML as the representation. After an initialization phase that computes the partition function of the TPKB in time linear in its size, queries are answered in sublinear time using a relational database implementation that allows for large disk-resident KBs and is easily parallelized. Maximum-likelihood parameter learning is done in closed form, and takes time linear in the size of the training data, which can therefore also be very large and disk-resident. Together these inference and learning algorithms make it possible to learn and reason with very large first-order probabilistic KBs.

As a test of our architecture and algorithms, we learn the structure and parameters of a TPKB from a multitude of very large data sets, and use it to answer a variety of queries. The resulting TPKB is larger than previous statistical relational models by at least an order of magnitude, with millions of objects and billions of parameters, yet allows for subsecond query answering times. Extensive experiments show the efficiency and effectiveness of the proposed framework for entity linking and resolution.

RELATED WORK

Probabilistic description logic programs [17] combine DL programs under the answer set and well-founded semantics with independent choice logic [26]. Particular variants of light-weight description logics with probabilistic seman-

tics [11, 22, 24] have been proposed. However, these formalisms are too expressive to be tractable for the types of queries needed in large-scale applications, do not allow the modeling of data properties, and do not address the problem of parameter and structure learning.

TPKBs are related to PROBLOG [28] a probabilistic logic programming language that is generally intractable. URDF [19] is based on weighted MAX-SAT algorithms and is in principle intractable. It also does not support queries that ask for a distribution over objects, classes and relations given a subset of their attributes. Infinite relational models (IRMs) are non-parametric hierarchical models but are not tractable in general [13]. We leverage the hierarchy of TPKBs to estimate parameters on the level of classes and relation more robustly. This is related to shrinkage [18].

TPKBs are different from probabilistic databases [31] in that they represent general knowledge attached to class and relation hierarchies and do not assume tuple independence, that is, the independence of facts in the knowledge base. Moreover, inference in a TPKB is always at most linear in the size of the TPKB whereas inference complexity in PDBs depends on the query expression and can be costly.

Open information extraction [7] and other IE projects [2, 15] often use ad-hoc approaches and heuristics and do not provide a consistent joint distribution and query language. There exist several statistical relational systems employing relational database technology to facilitate queries over structured data [23, 33, 25]. However, the proposed systems are intractable. Recent work on statistical relational learning has focused on tractable probabilistic graphical models, that is, probabilistic models for which inference is efficient by design. Examples are relational sum-product networks [20], particular tractable fragments of probabilistic logics [32], and probabilistic soft logic [14]. None of these languages feature disk-based, sublinear inference algorithms or has been used to build a real-world tractable probabilistic knowledge base. There is related work in the context of information extraction and relation learning. Notable representative publications of this line of research are tensor factorizations of YAGO [21] and universal schemas [29]. These approaches do not facilitate a consistent probabilistic semantics and expressive query languages.

TRACTABLE PROBABILISTIC KNOWLEDGE BASES

Tractable Markov logic (TML) [4] is a subset of Markov logic exploiting probabilistic class and part hierarchies to control complexity. In TML the domain is decomposed into parts, each part is drawn probabilistically from a class hierarchy, the part is further decomposed into parts according to its class, and so on. More recently, TML was extended

so as to also handle existence uncertainty [34]. Tractable probabilistic knowledge bases (TPKBs) accomplish this by defining a possible world as a set of objects and the truth values of the atoms in which they occur as arguments, rather than just as a set of truth values over a fixed universe of objects. TPKBs form the basis for the new algorithms in this paper. We extend them with attributes that can have discrete and continuous distributions. Attributes allow us to model typical properties of objects in real-world knowledge bases such as images, geographical locations, and mentions.

Syntax

A TPKB is a set of class and object declarations. A class declaration specifies the subclasses, parts, attributes, and relations of the class, along with their distributions. It has the following form:

```

Class C {
  subclasses  $S_1 w_1, \dots, S_n w_n$ ;
  parts  $P_1[c_1] C_1, \dots, P_\ell[c_\ell] C_\ell$ ;
  attributes  $A_1 D_1 u_1, \dots, A_m D_m u_m$ ;
  relations  $R_1(\dots) v_1, \dots, R_k(\dots) v_k$ ;
}

```

Subclass declarations specify the direct subclasses (children) of C and their real-valued weights. Direct subclasses are assumed to be mutually disjoint and form a partition of the superclass.

Part declarations specify the parts every instance of C must have. Part declarations consist of a part name P_i , the part's class C_i , and the number c_i of parts of that type where c_i is optional and 1 by default. Two classes such that one is a descendant of the other in the class hierarchy never have a part with the same name.

Every attribute A has a domain D_i (e.g., $\{0, \dots, 10\}, \mathbb{R}$) and a weight function $u_i : D_i \rightarrow \mathbb{R}$. The weight function for a continuous domain must be efficiently integrable.

Each relation $R_i(\dots)$ has the form $R_i(P_a, \dots, P_z)$ where each of P_a, \dots, P_z is a part of C , that is, one of P_1, \dots, P_ℓ , and the v_i 's are real-valued weights. Weights determine the probability that an object belongs to a direct subclass or that a relation is true as specified below. If a relation weight does not appear, then the relation is hard, that is, it must hold in every possible world.

Objects of the TPKB are introduced as parts, subparts, sub-subparts, etc., of a single *top object*. The top object is the sole instance of its class which is named the *top class*. The top class does not have superclasses. Given a set of class declarations, we can define the following concepts.

The *class hierarchy* is a directed graph whose nodes correspond to classes and whose edges correspond to direct

subclass relationships. The class hierarchy must be a forest.

The *part decomposition* is a directed graph whose nodes correspond to parts and with edges from each object to each of its possible parts. The part decomposition must be a tree with the top object as root node.

