Probabilistic Knowledge Discovery for the Web of Data

Abstract

Markov Logic Networks (MLNs) join probabilistic modeling with first-order logic and have been shown to integrate well with the Semantic Web foundations. While several approaches have been devised to tackle the subproblems of rule mining, grounding, and inference, no comprehensive framework was proposed so far. In this engineering paper, we fill this gap by introducing a massively-parallel framework for knowledge discovery specifically on RDF datasets. Our framework imports knowledge from referenced graphs, creates similarity relationships among similar literals, and relies on state-of-the-art techniques for rule mining, grounding, and inference computation. We show that our best configuration scales well and achieves at least comparable results with respect to other statistical-relational-learning algorithms on link prediction.

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 [Spohr *et al.*, 2011; Nikolov *et al.*, 2012; Rowe *et al.*, 2011], Evolutionary Algorithms [Wang *et al.*, 2006; Ngonga Ngomo and Lyko, 2012], Generative Models [Bhattacharya and Getoor, 2006], and Statistical Relational Learning (SRL) [Singla and Domingos, 2006]. In particular, the main objective of the application of such algorithms is to address the fourth Linked Data principle, which preaches to the Semantic Web community to *"include links to other URIs, so that they [the visitors] can discover more things"* [Berners-Lee, 2006]. Two years later, [Domingos *et al.*, 2008] proposed Markov Logic Networks (MLNs) – a well-known approach to Knowledge Discovery in knowledge bases [Richardson and Domingos, 2006] – to be a promising framework for the Semantic Web. 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 [Kok *et al.*, 2009; Niu *et al.*, 2011a; Noessner *et al.*, 2013; Bodart *et al.*, 2014]. 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 engineering paper, we present MLNF[2], a framework based on Markov Logic Networks for rule mining and link discovery. To the best of our knowledge, MLNF is the first framework to implement the entire workflow for knowledge discovery on RDF datasets. Making use of RDFS/OWL semantics, MLNF 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 all the components of our framework scale well. We evaluate MLNF on two benchmark datasets for link prediction and show that it can achieve comparable results w.r.t. other statistical-relational-learning (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 MLNF 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 [Ngonga Ngomo *et al.*, 2011; Ngonga Ngomo and Lyko, 2012; Shvaiko *et al.*, 2016].

[1] Retrieved on February 15, 2017 from `http://lodstats.aksw.org/`.

[2] We hide the real framework name to preserve anonymity.

Also evolutionary algorithms have been used to the same scope [Martinez-Gil *et al.*, 2008]. 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 [Ngonga Ngomo and Lyko, 2012]. 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 [Bhattacharya and Getoor, 2006] and topic modeling [Röder *et al.*, 2016].

SRL techniques such as Markov-logic [Richardson and Domingos, 2006] and tensor-factorization models [Nickel *et al.*, 2014] have been proposed for link prediction and triple classification; the formers have also been applied on problems like entity resolution [Singla and Domingos, 2006]. Among the frameworks which operate on MLNs, we can mention NetKit-SRL [Macskassy and Provost, 2005], ProbCog [Jain and Beetz, 2010], Alchemy [Kok *et al.*, 2009], Tuffy [Niu *et al.*, 2011a], Felix [Niu *et al.*, 2011b], Markov theBeast [Riedel, 2008], ArThUR [Bodart *et al.*, 2014], and RockIt [Noessner *et al.*, 2013]. Several approaches which rely on translations have been devised to perform link prediction via generation of embeddings [Bordes *et al.*, 2013; Lin *et al.*, 2015; Wang *et al.*, 2015; Xiao *et al.*, 2016]. 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 [Dong *et al.*, 2014].

This work is also related to link prediction in social networks [Liben-Nowell and Kleinberg, 2007; Scellato *et al.*, 2011]. Being social networks the representation of social interactions, they can be seen as RDF graphs having only one property. Recently, approaches such as DeepWalk [Perozzi *et al.*, 2014] and node2vec [Grover and Leskovec, 2016] showed impressive scalability to large graphs.

The link discovery frameworks Silk [Volz *et al.*, 2009] and LIMES [Ngonga Ngomo and Auer, 2011] present a variety of methods for the discovery of links among different knowledge bases [Jentzsch *et al.*, 2010; Ngonga Ngomo and Lyko, 2012; Sherif and Ngonga Ngomo, 2015].

