Reflections on: Knowledge Graph Fact Prediction via Knowledge-Enriched Tensor Factorization

Ankur Padia, Konstantinos Kalpakis, Francis Ferraro, and Tim Finin

University of Maryland, Baltimore County, Baltimore, MD, USA
{ankurpadia, kalpakis, ferraro, finin}@umbc.edu

Abstract. We present a family of four novel methods for embedding knowledge graphs into real-valued tensors that capture the ordered relations found in RDF. Unlike many previous models, these can easily use prior background knowledge from users or existing knowledge graphs. We demonstrate our models on the task of *predicting new facts* on eight different knowledge graphs, achieving a 5% to 50% improvement over existing systems. Through experiments, we derived recommendations for selecting the best model based on knowledge graph characteristics. We also give a provably-convergent, *linear* tensor factorization algorithm.

1 Introduction

Knowledge graphs are increasingly important due to their effectiveness in supporting a wide range of applications. While immensely useful in their current state, much work remains to be done to detect and correct errors they contain and add missing relations. Representation learning [2] provides a way to augment or reduce reliance on manually constructed ontology axioms and rules by using knowledge graph instances to discover common patterns and then use them to suggest changes to the graph. One popular approach is based on learning how to *embed* the entities and relations in a graph into a real-valued vector space, allowing the entities and relations to be represented by dense, real-valued vectors. There is considerable interest in understanding how graph embeddings can be used to augment knowledge graphs [11,19].

Current state-of-the-art systems compute embeddings to support a specific task, which might be *link ranking* (aka *link recommendation*), or *fact prediction* (Table 1). Link ranking tries to augment the graph by recommending relations that could hold between a subject–object pair assuming (1) at least one relation exists between the pair and (2) we can determine a good threshold for choosing links that hold. The fact prediction task identifies potential facts and *classifies* them as correct or incorrect, learning a model that can find relations that are likely to hold. This task is more specific than link ranking and more directly solves an important problem. Since we are only interested in extending a knowledge graph with relations likely to be true, our approach is designed to solve it directly. Fact prediction models can also be used to filter triples produced by information extraction or inference system.

Embedding entities and relations into a vector space has been shown to achieve good results. Tensor–based approaches like RESCAL [14] jointly learn the latent representation of entities and relations by factorizing the tensor representation of the knowledge graph. This can be further improved by imposing constraints on the factors, such

Tasks	Alternate terminology	Definition	Example
Link ranking (ranking)	Link prediction Link recommendation	**Input**: Given relation r and entity e_i. (e_i,r, ?) **Output**: Rank possible entities e_j **– or –** **Input**: Given entity pair e_i and e_j. (e_i,?, e_j) **Output**: Rank possible relations, *r*	**Input**: Where is Statue of Liberty located? **Output**: (1) Germany (2) United States (3) **New York (city)** (4) New York (state) (5) Brazil
Fact prediction (classification)	Link classification Fact classification	**Input**: triple (aka fact), e_i, r, and e_j. **Output**: 0 (No) or 1 (Yes)	**Input**: Is the Statue of Liberty located in Germany? **Output**: 0 (No)

Table 1. We focus on a binary classification *fact prediction* task rather than *link ranking*

as non-negativity, to achieve better performance on sparse graphs. RESCAL and its variants [9,10,21] have achieved state-of-the-art results in predicting missing relations on real-world knowledge graphs. However, their extensions require additional schema information, which may be unknown or require significant effort to provide. Neural network based approaches, like TransE [4] learn an embedding of a knowledge graph by minimizing the ranking loss to rank likely links higher than unlikely ones.

However, these models do not exploit the similarity among relations when computing embeddings nor have they studied the role that relation similarities have on regularizing and constraining the relation embeddings and the subsequent effect on fact prediction. We addresses these deficiencies and make four contributions: we (1) develop a new graph embedding framework exploiting prior knowledge of relation-similarity; (2) provide four new embedding models; (3) evaluate the models and previous systems on eight real-world knowledge graphs for fact prediction; and (4) prove convergence for a factorization algorithm matching or outperforming baselines. In this extended abstract, we describe one of the model, Quadratic+Constraint (**QC**) model in detail and refer interested readers to the full paper [18].