In addition to class declarations, TPKBs have object declarations which introduce evidence by specifying the names of the object as well as its subclass memberships, attribute values, and relations. An object declaration has the form

```
Path Name {
  S1, ..., Sn;
  A1 = D1, ...;
  R1(...), ..., ¬R1(...), ...;
}
```

where Path is the object's path from the top object in the part decomposition and Name the name given to the object. For instance, for a TPKB with top object World, World.Country₁.State₅.Capital Olympia { ... } is the declaration for object World.Country₁.State₅.Capital giving it the name Olympia. If, for instance, Country₁ was previously declared to be World.USA and State₅ to be USA.Washington, then Path could also be Washington.Capital.

The statement S₁, ..., S_n; expresses that S₁, ..., S_n are the only classes the object can be an instance of. Each statement A_i = D, with D ∈ D_i, is an attribute fact and denotes that the attribute is known to have value D. For every attribute A_i of the object there exists at most one fact A_{a_i} = D. Each statement R_i(...) is a relation fact and denotes that the relation is known to be true, and each statement ¬R_i(...) is a relation fact that denotes that the relation is known to be false. Object declarations are optional and may be empty, that is, they may have the form Path Name { }. Empty object declarations introduce evidence by declaring the object to exist in all possible worlds. Object declarations have the purpose of naming objects and introducing evidence and must be compatible with the class declarations. Table 1 depicts a typical set of class and object declarations.

Semantics

The possible objects of a TPKB are the top object and every other object in the part decomposition. The Herbrand universe of a TPKB is the set of constants representing classes, paths representing objects, and constants representing attribute values. An *n*-ary predicate grounded with elements of the Herbrand universe is an *atom*. We define the binary predicate Is, where atom Is(O, C) is true if object O is of class C and false otherwise. Likewise, A(O, D) is an atom which is true if attribute A has value D for object O. Finally,



```
Class Country {
  parts Capital City, President Person;
  attributes area ℝ u1, image Images u2;
}
Class Person {
  subclasses Politician 0.6, ..., Actor 1.1;
  attributes age ℕ u3, image Images u4;
}
Class City {
  parts Mayor Person, PoliceChief Person;
  attributes area ℝ u5, image Images u6;
  relations reportsTo(PoliceChief,Mayor) 2.3;
}
World.Country1 USA {
  area = 9857306;
}
USA.Capital WashingtonDC {
  area = 77, image=;
}
WashingtonDC.Mayor MurielBowser {
  age = 42, image=;
}
WashingtonDC.PoliceChief CathyLanier { }
```

Table 1: Example class and object declarations of a TPKB. area is an attribute that models the area of an entity in square kilometers, and images a collection of images depicting entities.

R(O, ...) is an atom which is true if the relation R(...) is true for object O and false otherwise. We refer to class membership, attribute, and relation atoms and their negations as literals. A TPKB is a DAG of objects and their properties (classes, attributes, relations), and a possible world is a subtree of the DAG with values for the attributes and relations. More formally, a possible world **W** of a TPKB with top object O₀ and top class C₀ is a set of literals such that:

1. Is(O₀, C₀) ∈ **W**.
2. If Is(O, C) ∈ **W**, then
 - (a) if C has direct subclasses, then Is(O, S) ∈ **W** for exactly one direct subclass S and ¬Is(O, S') ∈ **W** for every other direct subclass S';
 - (b) for every part P of O declared to be of class C', Is(O.P, C') ∈ **W**;
 - (c) for every attribute A_i declared for class C there is exactly one D ∈ D_i with A_i(O, D) ∈ **W** and ¬A_i(O, D') ∈ **W** for every other D' ∈ D_i; and
 - (d) for every relation R(...) declared for class C either R(O, ...) ∈ **W** or ¬R(O, ...) ∈ **W**; if R is hard, then R(O, ...) ∈ **W**.
3. No other literals are in **W**.

The set of objects that exist in a possible world is exactly the objects that occur as arguments of the class membership, relation, and attribute predicates. We write \mathcal{W} to denote the set of all possible worlds.

We write $\text{Subs}(C)$, $\text{Parts}(C)$, $\text{Atts}(C)$ and $\text{ReIs}(C)$ to denote the direct subclasses, parts, attributes, and relations declared for class C .

A possible subworld for object O and class C is defined as above with O_0 replaced by O and C_0 replaced C . The unnormalized distribution ϕ over possible subworld \mathbf{W} without evidence, that is, without object declarations, is defined recursively as $\phi(O, C, \mathbf{W}) = 0$ if $\neg \text{Is}(O, C) \in \mathbf{W}$ or if a relation R of C is hard and $\neg R(O, \dots) \in \mathbf{W}$ and otherwise as

$$\phi(O, C, \mathbf{W}) = \left(\sum_{S_i \in \text{Subs}(C)} e^{w_i} \phi(O, S_i, \mathbf{W}) \right) \times \left(\prod_{P_i \in \text{Parts}(C)} \phi(O.P_i, C_i, \mathbf{W})^{n_i} \right) \times \left(\prod_{A_i \in \text{Atts}(C)} \alpha(O, A_i, \mathbf{W}) \right) \times \left(\prod_{R_i \in \text{ReIs}(C)} \rho(O, R_i, \mathbf{W}) \right),$$

where $\alpha(O, A_i, \mathbf{W}) = e^{u_i(D)}$ if $A_i(O, D) \in \mathbf{W}$. Moreover, $\rho(O, R_i, \mathbf{W}) = e^{v_i}$ if $R_i(O, \dots) \in \mathbf{W}$ and $\rho(O, R_i, \mathbf{W}) = 1$ if $\neg R_i(O, \dots) \in \mathbf{W}$. If an object has no subclasses, parts, attributes, or relations, then the corresponding products in the above equation evaluate to 1.

