Knowledge Graph Embedding via Metagraph Learning

Chanyoung Chung
chanyoung.chung@kaist.ac.kr
School of Computing, KAIST
Republic of Korea

Joyce Jiyoung Whang*
jjwhang@kaist.ac.kr
School of Computing, KAIST
Republic of Korea

ABSTRACT

Knowledge graph embedding aims to represent entities and relations in a continuous feature space while preserving the structure of a knowledge graph. Most existing knowledge graph embedding methods either focus only on a flat structure of the given knowledge graph or exploit the predefined types of entities to explore an enriched structure. In this paper, we define the metagraph of a knowledge graph by proposing a new affinity metric that measures the structural similarity between entities, and then grouping close entities by hypergraph clustering. Without any prior information about entity types, a set of semantically close entities is successfully merged into one super-entity in our metagraph representation. We propose the metagraph-based pre-training model of knowledge graph embedding where we first learn representations in the metagraph and initialize the entities and relations in the original knowledge graph with the learned representations. Experimental results show that our method is effective in improving the accuracy of state-of-the-art knowledge graph embedding methods.

CCS CONCEPTS

• **Computing methodologies** → **Semantic networks**; • **Information systems** → **Clustering**.

KEYWORDS

knowledge graph; embedding; clustering; metagraph; pre-train

ACM Reference Format:
Chanyoung Chung and Joyce Jiyoung Whang. 2021. Knowledge Graph Embedding via Metagraph Learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3404835.3463072

1 INTRODUCTION

A knowledge graph represents human knowledge as a directed graph where a vertex indicates an entity and a directed edge indicates a relation between two entities. Each fact is represented as a triplet which consists of a head entity, a relation, and a tail entity. Knowledge graph embedding is a representation learning technique which projects entities and relations into a continuous feature space. Once the entities and the relations are represented as feature vectors, we can utilize these feature vectors to solve diverse problems such as link prediction and triplet classification [26].

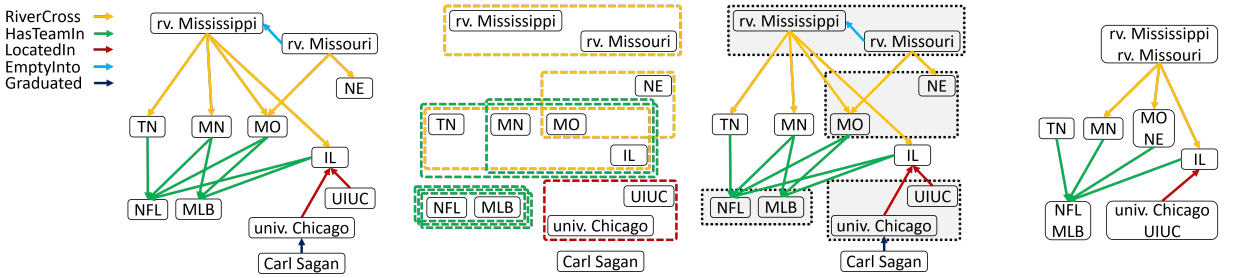
A number of different knowledge graph embedding methods have been proposed over the recent years. However, most existing methods focus only on a flat structure of a knowledge graph [2, 13, 23, 25, 31]. That is, they just consider an individual entity independently, and do not consider a global view or a clusterable structure of a knowledge graph. Some methods have attempted to incorporate entity types into knowledge graph embedding [10, 12, 29]. However, all these methods rely on predefined entity types which are manually and heuristically curated with extra cost [12]. Those entity types are sometimes incomplete, i.e., only a subset of entities has types, as noted in [10] or available only for certain datasets [29].

