Interactive Reasoning in Uncertain RDF Knowledge Bases

Timm Meiser, Maximilan Dylla, Martin Theobald
Max Planck Institute for Informatics
Saarbrücken, Germany
tmeiser@mpi-inf.mpg.de, mdylla@mpi-inf.mpg.de, mtb@mpi-inf.mpg.de

ABSTRACT

Recent advances in Web-based information extraction have allowed for the automatic construction of large, semantic knowledge bases, which are typically captured in RDF format. The very nature of the applied extraction techniques however entails that the resulting RDF knowledge bases may face a significant amount of incorrect, incomplete, or even inconsistent (i.e., *uncertain*) factual knowledge, which makes query answering over this kind of data a challenge. Our reasoner, coined URDF[1], supports SPARQL queries along with rule-based, first-order predicate logic to infer new facts and to resolve data uncertainty over millions of RDF triplets directly *at query time*. We demonstrate a fully interactive reasoning engine, combining a Java-based reasoning backend and a Flash-based visualization frontend in a dynamic client-server architecture. Our visualization frontend provides interactive access to the reasoning backend, including tasks like *exploring* the knowledge base, rule-based and statistical *reasoning*, *faceted browsing* of large query graphs, and *explaining answers* through lineage.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process*

General Terms

Algorithms, Design, Management

Keywords

Uncertain RDF, Interactive Reasoning, Visualization

1. INTRODUCTION

A plethora of free semantic knowledge bases (including DBpedia and its inter-linked data sources, KnowItAll [8], ReadTheWeb [7] and YAGO [9, 19]) as well as commercial endeavors (including FreeBase.com, TrueKnowledge.com, Sig.ma, EntityCube, Wolfram Alpha, or Google Squared) have emerged out of a variety of Web-based information extraction frameworks in recent years. Many of these projects employ Wikipedia as common source for extraction, with their techniques ranging from extracting contents of highly structured infoboxes [8] to free-text information extraction and natural language processing [7]. Due to the very nature of the applied (unsupervised or semi-supervised) extraction techniques, but also due to the collaborative and partly highly transient way in which Web sources like Wikipedia evolve, the resulting knowledge bases may contain a significant amount of false or contradictory facts. On the other hand, they often remain highly incomplete and may miss a large amount of facts about real-world entities. As a potential remedy, the underlying extraction tools sometimes provide confidence values attached to the extracted facts. Yet it remains far from obvious how to incorporate issues like incompleteness, incorrectness, and inconsistency into an interactive information system, in order to provide consistent query answers over this type of uncertain data.

Our reasoning engine provides a SPARQL-compliant query interface and employs Datalog-style, deductive grounding techniques of first-order logical rules (Horn clauses). Specifically, we distinguish between *soft rules* and *hard rules* for

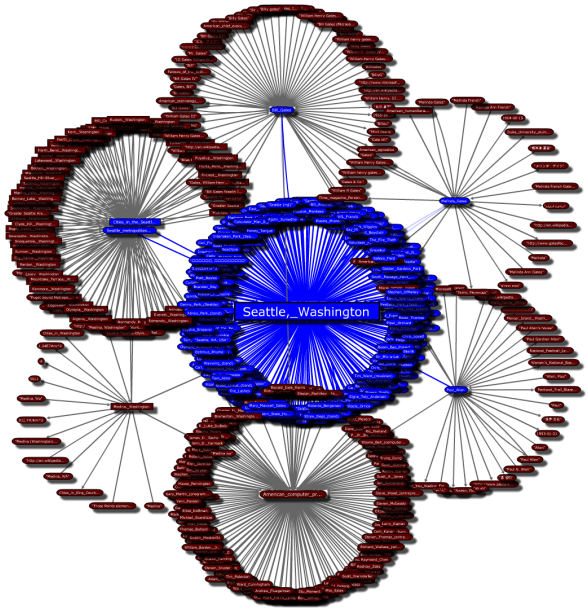

Figure 1: An exploration of the knowledge base around the entity "Seattle,_Washington", showing both base and derived facts.

[1] `http://urdf.mpi-inf.mpg.de`

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...$10.00.

reasoning. Soft rules can be employed in a deductive fashion to derive new facts from existing ones (including facts derived from other deduction steps), or to reinforce the confidence of existing base facts. Hard rules, on the other hand, enforce additional consistency constraints among both base and derived facts. The resulting logical dependencies among base and derived facts are captured via Boolean lineage formulas. Processing queries consists of two phases: 1) grounding the query against the rules and base facts contained in the knowledge base, and 2) a subsequent (propositional or probabilistic) consistency reasoning step. All reasoning-related computations—including grounding and consistency reasoning—are handled by URDF at query time.