The unnormalized probability of a possible world \mathbf{W} is now $\phi(O_0, C_0, \mathbf{W})$. The sub-partition function for object O and class C is

$$Z_{\mathcal{K}_{O,C}} = \sum_{\mathbf{W} \in \mathcal{W}} \phi(O, C, \mathbf{W}).$$

If O is the top object and C the top class, we simply write $Z_{\mathcal{K}}$. The probability of a possible world \mathbf{W} is

$$P(\mathbf{W}) = \frac{\phi(O_0, C_0, \mathbf{W})}{Z_{\mathcal{K}}}.$$

Sum-product networks (SPNs) are a class of deep probabilistic models in which one can perform fast exact inference [27]. An SPN is recursively constructed by forming sums and products of univariate distributions, and its partition function can be computed in time linear in its size. Every TPKB maps into a corresponding SPN such that the distribution of the TPKB is equivalent to the distribution of the SPN. The construction follows the recursive definition of the distribution over possible worlds $\phi(O_0, C_0, \cdot)$. Figure 1 depicts the SPN corresponding to the example TPKB

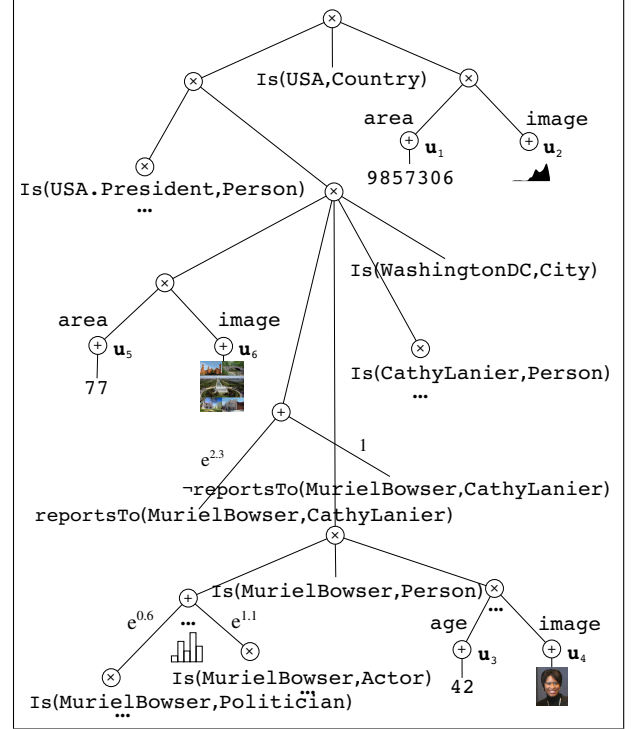


Figure 1: The SPN corresponding to the TPKB in Table 1.

of Table 1. The *size* of a TPKB is the number of objects times the number of classes each object is possibly an instance of. The proof of the following theorem can be found in the appendix.

Theorem 1. *The partition function of a TPKB can be computed in time linear in its size.*

We can also show that TPKBs are at least as expressive as the class of SPNs.

Theorem 2. *For every SPN there exists a TPKB with the same distribution and the size of the TPKB is linear in the size of the SPN.*

Hence, TPKBs include a large number of high-treewidth graphical models. These high-treewidth distributions are common in the real world, occurring whenever two subclasses of the same class have different subparts. TPKBs are non-trivial relational models that capture relational dependencies, multiple objects, existence uncertainty, etc. Also, even though data is not i.i.d. in general, closed-form maximum likelihood learning is possible given the TPKB's structure and complete data. Lifting in TPKBs is straightforward and makes the complexity of inference independent of the number of evidence-free objects.

The key assumption that TPKBs make to ensure tractability is that an object's attributes and relations are independent given its class. As a result, some distributions cannot be represented compactly. For example, an Ising model with

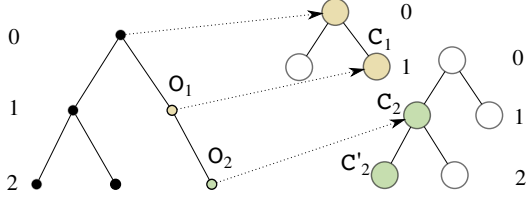


Figure 2: A part decomposition (left) with 6 objects and the class hierarchy (right) with two trees and 8 classes. Dashed arrows indicate which object was declared with which class. The numbers indicate the depth ℓ of the layers in the part decomposition (left) and of the trees of the class hierarchy (right).

arbitrary structure may require an exponential number of classes to represent.

INFERENCE

Object declarations introduce evidence in form of subclass, attribute, and relation facts and therefore influence the distribution of a TPKB by reducing the number of possible worlds with non-zero probability. The TPKB’s partition function is now the sum of the unnormalized probabilities of all possible worlds not contradicting the evidence. For a TPKB \mathcal{K} and a set of facts E , we write $Z_{\mathcal{K}}(E)$ for the corresponding partition function.

There are several possible types of queries. For instance, we can ask for $P(Q | E)$, where E and Q are sets of facts pertaining to objects in \mathcal{K} . The answers to these queries can always be tractably computed as ratios of partition functions:

$$P(Q | E) = \frac{Z_{\mathcal{K}}(Q \cup E)}{Z_{\mathcal{K}}(E)}.$$

We can also perform MAP inference, that is, ask for the most probable possible world given evidence. This is possible in a TPKB simply by replacing sums over subclasses with maxes and performing a traceback.