2 Similarity-driven knowledge graph embedding

Our approach for similarity-driven knowledge graph embedding relies on learning entity and relation embedding such that when "combined" (multiplied), the result is a close approximation of the original facts and relation occurrences observed in the graph. We augment the learning process with a *relational similarity* matrix that provides a holistic judgment of how similar pairs of relations are, adding addtional constraints to the learned embeddings.

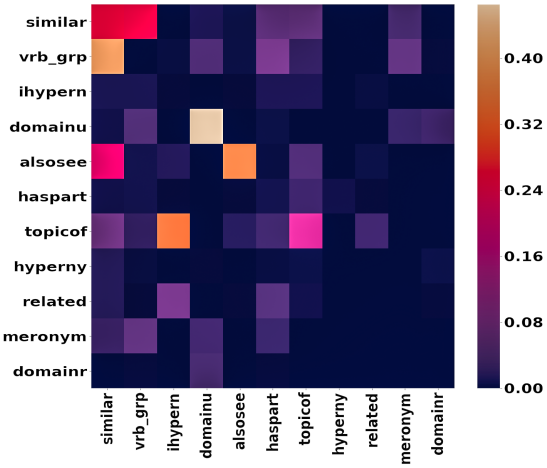

Fig. 1. Heatmap for WIN18RR's similarity matrix using transitivity (relation names abbreviated)

We represent a multi-relational knowledge graph of N_r binary relations among N_e entities by the order-3 tensor $\mathcal{X}$ of dimension

$N_e \times N_e \times N_r$ where each $N_e \times N_e$ is an adjacency matrix for relation k (called $slice$) has values of 1 when the i^{th} entity (e_i) is connected to the j^{th} entity (e_j) with relation k, else 0. This binary tensor is often very large and sparse. Our goal is to construct dense, informative p-dimensional embedding for entities and relations. We represent the collection of p-dimensional entity embeddings by $\mathcal{A}$ of size $N_e \times p$, a compact order-3 tensor $\mathcal{R}$ of size $p \times p \times N_r$, and the similarity matrix by C of size $N_r \times N_r$ (symbols are same as described in [18]).

Our objective is to reconstruct each of the k relation slices of $\mathcal{X}$ as the product

$$\mathbf{X}_k \approx \mathcal{A}_\alpha \mathbf{R}_k \mathcal{A}_\beta^\intercal. \tag{1}$$

Here both $\mathcal{A}_\alpha$ and $\mathcal{A}_\beta$ are matrices: each row is the embedding of an entity. By changing the exact form of $\mathcal{A}$—that is, the number of different entity matrices, or the different ways to index $\mathcal{A}$—we can then arrive at different models, each encapsulating both mathematical and philosophical differences. In this extended abstract, we describe just one of the four models, with full details on all available at [18]. We examine the case of having only a single entity embedding matrix, represented as $\mathbf{A}$—that is, $\mathcal{A}_\alpha = \mathcal{A}_\beta = \mathbf{A}$. This results in a quadratic reconstruction problem, as we approximate $\mathbf{X}_k \approx \mathbf{A}\mathbf{R}_k\mathbf{A}^\intercal$. We also examine the case of having two separate entity embedding matrices, represented as $\mathbf{A}_1$ and $\mathbf{A}_2$. This results in a reconstruction problem that is linear in the entity embeddings, as we approximate $\mathbf{X}_k \approx \mathbf{A}_1\mathbf{R}_k\mathbf{A}_2^\intercal$.

We learn $\mathbf{A}$, and $\mathcal{R}$ by minimizing the augmented reconstruction loss

$$\min_{\mathbf{A},\mathcal{R}} \underbrace{f(\mathbf{A},\mathcal{R})}_{\text{reconstruction loss}} + \overbrace{g(\mathbf{A},\mathcal{R})}^{\text{numerical regularization of the embeddings}} + \underbrace{f_s(\mathbf{A},\mathcal{R},C)}_{\text{knowledge-directed enrichment}}. \tag{2}$$

The first term of (2) reflects each of the k relational criteria given by (1). The second employs standard numerical regularization of the embeddings, such as Frobenius minimization, that enhances the algorithm's numerical stability and supports the interpretability of the resulting embeddings. The third term uses our relational similarity matrix C to enrich the learning process with our extra knowledge.

