Enhancing Knowledge Graph Completion By Embedding Correlations

Soumajit Pal*
Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
soumajitpal@gmail.com

Jacopo Urbani
Department of Computer Science
Vrije Universiteit Amsterdam, The Netherlands
jacopo@cs.vu.nl

ABSTRACT

Despite their large sizes, modern Knowledge Graphs (KGs) are still highly incomplete. Statistical relational learning methods can detect missing links by "embedding" the nodes and relations into latent feature tensors. Unfortunately, these methods are unable to learn good embeddings if the nodes are not well-connected. Our proposal is to learn embeddings for correlations between subgraphs and add a post-prediction phase to counter the lack of training data. This technique, applied on top of methods like TransE or HolE, can significantly increase the predictions on realistic KGs.

1 INTRODUCTION

Motivation. Currently, large amounts of semi-structured knowledge have been released on the Web in the form of Knowledge Graphs (KGs). Knowledge graphs are playing a prominent role in the transition from keyword-based to entity-based Web search. Several industrial organizations have been constructing KGs as part of their core business (e.g., Google's Knowledge Vault [3], Microsoft's Satori, etc.) and many not-for-profit projects have released large KGs on the Web as open linked data (e.g. YAGO [8]).

Problem. Despite modern KGs are fairly large, it is well known that they are still highly incomplete [3]. To solve this problem, we can apply Statistical Relational Learning (SRL) methods for predicting potential links between existing entities, and hence complete the graph. These methods learn numerical models and perform the predictions by means of algebraic operations. Many of such SRL methods have been proposed in literature ([1, 10, 11, 13] or see the survey at [9]) and the results are encouraging. For instance, *TransE* [1], one of the most popular method that "embeds" entities and relations into vectors of numerical latent features, is able to predict valid links for given entity/predicate queries (e.g., $livesIn(Bob, ?)$) on subsets of Freebase and Wordnet in respectively 75% and 94% of the cases[1]. This level of accuracy is impressive; unfortunately, SRL methods are unable to return good predictions if the graph is not sufficiently dense [4]. For instance, we applied TransE on a sample of YAGO (which is a graph significantly more sparse than the previous ones) and the same experiment returned a disappointing 22%.

In a sparse graph, the entities are not well connected, and consequently the embeddings are not sufficiently updated due to the lack of evidence. Modern KGs contain many of such low-connected entities, because typically their node degree distribution follow a power-law behavior. Therefore, it is important to study how we can improve the quality of prediction when the evidence is scarce.

Contribution. So far, existing latent-feature methods improve the quality of the predictions either by enhancing the learning or embeddings' structure (e.g. HolE [10] or variants of TransE like TransH [16] and TransR [7]) or by including external evidence in the process – like word embeddings [15], entities or taxonomies [6, 17] or rules [2, 12, 14]. In this paper, we propose a new type of optimization which consists of enriching the graph with new, special nodes that capture correlations between set of entities. Our intention is to embed such correlations into latent feature tensors, and exploit their semantics during the link prediction phase to boost the rankings of less-connected entities.

The special nodes are meant to represent star-shaped subgraphs of low-connected entities; thus they should inherit some of the links of the members of the subgraph. However, adding too many links would be counter productive because they will disrupt the original structure of the graph. To avoid this problem, we propose a procedure which judiciously add a minimal number of links that capture an interesting correlation.

Overall, our proposal has two main benefits: First, by adding nodes that represent subgraphs we enable a more efficient learning of correlations between entities that are not directly connected. We observed that in some cases this operation alone results in an improvement of the predictions. Second, by boosting the ranking of the subgraph nodes we give a chance to the less-connected entities to emerge as potential answers. This further improves the performance. We evaluated our contribution in combination with both TransE and HolE, which are two popular and state-of-the-art techniques. We report on significant improvements the performance on real-world datasets like Freebase or YAGO.

2 SRL AND SCARCITY OF EVIDENCE

Knowledge graphs consist of labeled directed multigraphs where the vertices represent entities and the labeled arcs establish semantic relations, e.g. $livesIn(Bob, London)$. Let $\mathcal{G} = (V, A, R)$ be a generic knowledge graph where V is the set of entities, A the set of arcs and R the set of labels for the arcs. We can represent the content of $\mathcal{G}$ with a set of binary relations $r(h, t)$ where $h, t \in V$ and there is an arc $\overrightarrow{(h, t)} \in A$ labeled with $r \in R$.

