Learning Entity Type Embeddings for Knowledge Graph Completion

Changsung Moon
North Carolina State University
Raleigh, NC 27606
cmoon2@ncsu.edu

Paul Jones
North Carolina State University
Raleigh, NC 27606
pjones@ncsu.edu

Nagiza F. Samatova*
North Carolina State University
Raleigh, NC 27606
Oak Ridge National Laboratory
Oak Ridge, TN 37831
samatova@csc.ncsu.edu

ABSTRACT

Missing data is a severe problem for algorithms that operate over knowledge graphs (KGs). Most previous research in KG completion has focused on the problem of inferring missing entities and missing relation types between entities. However, in addition to these, many KGs also suffer from missing entity types (i.e. the category labels for entities, such as */music/artist*). Entity types are a critical enabler for many NLP tasks that use KGs as a reference source, and inferring missing entity types remains an important outstanding obstacle in the field of KG completion. Inspired by recent work to build a contextual KG embedding model, we propose a novel approach to address the entity type prediction problem. We compare the performance of our method with several state-of-the-art KG embedding methods, and show that our approach gives higher prediction accuracy compared to baseline algorithms on two real-world datasets. Our approach also produces consistently high accuracy when inferring entities and relation types, as well as the primary task of inferring entity types. This is in contrast to many of the baseline methods that specialize in one prediction task or another. We achieve this while preserving linear scalability with the number of entity types. Source code and datasets from this paper can be found at (https://github.ncsu.edu/cmoon2/kg).

CCS CONCEPTS

•Computing methodologies →Machine learning; Machine learning approaches; Learning latent representations;

KEYWORDS

Knowledge Graph Completion, KG Embedding Method, Vector Embedding, Entity Type Prediction

1 INTRODUCTION

Knowledge graphs (KGs) can be represented as a multigraph with nodes corresponding to entities and edge labels corresponding to relation types. They store information as triples, with each triple consisting of two entities (subject and object) and the relationship (predicate); for example: (*Spock*, *characterIn*, *Star Trek*). KGs are often large (up to 1B triples) and they invariably suffer from missing data issues. For these reasons, it is critical to develop methods that address the task of KG completion (i.e. the inference of missing data) at scale. Most previous work in KG completion has focused on inferring missing entities and missing relation types. However, many KGs also suffer from missing entity types (or categories). For example, 10% of entities in the FB15k dataset, which have the type */music/artist* do not have the type */people/person* in the Freebase knowledge graph. Entity type information in KGs is widely used in various NLP tasks such as relation extraction [4], entity linking [6], and question answering [1, 16]. Missing entity types can undermine algorithm effectiveness in these kinds of tasks.

Many existing methods [2, 3, 7, 11–15] in KG embeddings focus on learning vector representations (or 'embeddings') for entities and for relation types. However, we observe that similar methods can also be used to learn embeddings for entity types. The relationship between an entity and its type can be represented by a triple such as (*entity*, *predicate*, *entity_type*), in which the *entity_type* is referred to as the object. However, when using this construct to represent entities and entity types, only one value for the *predicate* field is typically used. This lack of diversity of relation types means that KG embedding methods have particular difficulty making entity type prediction. To address this issue, a method [10] was proposed to infer missing entity types by combining information on the hierarchical structure of entity types from the Freebase KG, with external context information from Wikipedia. However, this approach is complicated and requires access to the additional information source(s), which might not be possible for some KGs.

To address these issues, we propose an embedding methodology that can handle the lack of diversity of relation types and does not need external information. Our key observation is that trained entity representations from a state-of-the-art KG embedding method [9] can be clustered well according to their entity types in the embedding space, and therefore the embeddings can be used to predict the associated types. We describe a novel approach based on this observation, and evaluate it against several baselines.

2 PROBLEM STATEMENT

In KGs, there is a relation type (such as "*rdf:type*") that indicates that the object is actually the entity type of the subject in a triple. For instance, such a set of triples may look like those in Table 1.

*Corresponding author

CIKM'17, November 6–10, 2017, Singapore.
 ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3133095

Table 1: Example of Triples used to represent Entity Types

Subject (Ent.)	Predicate (Rel. Type)	Object (Ent. Type)
("Michael Jackson",	"rdf:type"	"/music/artist")
("Michael Jackson",	"rdf:type"	"/film/actor")
("New York City",	"rdf:type"	"/location/citytown")
...	...	...

