# Probabilistic Knowledge Discovery for the Web of Data [PREPRINT]

Tommaso Soru
tsoru@informatik.uni-leipzig.de

Diego Esteves
esteves@informatik.uni-leipzig.de

Edgard Marx
marx@informatik.uni-leipzig.de

Axel-Cyrille Ngonga Ngomo
ngonga@informatik.uni-leipzig.de

AKSW, Fakultät für Mathematik und Informatik
Universität Leipzig, Germany

## ABSTRACT

Among the approaches for knowledge discovery in non-relational databases which have been proposed so far, many are based on Markov Logic Networks (MLNs). MLNs join probabilistic modeling with first-order logic and have been shown to integrate well with the Semantic Web foundations. However, the use of their original implementation on large datasets has been discouraged, as current frameworks suffer from a high computational complexity. In this paper, we fill this gap by proposing MANDOLIN, a massively-parallel framework for knowledge discovery specifically on RDF datasets. Our approach relies on state-of-the-art techniques for the subtasks of rule mining, weight learning, grounding, and inference computation. Making use of RDFS/OWL semantics, MANDOLIN imports knowledge from referenced graphs and creates similarity relationships among similar literals. We show that this additional information allows the discovery of links even between different knowledge bases. Finally, we show that our approach is more scalable than other MLN frameworks and achieves at least comparable results with respect to other statistical-relational-learning algorithms on link prediction.

## Keywords

Linked Data; Machine Learning; Statistical Relational Learning; Knowledge Discovery; Markov Logic Networks; Probabilistic Knowledge Bases; Link Prediction.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

## 1. INTRODUCTION

The *Linked Data cloud* has grown considerably since its inception. To date, the total number of facts exceeds 130 billions, spread in over 2,500 available datasets.[1] This massive quantity of data has thus become an object of interest for disciplines as diverse as Machine Learning (ML) [52], Evolutionary Algorithms [54, 33], Generative Models [2], and Statistical Relational Learning (SRL) [51]. In particular, the main objective of the application of such algorithms is to address the fourth Linked Data principle. This ten-year-old principle preaches to the Semantic Web community to *"include links to other URIs, so that they [the visitors] can discover more things"* [1]. Two years later, Domingos et al. proposed Markov Logic Networks (MLNs) [43]—a well-known approach to Knowledge Discovery in knowledge bases—to be a promising framework for the Semantic Web [8]. Bringing the power of probabilistic modeling to first-order logic, MLNs associate a weight to each formula (i.e., first-order logic rule) and are able to natively perform probabilistic inference. Several tools based on MLNs have been designed so far [21, 39, 41, 3]. Yet, none of the existing MLN frameworks develops the entire pipeline from the generation of rules to the discovery of new relationships in a dataset. Moreover, the size of the Web of Data represents today an enormous challenge for such learning algorithms, which often have to be re-engineered in order to scale to larger datasets. In the last years, this problem has been tackled by proposing algorithms that benefit of massive parallelism. Approximate results with some confidence degree have been preferred over exact ones, as they often require less computational power, yet leading to acceptable performances.

In this paper, we present MANDOLIN, a framework based on Markov Logic Networks for the Discovery of Links. To the best of our knowledge, MANDOLIN is the first framework to implement the entire workflow for

[1] http://lodstats.aksw.org/

knowledge discovery on RDF datasets. We base our research on top of state-of-the-art techniques for the subtasks of rule mining, weight learning, grounding, and inference computation. Making use of RDFS/OWL semantics, Mandolin can (i) import knowledge from referenced graphs, (ii) compute the forward chaining, and (iii) create similarity relationships among similar literals. We show that this additional information allows the discovery of links even between different knowledge bases. Finally, we show that our approach is more scalable than other MLN frameworks. We evaluate Mandolin on two benchmark datasets for link prediction and show that it can achieve comparable results w.r.t. other SRL algorithms and outperform them on two accuracy indices.

This paper is structured as follows. The next section presents the related work. We then start with some preliminaries in Section 3; afterwards, we describe Mandolin in details in Section 4. Section 5 shows the experiments, which are discussed in Section 6. At last, we conclude.

# 2. RELATED WORK

Machine-learning techniques have been successfully applied to ontology and instance matching, where the aim is to match classes, properties, and instances belonging to different ontologies or knowledge bases [32, 33, 50]. Also evolutionary algorithms have been used to the same scope [29]. For instance, genetic programming has shown to find good link specifications (i.e., similarity-based decision trees) in both a semi-supervised and unsupervised fashion [33, 34]. Generative models are statistical approaches which do not belong to the ML and SRL branches. Latent Dirichlet allocation is an example of application to entity resolution [2] and topic modeling [45].