Computing relational similarity matrix $\boldsymbol{C}$. We can view a knowledge graph's nodes and edges as representing a flow of information, with subjects and objects acting as information producers and consumers, respectively. Tensor factorization captures this interaction [14]. Relations that occur on more of the same entities are more likely to have *some* notion of being similar. Each element of the $N_r \times N_r$ matrix $\boldsymbol{C}$ represents the similarity between a pair of relations, i.e., slices $\boldsymbol{X}_i$ and $\boldsymbol{X}_j$, and is computed using the following equation for all pairs of relations ($1 \leq i, j \leq N_r$):

$$(\textit{Symmetric}) \quad C_{i,j} = \frac{|(S(X_i) \cup O(X_i)) \cap (S(X_j) \cup O(X_j))|}{|(S(X_i) \cup O(X_i)) \cup (S(X_j) \cup O(X_j))|} \tag{3}$$

where $S(\boldsymbol{X}_i)$ is the set of subjects of the matrix $\boldsymbol{X}$ holding the i^{th} relation, and similarly for the object $O(\boldsymbol{X}_i)$. $|S(\boldsymbol{X})|$ gives the cardinality of the set.

We measure relation similarity using the overlap in the entities observed with each relation. In addition to computing similarity matrices based on relation *symmetry* using Eq. 3, we support four additional measures: (1) *Agency*: number of times both the relations have share the same subject, (2) *Patient*: number of times both the relations have share the same object, (3) *Transitivity*: number of times object of relation i is the subject of relation j, and (4) *Reverse Transitivity*: number of times subject of relation i is the object of relation j. We experimented with all of the similarity functions and report the evaluation result in Section 3. For most of our experiments we used the similarity obtained from transitivity, as we found it gave the best overall performance. Figure 1 shows the computed similarity matrix for the WordNet dataset.

The QC model: Quadratic+Constraint. The **QC** model casts the decomposition of the order-3 tensor $\mathcal{X}$ into a compact relational tensor $\mathcal{R}$ and quadratic entity matrix $\mathbf{A}$ and solve the following problem.

$$\min_{\mathbf{A},\mathbf{R}_k} f(\mathbf{A},\mathbf{R}_k) + g(\mathbf{A},\mathbf{R}_k) + f_{\text{Lag}}(\mathcal{R},\mathbf{C}) \tag{4}$$

The terms of our objective are expressed as follows.

$$f(\mathbf{A},\mathbf{R}_k) = \tfrac{1}{2}\sum_k ||\mathbf{X}_k - \mathbf{A}\mathbf{R}_k\mathbf{A}^T||_F^2 \tag{5}$$

$$g(\mathbf{A},\mathbf{R}_k) = \tfrac{1}{2}\lambda_a||\mathbf{A}||_F^2 + \tfrac{1}{2}\lambda_r \textstyle\sum_k ||\mathbf{R}_k||_F^2 \tag{6}$$

$$f_{\text{Lag}} = \sum_i\sum_j \lambda_{ij}(1 - ||\mathbf{R}_i - \mathbf{R}_j||_F^2 + \mathbf{C}_{ij}). \tag{7}$$

Here $\mathbf{A}$ is a $n \times p$ matrix where each row represents the entity embeddings and $\mathbf{R}_k$ is a $p \times p$ matrix representing the embedding for the k^{th} relation capturing the interaction between the entities. The first term f forces the reconstruction to be similar to the original tensor and the second regularizes the unknown $\mathbf{A}$ and $\mathbf{R}_k$ to avoid overfitting.