*Work done as master student at the Vrije Universiteit Amsterdam.
[1]These numbers refer to the filtered top-10 hit-ratio. See Tab. 2 in [10].

CIKM'17, November 6–10, 2017, Singapore.
© 2017 ACM. ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3133143

Given $\mathcal{G}$ as input, SRL methods are used to predict the likelihood that there is a labeled relation between two entities in $\mathcal{G}$. There are several types of SRL methods: Here we will focus on SRL methods which learn latent feature models and use them to make predictions. In general, these methods proceed by assigning one numerical vector $\mathbf{v} \in \mathbb{R}^d$ to each entity $v \in V$ and one vector $\mathbf{r} \in \mathbb{R}^d$ to each entity $r \in R$ where d is a hyperparameter that indicates the size of the vectors. These vectors are supposed to contain values for latent features and these are learned by minimizing a loss function using gradient-descent. For instance, TransE minimizes the margin-based loss function:

$$\mathcal{L} = \sum_{r(h,t) \in \mathcal{G}} \sum_{r(h',t') \notin S'_{r(h,t)}} [\gamma + dist(\mathbf{h}+\mathbf{r},\mathbf{t}) - dist(\mathbf{h'}+\mathbf{r},\mathbf{t'})]_+ \quad (1)$$

where $[.]_+$ denotes the positive part of ., $\gamma > 0$ is the margin, $dist$ is either the L_1 or L_2 norm, and S' is a set of negative samples constructed by replacing one term of a given tuple $r(h,t)$ with a random term [1]. HolE proposes a more sophisticated approach which uses circular correlation between the entity embeddings as $dist$. More specifically, $dist(\mathbf{h}+\mathbf{r},\mathbf{t})$ is replaced by $\sigma(\mathbf{r}^\top(\mathbf{h} \star \mathbf{t}))$ where σ is the logistic function, and $\mathbf{a} \star \mathbf{b} = \mathcal{F}^{-1}(\mathcal{F}(\mathbf{a}) \odot \mathcal{F}(\mathbf{b}))$ with $\mathcal{F}$ being the fast Fourier Transform and $\odot$ the Hadamard product.

After the model is trained, graph completion can be performed by identifying all entities that might be linked to a given entity and relation label. In other words, the goal is to find all answers to the query $r(h,?)$ (or $r(?,t)$) which are not explicit in the KG. This task is performed by calculating a likelihood score for any entity in the graph, and picking the entities with the best score. For instance, TransE determines the likelihood that $r(h,t)$ exists with the function

$$score(r,h,t) := -dist(\mathbf{h}+\mathbf{r},\mathbf{t}) \quad (2)$$

If the query is $r(h,?)$, then Eq. 2 is applied with all $t \in V$ and only the t's with the highest scores are candidates for new links. The case with $r(?,t)$ is entirely analogous.

The problem arises when the answers are nodes with low degree. We call these nodes "low-connected". In this case, the prediction will be harder because the training algorithm will have less evidence for these entities and thus it has less chances update their embeddings. In the case of TransE, this problem is accentuated because updates are triggered by violations (i.e., when $[.]_+ > 0$) and these occur only when the distance between the arc's ends differs from the distance with other random nodes for less than the given margin. This further reduces the chances of updates.

We give a more intuitive explanation of this problem with the fictional graph of Fig. 1a. Here, "California" and "Berkeley" are well-connected nodes while the students and courses aren't. Since Eq. 1 iterates over the edges, the embeddings for the first two nodes will have more chances to be updated than the other ones. Also, there is a clear correlation between living in California and studying at Berkeley in the sense that it is likely that Berkeley's students live in California. It is desirable that the model learns this correlation so that when we ask for all students that live in California, then the model can rank highly also the students for which we do not know the living location but we know they attend Berkeley.

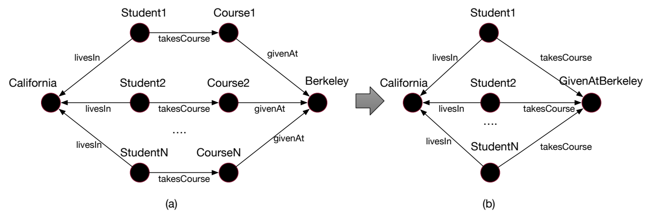


Figure 1: Example of Knowledge Graph (a) and with the addition of subgraph node (b).

Unfortunately, in this example learning such correlation is difficult because changes in one well-connected embedding can be "propagated" to the embeddings on other side of the graph only through the links between low-connected entities, which are considered at training time less times and whose update might not be large enough to trigger a violation. Our approach aims at overcoming this problem so that correlations like the one between studying at Berkeley and living in California can also be learned.