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 Probabilistic Knowledge Bases

First-order knowledge bases are composed by statements and formulas expressed in first-order logic [Genesereth and Nilsson, 1987]. In probabilistic knowledge bases, every statement (i.e., edge) has a weight associated with it [Wuthrich, 1995]. 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 [Leitão *et al.*, 2007; Shekarpour *et al.*, 2014; Bühmann *et al.*, 2014].

3.3 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 [Domingos and Richardson, 2007]. 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 ϕ. 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 kth clique. The joint distribution of a Markov Network is defined by:

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}) \tag{2}$$

where Z is the partition function $Z = \sum_{x \in \mathcal{X}} \prod \phi_k(x_{\{k\}})$. Conveniently, (2) can be rewritten as a *log-linear* model as:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j f_j(x)\right) \tag{3}$$

where f_j is a binary feature associated to each state of the clique. Since (2) and (3) are equivalent, the weight of the jth 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:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x)\right) \tag{4}$$

where $n_i(x)$ is the number of true groundings of F_i in x [Richardson and Domingos, 2006]. Please note that, despite (3) and (4) might look similar, the two summations iterate on different entities, i.e. cliques and MLN formulas, respectively.

4 The Framework

The MLNF 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.

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* [Gravano *et al.*, 2001] 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 [Xiao *et al.*, 2008] within a similarity threshold θ. 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://mlnf.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 θ.

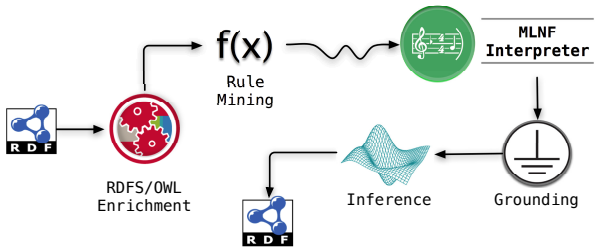

Figure 1: Overview of the MLNF modules.

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, MLNF 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 and Interpretation

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. MLNF 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 MLNF, 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 $l(e) = \texttt{a}$. 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 (σ) for rules of class 3 is so defined:

$$\sigma(a(z,x) \wedge b(z,y) \Longrightarrow c(x,y)) := \\ |\{(x,y) \in E : \exists z : a(z,x) \wedge b(z,y) \wedge c(x,y)\}| \quad (5)$$

However, as these values are absolute, a more proper measure was proposed [Galárraga *et al.*, 2015] to maintain independence from the size of the graph. The *head coverage* (η) of a rule $F \in \mathcal{F}$ is a normalized version of support and is defined as follows.

$$\eta(F) := \frac{\sigma(F)}{|\{e \in E : l(e) = \mathsf{c}\}|} \quad (6)$$

Finally, a measure of the confidence (κ) of a rule F is introduced. This index is also referred to as *Partial Completeness Assumption (PCA) confidence* [Galárraga *et al.*, 2015]:

$$\kappa(F) := \\ \frac{\sigma(F)}{|\{e = (x,y) \in E : \exists z_1, \ldots, z_m, y' : l(e) = c \wedge (x,y') \in E\}|} \quad (7)$$

(6) and (7) 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 [Galárraga *et al.*, 2015].

In the interpretation 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. $F' = \{F \in \mathcal{F} : \eta(F) \geq \bar{\eta}\}$. We preferred to use PCA confidence over head coverage because previous literature showed its greater effectiveness [Dong *et al.*, 2014; Galárraga *et al.*, 2015].

4.3 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 graph consisting of two types of nodes: factors and variables where the factors connect all the variables in their scope. Given a set of factors $\phi = \{\phi_1, \ldots, \phi_N\}$, ϕ_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 [Chen and Wang, 2013]. Tuffy, for example, stores the ground network data into a DBMS loaded on a RAM-disk for best performances [Niu *et al.*, 2011a]; however, growing exponentially, the RAM cannot contain the ground network data, resulting in the program going out of memory. For this reason, in the MLNF 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 [Chen and Wang, 2014], 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 [Chen and Wang, 2013].