SRL techniques such as Markov-logic [43] and tensor-factorization models [37, 38, 35] have been proposed for link prediction and triple classification; the formers have also been applied on problems like entity resolution [51]. Among the frameworks which operate on MLNs, we can mention NetKit-SRL [28], ProbCog [18], Alchemy [21], Tuffy [39], Felix [40], Markov theBeast [44], ArThUR [3], and RockIt [41]. Several approaches which rely on translations have been devised to perform link prediction via generation of embeddings [4, 56, 26, 55, 58]. The Google Knowledge Vault is a huge structured knowledge repository backed by a probabilistic inference system (i.e., ER-MLP) that computes calibrated probabilities of fact correctness [10].

This work is also related to link prediction in social networks [25, 47]. Being social networks the representation of social interactions, they can be seen as RDF graphs having only one property. Recently, approaches such as DeepWalk [42] and node2vec [17] showed impressive scalability to large graphs.

The link discovery frameworks LIMES [31] and Silk [53] present a variety of methods for the discovery of links among different knowledge bases [30, 19, 52, 33, 49, 15]. As presented in the next section, instance matching is a sub-problem of link discovery where the sought property is an equivalence linking instances. Most instance matching tools [24, 20, 11] take or have taken part in the *Ontology Matching Evaluation Initiative* (OAEI).

# 3. PRELIMINARIES

## 3.1 Link prediction

With respect to the link prediction problem, let us consider a directed labelled graph $G = (V, E)$ with labelling function $l : V \cup E \to U$, where $U$ is the set of all possible labels. In the RDF syntax, $U$ is the union of all URIs, literals, and blank nodes. A link prediction algorithm can be modelled as a function $lp$ which transforms the original graph $G$ to an enriched graph $G' = (V', E') = lp(G)$. The set of predicted links (i.e., edges) is thus represented by:

$$P = E' \setminus E. \tag{1}$$

## 3.2 Instance matching

The instance matching problem is defined on top of two directed labelled graphs $G_s = (V_s, E_s)$ and $G_t = (V_t, E_t)$ with their own labelling functions $l_s : V_s \cup E_s \to U$ and $l_t : V_t \cup E_t \to U$. The scope of this task is to find a mapping $M = (V_s, V_t)$ such that $(x, y) \in M$ iff $x$ and $y$ are the same thing in the real world. This mapping can also be partially known (i.e., semi-supervised setting); in this case, we have a 2-element subset $M_{train} \subseteq M$.

However, we can reduce the instance matching problem to a link prediction problem where the input graph is the merger of $G_s$ and $G_t$ adding the mapping to the edges, and the set of predicted links contains nothing but equivalences. In the Semantic Web, such relationship of equivalence is usually identified as `owl:sameAs` when linking instances, `owl:equivalentClass` when linking classes, and `owl:equivalentProperty` when linking properties. Therefore, given $G = (V_s \cup V_t, E_s \cup E_t \cup M_{train})$ and a link prediction function $lp$, the enriched graph $G' = (V, E') = lp(G)$ will feature a set of predicted equivalence links represented as follows:

$$P = \{e \in E' \setminus E : l(e) \in Q\} \tag{2}$$

where $Q$ is the set of labels of the equivalence properties.

## 3.3 Probabilistic Knowledge Bases

First-order knowledge bases are composed by statements and formulas expressed in first-order logic [14]. In probabilistic knowledge bases, every statement (i.e., edge) has a weight associated with it []. The weighting function can be represented as $\omega : E \to [0, 1]$. This means that a relationship might exist within some confidence or probability degree. In the current Semantic Web vision, any existing relationship (i.e., triple) is an edge having weight 1. However, the probabilistic interpretation of RDF graphs has shown to be able to help solving many problems, such as instance matching, question answering, and class expression learning [23, 48, 5].

## 3.4 Markov Logic Networks

As mentioned in Section 1, MLNs join first-order logic with a probabilistic model by assigning a weight to each formula. The semantic outcome is the representation of the probability distribution over possible worlds [9]. In the following, we present the difference between a *Markov Network* and a *Markov Logic Network*.

A *Markov Network* is a model for the joint distribution of a set of variables $X = (X_1, X_2, X_3, \ldots, X_n) \in \mathcal{X}$. It is composed by an undirected graph $\mathcal{G}$ and a set of potential functions $\phi$. A potential function $\phi_k : \mathcal{X}_k \to \mathbb{R}_0^+$ exists for each clique in the graph and assigns a non-negative value to each state $x_{\{k\}}$ of the corresponding $k$th clique. The joint distribution of a Markov Network is defined by:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{3}$$

where $Z$ is the partition function $Z = \sum_{x \in \mathcal{X}} \prod \phi_k(x_{\{k\}})$. Conveniently, (3) can be rewritten as a *log-linear* model as:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \tag{4}$$

where $f_j$ is a binary feature associated to each state of the clique. Since (3) and (4) are equivalent, the weight of the $j$th clique is thus $w_j = \log \phi_k(x_{\{k\}})$. The representation is exponential in the size of the cliques.