3 ADDING SUBGRAPH NODES

Our solution is conceptually simple: In order to strengthening the learning and prediction of low-connected entities, we add to the KG an additional set of special nodes which represent *sets* of entities. These sets contain entities which point to a common neighbor in the KG; therefore they form a star-shaped subgraph centered around the common neighbor. We call such special nodes *subgraph nodes*. In order to learn correlations, we add links between the neighbours of the subgraphs' members and the subgraph nodes. In this way, we are connecting the common neighbour of the subgraph to the nodes which are two links away. The following example illustrates the intuition behind this method.

Example 1. *Let us consider the graph in Fig. 1a. In this case, we can represent the subgraph of all courses given at Berkeley with a new node called* GivenAtBerkeley*, and add links from all students that follow courses at Berkeley to this node. The result can be seen in Fig. 1b: The node "GivenAtBerkeley" acts as a sort of "bridge" which encodes the correlation between the group of students that attend Berkeley and the courses given at Berkeley.*

The procedure for adding such special nodes is not straighforward as it seems. *First*, the subgraphs need to be sufficiently large in order to cluster together a sufficient number of entities. *Second*, we need to be careful because if we add too many links, then the resulting graph will be structurally too different and this will impact negatively the embeddings. In fact, every time we add a link the training algorithm will bring "closer" the embeddings of the two nodes incidents to the link and their neighbours. With too many links, the embeddings will become too close to each other and the ranking will be compromised.

In our proposal, we address the first problem by considering only subgraphs which are sufficiently large. Let $S_{(r,t)}$ be a set of all nodes in the KG which have an outgoing arc to t which is labeled

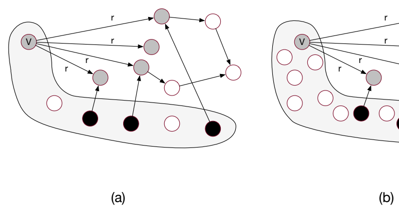


Figure 2: Example on the most appropriate subgraph for a entity v and a relation r by the second and third criteria.

with r. We add to the KG a node only if $|S_{(r,t)}| > \tau$, where τ is a hyperparameter (we used 10 in our experiments).

We address the second problem with a special procedure that adds only targeted links. This procedure visits each entity v in the original graph and consider all subgraphs which contain v. For each relation label r used by the links of v, we rank the subgraph nodes by a ranking function (θ, described below) and pick the subgraph node with the highest score. Then, we create a "r"-link from the neighbors of v only to that subgraph node.

Example 2. *Let us assume that the node $Course1$ is part of two different subgraphs, $GivenAtBerkeley$ and $AboutML$, and that this node is connected with the links $\langle Student1, takesCourse, Course1\rangle$ and $\langle Teacher1, teaches, Course1\rangle$. In this case, our procedure would consider the labels $takesCourse$ and $teaches$ and for each of them it will select one of the two subgraphs according to the ranking function θ. For example, the ranking function could select $GivenAtBerkeley$ for $takesCourse$ and $AboutML$ for $teaches$. Consequently our procedure would add two links $\langle Student1, takesCourse, GivenAtBerkeley\rangle$ and $\langle Teacher1, teaches, AboutML\rangle$.*

θ function. Now, we discuss how the subgraph nodes are selected. The function θ receives in input an entity v, one relation label r, and a non-empty set of subgraphs for which v is a member. It select one subgraph using three criteria:

First criterion: The function gives priority to the subgraphs $S_{(r',t)}$ where $r' \neq r$ to avoid re-stating with a new link information that is already contained in the graph.

Second criterion: The subgraphs are further ranked depending on how many members of the subgraph under exam are neighbors of the entities that are linked to v by r. We illustrate this operation in Fig. 2(a). Here, the neighbors of v are colored in dark grey while the members of the subgraph are within the light-grey area. The members of the subgraph that are directly connected to the neighbors of v are colored in black. The black nodes give an indication on how related the subgraph is to the neighbors of v. The function will choose the subgraph where the number of black nodes is maximal to reduce the risk that we are bridging unrelated entities.

Third criterion: If there are subgraphs with an equal number of "black" members, see e.g. Figs 2(a) and (b), then the function chooses the smallest subgraph. This case occurs when, for instance, one subgraph is more specific than another one (e.g. $(type, Student)$ and $(type, Person)$). If, after this step, there are still multiple elements to choose from, then the choice becomes arbitrary.