4.4 Inference

MAP Inference in Markov networks is a P#-complete problem [Roth, 1996]. However, the values in (4) can be approximated using techniques such as Gibbs sampling – which showed to perform best [Noessner *et al.*, 2013] – and belief propagation [Richardson and Domingos, 2006]. We employ the use of Gibbs sampling for the computation of the set of links P defined in (1). Typically, the number of iterations for the Gibbs sampler is $\gamma = 100 * |E|$ [Noessner *et al.*, 2013]. 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 3.3), 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 $\tau \in [0,1]$.

Another peculiarity of our framework is the incremental learning setting. After the set of links P has been discovered, MLNF 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.

4.5 Implementation

The MLNF framework was mainly developed in *Java 1.8* and its source code will be made available online[3] 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* [Noessner *et al.*, 2013] instead of *GraphLab* [Low *et al.*, 2010], 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 [Chen and Wang, 2014].

5 Experiments

5.1 Evaluation setup

Any directed labelled graph can be easily transformed into an RDF graph by simply creating a namespace and prepending

[3] Binaries and data can be temporarily downloaded at `http://bit.ly/2lCn5eV`.

	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
MLNF	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.

it to entities and properties in statements. We thus created an RDF version of a benchmark for knowledge discovery used in [Bordes *et al.*, 2013; Lin *et al.*, 2015; Wang *et al.*, 2015; Xiao *et al.*, 2016; Nickel *et al.*, 2015]. 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 MLNF on link prediction. Finally, we employed two large-scale datasets to evaluate the scalability of our approach. All experiments were carried out on a 64-core server with 256 GB RAM.

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:

$$MRR = \frac{1}{Q} \sum_{i=1}^{|Q|} \frac{1}{rank_i} \tag{8}$$

MRR has been preferred over mean rank because it is less sensitive to outliers [Nickel *et al.*, 2015]. 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 MLNF 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 [Nickel *et al.*, 2015] recorded the highest performance w.r.t. MRR and Hits@1; the two approaches achieved almost the same value on Hits@3. Here, MLNF 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 MLNF 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[4], 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* ($\bar{\eta}$), used to filter rules;
- *Gibbs sampling iterations* (γ).

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: $\gamma = 1M, 2M, 3M, 5M, 10M, 50M, 100M$. Our findings showed that the runtime is, excluding an overhead, linear w.r.t. the number of iterations. On both datasets, the Hits@10 index tends to stabilize at around $\gamma = 5M$, however higher accuracy can be found by increasing this value.

5.3 Large-scale datasets

We performed tests on large-scale datasets such as DBpedia[5] and Wikidata[6]. The results (see Table 2) showed that, in all cases, MLNF is the only framework that was able to terminate the computation. As DBpedia and Wikidata are not manually generated, i.e. they cannot be considered gold standards, we could not measure any accuracy, however Table 2 shows that our system can deal even with such large datasets.

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

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

[5] Version 2015-10 from `http://dbpedia.org`.

[6] Version 20150330 from `http://wikidata.org`.

Dataset	Runtime (s)	($\|\mathcal{F}'\|$)	($\|P\|$)
DBpedia	85,334	1,500	179,201
Wikidata	46,444	1,500	83,599

Table 2: Results for $\gamma = 0.9$ and $\bar{\eta} = 10M$. Runtime, number of rules after the filtering, and number of predicted links on the larger datasets are shown.

clauses, new relationships can only be discovered if they belong to a 3-vertex clique where the other two edges are already in the graph. Therefore, MLNF needed one more step to discover them on a less connected graph such as FB15k.

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 overperformed 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 MLNF, a framework 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 scale on large graphs. We plan to extend this work in order to refine domain and range in rules and build functionals using OWL rules and evaluate their effectiveness on the predicted links.

References

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

[Bhattacharya and Getoor, 2006] Indrajit Bhattacharya and Lise Getoor. A latent dirichlet model for unsupervised entity resolution. In *SDM*, volume 5, 7. SIAM, 2006.

[Bodart *et al.*, 2014] Axel Bodart, Keyvin Evrard, James Ortiz, and Pierre-Yves Schobbens. Arthur: A tool for markov logic network. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*. Springer, 2014.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, 2013.