The f_{lag} term represents the model's knowledge-directed enrichment component. In order to incorporate similarity constraints, we solve the dual objective for a constraint in Eq. 8 such that relations with high similarity are near one another. We convert a constraint problem into an unconstrained problem via Lagrange multipliers λ_{ij}.

$$||\mathbf{R}_i - \mathbf{R}_j||_F^2 = 1 - C_{ij}, 1 \leq i, j \leq n. \tag{8}$$

Computing Factor Matrices, A, $\mathbf{R}_k$ and Lagrange Multipliers $\boldsymbol{\lambda_{ij}}$. We compute the unknown factor matrices using the Adam optimization [8] stochastic gradient descent extension. Each unknown is updated in the alternative fashion, in which each parameter is updated while treating the others as constants. The model's unknown parameters in $\mathbf{A}$ and $\mathbf{R}_k$ are updated with different learning rates. We empirically found that the error value of the objective function decreases after few iterations. Taking the partial derivative of the Eq. 4 with respect to $\mathbf{A}$ and equating to zero, we obtain the update rule for $\mathbf{A}$. Since we are indirectly constraining the embeddings of $\mathbf{A}$ through slices of the compact relation tensor $\mathcal{R}$, we obtain the same update rule for $\mathbf{A}$ as in RESCAL. By equating the partial derivatives of Eq. 4 with respect to the unknowns $\mathbf{R}_k$ and λ_{ij} to 0, we obtain updates for $\mathbf{R}_k$ and λ_{ij} as describe in the [18].

Name	# Entities	# Relations	# Facts	Avg. Degree	Graph Density
Kinship	104	26	10,686	102.75	0.98798
UMLS	135	49	6,752	50.01	0.37048
FB15-237	14,541	237	310,116	21.32	0.00147
DB10k	4,397	140	10,000	2.27	0.00052
FrameNet	22,298	16	62,344	2.79	0.00013
WN18	40,943	18	151,442	3.70	0.00009
FB13	81,061	13	360,517	4.45	0.00005
WN18RR	40,943	11	93,003	2.27	0.00005

Table 2. Statistics of the eight datasets used in evaluation experiments

3 Experimental evaluation

We evaluated the performance of the entity and relation embeddings on the *fact prediction* task, which distinguishes correct from incorrect triples, and compared the results with state-of-the-art tensor decomposition models RESCAL and NN-RESCAL, and two popular neural-based benchmarks, DistMult (which considers relation embedding matrices to be diagonal) and ComplEx (which represents entities and relation in complex vector space). Here we present results only for the tensor-based models; additional evaluation data and comparisons with neural-based models are in [18].

Datasets. Table 3 summarizes the key statistics of the datasets: the number of entities (N_e), relations (N_r) and facts (non-zero entries in the tensor), the average degree of entities across all relations (the ratio of facts to entities) and the graph density (the number of facts divided by square of the number of entities). A smaller average degree or graph density indicates that the knowledge graph is sparser.

We used eight datasets in our evaluation, including both previous graph-embedding benchmarks and new ones. They include (1) **Kinship** [7], which encodes complex family links among 104 members of a tribe, (2) **UMLS**, a biomedical dataset [7] based on the Unified Medical Language System, (3,4) **WN18** [3] and **WN18RR** [5] (reverse removed), linguistic datasets of relations between words like *hypernym*, *holonym*, *meronym* and *hyponym*, (5,6,7) three general world knowledge graphs: **FB13** [3] , **DB10k** and **FB15-237** which contained more relations compared to **FB13**, and (8) **FrameNet** [1], a lexical database describing how language can be used to evoke complex representations of *frames* describing events, relations or objects and their participants. For example, the *Commerce_buy* frame represents the interrelated concepts surrounding stereotypical commercial transactions. Frames have roles for expected participants (e.g., *Buyer*, *Goods*, *Seller*), modifiers (e.g., *Imposed_purpose* and *Period_of_iterations*), and inter-frame relations defining *inheritance* and *usage* hierarchies (e.g., *Commerce_buy* inherits from the more general *Getting* and is inherited by the more specific *Renting*.