The problem of entity type prediction can be formally defined as follows: Let $E = \{e_1, e_2, \ldots, e_n\}$ be a set of all entities, and let $T = \{t_1, t_2, \ldots, t_m\}$ be a set of all entity types. A triple can be denoted by (ϵ, p, τ) where $\epsilon \in E$, p is the predicate "*rdf:type*" and $\tau \in T$. The problem of entity type prediction is to determine a score $\psi(\tau = t_i \mid \epsilon)$, for all $t_i \in T$, where we would like the score to roughly correspond to a probability of each entity type.

We would then like to use these ψ scores to make '*Hits@N*' predictions that indicate the top-N most probable entity types. We could then determine the accuracy of predictions in simple percentage terms; however, a more robust statistical measure of the quality of our predictions can be obtained from the mean reciprocal rank (MRR) metric, which is computed as follows:

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

where Q is a set of test triples, and $rank_i$ is the rank position of the true entity type for the i-th triple.

3 METHOD

First of all, we choose a specific KG embedding model, ConTE [9], to test the feasibility of using it for the task of entity type prediction. The authors of this method show more scalable and better accuracy performance for the prediction of both missing entities and relation types than other methods. ConTE learns the embeddings of entities and relation types while taking contextual relation types into account. The basic idea of ConTE is that it includes outgoing (from the subject) and incoming (to the object) relation types of a triple as contextual relation types C into the embedding learning process for a triple (s, p, o) as follows:

$$\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{e}_c \approx \boldsymbol{e}_p \qquad \text{for } \forall c \in C \tag{2}$$

where $\boldsymbol{e}_s, \boldsymbol{e}_o, \boldsymbol{e}_c$ and $\boldsymbol{e}_p \in \mathbb{R}^k$ are vector embeddings of a subject s, an object o, a contextual relation type c, and a predicate p, respectively. If the triple (s, p, o) exists in the dataset, $\boldsymbol{e}_p$ should be close to $\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{e}_c$; otherwise $\boldsymbol{e}_s + \boldsymbol{e}_o + \boldsymbol{e}_c$ should be far away from $\boldsymbol{e}_p$.

In this section, we show that the entity vectors trained by ConTE cluster well according to their entity types. Figure 1 shows a scatter plot of Freebase entities, which have one of six example entity types. We trained the model for the FB15k training set and used the *t-SNE* algorithm [8] to project and visualize entity embeddings in a 2-dimensional space. The figure shows that entities with the same entity type tend to appear in well-defined clusters in the embedding space. For example, in the figure, the blue dots indicate the entities with the */film/film* type. The *t-SNE* plot shows that this model can make the film entities appear close to each other in the embedding space. In addition, the group of entities with */tv/tv_actor* and those with */book/author* are closer to each other than entities with other types, and they show some overlap. These entities have

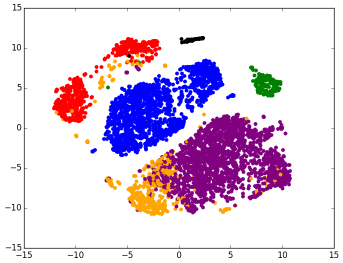

Figure 1: A t-SNE plot of entities with an entity type (Red: */education/educational_institution*, Blue: */film/film*, Purple: */tv/tv_actor*, Orange: */book/author*, Green: */tv/tv_program*, Black: */music/instrument*)

some common types including */person/person*, which is the reason that they are close to each other in the embedding space.

In our proposed approach, we build on the observation that the ConTE model embeds entities in such a way that they are close to each other in the vector space when they have the same or similar types. First of all, we learn embeddings of entities and relation types with ConTE on KGs that don't have entity types. Both the trained embeddings of entities and a taxonomy of entity types form the inputs to our method that learns embeddings of entity types. Our method is called ETE, which stands for Entity Type Embeddings. In the vector space, the method embeds entity types close to their entities. The final ETE model is used to infer missing entity types as well as missing entities and relation types.

3.1 Learning Embeddings of Entity Types

Given an entity ϵ, we learn vector embeddings of the entity types of ϵ. The set of entity types of ϵ is denoted by T_ϵ. The key idea of our method is based on the observation that missing entity types of an entity can be found from other entities that are close to the entity in the vector space.