Inspired by the fact that there exist many semantically close entities in a knowledge graph, we hypothesized that those semantic closeness should be able to be inferred by the structural similarity between entities. For example, if two entities share the same tail entity with the same relation, they might belong to the same semantic category. By taking into account this kind of patterns accumulatively, we can appropriately define the affinity between entities. We formalize this idea by introducing the concept of hypergraph that allows us to connect an arbitrary number of vertices by a hyperedge (Section 3.1). Once a knowledge graph is converted into the corresponding hypergraph, we apply a hypergraph clustering scheme with appropriate normalization to group similar entities (Section 3.2). Once groups of similar entities are detected, we form super-entities by merging entities in the same group. That is, each cluster is now transformed into a super-entity. To define triplets between super-entities, we consider the connections between each pair of head and tail entities from one super-entity to another. We define the metagraph of a knowledge graph to be the graph that consists of the super-entities and the relations between them. The metagraph can be considered as a compact representation of the knowledge graph, which enables us to capture the core structure of the graph (Section 3.3). Figure 1 summarizes the overall procedure.

By learning representations of entities and relations in the metagraph, we can efficiently initialize the entities and relations in the original knowledge graph by simply projecting the learned representations into the corresponding entities and relations. Since all the entities belonging to the same group have the same initial representations by this projection, it is naturally encouraged that the entities in the same group have close representation vectors, which is desirable because those entities are likely to be semantically close (Section 3.4). Experimental results show that our metagraph-based pre-training strategy improves the link prediction performance of state-of-the-art knowledge graph embedding methods.

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SIGIR '21, July 11–15, 2021, Virtual Event, Canada
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3463072



(a) Knowledge Graph (KG) (b) Hypergraph of the KG (c) Grouping Entities (d) Metagraph of the KG

Figure 1: From a Knowledge Graph to its Metagraph. (a): the original knowledge graph. (b): the hypergraph representation of the knowledge graph where each hyperedge connects a set of entities that share the same head or the same tail with the same relation (Section 3.1). (c): similar entities are grouped by hypergraph clustering (Section 3.2). (d): the metagraph of the given knowledge graph where each group is considered to be a super-entity and representative between-group triplets are likely to be preserved while unrepresentative triplets are likely to be dropped (Section 3.3).

2 RELATED WORK

There have been some attempts to identify entity types in knowledge graphs [16, 18, 19, 34]. We note that these existing methods assume that the entity types are predefined [16, 34] or some rich features are provided to infer the semantic categories of entities [18, 19]. On the other hand, our hypergraph-based clustering scheme does not require any prior information about entity types or other external features. Furthermore, our method can create different numbers of clusters depending on a desired granularity.

The concept of metagraph has been investigated in the context of heterogeneous network embedding [7, 8, 32] where different node types are already defined in the dataset. To incorporate heterogeneous node types into a network embedding model, various types of metagraphs [8, 32] and metapaths [7] have been proposed. Different from these methods [7, 8, 32], our method automatically constructs the metagraph based on the structural similarity between entities. To the best of our knowledge, our work is the first study that introduces the concept of metagraph to knowledge graphs.

In practice, knowledge graph embedding models are usually initialized by random Gaussian distributions [11, 24] or Glorot initialization [9] because existing pre-training models require a rich language model [33] such as BERT [6] or extra lexical descriptions [14] or word embedding [22]. On the other hand, our metagraph-based initialization does not require this extra information.

Multi-level graph embedding framework has been recently studied in the context of graph embedding [4, 5], albeit has not been considered for knowledge graphs. While both HARP [4] and GraphZoom [5] deal with standard homogeneous graphs, our method targets knowledge graphs which make the coarsening step much more challenging due to diverse types of entities and relations. We believe that we can easily extend our method to a multi-level framework by recursively applying the clustering operation, which might allow us to further improve the performance of our method.

3 META-KGE: METAGRAPH-BASED KNOWLEDGE GRAPH EMBEDDING

We propose the metagraph-based knowledge graph embedding framework which consists of four steps, each of which is described in the following subsection.

Algorithm 1: Converting a knowledge graph into a hypergraph where similar entities are connected via a hyperedge.