Formally, a *Markov Logic Network* can be described as a set $(F_i, w_i)$ of pairs of formulas $F_i$, expressed in first-order logic, and their corresponding weights $w_i \in \mathbb{R}$. The weight $w_i$ associated with each formula $F_i$ softens the crisp behavior of boolean logic as follows. Along with a set of constants $C$, a MLN can be viewed as a template for building a Markov Network. Given $C$, a so-called *Ground Markov Network* is thus constructed, leaving to each grounding the same weight as its respective formula. The distribution of all possible worlds is then defined as:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) \tag{5}$$

where $n_i(x)$ is the number of true groundings of $F_i$ in $x$ [43]. Please note that, despite (4) and (5) might look similar, the two summations iterate on different entities, i.e. cliques and MLN formulas, respectively.

## 3.5 Research questions

With this work, we aim at answering to the following research questions:

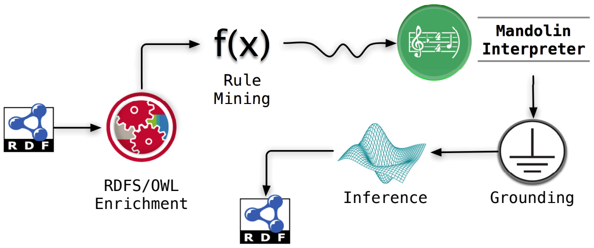

**Figure 1: Overview of the Mandolin modules.**

(Q1) Can MLNs outperform other SRL techniques on link prediction?

(Q2) Can we optimize the trade-off between their computation needs and accuracy?

(Q3) Can MLNs scale to large datasets?

(Q4) Can we join RDF semantics and Markov rules to improve the predictions?

(Q5) Can we use MLNs as an approach to RDF instance matching?

In the following section, we present our proposed resolution to the research questions above, whereas in Section 5, we will show the results we obtained.

# 4. MANDOLIN

The MANDOLIN framework is composed by five modules: *RDFS/OWL enrichment*, *Rule mining*, *Interpretation*, *Grounding*, and *Inference*. As can be seen in Figure 1, the modules are aligned in a sequential manner. Taking a union of RDF graphs $G = \bigcup_i G_i$ as input, the process ends with the generation of an enriched graph $G'$.

## 4.1 RDFS/OWL enrichment

The *RDFS/OWL enrichment* module activates optionally and features three different operations: *Similarity join*, *Ontology import*, and *Forward chaining*. Its function is to add a layer of relationships to the input graph $G$.

### *4.1.1 Similarity join.*

A node in an RDF graph may represent either a URI, a literal, or a blank node. While URI or a blank node have no restrictions w.r.t. their end in the triple, a literal can only be put as object (i.e., have only incoming edges). Literals can be of different datatype (e.g., strings, integers, floats). In order to generate the similarity relationships, we first collect all literals in the graph into as many buckets as there are datatype properties. We chose to use the Jaccard similarity on *q-grams* [16] to compare strings. To tackle the quadratic time complexity for the extraction of similar candidate pairs, we apply a positional filtering on prefixes and suffixes [57] as implemented in LIMES [30] within a similarity threshold $\theta$. Once the candidate pairs are extracted, from the original datatype property URI we construct a new property URI for each bucket and generate new triples using such URI.

For example, let us set $\theta = 0.6$ and consider the following triples:

```
:New_York_City :isIn :New_York
:New_York :isIn :USA
:New_York_City foaf:name "New York City"@en
:New_York foaf:name "New York"@en
:USA foaf:name "USA"@en
```

we first collect strings `New York City`, `New York`, and `USA`. After the filtering, the only extracted pair for property `foaf:name` is (`New York City`, `New York`), having a Jaccard similarity of $8/13 = 0.61$. We generate a new URI featuring the *SHA1* hash function of the original property URI and the threshold value for a new property, such as:

```
http://mandolin.aksw.org/similarity/
    d0c70c5ef3a2cd1e38e266bcf5e2d607e4bbd47f/0.6
```

which is then used, for each extracted pair, to connect the two subjects linked to them via `foaf:name`, i.e. `:New_York_City` and `:New_York`. Intuitively, there exists a hierarchy among properties carrying the same hash function, where properties with a higher threshold are sub-properties of the ones with lower threshold. As large multi-domain datasets such as DBpedia contains $n = 5,729$ properties, we estimate the probability of a hash collision as $p = \frac{n(n-1)}{2^{b+1}} \approx 10^{-41}$.

The procedure above is repeated for each datatype property. In case of numerical or time values, we sort them by value and create a similarity relationship whenever the difference of two values is less than the threshold $\theta$.

### 4.1.2 Ontology import and Forward chaining

RDF datasets in the Web are published so that their content can be accessible from everywhere. The vision of the Semantic Web expects URIs to be referenced from different knowledge bases. In any knowledge-representation application, in order to process the semantics associated with an URI, one option is to import the ontology (or the available RDF data) which defines such entity. To accomplish this, Mandolin dereferences external URIs, imports the data into its graph $G$, and performs forward chaining (i.e., semantic closure) on the whole graph. For instance, importing the declaration of `foaf:name`, we notice it is a sub-property of `rdfs:label`. Which means that, after performing the forward chaining, all `foaf:name` relationships exist also as `rdfs:labels`. This additional information can be useful for the Markov logic, since it fosters connectivity on $G$.