Tensor creation, parameter selection and performance metric. We created a 0-1 tensor for each dataset, as described in [18]. If entity *s* has relation *r* with entity *o*, then the value of the (*s, r, o*) entry in the tensor is set to 1, otherwise it is set to 0. Each of the created tensors is used to generate a slice-similarity matrix using Eq. 3. We fix the parameters for different datasets using coordinate descent, changing only one hyperparameter at a time and always making a change from the best configuration of hyperpa-

Model Name	Kinship	UMLS	WN18	FB13	DB10	Framenet	WN18RR	FB15-237
Previous tensor factorization models								
RESCAL	93.24	**88.53**	62.13	65.37	61.27	82.54	66.63	92.56
NN-RESCAL	92.19	88.37	83.93	79.13	**81.72**	82.6	68.49	93.03
Linear/Quadratic Regularized/Constrained tensor factorization models								
LR	**93.99**	88.22	81.86	80.07	80.79	78.11	69.15	90.00
QR	93.89	88.11	84.41	79.12	80.47	82.34	66.73	**93.07**
LC	92.87	84.71	80.18	75.79	80.67	73.64	66.46	81.88
★ **QC**	93.84	86.17	**91.07**	**85.15**	81.69	**86.24**	**72.62**	86.47

Table 3. Fact prediction AUC performance for all models. ★ indicating best overall. **LR** and **LC** are linear regularized/constrained models; **QR** and **QC** are quadratic regularized/constrained.

rameters found so far. The latent dimension is equal to the number of relations. We use the same performance metric as RESCAL on three samples: *stratified-uniform* (sampling 60% correct and 40% incorrect from each relation), *stratified-weighted* (sampling 60% correct and 40% incorrect from the public dataset with few relations are mentioned more frequently then other), and balanced-weighted (dataset mentioned in [20]).

Results and discussion. In this section we analyze and the results of our models, which include a quantitative comparison with other tensor-based models and the impact of knowledge graph sparsity on the models. Table 3 has the AUC performance of our models which range from 5% to 50%.

The Kinship and UMLS datasets have a significantly higher graph density compared to the others, as shown in Table 3. Combining this observation with the results in Table 3, we notice that graphs with lower density result in larger performance variability across both the baseline systems and our models. This suggests that when learning knowledge graph embeddings on *dense* graphs, basic tensor methods like RESCAL can give acceptable performance, but that for lower density graphs, different embedding learning mechanisms may be better. Focusing on the datasets with lower graph density, we see that while the **LC** and **LR** models often matched or surpassed RESCAL, they achieved comparable or lower performance compared to their corresponding quadratic models (**QC** and **QR**). This is due to the fact that the distinction of the subject and object made by $\mathbf{A}_1$ and $\mathbf{A}_2$ embeddings tends not to hold in many of the standard datasets. That is, objects can behave as subjects (and vice versa), as in the WN18 dataset. Hence the distinction between the subject and the object may not always be needed.

The performance difference between the quadratic and linear versions is high for WN18 and FB13, though the difference is relatively small for DB10k. This is largely because the DBpedia dataset includes many datatype properties, i.e., properties whose values are literals rather than entities.

In most cases, non-negative RESCAL outperforms the linear models. The **QC** model significantly outperforms RESCAL and performs relatively better compared to our other three models, emphasizing the importance of the flexible penalization that the Lagrange multipliers provide. Compared to all others, the **QC** model performs better in most of the cases, since the Lagrange multiplier introduces flexibility in penalizing the latent relational embeddings while learning. We also did significance tests using a

Wilcoxon signed-rank test for all algorithms and datasets at significance level of 1% and found the **QC** model to perform better compared to the others. In summary, both the quadratic and linear models are important depending on the data, with the **QC** model performing the best overall and the Linear models performing comparably.

Effect of knowledge graph density. We explored how our models handle knowledge graphs of different density by reducing the number of subjects while keeping the objects constant, making the graph sparser. We used the FB13 dataset, which has nearly 16K objects and 76K subjects, so each object entity is connected to nearly five subjects on average. Figure 2 shows the behavior of different tensor based models on reducing number of subjects at step size of 2% with higher the value increasingly removing the number of subjects. We see that removing subjects gradually reduces performance. Compared to other models, **LC** improves significantly faster when subjects are added back, irrespective of the similarity metric, eventually achieving comparable performance with other tensor based models. The **QC** model performs the best irrespective of the graph's density.