To incorporate this observation into our model, we train embeddings of each entity type of T_ϵ to be closer to the embeddings of the entity ϵ as follows:

$$\boldsymbol{e}_\epsilon \approx \boldsymbol{e}_\tau \qquad \text{for } \forall \tau \in T_\epsilon \tag{3}$$

where $\boldsymbol{e}_\epsilon$ and $\boldsymbol{e}_\tau \in \mathbb{R}^k$ are the embedding vectors of the entity ϵ and the entity type τ, respectively. When the triple (ϵ, p, τ) exists in the training set, $\boldsymbol{e}_\tau$ should be close to $\boldsymbol{e}_\epsilon$; when the tuple does not exist, $\boldsymbol{e}_\tau$ should be far away from $\boldsymbol{e}_\epsilon$. When we train vectors for (ϵ, p, τ), we update only $\boldsymbol{e}_\tau$, not $\boldsymbol{e}_\epsilon$ that has been trained by the ConTE model. This update approach means that our ETE model is optimized for inferring missing entities and relation types as well as entity types since the trained embeddings of entities and relation types are preserved. To calculate the dissimilarity $d(\boldsymbol{e}_\epsilon, \boldsymbol{e}_\tau)$ between two vectors $\boldsymbol{e}_\epsilon$ and $\boldsymbol{e}_\tau$, we use the L_1-norm as follows:

$$d(\boldsymbol{e}_\epsilon, \boldsymbol{e}_\tau) = \sum_{i=1}^{k} \mid \boldsymbol{e}_\epsilon[i] - \boldsymbol{e}_\tau[i] \mid \tag{4}$$

where k is the number of dimensions of a vector. We use $-d(\boldsymbol{e}_\epsilon, \boldsymbol{e}_\tau)$ as the similarity function.

Figure 2 shows our model with an example of an entity *Elvis_Presley* and its entity types $T_{Elvis\_Presley} = \{People\_from\_Mississippi,$

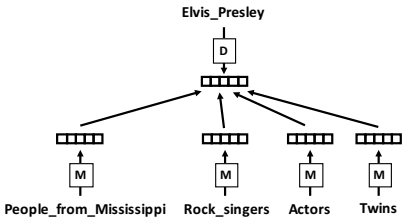

Figure 2: A framework for learning vectors of entity types

Rock_singers, *Actors*, *Twins*, ...}. The matrix D is the output from the ConTE model. Each row represents the trained embeddings for each entity. Each entity type is also mapped to a unique vector, which is represented by a row in the matrix M. Our model scales linearly with the number of entity types as a new entity type can be easily added by adding a new row into M. We compute the similarities between the entity vector $\boldsymbol{e}_{Elvis\_Presley}$ and each entity type vector $\boldsymbol{e}_{\tau \in T_{Elvis\_Presley}}$. Our model optimizes $\boldsymbol{e}_\tau$ to maximize the similarity values. In addition, we use a negative sampling approach and stochastic gradient descent (SGD) with AdaGrad [5] as our optimization approach to improve convergence performance.

3.2 Negative Sampling

In the case of our problem, the positive triples consist of pairs of an entity and an entity type that exist in a training set. The negative samples are non-existent triples in the training set. Our negative sampling is defined as follows:

$$S'_{(\epsilon,p,\tau)} = \{(\epsilon, p, \tau') \mid \tau' \in T\} \tag{5}$$

For each triple (ϵ, p, τ) in the training dataset, we generate the set of negative samples with the entity type replaced by random entity types. Our model is optimized to ensure the similarity between the entity and its type in the training triple (ϵ, p, τ) will be higher than those in the triples (ϵ, p, τ') from the negative samples.

3.3 Margin-Based Ranking Loss Function

We minimize the following margin-based ranking loss function over the training set:

$$\mathcal{L} = \sum_{(\epsilon,p,\tau)\in S} \left(\sum_{(\epsilon,p,\tau')\in S'_{(\epsilon,p,\tau)}} max(0, \gamma + d(\boldsymbol{e}_\epsilon, \boldsymbol{e}_\tau) - d(\boldsymbol{e}_\epsilon, \boldsymbol{e}_{\tau'})) \right) \tag{6}$$

where $\gamma > 0$ is the margin. This loss function ensures the similarity between vectors of the entity and the entity type from positive triples will be higher than the similarity between vectors from negative samples.