Post-prediction. The subgraph nodes become useful after we ranked the predictions with Eq. 2 because they give hints on what the real answers might be. To illustrate this point, consider again Fig. 1 and assume that we would like to find all possible answers for the query $takesCourse(Student2, ?)$. If the model returns the correlation node $S_{(givenAt, Berkeley)}$ among the top answers, then we can interpret this result as a hint that the correct answer should be one course from Berkeley and consequently increase the score of the members of this subgraph.

	FB15K	**LUBM**	**YAGO**
no. of. Subgraphs Detected	6,254	1,152	6,760
no. of. Additional Links	118,354	28,620	62,015
Processing Time (sec)	2,002	135	2,735

Table 1: Details about the special nodes to the KGs.

To exploit such hints, we apply the following procedure to the ranked list of answers produced with Eq. 2:

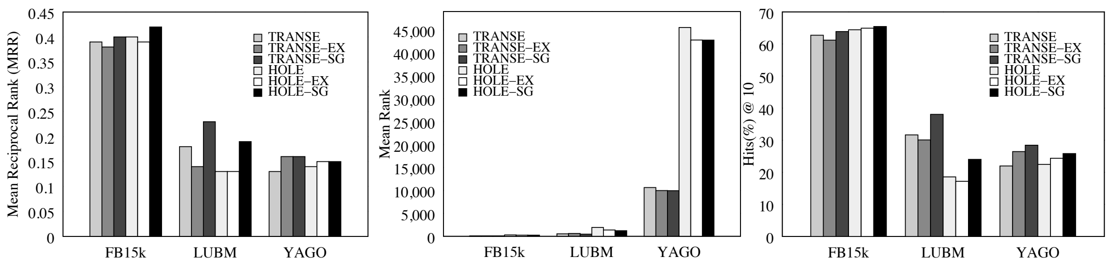
- We select the top-k_3 ranked subgraph nodes;
- From the top-k_3 ones, we pick the ones whose sets contain at least a given percentage of known answers of the query;
- We increase the score of the entities of the picked subgraphs. If an entity is part of multiple subgraphs, then it its score is increased multiple times;
- We increase the scores of the top k_4 entities in the original ranked list using a sufficiently high value to ensure that their positions remain unchanged.

This procedure requires three input parameters k_3, k_4, B. Ideal values for these parameters can be found as usual using grid search.

4 EVALUATION

We present a preliminary evaluation of our method in combination with TransE and HolE. We chose TransE because it is simple and widely used, and HolE because it is one of the most recent state-of-art KG completion models. Supplementary material is available at `https://github.com/karmaresearch/statlearning`. We used three benchmark datasets: i) *FB15K*, a subset of Freebase used in almost all related publications. It contains 14,951 entities, 1345 relation types and 592,213 relations. ii) *LUBM*, which is an artificial KG that is widely used to evaluate reasoning [5]. This dataset contains 17,277 entities, 23 relation types and 70,226 relations. iii) *YAGO*, which is a sample of YAGO3 [8] with 266,640 entities, 37 relation types and 562,817 relations. We selected YAGO and LUBM because they are both popular KGs which are much more sparse than Freebase. The experiments were ran on a machine with a Intel Xeon 32-core and 256GB RAM.

We proceeded as follows: First, we launched both TransE and HolE with optimized parameters on FB15K, LUBM and YAGO using the standard training/valid/test division. These will be our baselines. Then, we launched our procedure to extend the three KGs with the special nodes (see Tab. 1 for details) and re-trained new models. Then, we tested the quality of the predictions on the enriched KGs with and without the procedure to boost the rankings. As metrics, we chose, as usual, MRR (mean reciprocal rank), the mean ranks and the ratio of correct entries in the first 10 results, denoted as Hits@10. In the following, we report the "filtered" results, which exclude from the ranked list all the answers that are already in the KG. The unfiltered results follow the same behavior.



(a) MRR (higher is better). **(b) Mean Rank (lower is better).** **(c) Hits@10 (%, higher is better).**

Figure 3: Comparison vs the baseline on various metrics. The results with the correlation nodes but no post-prediction procedure are marked with “-EX”. The results with the post-prediction procedures are marked with “-SG”.

The results are reported in Fig. 3. First, we compare the results between the baselines and our enriched graph but without any post-prediction processing. In this case, we observe mixed results. In some cases, we obtained a general improvement, like with YAGO and on the mean rank with HolE on LUBM and FB15K (the last two improvements – of 41% and 15%, are not visible in the graph). In other cases, the performance was unchanged or slightly worse.