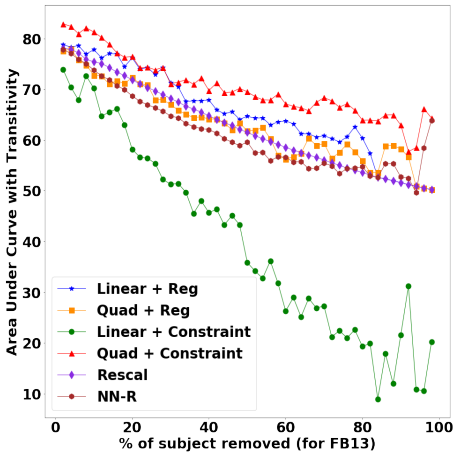

Fig. 2. Change in performance with sparse knowledge graph

Effect of similarity encoding. Figure 3 shows the relative changes in performance of each similarity metric compared to RESCAL, grouped by how we encode the knowledge. The gray boxes show the percent change of the two RESCAL versions. Most encoding approaches perform well, but the encoding can yield a significant performance gain for certain datasets. In DB10 (top left) using **LR** the *agency* and *symmetric* encodings give poor performance. Changing the encoding to *transitivity* or *reverse_transitivity* yields a large performance gain. On the other hand, for WN18RR both *transitivity* and *reverse_transitivity* with the **LR** model perform poorly. The **LC** model performs similarly for all kinds of encoding. Moreover, **QC** performs consistently well compared to all the baselines without being affected by the similarity encoding. While we find that different kinds of similarity encoding methods *can*, and do, influence performance, we see the effect of how that knowledge is encoded in these datasets. For example, whether an encoding uses a symmetric or transitive approach may be less important than whether or not accurate knowledge is encoded at all. The knowledge enrichment that the encoding provides can result in effective model generalization beyond what simple, knowledge-poor regularizations, like Frobenius norm regularization, give.

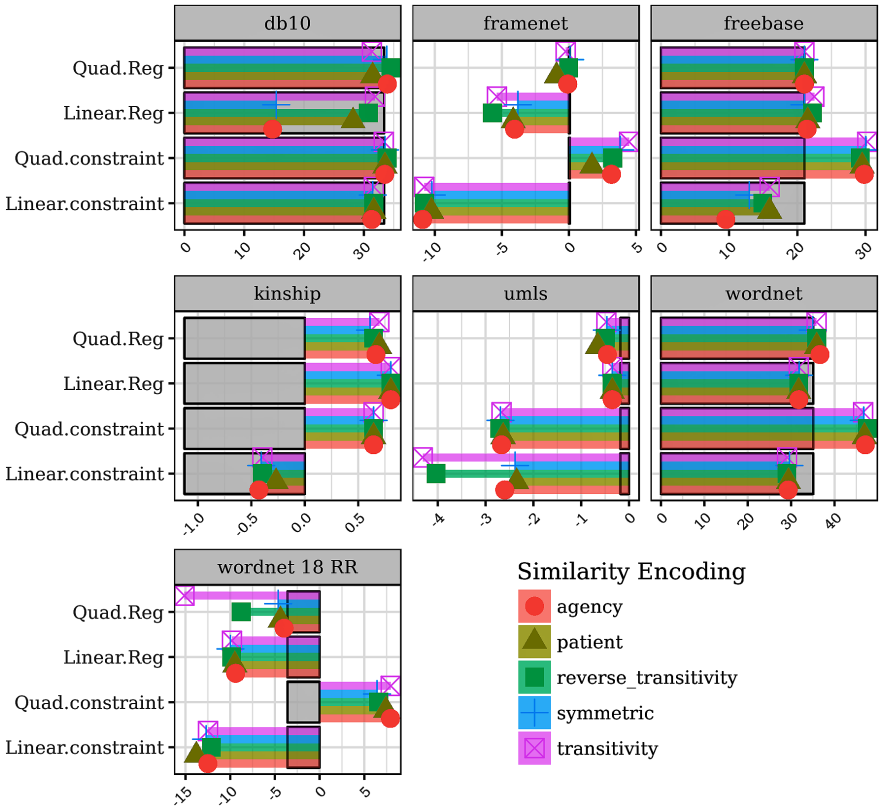

Fig. 3. % AUC change over RESCAL for five different similarity encodings.