[Bühmann *et al.*, 2014] Lorenz Bühmann, Daniel Fleischhacker, Jens Lehmann, Andre Melo, and Johanna Völker. Inductive lexical learning of class expressions. In Krzysztof Janowicz, Stefan Schlobach, Patrick Lambrix, and Eero Hyvönen, editors, *Knowledge Engineering and Knowledge Management*, volume 8876 of *Lecture Notes in Computer Science*, pages 42–53. Springer International Publishing, 2014.

[Chen and Wang, 2013] DZW Yang Chen and Daisy Zhe Wang. Web-scale knowledge inference using markov logic networks. In *ICML workshop on Structured Learning: Inferring Graphs from Structured and Unstructured Inputs*. Association for Computational Linguistics, 2013.

[Chen and Wang, 2014] Yang Chen and Daisy Zhe Wang. Knowledge expansion over probabilistic knowledge bases. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*. ACM, 2014.

[Domingos and Richardson, 2007] Pedro Domingos and Matthew Richardson. 1 markov logic: A unifying framework for statistical relational learning. *Statistical Relational Learning*, 2007.

[Domingos *et al.*, 2008] Pedro Domingos, Daniel Lowd, Stanley Kok, Hoifung Poon, Matthew Richardson, and Parag Singla. Just add weights: Markov logic for the semantic web. In *Uncertainty Reasoning for the Semantic Web I*. Springer, 2008.

[Dong *et al.*, 2014] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *SIGKDD*. ACM, 2014.

[Galárraga *et al.*, 2015] Luis Galárraga, Christina Teflioudi, Katja Hose, and Fabian M Suchanek. Fast rule mining in ontological knowledge bases with amie+. *The VLDB Journal*, 24(6), 2015.

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

[Gravano *et al.*, 2001] Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava. Approximate string joins in a database (almost) for free. In *VLDB*, VLDB '01, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.

[Grover and Leskovec, 2016] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD 2016*, 2016.

[Jain and Beetz, 2010] Dominik Jain and Michael Beetz. Soft evidential update via markov chain monte carlo inference. In *Annual Conference on Artificial Intelligence*. Springer, 2010.

[Jentzsch *et al.*, 2010] Anja Jentzsch, Robert Isele, and Christian Bizer. Silk-generating rdf links while publishing or consuming linked data. In *Proceedings of the 2010 International Conference on Posters & Demonstrations Track-Volume 658*. CEUR-WS. org, 2010.

[Kok *et al.*, 2009] Stanley Kok, Marc Sumner, Matthew Richardson, Parag Singla, Hoifung Poon, Daniel Lowd, Jue Wang, and Pedro Domingos. The Alchemy system for statistical relational {AI}. Technical report, Department of Computer Science and Engineering, University of Washington, 2009.

[Leitão *et al.*, 2007] Luís Leitão, Pável Calado, and Melanie Weis. Structure-based inference of xml similarity for fuzzy duplicate detection. In *CIKM*. ACM, 2007.

[Liben-Nowell and Kleinberg, 2007] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7), 2007.

[Lin *et al.*, 2015] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[Low *et al.*, 2010] Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M Hellerstein. Graphlab: A new framework for parallel machine learning. arxiv preprint. *arXiv preprint arXiv:1006.4990*, 1, 2010.

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

[Martinez-Gil *et al.*, 2008] Jorge Martinez-Gil, Enrique Alba, and Jose F Aldana 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*. CEUR-WS. org, 2008.

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

[Ngonga Ngomo and Lyko, 2012] Axel-Cyrille Ngonga Ngomo and Klaus Lyko. Eagle: Efficient active learning of link specifications using genetic programming. In *Proceedings of ESWC*, 2012.

[Ngonga Ngomo *et al.*, 2011] Axel-Cyrille Ngonga Ngomo, Norman Heino, Klaus Lyko, René Speck, and Martin Kaltenböck. Scms - semantifying content management systems. In *ISWC 2011*, 2011.

[Nickel *et al.*, 2014] Maximilian Nickel, Xueyan Jiang, and Volker Tresp. Reducing the rank in relational factorization models by including observable patterns. In *Advances in Neural Information Processing Systems*, 2014.

[Nickel *et al.*, 2015] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. Holographic embeddings of knowledge graphs. *AAAI*, 2015.