For real-world knowledge bases with millions of objects and tens of thousand of relations and attributes, disk-based and parallel inference algorithms with sublinear running time are required. We now introduce a novel inference algorithm with these characteristics. To the best of our knowledge, this is the first sublinear exact inference algorithm that scales to millions of objects and is fully implemented using a relational database. The algorithm performs several relational queries, which allows us to take advantage of query evaluation plans, index structures, and caching strategies. Intuitively, the structure of the query expressions resembles the corresponding sum-product network where sum nodes correspond to join-and-sum operations and product nodes to join-and-multiply operations.

There are two distinct inference algorithms. The first one computes the partition function and, as a by-product, all sub-partition functions for objects and their possible classes (stored in table T_Z) and the values of all sum nodes (stored in table T_{+S}). It does this once offline, in time linear in the size of the TPKB.

The second algorithm computes, given evidence E , the *ratio* of the partition function with and without E (the marginal probability of E), by propagating ratios of sub-partition functions with and without evidence.

To illustrate the algorithm, let us consider Figure 2 which depicts the part decomposition and the class hierarchy of a small TPKB. Every fact given by evidence E specifies either (a) the value of an attribute; (b) the truth value of a relation; (c) or the possible classes of an object. Evidence changes the sub-partition functions involving objects to which these attributes, relations, or subclasses apply. For instance, in Figure 2, if the sub-partition function for object O_2 and its possible class C'_2 is $e^{w_s} e^{w_p} e^{w_a} e^{w_r}$, and if evidence changes the factor pertaining to attributes from e^{w_a} to $e^{w'_a}$ and leaves all other factors the same, then the ratio of the sub-partition functions with and without evidence is $e^{w'_a} / e^{w_a}$. The evidence also impacts the ratios of sub-partition functions with and without evidence, $Z_{\mathcal{K}}^{(O_2, C)}(E) / Z_{\mathcal{K}}^{(O_2, C)}$, for every superclass C of C'_2 the object O_2 can be an instance of (here: class C_2). In Figure 2 these classes are depicted as shaded nodes in the second tree of the class hierarchy. Moreover, since O_2 is a part of object O_1 , the ratios of sub-partition functions for object O_1 in layer 1 and the class it was declared with also change, and so on. Fortunately, we only need to compute the ratios for objects that are ancestors of the object whose sub-partition function is impacted by the evidence.

The inference algorithm propagates these ratios of sub-partition functions bottom-up through the TPKB’s structure to obtain $Z_{\mathcal{K}}(E) / Z_{\mathcal{K}} = P(E)$. The relational queries computing these ratios of sub-partition functions only involve tables changed by the evidence and, therefore, the complexity of the algorithm depends on the evidence and not the size of the TPKB.

The inference algorithm is shown in Algorithm 1. To avoid confusion with relations and attributes of the TPKB, we use the terms *table* for relations and *columns* for attributes in the relational algebra. Table $T_Z(\text{Path}, \text{Class}, \text{subZ})$ with unique key $(\text{Path}, \text{Class})$ represents the TPKB’s sub-partition functions. Hence, $(O, C, Z) \in T_Z$ if and only if O can be an instance of class C and $Z_{\mathcal{K}}(O, C) = Z$. Analogously, we store the ratios of sub-partition function with and without evidence E in table T_Z^E with the same schema. We often write $T_Z(O, C)$ to denote the value of the unique sub-partition function for object O and its possible class C .

Layer ℓ of the part decomposition consists of all objects with depth (distance to the root node) ℓ . Table

Algorithm 1 Computes the probability of evidence E at each sub-partition function of objects in layer ℓ .

```

1:  $T \leftarrow T_{\text{Is}}^{\ell, E}$ 
2:  $k \leftarrow$  maximum height of class hierarchy
3: if  $\ell <$  height of part decomposition then
4:    $T' \leftarrow T_{\text{Part}}^{\ell} \bowtie T_Z^E$ 
5:    $T_{\times P}^E \leftarrow \pi_{(\text{Path}, \text{Class}, \text{subZ})}(\pi_{(\text{Path}, \text{Class})} \mathbf{G}_{\text{MUL}(\text{subZ})}(T'))$ 
6:    $T \leftarrow \otimes (T, T_{A_1}^E, \dots, T_{A_m}^E)$ 
7:    $T \leftarrow \otimes (T, T_{R_1}^E, \dots, T_{R_n}^E)$ 
8:    $T \leftarrow \otimes (T, T_{\times P}^E)$ 
9:    $T' \leftarrow \emptyset$ 
10: for each  $t \leftarrow (O, C, C', Z) \in T$  do
11:    $T_Z^E \leftarrow T_Z^E \cup \{(O, C, Z)\}$ 
12:   if  $C = C'$  then
13:      $T \leftarrow T - \{t\}$ 
14:   else if  $C$  is in layer  $k$  of the class hierarchy then
15:      $t \leftarrow (O, C, C', T_Z(O, C)(1 - Z))$ 
16:      $T' \leftarrow T' \cup \{t\}$ 
17:   if  $T \neq \emptyset$  then
18:      $T \leftarrow T - T'$ 
19:      $T' \leftarrow \oplus_{(\text{Path}, \text{Superclass})}(\otimes(T', T_{\text{Sub}}^{k-1}))$ 
20:      $T' \leftarrow \pi_{(\text{Path}, \text{Superclass} \rightarrow \text{Class}, \text{Class}', \text{subZ})}(T')$ 
21:     for each  $(O, C, C', Z) \in T'$  do
22:        $T \leftarrow T \cup (O, C, (T_{+S}(O, C) - Z)/T_{+S}(O, C))$ 
23:      $k \leftarrow k - 1$ 
24:   goto 6

```

$T_{\text{Is}}^{\ell, E}(\text{Path}, \text{Class}, \text{Class}')$ represents, for every object O in layer ℓ , its possible leaf classes C in the class hierarchy, and the classes C' it is declared with. The table, however, only stores objects and classes whose sub-partition function is impacted by the evidence E . These sub-partition functions can be efficiently computed using the structure of the TPKB.