## 4.2 Rule mining

The mining of rules in a knowledge base is not a task strictly related to MLN systems. Instead, the set of MLN rules are usually given as input to the MLN system. Mandolin integrates the rule mining phase in the workflow exploiting a state-of-the-art algorithm.

The rule mining module takes an RDF graph as input and yields rules expressed as first-order Horn clauses. A Horn clause is a logic clause having at most one positive literal if written in disjunctive normal form (DNF). Any DNF clause $\neg a(x,y) \vee c(x,y)$ can be rewritten as $a(x,y) \Rightarrow c(x,y)$, thus featuring an implication. The part that remains left of the implication is called *body*, whereas the right one is called *head*. In Mandolin, a rule can belong to one of the following classes:

1. $a(x,y) \Longrightarrow c(x,y)$
2. $a(y,x) \Longrightarrow c(x,y)$
3. $a(z,x) \wedge b(z,y) \Longrightarrow c(x,y)$
4. $a(x,z) \wedge b(z,y) \Longrightarrow c(x,y)$
5. $a(z,x) \wedge b(y,z) \Longrightarrow c(x,y)$
6. $a(x,z) \wedge b(y,z) \Longrightarrow c(x,y)$

where, for our notation, a statement $a(x,y)$ is an edge $e = (x,y) \in E$ such that $l(e) = \mathtt{a}$. Note that the universal quantifiers have been omitted since the rules are declared in a non-propositional way. Intuitively, considering only a subset of Horn clauses decreases expressivity but also the search space. In large-scale knowledge bases, this strategy is preferred since it allows to scale.

Rules in knowledge bases can be ranked using several indices. The *support* of a rule is defined as the number of correct predictions in the data. For instance, the support ($\sigma$) for rules of class 3 is so defined:

$$\begin{aligned}&\sigma(a(z,x) \wedge b(z,y) \Longrightarrow c(x,y)) := \\ &\qquad |\{(x,y) \in E : \exists z : a(z,x) \wedge b(z,y) \wedge c(x,y)\}|\end{aligned} \tag{6}$$

However, as these values are absolute, a more proper measure was proposed [13] to maintain independence from the size of the graph. The *head coverage* ($\eta$) of a rule $F \in \mathcal{F}$ is a normalized version of support and is defined as follows.

$$\eta(F) := \frac{\sigma(F)}{|\{e \in E : l(e) = \mathtt{c}\}|} \tag{7}$$

Finally, a measure of the confidence ($\kappa$) of a rule $F$ is introduced. This index is also referred to as *Partial Completeness Assumption (PCA) confidence* [13]:

$$\kappa(F) := \frac{\sigma(F)}{|\{e = (x,y) \in E : \exists z_1, \ldots, z_m, y' : l(e) = c \wedge (x,y') \in E\}|} \tag{8}$$

(7) and (8) play an important role in the rule mining phase, as their values indicate whether or not a rule has

to be pruned from the results. For the search of rules in the graph, we rely on the *AMIE+* algorithm described in [12].

### 4.3 Interpretation

In this module, the set of rules outputted by the rule miner are collected, filtered, and translated for the next phase, i.e. the grounding. At the end of the mining phase, we perform a selection of rules based on their head coverage, i.e. $F' = \{F \in \mathcal{F} : \eta(F) \geq \bar{\eta}\}$. In most MLN frameworks, weights associated to rules are provided manually [18, 41, 3]; some systems can learn them from data [21, 39]. However, as in our framework the rules come with their respective measures, a weight learning phase is not needed. We thus assign a weight to a rule using the following formula:

$$w(F) := k * \kappa(F) \tag{9}$$

with $k = 100$. We preferred to use PCA confidence over head coverage because previous literature showed its greater effectiveness [10, 12].

### 4.4 Grounding

Grounding is the phase where the ground Markov network (factor graph) is built starting from the graph and a set of MLN rules. A factor graph is a set of factors $\phi = \{\phi_1, \ldots, \phi_N\}$ where $\phi_i$ is a function over a random vector $X_i$ which indicates the value of a variable in a formula. As the computational complexity for grounding is NP-complete, the problem of scalability has been addressed by using relational databases. However, frameworks such as *Tuffy* or *Alchemy* showed they are not able to scale, even in datasets with a few thousands statements [6]. Tuffy, for example, stores the ground network data into a DBMS loaded on a RAM-disk for best performances [39]; however, growing exponentially, the RAM cannot contain the ground network data, resulting in the program going out of memory. For this reason, in the MANDOLIN module for grounding, we integrated *ProbKB*, state-of-the-art algorithm for the computation of factors. The main strength of this approach is the exploitation of the simple structure of Horn clauses [7], differently from other frameworks where any first-order logic formula is allowed. It consists of a two-step method, i.e. (1) new statements are inferred until convergence and (2) the factor network is built. Each statement is read in-memory at most 3 times; differently from Tuffy, where it is read every time it appears in the knowledge base [6].