However, if we look at the performance where the post-prediction procedure was executed (the “-SG” column in Fig. 3), then we observe a general and consistent improvement of the performance. The MRR increases in all three datasets with both TransE and HolE (Fig. 3a). The largest improvement occurs with LUBM where the MRR improves of 28% with TransE and of 46% with HolE. On YAGO, the improvement is of 23% (TransE) and 7% (HolE). On FB15K, the MRR is only marginally improved (2.5% with TransE and 5% with HolE) since this graph is dense and the baselines are already performing well. From Fig. 3b, we notice that the mean rank is significantly higher on YAGO. This is because YAGO has many more entities (266K) than the other two and the predictions are worse. In general, the mean rank improved of 12% (TransE) and 37% (HolE) on LUBM, of 6% (both TransE and HolE) on YAGO, and 4% (TransE) and 15% (HolE) on FB15K. Finally, we notice that also the Hit@10 benefits from our method (Fig. 3c). The improvement is of 20% (TransE) and of 29% (HolE) on LUBM and of 6.5% (both) on YAGO. We observe also a small improvement on FB15K, which has already a high hit-rate (1.8% with TransE, 1.5% with HolE).

5 OUTLOOK

The main research stream in statistical knowledge graph completion has so far either focused on optimizing of the learning algorithm or on more complex data structures for the embeddings to capture the structure of the KG in the high-dimensional feature space. In this paper, we propose a new type of optimization that consists of enriching the KG with a new set of subgraph nodes in order to facilitate the learning of features that are difficult to capture due to the scarcity of evidence. Our preliminary evaluation showed that our proposal led to an improvement of the link prediction on all datasets and with all metrics. The fact that the improvement occurred with multiple off-the-shelf learning algorithms is remarkable. Nevertheless, future work is necessary, e.g., for avoiding the cases where accuracy decreased without the post-prediction phase.

Two main conclusions can be drawn from this work: First, enriching the KG with embeddings of more complex graph structures than single nodes is a viable and complementary research direction to enhance the quality of link prediction. Second, it is useful to apply some post-processing on the ranked results to see whether some answers should be included. Our experiments show that this results in a significant increase of the quality of the predictions.

REFERENCES

[1] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of NIPS*. 2787–2795.

[2] T. Demeester, T. Rocktäschel, and S. Riedel. 2016. Lifted Rule Injection for Relation Embeddings. *ArXiv e-prints* (June 2016). arXiv:cs.LG/1606.08359 Available at `http://arxiv.org/abs/1606.08359`.

[3] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-scale Approach to Probabilistic Knowledge Fusion. In *Proceedings of SIGKDD*. 601–610.

[4] Matthew Gardner, Partha Pratim Talukdar, Jayant Krishnamurthy, and Tom M. Mitchell. 2014. Incorporating Vector Space Similarity in Random Walk Inference over Knowledge Bases. In *EMNLP*. 397–406.

[5] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. 2005. LUBM: A Benchmark for OWL Knowledge Base Systems. *Journal of Web Semantics* 3, 2-3 (2005), 158–182.

[6] Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. 2015. Entity Hierarchy Embedding. In *Proceedings of ACL*. 1292–1300.

[7] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings AAAI*. 2181–2187.

[8] Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. 2014. YAGO3: A Knowledge Base from Multilingual Wikipedias. In *Proceedings of CIDR*.

[9] Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016. A Review of Relational Machine Learning for Knowledge Graphs. *Proc. IEEE* 104, 1 (2016), 11–33.

[10] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio. 2016. Holographic Embeddings of Knowledge Graphs. In *Proceedings of AAAI*. 1955–1961.

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *ICML*. 809–816.

[12] Tim Rocktaschel, Sameer Singh, and Sebastian Riedel. 2015. Injecting Logical Background Knowledge into Embeddings for Relation Extraction. In *Proceedings of NAACL*. 1119–1129.

[13] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Proceedings of NIPS*. 926–934.

[14] William Yang Wang and William W. Cohen. 2016. Learning First-Order Logic Embeddings via Matrix Factorization. In *Proceedings of IJCAI*. 2132–2138.

[15] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph and Text Jointly Embedding.. In *Proceedings of EMNLP*. 1591–1601.

[16] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI*. 1112–1119.

[17] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. In *Proceedings of AAAI*. 2659–2665.