Table T_{Part} represents the part decomposition of the TPKB. This table is used in lines 4 and 5 of Algorithm 1 to compute the table $T_{\times P}^E(\text{Path}, \text{Class}, \text{subZ})$ with $(O, C, Z) \in T_{\times P}^E$ if and only if object O is in layer ℓ of the part decomposition, O is declared as instance of class C , and Z is the product of the ratios of sub-partition functions with and without evidence of O 's parts according to C .

Lines 6, 7, and 8 evaluate the attribute, relation, and part product nodes of the TPKB's SPN (the three product nodes in Figure 1). The table $T_A^E(\text{Path}, \text{Class}, \text{subZ})$ stores, for each object and its possible classes, the ratio of the exponentiated summed weights of the attribute A with and without evidence E . The tables $T_{R_i}^E(\text{Path}, \text{Class}, \text{subZ})$ are the analogous tables for relations. The operator \otimes computes a left outer join on the columns Path and Class , and multiplies the values of the tables' subZ columns. The join queries of lines 5 and 6 only involve attribute and relation tables changed by the evidence.

Line 11 stores the computed ratios of sub-partition functions with and without evidence to table T_Z^E . In Line 12 it is tested, for each object O , whether we have reached the node of the class O was declared with. If this is the case, we remove the corresponding tuple from T since the computation of sub-partition functions that involve these objects is finished.

Lines 19 and 20 evaluate the SPN's sum nodes. First, we perform a left outer join of the tables T and T_{Sub}^k to multiply the children of each sum node with their respective weight. Table $T_{\text{Sub}}(\text{Class}, \text{Superclass}, \text{weight})$ represents the class hierarchy and the subclass weights. For each Superclass in layer k of the class hierarchy, $T_{\text{Sub}}^k(\text{Class}, \text{Superclass}, \text{weight})$ stores the weight for each of its subclasses. Second, the operator \oplus performs a grouping on the columns Path and Superclass and sums the values of column subZ for each of the groups. Line 20 renames the column Superclass to Class and projects out superfluous columns. The result of these operations are the values Z by which the sub-partition function at the sum nodes is reduced by the evidence. Line 22 computes the ratio of the sum node for object O and class C with and without evidence: $(T_{+S}(O, C) - Z)/T_{+S}(O, C)$.

Steps 6-22 are repeated until we have computed the ratio of the sub-partition function with and without evidence for every object in layer ℓ and each of its possible classes.

After running Algorithm 1 for $\ell = h, \dots, 0$, where h is the height the part decomposition, we have that $T_Z^E(O_0, C_0) = Z_{\mathcal{K}}(E)/Z_{\mathcal{K}} = P(E)$.

Theorem 3. *Let \mathcal{K} be a TPKB and let $n(E)$ be the number of sub-partition functions changed by a set of facts E . Algorithm 1 computes the probability $P(E)$ in time $O(n(E))$ and independent of the size of \mathcal{K} .*

Proof sketch. This can be shown with a nested proof by induction on (a) the height of the part decomposition and (b) the height of the class trees of each object. \square

Hence, inference time depends on the number of sub-partition functions changed by the evidence. This approach is different from caching previously computed values [10]. While we exploit this idea as well, the major advancement is the propagation of only those ratios of partition functions that are changed by the evidence.

Algorithm 1 can be parallelized by leveraging the structure of the TPKB's part decomposition and class hierarchy. First, the executions of Algorithm 1 on distinct trees of the class hierarchy are independent and can be performed in parallel. Moreover, for each layer of the part decomposition, the execution of the algorithm for an object in the layer is independent of the execution for any other object in the same layer. More generally, for any two objects, Algorithm 1 can be executed independently until we reach a

layer with the parent of the two objects. By using relational database systems for query processing, we can take advantage of existing parallelization strategies that detect these types of decompositions. We show below that this reduces the query evaluation time on multi-core architectures.

LEARNING

The TPKB’s parameters are the weights associated with attributes, relations, and subclasses. The training data is an i.i.d. sample of possible worlds. Hence, a sample $\{\mathbf{W}_1, \dots, \mathbf{W}_N\}$ is a multiset of N i.i.d. possible worlds. Note that, even if the sample consists of only one possible world (a mega-example), we can often still obtain robust parameter estimates due to the structure of a TPKB which features a large number of objects with the same class and, therefore, the same attributes and relations.

We derive closed-form maximum likelihood (ML) estimators of the parameters. We choose ML parameters that lead to a partition function with value 1. Since for each possible world, the classes for all objects are known and every object can only be an instance of one of a set of sibling classes, the expression for the likelihood is a product of factors. The log-likelihood, therefore, is a sum of terms and differentiating it with respect to a particular parameter leaves only the terms involving one particular attribute, at which point the MLE can be found in closed form for exponential family distributions.

Let $n(C)$ be the number of objects O in all possible worlds in the training data such that O is an instance of class C . For each class C , the probabilities of its subclasses S_1, \dots, S_n are governed by a categorical distribution. The standard ML estimates of $P(S_i|C)$, therefore, are $n(S_i)/n(C)$. Due to the parameterization of TPKBs, the MLEs are the logarithms of these estimates. Hence $\text{MLE}(w_i) = \log(n(S_i)/n(C))$.