### 4.5 Inference

Inference in Markov networks is a P-complete problem [46]. However, the values in (5) can be approximated using techniques such as Gibbs sampling – which showed to perform best [41] – and belief propagation [43]. We employ the use of Gibbs sampling for the computation of the set of links $P$ defined in (1) and (2). Typically, the number of iterations for the Gibbs sampler is $\gamma = 100{*}|E|$ [41]. Due to scalability reasons, we approximate the probability values by limiting the number of iterations.

Every statement $a(x, y)$ is associated to a node in the factor graph. Therefore, its probability is proportional to the product of all potential functions $\phi_k(x_{\{k\}})$ applied to the state of the cliques touching that node. Once we compute the probabilities of all sampled candidates, instead of dividing the product by the partition function constant $Z$ (see Section 3.4), we normalize them so that the minimum and maximum value are 0 and 1 respectively. The final set $P$ of predicted links is then defined as those statements whose probability is greater than a threshold $\tau \in [0, 1]$.

### 4.6 Incremental learning

Another peculiarity of our framework is the incremental learning setting. After the set of links $P$ has been discovered, MANDOLIN can optionally run again on the enriched graph $G'$ to yield a more enriched graph $G''$. We use a validation set to let the algorithm estimate the performances and decide whether to stop. In case the performance of the last iteration is the highest found so far, another iteration is carried out.

### 4.7 Implementation

The MANDOLIN framework was mainly developed in *Java 1.8* and its source code is available online[2] along with all datasets used for the evaluation. Libraries and research projects involved are *Apache Jena*, *Pellet*, *AMIE+*, *ProbKB*, and the *Gurobi* optimizer. To implement the Gibbs sampling, we chose to use *RockIt* [41] instead of *GraphLab* [27], since the latter project had just been rebranded and its code privatized. We rely on a *PostgreSQL* database for the storage of the networks. How the input graph and MLN rules are stored and the factor graph computed via a *Pl/PgSQL* script is described in [7].

## 5. EXPERIMENTS

### 5.1 Evaluation setup

Any directed labelled graph can be easily transformed to an RDF graph by simply creating a namespace and prepending it to entities and properties in statements. We thus created an RDF version of a well-known benchmark for knowledge discovery used in [36]. The benchmark consists of two datasets: *WN18*, built upon the WordNet glossary, and *FB15k*, a subset of the Freebase collaborative knowledge base. Using these datasets, we evaluated MANDOLIN on link prediction. Finally, we employed three large-scale datasets (DBLP-ACM [22] as well as DBpedia 3.8 and Wikidata 20161020) to evaluate the scalability of our approach. All experiments were carried out on a 64-core server with 256 GB RAM.

[2] http://mandolin.aksw.org/

## 5.2 Link prediction evaluation

We evaluated the link prediction task on two measures, *Mean Reciprocal Rank* (MRR) and *Hits@k*. The benchmark datasets are divided into training, validation, and test sets. We used the training set to build the models and the validation set to find the hyperparameters, which are introduced later. Afterwards, we used the union of the training and validation sets to build the final model. For each test triple $(s, p, o)$, we generated as many corrupted triples $(s, p, \tilde{o})$ as there are nodes in the graph such that $o \neq \tilde{o} \in V$. We computed the probability for all these triples, when this value was available; if not, we assumed it 0. Then, we ranked the triples in descending order and checked the position of $(s, p, o)$ in the rank. The MRR is thus the reciprocal rank, averaged to all test triples:

$$MRR = \frac{1}{Q} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{10}$$

MRR has been preferred over mean rank because it is less sensitive to outliers [36]. The Hits@k index is the ratio (%) of test triples that have been ranked among the top-$k$. We compute the Hits@1, 3, and 10 with a filtered setting, i.e. all corrupted triples ranked above $(s, p, o)$ which are present in the training sets are discarded before computing the rank.

The results for link prediction on the *WN18-FB15k* benchmark are shown in Table 1. We compare MANDOLIN with other SRL techniques based on embeddings and tensor factorization. On *WN18*, we overperform all other approaches w.r.t. the Hits@10 index (96.0%). However, HOLE [36] recorded the highest performance w.r.t. MRR and Hits@1; the two approaches achieved almost the same value on Hits@3. Here, MANDOLIN recorded its own highest performance after 1 iteration of incremental learning; the second iteration saw a drop of −8% in all measures. On the Freebase dataset, three different approaches hold the highest values. HOLE performed best on MRR and Hits@3, whereas MANDOLIN on Hits@1, and TRANSE on Hits@10. On this dataset, our incremental learning setting showed its effectiveness, yielding a +12% Hits@10 from the first to the second iteration, which recorded the highest local value. Examples of rules learned can be found at the project website.