2. RELATED WORK

State-of-the-art RDF engines (see, e.g., [1, 11]) primarily focus on conjunctive queries on top of a relational encoding of RDF data. These engines generally follow a deterministic data and query processing model and do not have a notion of rule-based inference or uncertain reasoning. Approaches for managing uncertainty in the context of probabilistic databases [3, 4, 5, 18, 17] focus on relational data with fixed schemata and often employ strong independence assumptions among data objects (the "base tuples"). ULDBs [4] provide a lineage-based representation formalism for probabilistic data, which has been shown to be closed and complete under any combination of SQL operators and across arbitrary levels of materialized views. Here, the lineage (aka. "history" in [18] or "repair-key" operator in [3]) of a derived tuple is captured as a Boolean formula, which recursively unrolls the logical dependencies from the derived tuple back to the base tuples. In probabilistic databases, SQL is employed for formulating queries and for defining views. SQL statements generally yield "hard" Boolean constraints among tuples, but lack the notion of "soft" dependencies among data items. Moreover, capturing correlated tuples [16] with probabilistic graphical models such as Bayesian Nets [6, 21] and Markov Random Fields [17] is finding increasing attention also in the database community. Also in the context of these graphical models, lineage remains the key for a closed representation model [10].

Statistical Relational Learning (SRL) has been gaining an increasing momentum in the machine learning, database, and knowledge management communities recently. Markov Logic Networks (MLNs) [14] are one of the most generic approaches for combining first-order logic and probabilistic graphical models. MLNs work by grounding a set of first-order logical rules against a knowledge base, and by sampling states ("worlds") over a Markov network that represents the grounded (i.e., propositional) formulas. Markov Logic however does not easily scale to very large knowledge bases. Even the most efficient, recent database approaches which adopt inference methods for probabilistic graphical networks, such as [21], [12] or [22], are designed for batch processing and are not well suited for interactive querying.

3. SYSTEM ARCHITECTURE

The reasoning backend is deployed under an Apache Tomcat Web server and accesses a PostgreSQL database which captures the knowledge base. Our visualization frontend has been developed under Adobe Flex to run via Flash in almost any common Web browser, thus supporting a variety of operating systems (e.g., Windows, Linux, Android) and clients (e.g., laptops, tablets). Using the Flex framework (via a combination of ActionScript and MXML components), our visualization follows a so-called Rich Internet Application (RIA) architecture and delivers a desktop-like look-and-feel in a regular browser. The data transfer between the visualization frontend and the reasoning backend is handled by the Adobe-specific data exchange service BlazeDS. This service allows for a fast binary data exchange between server and client via object serialization. Moreover, we integrate the Flare[2] visualization libraries for rendering graphs with different layouts. An example graph, as it can be obtained by exploring the YAGO [9, 19] knowledge base around the entity "Seattle,_Washington", is depicted in Figure 1.

4. REASONING BACKEND

We adopt the common *possible worlds* [2] model as basis for our data model and for defining the semantics of queries. Intuitively, every fact and first-order soft rule in the knowledge base corresponds to a binary random variable, which may hold in the knowledge base (i.e., be "true") with some marginal probability. Via soft and hard rules, facts may be correlated with each other, such that rules (when grounded) may impose intricate dependencies among facts. In the following, we formally consider a knowledge base $\mathcal{KB} = \langle \mathcal{F}, \mathcal{C}, \mathcal{S} \rangle$ as a triplet consisting of propositional facts $\mathcal{F}$, first-order soft rules $\mathcal{C}$ (clauses used for deduction), and first-order hard rules $\mathcal{S}$ (strict consistency constraints).

Base Facts. We presume a set of RDF base facts $\mathcal{F}_{db} \subseteq \mathcal{F}$ to be contained as extensional knowledge in our database. These facts are not derived from other facts and thus are assumed as input (e.g., from an information extraction tool). Base facts may be uncertain, i.e., every fact $f \in \mathcal{F}_{db}$ has a confidence weight $w(f) \in [0, 1]$ attached, which corresponds to a prior probability of this fact being true.