Analogously, we can derive the ML estimates for attributes and relations. For categorical attributes, let $n_A(C, D)$ be the number of objects O in all possible worlds in the training data such that O is an instance of class C and has value D for attribute A . Moreover, let $n_A(C)$ be the number of objects O in all possible worlds such that O is an instance of class C and has *some* value for attribute A . If attribute A is declared for class C and for none of C ’s superclasses, then $\text{MLE}(\mathbf{u}(D)) = \log(n_A(C, D)/n_A(C))$. If attribute A is declared for class C , then it is also (implicitly) declared for each of its subclasses. Let S be such a subclass of C . Then

$$\text{MLE}(\mathbf{u}(D)) = \log \frac{n_A(S, D)/n_A(S)}{n_A(C, D)/n_A(C)}.$$

These are ML estimates precisely because we set the empirical probability, that is, the probability of an instance of class S having attribute value D , to the exponentiated sum of the weight estimates for S and all its superclasses. If the distribution is identical for a class and one of its subclasses,

```

Class Capital {
  parts s City, o Country;
  attributes distance  $\mathbb{R}$   $\mathbf{u}_1$ ;
}
Class Country {
  subclasses USA 1.1, ..., Italy 0.4;
  attributes area  $\mathbb{R}$   $\mathbf{u}_3$ , image Images  $\mathbf{u}_4$ ;
}
Class Paris {
  founded  $\mathbb{N}$   $\mathbf{u}_8$ ;
}

```

Table 2: Example class and object declarations of the TPKB based on the DBPEDIA ontology.

the MLE is 0 and we do not have to explicitly represent the attribute parameters for the subclass. This is advantageous as it results in sparser TPKBs. Note also that this parameterization is useful as we may not know an object’s most specific class during inference time.

The ML estimates for attributes with continuous distributions are derived analogously. As before, the ML estimate is based on the sufficient statistics for the attribute at class C and its superclass for which the attribute is also declared. For instance, for Gaussian variables, we compute the mean and standard deviation, and the MLE for the weight function for S subclass of C is

$$\text{MLE}(\mathbf{u}(x)) = \log \frac{(x - \mu_S)^2 / \sigma_S^2}{(x - \mu_C)^2 / \sigma_C^2}.$$

Relations are similar to Boolean attributes except that for relations we represent the positive case only. The weight of a relation at a class is the log ratio of the positive and negative count differences.

In order to smooth distributions that are defined on multiple levels of the class hierarchy, we recursively average the estimate for each class with the estimate for the superclass, with the combination weights determined by a form of cross-validation [18]. There are more sophisticated variants of shrinkage and other forms of hierarchical smoothing, but we leave this to future work.

So far, we have assumed that the training data is complete, that is, that for every object its class, its attribute values, and the truth values of relations are given. If attribute values, classes, or relation values are missing in the training data, we can use the EM algorithm [3], alternating between inferring the distributions of the missing values and finding the MLEs.

Structure learning involves the learning of the part decomposition and the class hierarchy. While this is beyond the scope of this paper, possible directions include adapting LearnSPN [9] and LearnRSPN [20] to TPKBs.

TPKBS FOR INFORMATION EXTRACTION

There are several information extraction projects such as NELL [2], DBPEDIA [16, 1], the Google Knowledge Vault [5], and REVERB [7]. These systems parse large amounts of text and semi-structured data sources. Typical extractions are of the form $\langle m_s, m_p, m_o \rangle$ where m_s is a *mention* of the subject, m_p of the predicate, and m_o of the object of a sentence. For instance, $\langle \text{Obama}, \text{graduated from}, \text{Columbia} \rangle$. A common problem is the grounding of these extractions in a canonical KB such as DBPEDIA. It is a challenging problem due to the inherent ambiguities of extractions and the problem of representing context. For instance, in the above example extractions, the mention Obama might refer to numerous individuals; Columbia to the district, the university, or a number of other possible entities; and graduated from might express that someone graduated from high school, college, etc. Only in conjunction with background knowledge is it possible to infer that the extraction mentions Barack Obama, 44th President of the US, and his being a graduate of Columbia University in New York.

We propose to use TPKBs to address the problem of joint entity linking. The TPKB has three objects s , p , and o , representing the latent canonical subjects, predicates, and objects of extractions. Every possible class of the object p represents a relation type. For instance, DBPEDIA's canonical relation type `alumni`, which expresses a person having graduated from a university, is such a class. The representation of relation types as classes of object pairs is advantageous due to the type-token distinction: in order to perform entity linking one has to model objects as classes, not constants, since one predicts the class membership of s and o from their attributes. Since objects are represented as classes, we cannot use TPKB relations and, instead, represent relation types as classes of object pairs.

The objects s and o are declared as parts of object p . The classes of s and o depend on the class in which they were declared. For instance, in class `alumni`, s is declared to be of class `Person` and o to be of class `University`. In class `capital`, s is declared to be of class `populatedPlace` and o of class `City`. Objects s and o are declared as parts of p only in classes representing relations without subrelations.

The class hierarchy consists of three trees, representing the class structure of objects o , s , and p , respectively. The leaf classes of the class trees of s and o are classes that represent canonical entities such as `BarackObama`. The structure of this TPKB allows us to model the dependencies between attributes, classes, and relation types in a tractable and principled way. It also allows us to disambiguate extractions based on geographical, temporal, and other numerical attributes in a principled manner.

Once the parameters of the TPKB are estimated, we can perform entity linking. For instance, given the extraction $\langle \text{Obama}, \text{graduated from}, \text{Columbia} \rangle$ we compute the probability of the mention Obama referring to the canonical entity `BarackObama` using the query $P(Q | E)$ with $Q = \{\text{Is}(s, \text{BarackObama})\}$ and $E = \{\text{mention}(s, \text{Obama}), \text{mention}(p, \text{graduated from}), \text{mention}(o, \text{Columbia})\}$, where `mention` is an attribute modeling mentions.