Since the two datasets above contain no datatype values and no statements using the RDF schema[3], we did not activate the RDF-specific settings introduced in Section 4.1. However, beyond them, our framework depends on the following hyperparameters:

- *minimum head coverage* ($\overline{\eta}$), used to filter rules;
- *Gibbs sampling iterations* ($\gamma$).

To compute the optimal configuration on the trade-off between computational needs and overall performances, we performed further experiments on the link prediction benchmark. We investigated the relationship between number of Gibbs sampling iterations, runtime, and accuracy by running our approach using the following values: $\gamma = 1M, 2M, 3M, 5M, 10M, 50M, 100M$. As can be seen in Figure 2, the runtime is, excluding an overhead, linear w.r.t. the number of iterations. The Hits@10 index seems to stabilize at around $\gamma = 5M$, however higher accuracy can be found by increasing this value.

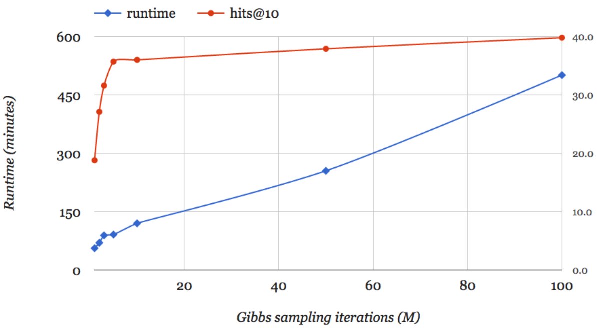

**Figure 2: Runtime and Hits@10 on *FB15k* with $\overline{\eta} = 0.9$.**

## 5.3 Scalability and instance matching

We performed tests on large-scale datasets such as DBLP–ACM, DBpedia[4] and Wikidata[5]. We compared our approach with other MLN frameworks, i.e. NetKit-SRL, ProbCog, Alchemy, and Tuffy. As these frameworks can learn rule weights but not rules themselves, we fed them with the rules found by our rule miner. We set $\gamma = 0.9$ and $\overline{\eta} = 10M$. The results (see Table 2) showed that, in all cases, MANDOLIN is the only framework that was able to terminate the computation. In the DBLP–ACM dataset, we were able to discover equivalence links among articles and authors that had not been linked in the original datasets. After dividing the mapping $M$ into two folds $(90\% - 10\%)$, we used the larger as training set. We were able to predict 71.0% of the correct `owl:sameAs` links in the remaining test set. DBpedia and Wikidata do not provide a ground truth but we could show that we scale well even on such large datasets (see Table 2).

# 6. DISCUSSION

We have witnessed a different behavior of our algorithm when evaluated on the two datasets for link prediction. In particular, the incremental learning setting showed to be beneficial only on the Freebase dataset. This might be explained by the different structure of the graphs: Relying on first-order Horn clauses, new relationships can only be discovered if they belong to a

[3] https://www.w3.org/TR/rdf-schema/

[4] http://dbpedia.org

[5] http://wikidata.org

| | WN18 | | | | FB15k | | | |
|---|---|---|---|---|---|---|---|---|
| | MRR | Hits@1 | Hits@3 | Hits@10 | MRR | Hits@1 | Hits@3 | Hits@10 |
| TransE | 0.495 | 11.3 | 88.8 | 94.3 | 0.463 | 29.7 | 57.8 | **74.9** |
| TransR | 0.605 | 33.5 | 87.6 | 94.0 | 0.346 | 21.8 | 40.4 | 58.2 |
| ER-MLP | 0.712 | 62.6 | 77.5 | 86.3 | 0.288 | 17.3 | 31.7 | 50.1 |
| RESCAL | 0.890 | 84.2 | 90.4 | 92.8 | 0.354 | 23.5 | 40.9 | 58.7 |
| HolE | **0.938** | **93.0** | **94.5** | 94.9 | **0.524** | 40.2 | **61.3** | 73.9 |
| Mandolin | 0.892 | 89.2 | 94.3 | **96.0** | 0.404 | **40.4** | 48.4 | 52.6 |

**Table 1: Results for link prediction on the WordNet (WN18) and Freebase (FB15k) datasets.**

| Dataset | Runtime (s) | $(\lvert\mathcal{F}'\rvert)$ | $(\lvert P\rvert)$ |
|---|---|---|---|
| DBLP–ACM | 2,460 | 1,500 | 4,730 |
| DBpedia | 85,334 | 1,500 | 179,201 |
| Wikidata | 46,444 | 1,500 | 83,599 |

**Table 2: Runtime, number of rules after the filtering, and number of predicted links on the larger datasets are shown.**

3-vertex clique where the other two edges are already in the graph. Therefore, Mandolin needed one more step to discover them on a less connected graph such as FB15k. A more detailed view on the learned rules is provided on the project website (see Section 4.7).