Input: a knowledge graph $G = (\mathcal{V}, \mathcal{R}, \mathcal{E})$
Output: a hypergraph $G_H = (\mathcal{V}, \mathcal{E}_H)$ where $\mathcal{E}_H$ is a multiset

Initialize $\mathcal{E}_H = \emptyset$.
for $v_i \in \mathcal{V}$ **do**
  $\mathcal{R}_i = \{r | (h, r, v_i) \in \mathcal{E} \vee (v_i, r, t) \in \mathcal{E}, h \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{V}\}$.
  for $r_j \in \mathcal{R}_i$ **do**
    $\mathcal{S}_H = \{h \mid (h, r_j, v_i) \in \mathcal{E}, h \in \mathcal{V}\}$.
    $\mathcal{S}_T = \{t \mid (v_i, r_j, t) \in \mathcal{E}, t \in \mathcal{V}\}$.
    if $|\mathcal{S}_H| > 1$ **then**
      $\mathcal{E}_H = \mathcal{E}_H \cup \{\mathcal{S}_H\}$.
    end if
    if $|\mathcal{S}_T| > 1$ **then**
      $\mathcal{E}_H = \mathcal{E}_H \cup \{\mathcal{S}_T\}$.
    end if
  end for
end for

3.1 Defining Affinity between Entities by Hypergraph Representation

Entities in a knowledge graph can be grouped such that semantically close entities are assigned to the same group. Since this semantic closeness is reflected on the structure of the graph, we can define affinity between entities based on the structural similarity. A knowledge graph G is defined by three sets $\mathcal{V}$, $\mathcal{R}$, and $\mathcal{E}$ where $\mathcal{V}$ is a set of entities, $\mathcal{R}$ is a set of relations, and $\mathcal{E}$ is a set of triplets. Assume that there are n different entities and n' different relations. Formally, a knowledge graph can be represented by $G = (\mathcal{V}, \mathcal{R}, \mathcal{E})$ where $\mathcal{V} = \{v_1, v_2, \cdots, v_n\}$, $\mathcal{R} = \{r_1, r_2, \cdots, r_{n'}\}$, and $\mathcal{E} = \{(h, r, t) \mid h \in \mathcal{V}, r \in \mathcal{R}, t \in \mathcal{V}\}$. A triplet (h, r, t) consists of a head entity h, a relation r, and a tail entity t. By introducing the concept of hyperedge which can connect more than two nodes, we connect a set of entities via a hyperedge if they share the same head entity with the same relation or if they share the same tail entity with the same relation. For example, in Figure 1(b), the yellow hyperedge that includes 'TN', 'MN', 'MO', 'IL' is created because these four entities all share the head entity 'rv. Mississippi' with the same relation 'RiverCross'. By Algorithm 1, we convert a knowledge graph into a hypergraph so that similar entities are connected via a hyperedge. Let $\boldsymbol{H}$ indicate the incidence matrix of the

Table 1: Top 5 most similar entities to the target entity according to our affinity metric defined in (1) on NELL-995

Target entity	Top 5 most similar entities to the target entity (ties are all included)
emotion_thankfulness	emotion_graditude, emotion_admiration, emotion_happiness, emotion_joy, emotion_deep_love, emotion_jealousy, emotion_thanks
software_microsoft_word	software_internet_explorer, software_microsoft_frontpage, software_microsoft_powerpoint, software_notepad, software_autocad
sport_american_football	sport_ski, sport_scout, sport_skiing, sport_golf, sport_judo
university_harvard	university_harvard_university, university_harvard_law, school_oxford, university_harvard_law_school, university_john_f__kennedy_school
furniture_queen_bed	furniture_queen, furniture_king_beds, furniture_king_size_beds, furniture_twin_beds, furniture_queen_size_beds