Experiments

We derived the class trees for objects s and o directly from the DBPEDIA ontology [16, 1]. The class tree for object p is derived from the relations in DBPEDIA, that is, every relation type in DBPEDIA is represented with a class object that p can be an instance of. In addition to the classes and relation types taken from the DBPEDIA ontology, we created one leaf class for each canonical entity. For instance, the DBPEDIA entity `BarackObama` is a leaf in the class hierarchy with attributes `birthYear`, `mention`, etc. Table 2 depicts a small selection of class declarations.

For several attributes such as `birthYear`, `elevation`, `geocoordinates`, etc. we used data from DBPEDIA to learn the parameters. To learn the attribute distributions for leaf classes, that is, classes modeling entities, we assumed a uniform distribution if, for one entity, more than one value was given for a particular attribute. For instance, if `Arnold` has values 1947 and 1946 for attribute `birthYear` then, following the maximum-likelihood principle, we assume that both values have probability 0.5. For the other classes, we pooled attribute values of the instances of each class and used histograms to model these distributions. For the attribute `mention` modeling mentions we used the WIKIPREP tool [8] to compute the conditional distribution of a canonical entity given a mention. We also learned the parameters of attributes for classes representing relation types. For instance, we introduced the attribute `diffBirthYear` which models a distributions over the absolute value of birth year differences.

The number of parameters of the resulting TPKB exceeds 1 billion and we model more than 1 million objects. We performed inference by running Algorithm 1 using a MySQL database system. Each query was answered in less than one second. The reduction in running time for computing the partition function was 31% for 2 cores, 47% for 4 cores, 49% for 6 cores, and 50% for 8 cores. We evaluated the learned TPKB empirically on two important problem classes.

Entity Linking

For the entity linking experiments we used an existing gold standard [6] for aligning NELL triples to DBPEDIA entities. For each NELL triple of the form $\langle m_s, m_p, m_o \rangle$, we

NELL relation	Precision@1			Recall@1		
	WL	TPKB1	TPKB2	WL	TPKB1	TPKB2
ActorStarredInMovie	0.81	0.85	0.92	0.82	0.82	0.31
AgentcollaboratesWithAgent	0.82	0.83	0.91	0.86	0.87	0.20
AnimalIsTypeofAnimal	0.86	0.86	0.99	0.86	0.86	0.71
AthleteLedSportsTeam	0.89	0.91	0.93	0.86	0.87	0.37
BankBankInCountry	0.82	0.87	0.93	0.76	0.76	0.10
CityLocatedInState	0.80	0.85	0.95	0.81	0.82	0.64
BookWriter	0.82	0.83	0.92	0.81	0.82	0.73
CompanyAlsoKnownAs	0.71	0.71	1.00	0.58	0.61	0.49
PersonLeadsOrganization	0.79	0.81	0.92	0.75	0.71	0.68
TeamPlaysAgainstTeam	0.81	0.81	1.00	0.81	0.83	0.70
WeaponMadeInCountry	0.88	0.91	1.00	0.88	0.89	0.65
LakeInState	0.90	0.91	1.00	0.90	0.90	0.84

System	Prec@1	Rec@1
PARIS	91.9	73.8
TPKB1	85.3	75.2
TPKB2	92.1	74.0

Table 3: Results for entity resolution experiments (left) and entity linking experiments (right). Bold numbers indicate significance (paired t-test; $p < 0.05$) compared to the baselines.

performed the following two queries. The query (TPKB1)

$$P(\{Is(s, x), Is(o, y)\} \mid \{mention(s, m_s), mention(o, m_o)\})$$

asks for the marginal probability of s being entity x and o being entity y , conditioned on the NELL triple’s mentions. The result is a list of substitutions for variables x and y and the corresponding marginal probabilities.

We proceeded to manually align the subject and object mention with their classes in the TPKB. For instance, for the triple $\langle \text{Obama, graduated from, Columbia} \rangle$, we aligned Obama to the class Politician and Columbia to the class University. We then performed the previous query except that we added the set $\{Is(s, C_s), Is(o, C_o)\}$ to the evidence (TPKB2). This query retrieves the probabilities of s being entity x and o being entity y , conditioned on the mentions and their class memberships. The results are given in Table 3 (left) and compared with a baseline given in [6]. There are other possible baselines such as matrix factorization methods [21, 29]. However, it is non-trivial to use these methods for the entity linking problem. This is because we have only mentions of entities and no data that links these mentions to relations and attributes in the canonical KBs. Precision@ k and Recall@ k is computed by retrieving the k most probable answer tuples. The results of the TPKB outperform the WikiLink baseline [6] substantially and are as efficient to compute. TPKB2, however, should only be used if high precision results are required.

Entity Resolution

Entity resolution is the problem of determining whether two entities in two knowledge bases are equivalent. To evaluate the TPKB for entity linking we repeated the experiment of linking YAGO [12] to DBPEDIA conducted to evaluate the PARIS matching system [30]. Both knowledge bases use Wikipedia identifiers for their objects which gives us a large set of gold standard pairs for evaluation purposes. We manually aligned a set of attributes (datatype

properties) and classes between YAGO and the learned TPKB. We sampled 100,000 objects in YAGO, retrieved the aligned attributes for each object (labels, numerical attributes, etc.) and ran, for each entity, the query above, where we condition on the given value of attribute mention (TPKB1); and all other attributes for which a manual alignment existed (TPKB2). Table 3 shows that TPKBs are able to accurately link entities and compare favorably with specialized algorithms.