The reason why approaches like RESCAL and ER-MLP have performed worse than others is probably overfitting. Embedding methods have shown to achieve excellent results, however no method significantly over-performed the others. We thus believe that heterogeneity in Linked Datasets is still an open challenge and structure plays a key role to the choice of the algorithm.

Although our MLN framework showed to be more scalable and to be able to provide users with *justifications* for adding triples through the rules it generates, we recognize reasoning is a powerful resource but not yet efficient. Following the examples in other cases (e.g., rule mining, inference), the reasoning task could be limited to the mere transitive closure. Avoiding the computation of all consistency and coherence axioms should considerably decrease the overall runtime.

## 7. CONCLUSION

In this paper, we described Mandolin, an approach for knowledge discovery specifically designed for the Web of Data. To the best of our knowledge, it is the first complete framework for RDF link prediction based on Markov Logic Networks which features the entire pipeline necessary to achieve this task. We showed that it is able to achieve results beyond the State of the Art for some measures on a well-known link prediction benchmark. Moreover, it can predict equivalence links across datasets and scale on large graphs.

We plan to extend this work in order to: (1) refine domain and range in rules; (2) build functionals using OWL rules and evaluate their effectiveness on the predicted links; (3) evaluate our approach on triple classification; (4) improve the run times with the use of a parallel DBMS and (5) a less powerful yet more efficient reasoner.

We would like to thank Yang Chen for his precious and kind help.

## 8. REFERENCES

[1] T. Berners-Lee. Linked data-design issues (2006). *Published online*, 2006.

[2] I. Bhattacharya and L. Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM*, volume 5, 7, page 59. SIAM, 2006.

[3] A. Bodart, K. Evrard, J. Ortiz, and P.-Y. Schobbens. Arthur: A tool for markov logic network. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 319–328. Springer, 2014.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795, 2013.

[5] L. Bühmann, D. Fleischhacker, J. Lehmann, A. Melo, and J. Völker. Inductive lexical learning of class expressions. In K. Janowicz, S. Schlobach, P. Lambrix, and E. Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, volume 8876 of *Lecture Notes in Computer Science*, pages 42–53. Springer International Publishing, 2014.

[6] D. Y. Chen and D. Z. Wang. Web-scale knowledge inference using markov logic networks. In *ICML workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*, pages 106–110. Association for Computational Linguistics, 2013.

[7] Y. Chen and D. Z. Wang. Knowledge expansion over probabilistic knowledge bases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 649–660. ACM, 2014.

[8] P. Domingos, D. Lowd, S. Kok, H. Poon,

M. Richardson, and P. Singla. Just add weights: Markov logic for the semantic web. In *Uncertainty Reasoning for the Semantic Web I*, pages 1–25. Springer, 2008.

[9] P. Domingos and M. Richardson. 1 markov logic: A unifying framework for statistical relational learning. *Statistical Relational Learning*, page 339, 2007.

[10] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–610. ACM, 2014.

[11] D. Faria, C. Pesquita, E. Santos, M. Palmonari, I. F. Cruz, and F. M. Couto. The agreementmakerlight ontology matching system. In *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*, pages 527–541. Springer, 2013.

[12] L. Galárraga, C. Teflioudi, K. Hose, and F. M. Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6):707–730, 2015.

[13] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. ACM, 2013.

[14] M. R. Genesereth and N. J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987.

[15] K. Georgala, M. A. Sherif, and A.-C. N. Ngomo. An efficient approach for the generation of allen relations. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI) 2016, The Hague, 29. August - 02. September 2016*, 2016.

[16] L. Gravano, P. G. Ipeirotis, H. V. Jagadish, N. Koudas, S. Muthukrishnan, and D. Srivastava. Approximate string joins in a database (almost) for free. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 491–500, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[17] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *KDD 2016*, 2016.

[18] D. Jain and M. Beetz. Soft evidential update via markov chain monte carlo inference. In *Annual Conference on Artificial Intelligence*, pages 280–290. Springer, 2010.

[19] A. Jentzsch, R. Isele, and C. Bizer. Silk-generating rdf links while publishing or consuming linked data. In *Proceedings of the 2010 International Conference on Posters & Demonstrations Track-Volume 658*, pages 53–56. CEUR-WS. org, 2010.

[20] E. Jiménez-Ruiz and B. C. Grau. Logmap: Logic-based and scalable ontology matching. In *International Semantic Web Conference*, pages 273–288. Springer, 2011.

[21] S. Kok, M. Sumner, M. Richardson, P. Singla, H. Poon, D. Lowd, J. Wang, and P. Domingos. The Alchemy system for statistical relational {AI}. Technical report, Department of Computer Science and Engineering, University of Washington, 2009.

[22] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *PVLDB*, 3(1):484–493, 2010.

[23] L. Leitão, P. Calado, and M. Weis. Structure-based inference of xml similarity for fuzzy duplicate detection. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pages 293–302. ACM, 2007.