Soft Rules. To tackle the inherent incompleteness of $\mathcal{F}_{db}$, we also assume a set of soft deduction rules $\mathcal{C}$ as input to our knowledge base. Just like base facts, soft rules may be uncertain, i.e., they may hold only for a fraction of entities in the real world, which is again reflected by a confidence weight $w(C) \in [0, 1]$ for each soft rule $C \in \mathcal{C}$. Soft rules have the shape of Datalog-style (but definite) Horn clauses. They could state, for example, that "married couples usually live at the same place", with a confidence of 0.38 of being correct, which we would write as follows:

$$livesIn(p_1, loc_1) \leftarrow marriedTo(p_1, p_2) \wedge livesIn(p_2, loc_1) \ _{[0.38]} \quad (1)$$

Grounding the rules in $\mathcal{C}$ over the facts $\mathcal{F}_{db}$ results in new intensional knowledge, which we denote by the set $\mathcal{F}_{in} \subseteq \mathcal{F}$. For intensional facts, we do not know their marginal probabilities upfront, i.e., we need to compute their posterior probabilities via lineage.

Hard Rules. The set of base and derived facts may contain inconsistent information. For example, if we observe two different birth dates for a person, clearly something went wrong either during extraction or when reasoning with soft rules. We can formally express such a consistency constraint as follows:

$$(date_1 = date_2) \leftarrow bornOn(p_1, date_1) \wedge bornOn(p_1, date_2) \quad (2)$$

[2] `http://flare.prefuse.org`

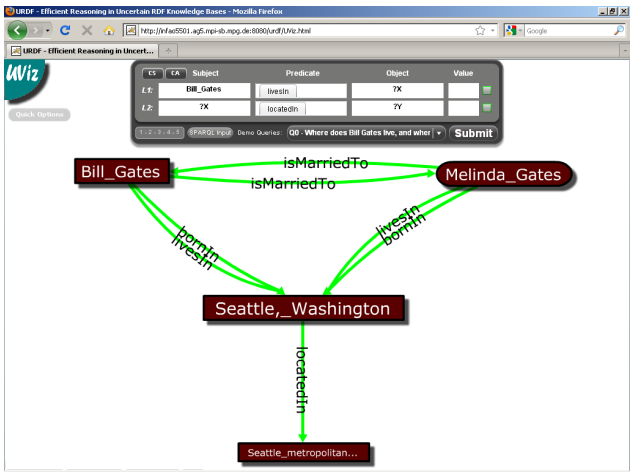

Figure 2: Dependency graph for the query *"Where does Bill Gates live, and where is this place located?"*

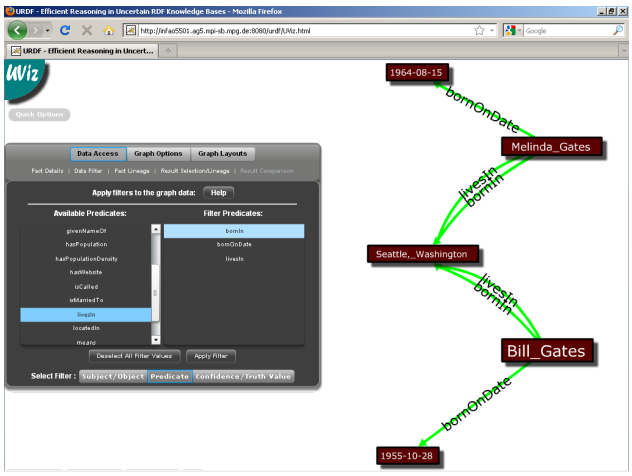

Figure 3: Faceted browsing panel and a filtered version of the graph of Figure 1.

In contrast to soft rules, hard rules (i.e., consistency constraints) have no intensional predicate as head literal and are not used for deriving new facts. Consistency constraints either have an arithmetic predicate (as above) or the constant *false* as head literal.

Lineage. The reasoner employs SLD resolution for grounding soft rules, while hard rules are not used for deduction and thus are grounded separately (see [20] for details on the grounding algorithm). Tracing lineage through SLD resolution resolves to capturing the logical dependencies of derived facts back to the base facts. In analogy to [4, 17], we represent lineage of derived facts as Boolean formulas.

- *Positive Lineage.* SLD resolution with soft rules creates positive lineage. We obtain *conjunctive* lineage whenever we combine (ground) two or more positive literals that co-occur in the body of a rule. We obtain *disjunctive* lineage whenever two or more rules (including base facts) imply the same derived fact.
- *Negations.* Hard rules offer a way to formulate negations in order to constrain the set of possible worlds. A special type of hard rules we consider in [20] partitions $\mathcal{F}$ into disjoint sets of mutually exclusive facts. In this case, each fact in such a "competitor set" may only be true if *none* of the facts it is in conflict with are true.

Propositional Reasoning. In propositional reasoning, we aim to find a truth assignment to facts in $\mathcal{F}$ that is consistent with all hard rules. In [20], we present a constrained and weighted MaxSAT solver that is specifically tailored to the combination of soft and hard rules we consider in our reasoning framework. It operates on a Boolean formula in conjunctive normal form (CNF), which is constructed as a conjunction of all clauses (including base facts, soft, and hard rules) that are grounded in response to a query.