resulting hypergraph G_H, and assume that there are m hyperedges in G_H. Each entry of $\boldsymbol{H} \in \{0,1\}^{n\times m}$ is defined to be $h_{ij} = 1$ if an entity v_i is included in the j-th hyperedge, and $h_{ij} = 0$ otherwise. Also, let us define the degree diagonal matrix $\boldsymbol{D} \in \mathbb{R}_+^{m\times m}$ where d_{ii} is the number of entities included in the i-th hyperedge. We define the affinity matrix $\boldsymbol{A}$ to be $\boldsymbol{A} \coloneqq \boldsymbol{H}\boldsymbol{D}^{-2}\boldsymbol{H}^T$ which encodes the pairwise similarity where each entry a_{ij} of $\boldsymbol{A}$ indicates the affinity between v_i and v_j, i.e., $a_{ij} \coloneqq \sum_{l\in\mathcal{L}} 1/{d_l}^2$ where $\mathcal{L}$ indicates the set of hyperedges which contain v_i and v_j simultaneously, and d_l indicates the number of entities in the hyperedge l. The intuition of this definition is that we assign a high affinity score between two entities if they are connected by multiple hyperedges (i.e., a large $\mathcal{L}$) or they appear together in rare patterns, i.e., they are connected by a hyperedge that contains only a few entities (i.e., a small d_l).[1]

3.2 Grouping Entities by Optimizing the Hypergraph Normalized Cut

Once we convert a knowledge graph into the hypergraph G_H, optimizing the hypergraph normalized cut [27, 35] on G_H is mathematically equivalent to optimizing the standard normalized cut [21] on $\boldsymbol{A}$ if we assume that each hyperedge has a weight of $1/d_l$. By considering the entity-level hypergraph normalized cut, we further refine the pairwise similarity as follows:

$$\hat{a}_{ij} = \frac{a_{ij}}{\sum_k a_{ik}} + \frac{a_{ij}}{\sum_k a_{kj}} \tag{1}$$

which indicates that we normalize the affinity score between two entities by their affinity scores to their incident entities. Table 1 shows some examples of similar entities according to our affinity metric on NELL-995 dataset [30] where we randomly choose target entities and list up five most similar entities to the target entities including all the ties. We see that the entities in the lists are semantically quite similar to the target entities.

Using (1) as the similarity metric, we apply an agglomerative hierarchical clustering with the average linkage strategy [17, 20] to group similar entities together. We cluster the given graph into $\lfloor np \rfloor$ clusters where n is the number of entities and $0 < p < 1$. Figure 1(c) shows an example of our clustering result when $p = 0.7$. Each grey box indicates a cluster, and we see that those clusters are reasonably formed by merging similar entities.

3.3 Metagraph Construction

Once similar entities are grouped together, each cluster is considered to be a super-entity. Even though an entity is not merged into any cluster after the clustering step, we also call this singleton as a cluster or a super-entity which only includes itself.

[1]If a hyperedge contains d_l entities, the hyperedge introduces ${d_l}^2$ non-zero entries in $\boldsymbol{H}\boldsymbol{H}^T$. To let each hyperedge introduce the same affinity score in total, we normalize the introduced non-zero entries by ${d_l}^2$.

Table 2: Datasets and Metagraph Information

		Train set			Validation set			Test set		
		$\lvert V\rvert$	$\lvert R\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert R\rvert$	$\lvert E\rvert$	$\lvert V\rvert$	$\lvert R\rvert$	$\lvert E\rvert$
FB15K	Original KG	14,951	1,345	483,142	13,292	916	50,000	13,584	961	59,071
	Metagraph	10,387	1,142	224,824	8,377	686	21,778	8,611	703	25,433
NELL-995	Original KG	74,432	200	149,678	765	12	543	3,747	12	3,992
	Metagraph	48,693	200	67,568	382	12	261	2,387	12	2,118
WN18	Original KG	40,943	18	141,442	7,802	18	5,000	7,845	18	5,000
	Metagraph	27,250	18	63,423	3,695	18	2,234	3,689	18	2,227