[Nikolov *et al.*, 2012] Andriy Nikolov, Mathieu dAquin, and Enrico Motta. Unsupervised learning of link discovery configuration. In *Extended Semantic Web Conference*. Springer, 2012.

[Niu *et al.*, 2011a] Feng Niu, Christopher Ré, AnHai Doan, and Jude Shavlik. Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. *Proceedings of the VLDB Endowment*, 4(6), 2011.

[Niu *et al.*, 2011b] Feng Niu, Ce Zhang, Christopher Ré, and Jude W. Shavlik. Felix: Scaling inference for markov logic with an operator-based approach. *CoRR*, abs/1108.0294, 2011.

[Noessner *et al.*, 2013] Jan Noessner, Mathias Niepert, and Heiner Stuckenschmidt. Rockit: Exploiting parallelism and symmetry for map inference in statistical relational models. *arXiv preprint arXiv:1304.4379*, 2013.

[Perozzi *et al.*, 2014] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *SIGKDD*. ACM, 2014.

[Richardson and Domingos, 2006] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine learning*, 62(1-2), 2006.

[Riedel, 2008] Sebastian 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.

[Röder *et al.*, 2016] Michael Röder, Axel-Cyrille Ngonga Ngomo, Ivan Ermilov, and Andreas Both. Detecting similar linked datasets using topic modelling. In *Proceedings of the 13th Extended Semantic Web Conference*, pages 3–19, 2016.

[Roth, 1996] Dan Roth. On the hardness of approximate reasoning. *Artif. Intell.*, 82(1-2), April 1996.

[Rowe *et al.*, 2011] Matthew Rowe, Sofia Angeletou, and Harith Alani. Predicting discussions on the social semantic web. In *Extended Semantic Web Conference*. Springer, 2011.

[Scellato *et al.*, 2011] Salvatore Scellato, Anastasios Noulas, and Cecilia 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*. ACM, 2011.

[Shekarpour *et al.*, 2014] Saeedeh Shekarpour, Edgard Marx, Axel-Cyrille Ngonga Ngomo, and Sören Auer. Sina: Semantic interpretation of user queries for Question Answering on interlinked data. *Web Semantics*, 1:–, 2014.

[Sherif and Ngonga Ngomo, 2015] Mohamed Ahmed Sherif and Axel-Cyrille Ngonga Ngomo. An optimization approach for load balancing in parallel link discovery. In *SEMANTiCS 2015*, 2015.

[Shvaiko *et al.*, 2016] Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Michelle Cheatham, and Oktie Hassanzadeh, editors. *Proceedings of the 10th International Workshop on Ontology Matching*, volume 1545 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016.

[Singla and Domingos, 2006] Parag Singla and Pedro Domingos. Entity resolution with markov logic. In *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006.

[Spohr *et al.*, 2011] Dennis Spohr, Laura Hollink, and Philipp Cimiano. A machine learning approach to multilingual and cross-lingual ontology matching. In *International Semantic Web Conference*. Springer, 2011.

[Volz *et al.*, 2009] Julius Volz, Christian Bizer, Martin Gaedke, and Georgi Kobilarov. Silk-a link discovery framework for the web of data. *LDOW*, 538, 2009.

[Wang *et al.*, 2006] Junli Wang, Zhijun Ding, and Changjun Jiang. Gaom: Genetic algorithm based ontology matching. In *2006 IEEE Asia-Pacific Conference on Services Computing (APSCC'06)*. IEEE, 2006.

[Wang *et al.*, 2015] Quan Wang, Bin Wang, and Li Guo. Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015.

[Wuthrich, 1995] Beat Wuthrich. Probabilistic knowledge bases. *IEEE Transactions on Knowledge and Data Engineering*, 7(5), 1995.

[Xiao *et al.*, 2008] Chuan Xiao, Wei Wang, Xuemin Lin, and Jeffrey Xu Yu. Efficient similarity joins for near duplicate detection. In *WWW*, WWW '08, New York, NY, USA, 2008. ACM.

[Xiao *et al.*, 2016] Han Xiao, Minlie Huang, Yu Hao, and Xiaoyan Zhu. Transg: A generative mixture model for knowledge graph embedding. *ACL*, 2016.