Probabilistic Reasoning. In contrast to propositional reasoning, probabilistic reasoning does not consider only a single consistent truth assignment to facts (i.e., *one* possible world), but—at least conceptually—*all* consistent worlds. In other words, the marginal probability of a derived fact $f \in \mathcal{F}_{in}$ is nothing else but the sum of probabilities of consistent worlds (i.e., those that do not violate any hard rule), for which the Boolean lineage formula of f evaluates to true. For general lineage formulas, probabilistic inference is known to be #P-complete. We thus employ Gibbs sampling or MC-SAT (depending on the shape of the hard rules, see [13]) for approximate inference when exact inference is intractable.

5. VISUALIZATION FRONTEND

Exploration Mode. This is our default visualization mode. For entering a query, the user can switch between a plain-text input field (to enter a SPARQL-style join pattern) and a structured form field (to enter individual subject-predicate-object patterns into predefined forms). When the query is submitted, the reasoner is triggered, and a graph containing all isomorphic embeddings of the query pattern in the knowledge base (consisting of both base and derived facts) is displayed. In addition, the complete dependency graph of the query can be shown. The dependency graph contains all facts that are relevant for answering the query, including facts used for grounding the rules (see Figure 2). The graph layouts in exploration mode correspond to entity-relationship graphs, where the nodes represent the entities in the knowledge base, and the edges represent RDF facts. Edges are colored in *green* or *red*, according to the truth value assigned to the facts by the MaxSAT solver. Different alpha values for the edges visually encode the confidences (in the range of $[0, 1]$) of the facts as they were computed by the possible-worlds-based probabilistic model. The visualization frontend supports the radial and force-directed layouts of Flare, which can both be customized in various ways. Moreover, several options to control and interact with the graph (such as panning and zooming) are supported. By double-clicking on a node, we can dynamically load more facts from the knowledge base in order to iteratively explore the neighborhood of a particular entity. A possible result of such an exploration is shown in Figure 1.

Faceted Browsing. The graphs displayed in exploration mode can quickly grow fairly large. The faceted browsing filter automatically extracts facets from the current graph by grouping the facts along various dimensions. An example for this functionality is depicted in Figure 3, where the user has specified a filter on the predicates *bornIn*, *isMarriedTo* and *livesIn* to restrict the graph of Figure 1, thus showing only facts that match either one of those predicates.

Explanation Mode. By clicking on an edge (i.e., fact) for which lineage is available, the visualization switches into

explanation mode. In this mode, the graph granularity changes from an entity-relationship graph into a lineage graph, where the nodes represent entire facts, and the edges denote the logical dependencies among those facts. Additional conjunction (AND) and disjunction (OR) nodes group outgoing edges according to the logical dependencies among facts obtained from the deductive grounding steps. Lineage may not be cyclic, but it may form a DAG structure over the grounded facts. For better readability, lineage DAGs are flattened into trees, where a same fact may occur in multiple branches of the tree. Fact nodes are again colored in *green* and *red*, according to the truth values assigned by the MaxSAT solver. An example explanation tree rooted at the fact *livesIn(Bill_Gates,Seattle,_Washington)* is shown in Figure 4. The upper OR node indicates that several derivations for this fact exist. The second level of the expanded lineage branch has been obtained from the same soft rule as shown in Rule (1). Several sub-trees have been collapsed for better readability (as indicated by the dots).

Comparison Mode. Finally, the comparison feature allows for comparing the answers of any two subsequently executed queries, including changes obtained from updating soft or hard rules. Nodes and edges, which are only contained in the first answer set, are colored in *black* borders; those contained in the second set are colored in *white* borders; and those shared by both sets have *yellow* borders.

6. DEMONSTRATION DESCRIPTION

Our demo currently runs interactively on the latest version of the YAGO [9] knowledge base, consisting of more than 10 million entities and more than 80 million facts about these entities. To demonstrate all features of our interactive reasoning framework, we have prepared more than 10 SPARQL queries, ranging from single-literal queries like *"Who is Woody Allen married to?"* to multi-join query patterns which require intricate reasoning steps. The demo will focus on queries and rules about people and locations (such as universities and people's advisors or alma maters, merged with gazetteer data from *geonames.org*), as these domains are particular strengths of YAGO. Following the scenario shown in Figures 1–4, we start by showing the difference between results obtained from plain SPARQL-style query processing in comparison to rule-based reasoning. We then iteratively explore the knowledge base around the initial result entities, thus constantly growing the graph. As the information displayed by the expanded graph becomes increasingly cluttered, we can take advantage of the faceted browsing filter which provides a fine-grained control over the visualized information. We finally demonstrate how the explanation and comparison modes can be employed to effectively visualize how rule updates affect the query answers. At all steps of the demonstration, there will be ample opportunity for the audience to propose new queries, update rules, and try out different features of the reasoner.