3.4 Prediction

Given an entity ϵ, we rank entity types for ϵ by using the following score function:

$$\psi(\tau = t_i \mid \epsilon) = -d(\boldsymbol{e}_\epsilon, \boldsymbol{e}_{t_i}) \qquad \text{for } \forall t_i \in T \tag{7}$$

This score function computes the similarity between each entity type t_i and the entity ϵ. The scores are ranked to find the N highest scored types, which are used for our *Hits@N* predictions. In addition, the quality of the ranking is measured using the mean reciprocal rank (MRR) statistic.

Table 2: Characterization of Datasets

Dataset	FB15k	YAGO43k	FB15kET	YAGO43kET
Rel. types	1,345	37	1,346	38
Entities	14,951	42,975	14,951	42,975
Entity types	-	-	3,851	49,980
Train. triples	483,142	331,687	619,760	709,859
Valid. triples	50,000	30,000	16,000	46,000
Test triples	59,071	30,000	16,000	46,000

4 EMPIRICAL EVALUATION

4.1 Datasets

To demonstrate the effectiveness of our proposed approach, we compare the accuracy performance on two real-world KGs. A description of the datasets is given in Table 2. FB15k [2] and YAGO43k [9] are subsets of Freebase and YAGO, respectively. They consist of only entities and relation types (i.e., all entity types are missing). We also collected entity types that are mapped to entities in both datasets from [15] for FB15k and the YAGO taxonomy[1] for YAGO43k. We added additional training triples for the collected entity types into the training sets of the two datasets. They consist of an entity (subject), an entity type (object) and the relation type "*rdf:type*" (predicate); for example (*Elvis_Presley*, *rdf:type*, *Rock_singers*). We call the merged datasets FB15kET and YAGO43kET, respectively. We validated and tested our model and all of baselines on the FB15kET and YAGO43kET validation and test sets, respectively. These sets consist of only triples of an entity (subject), a missing entity type (object) and "*rdf:type*" (predicate).

4.2 Parameter Selection

ETE and baseline methods depend mainly on three parameters. These parameters and the values we tested in our experiments are as follows:

- k – the number of dimensions: {20, 50, 100, 150, 200}
- γ – the margin: {0.1, 0.2, 0.5, 1.0, 2.0, 4.0}
- λ – the learning rate: {0.01, 0.05, 0.1, 0.15, 0.2, 0.25}

To select these parameter values, our method and the baselines were run on the validation sets of FB15kET and YAGO43kET, and the best combination of parameters (according to MRR) was selected for each method. We obtained the following combinations:

- RESCAL – $\{k = 150, \gamma = 0.2, \lambda = 0.1\}$
- RESCAL-ET – $\{k = 150, \gamma = 0.2, \lambda = 0.1\}$
- TRANSE – $\{k = 50, \gamma = 2.0, \lambda = 0.1\}$
- TRANSE-ET – $\{k = 50, \gamma = 2.0, \lambda = 0.1\}$
- HOLE – $\{k = 200, \gamma = 0.2, \lambda = 0.1\}$
- HOLE-ET – $\{k = 200, \gamma = 0.2, \lambda = 0.1\}$
- ETE – $\{k = 200, \gamma = 2.0, \lambda = 0.1\}$

We tested three baseline knowledge graph embedding methods, RESCAL [12], TRANSE [2] and HOLE [11], for entity type prediction. In addition, we also applied our entity-type embedding approach to these baselines in the same way as the CONTE method to determine how much they could be improved. We label these additional baselines RESCAL-ET, TRANSE-ET and HOLE-ET, respectively.

[1] http://www.yago-knowledge.org/

Table 3: Entity Type Prediction Accuracy on FB15kET

Method	MRR		Hits @		
	Raw	Filt	1	3	10
Rescal	0.07	0.19	9.71	19.58	37.58
Rescal-ET	0.10	0.24	12.17	27.92	50.72
TransE	0.15	0.45	31.51	51.45	**73.93**
TransE-ET	**0.17**	0.46	33.56	52.96	71.16
HolE	0.08	0.22	13.29	23.35	38.16
HolE-ET	0.15	0.42	29.40	48.04	66.73
ETE	**0.17**	**0.50**	**38.51**	**55.33**	71.93