4 Conclusions and future work

We described a new framework for learning knowledge-enriched entity and relation embeddings and four readily obtainable models that generalize existing efforts and demonstrate significant improvements over both state-of-the-art tensor decomposition and neural-based translation models. We motivated and empirically explored different methods for encoding prior knowledge into the tensor factorization algorithm, finding that using transitive relationship chains resulted in the highest overall performance. We further characterized the conditions under which each model performed well and concluded the **QC** model is typically the best choice. Finally, we proved in [18] that the **LR** model has the desirable property of convergence.

Our future work will use the KGFP framework for fact prediction in three different scenarios: (1) improving information extraction in tasks like the TAC Knowledge Base Population [6], (2) enhancing the knowledge graphs used in systems [13,12] that identify possible cybersecurity attacks, and (3) cleaning noisy knowledge graphs [16,17] by identifying and possibly correcting errors. The evaluation datasets and some code will be available at the KGFP repository [15].

References

1. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The berkeley framenet project. In: ACL. pp. 86–90. ACL (1998)
2. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. Transactions on Pattern Analysis and Machine Intelligence **35**(8) (2013)
3. Bordes, A., Glorot, X., Weston, J., Bengio, Y.: A semantic matching energy function for learning with multi-relational data. Machine Learning pp. 233–259 (2014)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: NIPS. pp. 2787–2795 (2013)
5. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. arXiv preprint arXiv:1707.01476 (July 2018), extended AAAI18 paper
6. Finin, T., Lawrie, D., McNamee, P., Mayfield, J., Oard, D., Peng, N., Gao, N., Lin, Y.C., MacKin, J., Dowd, T.: HLTCOE Participation in TAC KBP 2015: Cold Start and TEDL. In: 8th Text Analysis Conf. NIST (November 2015)
7. Kemp, C., Tenenbaum, J.B., Griffiths, T.L., Yamada, T., Ueda, N.: Learning systems of concepts with an infinite relational model. In: AAAI. pp. 381—-388. AAAI (2006)
8. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: Third Int. Conf. on Learning Representations (December 2014)
9. Krompass, D., Baier, S., Tresp, V.: Type-constrained representation learning in knowledge graphs. In: Int. Semantic Web Conf. Springer (2015)
10. Krompass, D., Nickel, M., Jiang, X., Tresp, V.: Non-negative tensor factorization with RESCAL. In: Tensor Methods for Machine Learning, ECML Workshop (2013)
11. Meilicke, C., Fink, M., Wang, Y., Ruffinelli, D., Gemulla, R., Stuckenschmidt, H.: Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In: International Semantic Web Conference. pp. 3–20. Springer (2018)
12. Mittal, S., Joshi, A., Finin, T.: Cyber-all-intel: An AI for security related threat intelligence. arXiv preprint arXiv:1905.02895 (2019)
13. Narayanan, S.N., Ganesan, A., Joshi, K.P., Oates, T., Joshi, A., Finin, T.: Early detection of cybersecurity threats using collaborative cognition. In: Int. Conf. on Collaboration and Internet Computing. IEEE (2018)
14. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: 28th Int. Conf. on machine learning (ICML-11). pp. 809–816 (2011)
15. Padia, A.: KGFP repository. https://github.com/Ebiquity/KGFP.git
16. Padia, A.: Cleaning Noisy Knowledge Graphs. In: Doctoral Consortium at the 16th Int. Semantic Web Conf. vol. 1962. CEUR Workshop Proceedings (October 2017)
17. Padia, A., Ferraro, F., Finin, T.: KGCleaner: Identifying and correcting errors produced by information extraction systems. arXiv preprint arXiv:1808.04816 (August 2018)
18. Padia, A., Kalpakis, K., Ferraro, F., Finin, T.: Knowledge Graph Fact Prediction via Knowledge-Enriched Tensor Factorization. Journal of Web Semantics (January 2019)
19. Ristoski, P., Rosati, J., Di Noia, T., De Leone, R., Paulheim, H.: RDF2Vec: RDF graph embeddings and their applications. Semantic Web (2019)
20. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in neural information processing systems. pp. 926–934 (2013)
21. Yang, B., Yih, W.T., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)