7. REFERENCES

[1] D. J. Abadi, A. Marcus , S.R. Madden, and K. J. Hollenbach. Scalable semantic web data management using vertical partitioning. *VLDB*, 2007.

[2] S. Abiteboul, P. Kanellakis, and G. Grahne. On the representation and querying of sets of possible worlds. *Theor. Comput. Sci.*, 78(1), 1991.

[3] L. Antova, C. Koch, and D. Olteanu. MayBMS: Managing incomplete information with probabilistic world-set decompositions. *ICDE*, 2007.

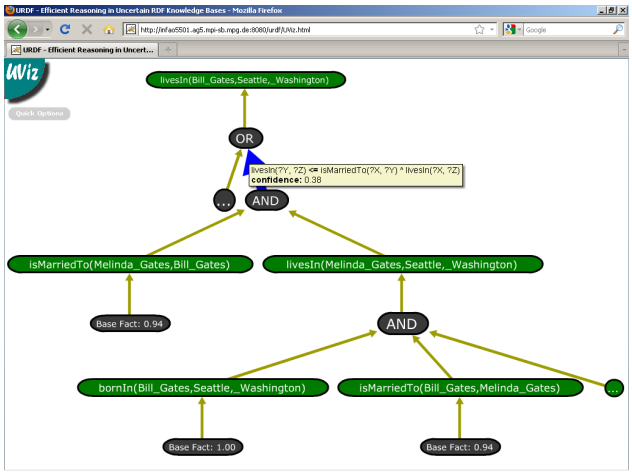

Figure 4: Explanation tree for the derived fact *livesIn(Bill_Gates, Seattle,_Washington).*

[4] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom. ULDBs: Databases with uncertainty and lineage. *VLDB*, 2006.

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

[6] H. C. Bravo and R. Ramakrishnan. Optimizing MPF queries: decision support and probabilistic inference. *SIGMOD*, 2007.

[7] A. Carlson, J. Betteridge, R. C. Wang, E. R. H. Jr., and T. M. Mitchell. Coupled semi-supervised learning for information extraction. *WSDM*, 2010.

[8] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-scale information extraction in KnowItAll. *WWW*, 2004.

[9] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, and G. Weikum. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. *WWW*, 2011.

[10] B. Kanagal and A. Deshpande. Lineage processing over correlated probabilistic databases. *SIGMOD*, 2010.

[11] T. Neumann and G. Weikum. RDF-3X: a RISC-style engine for RDF. *PVLDB*, 1(1), 2008.

[12] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: scaling up statistical inference in Markov logic networks using an RDBMS. *PVLDB*, 4(6), 2011.

[13] H. Poon, P. Domingos, and M. Sumner. A general method for reducing the complexity of relational inference and its application to MCMC. *AAAI*, 2008.

[14] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1-2), 2006.

[15] A. D. Sarma, M. Theobald, and J. Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. *ICDE*, 2008.

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

[17] P. Sen, A. Deshpande, and L. Getoor. PrDB: managing and exploiting rich correlations in probabilistic databases. *VLDB J.*, 18(5), 2009.

[18] S. Singh, C. Mayfield, R. Shah, S. Prabhakar, S. Hambrusch, J. Neville, and R. Cheng. Database support for probabilistic attributes and tuples. *ICDE*, 2008.

[19] F. M. Suchanek, G. Kasneci, and G. Weikum. YAGO - A Large Ontology from Wikipedia and WordNet. *Elsevier Journal of Web Semantics*, 6(3), 2008.

[20] M. Theobald, M. Sozio, F. Suchanek, and N. Nakashole. URDF: Efficient Reasoning in Uncertain RDF Knowledge Bases with Soft and Hard Rules. Tech-Report MPI-I-2010-5-002, 2010.

[21] D. Z. Wang, E. Michelakis, M. N. Garofalakis, and J. M. Hellerstein. BayesStore: managing large, uncertain data repositories with probabilistic graphical models. *PVLDB*, 1(1), 2008.

[22] M. L. Wick, A. McCallum, and G. Miklau. Scalable Probabilistic Databases with Factor Graphs and MCMC. *PVLDB*, 3(1), 2010.