Table 4: Entity Type Prediction Accuracy on YAGO43kET

Method	MRR		Hits @		
	Raw	Filt	1	3	10
Rescal	0.05	0.08	4.24	8.31	15.31
Rescal-ET	0.04	0.09	4.32	9.62	19.40
TransE	0.09	0.21	12.63	23.24	38.93
TransE-ET	**0.10**	0.18	9.19	19.41	35.58
HolE	0.05	0.16	9.02	17.28	29.25
HolE-ET	0.08	0.18	10.28	20.13	34.90
ETE	**0.10**	**0.23**	**13.73**	**26.28**	**42.18**

4.3 Experimental Results

For our experiments, we used the following evaluation protocol: for each triple (ϵ, p, τ) in the test set, first τ is replaced by τ', and we compute the score of (ϵ, p, τ') for $\forall \tau' \in T$, and then we rank all of these *corrupted* triples by the scores. It is possible that multiple corrupted versions of a triple exist in a dataset because an entity could have multiple entity types. In this case, only one corrupted triple is considered as the correct one for each test triple. To avoid this issue, we remove all of the corrupted triples from the ranking, except for the correct one. In Tables 3 and 4, Raw indicates the cases where we do not remove the additional corrupted triples; those where we only include the correct triple are shown as Filt (filtered).

For our experiments, the prediction accuracy of ETE was compared to three state-of-the-art KG embedding methodsbaselines: Rescal, TransE and HolE on the FB15kET and YAGO43kET datasets. We also extended Rescal, TransE and HolE in the same way as ETE that they are trained with the FB15k and YAGO43k training sets first and then used to train embeddings of entity types on the FB15kET and YAGO43kET training sets. The trained vectors of entity types are used for the test with FB15kET and YAGO43kET validation and test sets. We call the extended baselines Rescal-ET, TransE-ET and HolE-ET, respectively.

From Tables 3 and 4, the effectiveness of our approach can clearly be seen. The extended baselines Rescal-ET, TransE-ET and HolE-ET show better accuracy performance than the original methods in all cases, except for TransE-ET on YAGO43kET. Furthermore, these experimental results show that our method ETE consistently outperforms all the baseline methods.

5 CONCLUSION

We proposed an embedding method ETE for entity type prediction. The main benefits of our approach are: (1) higher prediction accuracy compared to state-of-the-art baseline algorithms, and (2) higher accuracy both for inferring missing entity types as well as for inferring missing entities and relation types. We achieve these benefits while preserving linear scalability with the number of entity types. From the results of our experiments, we show that our method consistently gives higher prediction accuracy than baseline methods on two kinds of real-world knowledge graph.

6 ACKNOWLEDGEMENTS

This material is based upon work supported in whole or in part with funding from the Laboratory for Analytic Sciences (LAS). Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the LAS and/or any agency or entity of the United States Government.

REFERENCES

[1] Antoine Bordes, Sumit Chopra, and Jason Weston. 2014. Question answering with subgraph embeddings. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*. 2787–2795.
[3] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Conference on Artificial Intelligence*.
[4] Kai-Wei Chang, Scott Wen-tau Yih, Bishan Yang, and Chris Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
[5] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.
[6] Yuan Fang and Ming-Wei Chang. 2014. Entity linking on microblogs with spatial and temporal signals. In *Transactions of the Association for Computational Linguistics*, Vol. 2. 259–272.
[7] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *AAAI Conference on Artificial Intelligence*. 2181–2187.
[8] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.
[9] Changsung Moon, Steve Harenberg, John Slankas, and Nagiza Samatova. 2017. Learning Contextual Embeddings for Knowledge Graph Completion. In *Pacific Asia Conference on Information Systems (PACIS)*.
[10] Arvind Neelakantan and Ming-Wei Chang. 2015. Inferring missing entity type instances for knowledge base completion: New dataset and methods. In *the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL)*. 515–525.
[11] Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic Embeddings of Knowledge Graphs. In *AAAI Conference on Artificial Intelligence*. 1955–1961.
[12] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*. 809–816.
[13] Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *Advances in Neural Information Processing Systems*. 926–934.
[14] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *AAAI Conference on Artificial Intelligence*. 1112–1119.
[15] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-16)*. 2965–2971.
[16] Xuchen Yao and Benjamin Van Durme. 2014. Information Extraction over Structured Data: Question Answering with Freebase. In *the Association for Computational Linguistics*. Citeseer, 956–966.