[24] J. Li, J. Tang, Y. Li, and Q. Luo. Rimom: A dynamic multistrategy ontology alignment framework. *IEEE Transactions on Knowledge and Data Engineering*, 21(8):1218–1232, 2009.

[25] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[26] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[27] Y. Low, J. Gonzalez, A. Kyrola, D. Bickson, C. Guestrin, and J. M. Hellerstein. Graphlab: A new framework for parallel machine learning. arxiv preprint. *arXiv preprint arXiv:1006.4990*, 1, 2010.

[28] S. A. Macskassy and F. Provost. Netkit-srl: A toolkit for network learning and inference. In *Proceeding of the NAACSOS Conference*, 2005.

[29] J. Martinez-Gil, E. Alba, and J. F. A. Montes. Optimizing ontology alignments by using genetic algorithms. In *Proceedings of the First International Conference on Nature Inspired Reasoning for the Semantic Web-Volume 419*, pages 1–15. CEUR-WS. org, 2008.

[30] A.-C. Ngonga Ngomo. Link discovery with guaranteed reduction ratio in affine spaces with minkowski measures. In *Proceedings of ISWC*, 2012.

[31] A.-C. Ngonga Ngomo and S. Auer. Limes - a time-efficient approach for large-scale link discovery on the web of data. In *Proceedings of IJCAI*, 2011.

[32] A.-C. Ngonga Ngomo, N. Heino, K. Lyko, R. Speck, and M. Kaltenböck. Scms - semantifying content management systems. In *ISWC 2011*, 2011.

[33] A.-C. Ngonga Ngomo and K. Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *Proceedings of ESWC*, 2012.

[34] A.-C. Ngonga Ngomo and K. Lyko. Unsupervised learning of link specifications: deterministic vs. non-deterministic. In *Proceedings of the Ontology Matching Workshop*, 2013.

[35] M. Nickel, X. Jiang, and V. Tresp. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, pages 1179–1187, 2014.

[36] M. Nickel, L. Rosasco, and T. Poggio. Holographic embeddings of knowledge graphs. *arXiv preprint arXiv:1510.04935*, 2015.

[37] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816, 2011.

[38] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web*, pages 271–280. ACM, 2012.

[39] F. Niu, C. Ré, A. Doan, and J. Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *Proceedings of the VLDB Endowment*, 4(6):373–384, 2011.

[40] F. Niu, C. Zhang, C. Ré, and J. W. Shavlik. Felix: Scaling inference for markov logic with an operator-based approach. *CoRR*, abs/1108.0294, 2011.

[41] J. Noessner, M. Niepert, and H. Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. *arXiv preprint arXiv:1304.4379*, 2013.

[42] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*, pages 701–710. ACM, 2014.

[43] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 62(1-2):107–136, 2006.

[44] S. Riedel. Improving the accuracy and efficiency of map inference for markov logic. In *Proceedings of the 24th Annual Conference on Uncertainty in AI (UAI '08)*, pages 468–475, 2008.

[45] M. Röder, A.-C. Ngonga Ngomo, I. Ermilov, and A. Both. Detecting similar linked datasets using topic modelling. In *Proceedings of the 13th Extended Semantic Web Conference*, pages 3–19, 2016.

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

[47] S. Scellato, A. Noulas, and C. Mascolo. Exploiting place features in link prediction on location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1046–1054. ACM, 2011.

[48] S. Shekarpour, E. Marx, A.-C. Ngonga Ngomo, and S. Auer. Sina: Semantic interpretation of user queries for Question Answering on interlinked data. *Web Semantics*, 1:–, 2014.

[49] M. A. Sherif and A.-C. Ngonga Ngomo. An optimization approach for load balancing in parallel link discovery. In *SEMANTiCS 2015*, 2015.

[50] P. Shvaiko, J. Euzenat, E. Jiménez-Ruiz, M. Cheatham, and O. Hassanzadeh, editors. *Proceedings of the 10th International Workshop on Ontology Matching*, volume 1545 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[51] P. Singla and P. Domingos. Entity resolution with markov logic. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 572–582. IEEE, 2006.

[52] T. Soru and A.-C. Ngonga Ngomo. Active learning of domain-specific distances for link discovery. In *Proceedings of JIST*, 2012.

[53] J. Volz, C. Bizer, M. Gaedke, and G. Kobilarov. Silk-a link discovery framework for the web of data. *LDOW*, 538, 2009.

[54] J. Wang, Z. Ding, and C. Jiang. Gaom: Genetic algorithm based ontology matching. In *2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*, pages 617–620. IEEE, 2006.

[55] Q. Wang, B. Wang, and L. Guo. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1859–1865, 2015.

[56] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer, 2014.

[57] C. Xiao, W. Wang, X. Lin, and J. X. Yu. Efficient similarity joins for near duplicate detection. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 131–140, New York, NY, USA, 2008. ACM.

[58] H. Xiao, M. Huang, Y. Hao, and X. Zhu. Transg: A generative mixture model for knowledge graph embedding. *arXiv preprint arXiv:1509.05488*, 2015.