MANDOLIN: Evaluation and Discussion

Tommaso Soru et al.

1 Experiments

1.1 Evaluation setup

Any directed labelled graph can be easily transformed into 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 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 MANDOLIN 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.

1.2 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:

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

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.

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.

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 [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, 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[1], we did not activate the RDF-specific settings. 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: $\gamma = 1M, 2M, 3M, 5M, 10M, 50M, 100M$. As can be seen in Figure 1, the runtime is, excluding an overhead, linear w.r.t. the number of iterations. 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. The Hits@10 index seems to stabilize at around $\gamma = 5M$, however higher accuracy can be found by increasing this value.

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

	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.

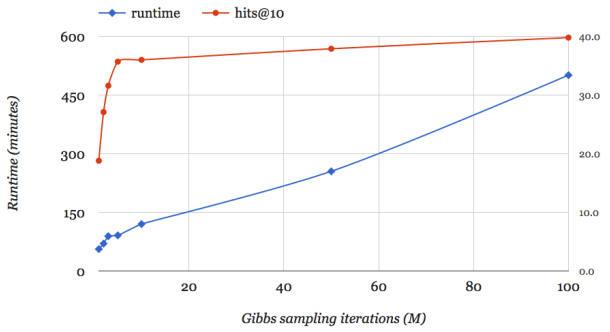

Figure 1: Runtime and Hits@10 on *FB15k* with $\bar{\eta} = 0.9$.

1.3 Large-scale datasets

We performed tests on large-scale datasets such as DBpedia[2] and Wikidata[3]. We compared our approach with other MLN frameworks, i.e. NetKit-SRL, ProbCog, Tuffy, and Alchemy. As the first three frameworks can learn rule weights but not rules themselves, we fed them with the rules found by our rule miner. Although Alchemy can learn rules, it uses an unoptimized approach considering all possible permutations, which is not feasible on medium- and large-sized dataset. Therefore, we also fed Alchemy with our rules. We set $\gamma = 0.9$ and $\bar{\eta} = 10M$. The results (see Table 2) showed that, in all cases, MANDOLIN 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.

Dataset	Runtime (s)	($\lvert\mathcal{F}'\rvert$)	($\lvert P\rvert$)
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.

[2]Version 2015-10 from `http://dbpedia.org`.
[3]Version 20150330 from `http://wikidata.org`.

2 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-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.

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.

References

[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.

[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.

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

[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.

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