The learned TPKB is both efficient and accurate, outperforming existing problem-specific approaches.

CONCLUSION

We presented a novel inference algorithm for TPKBs that is disk-based, parallel, and sublinear. We also derived closed-form maximum likelihood estimates for TPKB parameters. We used these results to learn a large TPKB from multiple data sources and applied it to information extraction and integration problems. The TPKB outperformed existing algorithms in accuracy and efficiency.

Future work will be concerned with more sophisticated smoothing approaches, the comparison of different learning strategies, and the problem of structure learning. We also plan to apply TPKBs to a wide range of problems that benefit from tractable probabilistic knowledge representations.

An open-source implementation of TPKBs is available at alchemy.cs.washington.edu/lite2.

ACKNOWLEDGMENTS

This research was partly funded by ONR grants N00014-13-1-0720 and N00014-12-1-0312, and AFRL contract FA8750-13-2-0019. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ONR, AFRL, or the United States Government.

References

- [1] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009.
- [2] 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 *Proc. AAAI*, pages 1306–1313, 2010.
- [3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.
- [4] P. Domingos and W. A. Webb. A tractable first-order probabilistic logic. In *Proc. AAAI*, pages 1902–1909, 2012.
- [5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmman, S. Sun, and W. Zhang. Knowledge Vault: A web-scale approach to probabilistic knowledge fusion. In *Proc. SIGKDD*, pages 601–610, 2014.
- [6] A. Dutta, C. Meilicke, M. Niepert, and S. P. Ponzetto. Integrating open and closed information extraction: Challenges and first steps. In *Proc. NLP-DBPEDIA@ISWC*, 2013.
- [7] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: The second generation. In *Proc. IJCAI*, pages 3–10, 2011.
- [8] E. Gabrilovich and S. Markovitch. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In *Proc. IJCAI*, pages 1606–1611, 2007.
- [9] R. Gens and P. Domingos. Learning the structure of sum-product networks. In *Proc. ICML*, pages 873–880, 2013.
- [10] V. Gogate and P. Domingos. Probabilistic theorem proving. In *Proc. UAI*, pages 256–265, 2011.
- [11] V. Gutiérrez-Basulto, J. C. Jung, C. Lutz, and L. Schröder. A closer look at the probabilistic description logic prob-el. In *Proc. AAAI*, pages 197–202, 2011.
- [12] 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.
- [13] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *Proc. AAAI*, pages 381–388, 2006.
- [14] A. Kimmig, S. H. Bach, M. Broecheler, B. Huang, and L. Getoor. A short introduction to probabilistic soft logic. In *Proc. NIPS Workshop on Probabilistic Programming: Foundations and Applications*, 2012.
- [15] J. Lehmann, D. Gerber, M. Morsey, and A.-C. N. Ngomo. Defacto - deep fact validation. In *Proc. ISWC*, pages 312–327, 2012.
- [16] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer. DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, 2014.
- [17] T. Lukasiewicz. Probabilistic description logic programs. *Int. J. Appr. Reas.*, 45, 2007.
- [18] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proc. ICML*, pages 359–367, 1998.
- [19] N. Nakashole, M. Sozio, F. M. Suchanek, and M. Theobald. Query-time reasoning in uncertain RDF knowledge bases with soft and hard rules. pages 15–20, 2012.
- [20] A. Nath and P. Domingos. Learning the structure of relational sum-product networks. In *Proc. AAAI*, pages 873–880, 2013.
- [21] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing YAGO: Scalable machine learning for linked data. In *Proc. WWW*, pages 271–280, 2012.
- [22] M. Niepert, J. Noessner, and H. Stuckenschmidt. Log-linear description logics. In *Proc. IJCAI*, pages 2153–2158, 2011.
- [23] F. Niu, C. Ré, A. Doan, and J. W. Shavlik. Tuffy: Scaling up statistical inference in Markov logic networks using an RDBMS. *PVLDB*, 4(6):373–384, 2011.
- [24] J. Noessner and M. Niepert. Elog: A probabilistic reasoner for OWL EL. In *Proc. RR*, pages 281–286, 2011.
- [25] J. Noessner, M. Niepert, and H. Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. In *Proc. AAAI*, pages 739–745, 2013.
- [26] D. Poole. The independent choice logic and beyond. In *Probabilistic inductive logic programming*. Springer-Verlag, 2008.
- [27] H. Poon and P. Domingos. Sum-product networks: A new deep architecture. In *Proc. UAI*, pages 337–346, 2011.
- [28] L. D. Raedt, A. Kimmig, and H. Toivonen. Problog: a probabilistic prolog and its application in link discovery. In *Proc. IJCAI*, pages 2468–2473, 2007.
- [29] S. Riedel, L. Yao, B. M. Marlin, and A. McCallum. Relation extraction with matrix factorization and universal schemas. In *Proc. HLT-NAACL*, pages 74–84, 2013.
- [30] F. M. Suchanek, S. Abiteboul, and P. Senellart. PARIS: Probabilistic alignment of relations, instances, and schema. *PVLDB*, 5(3):157–168, 2011.
- [31] D. Suciu, D. Olteanu, R. Christopher, and C. Koch. *Probabilistic Databases*. Morgan & Claypool Publishers, 1st edition, 2011.
- [32] G. Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Proc. NIPS*, pages 1386–1394, 2011.
- [33] D. Z. Wang, M. J. Franklin, M. N. Garofalakis, and J. M. Hellerstein. Querying probabilistic information extraction. *PVLDB*, 3(1):1057–1067, 2010.
- [34] W. Webb and P. Domingos. Tractable probabilistic knowledge bases with existence uncertainty. In *Proc. AAAI Workshop on Statistical Relational AI*, 2013.