Let C_i indicate the i-th super-entity where $i = 1, \cdots, \lfloor np \rfloor$. Let $G_M = (\mathcal{V}_M, \mathcal{R}_M, \mathcal{E}_M)$ be the metagraph of a given knowledge graph. Each super-entity C_i is considered as an individual entity in G_M. Given super-entities C_i and C_j, we add a triplet (C_i, r, C_j) to $\mathcal{E}_M$ with the probability of

$$\frac{|\{(h,r,t)|h \in C_i \wedge t \in C_j \wedge r \in \mathcal{R}\}|}{|C_i||C_j|} \tag{2}$$

which indicates that a triplet (C_i, r, C_j) is likely to be added to $\mathcal{E}_M$ if entities in C_i have many connections to those in C_j with the relation r. Conversely, if there are few connections from C_i to C_j with the relation r, (C_i, r, C_j) is likely to be dropped in $\mathcal{E}_M$. As a result, we encourage representative triplets to be preserved while unrepresentative triplets to be dropped in G_M. Also notice that we drop within-cluster triplets and only consider between-cluster triplets when we create G_M. Figure 1(d) shows an example of this process. The relation 'EmptyInto' is omitted because it only appears within a group. The relation from 'Carl Sagan' to 'univ. Chicago' is probabilistically dropped with the probability of 1/2.

3.4 Pre-training and Fine-tuning of a Knowledge Graph Embedding Model

Since the metagraph simplifies the structure of the original knowledge graph, we can efficiently run a knowledge graph embedding method on the metagraph. Once we learn the representations of super-entities and the relations on the metagraph, we initialize the corresponding entities and relations in the knowledge graph with the learned representations. In this process, entities in the same super-entity are initialized with the same representations. The omitted entities and relations in the metagraph are randomly initialized in the original knowledge graph. Once we initialize the representations in the original knowledge graph, we further fine-tune the representations of individual entities and relations to learn a finer structure of the knowledge graph.

4 EXPERIMENTAL RESULTS

We show experimental results of our method on the link prediction task using three real-world datasets: FB15K [1], NELL-995 [3, 30], and WN18 [15] shown in Table 2. We notice that NELL-995 is much sparser than the other two datasets. As described in Section 3.2, we cluster a given knowledge graph into $\lfloor np \rfloor$ clusters to create a

Table 3: Hyperparameters of each method

	FB15K	NELL-995	WN18
TransE	$\alpha = 5, \gamma = 5$	$\alpha = 5, \gamma = 5$	$\alpha = 0.1, \gamma = 10$
TransE@mgraph	$\alpha = 5, \gamma = 5$	$\alpha = 5, \gamma = 10$	$\alpha = 0.1, \gamma = 10$
meta-TransE	$\alpha = 5, \gamma = 2$	$\alpha = 1, \gamma = 5$	$\alpha = 0.1, \gamma = 10$
DistMult	$\alpha = 0.01, \beta = 0.1$	$\alpha = 0.1, \beta = 0.1$	$\alpha = 0.5, \beta = 2$
DistMult@mgraph	$\alpha = 0.01, \beta = 0.1$	$\alpha = 0.1, \beta = 0.1$	$\alpha = 1, \beta = 5$
meta-DistMult	$\alpha = 0.1, \beta = 0.1$	$\alpha = 0.1, \beta = 0.1$	$\alpha = 1, \beta = 1$
RotatE	$\alpha = 5 \cdot 10^{-5}, \gamma = 12, s = 1$	$\alpha = 10^{-4}, \gamma = 12, s = 2$	$\alpha = 10^{-4}, \gamma = 3, s = 0.5$
RotatE@mgraph	$\alpha = 10^{-4}, \gamma = 9, s = 0.5$	$\alpha = 10^{-4}, \gamma = 3, s = 1$	$\alpha = 10^{-4}, \gamma = 3, s = 0.1$
meta-RotatE	$\alpha = 5 \cdot 10^{-5}, \gamma = 12, s = 1$	$\alpha = 10^{-4}, \gamma = 12, s = 2$	$\alpha = 10^{-4}, \gamma = 3, s = 0.5$

metagraph. We set $p = 0.7$ for FB15K and WN18 whereas $p = 0.85$ for NELL-995 because these p values allow the size of the metagraph to be about half of the original knowledge graph in terms of the number of triplets in the train set. We perform clustering only using the train set. Table 2 shows the information about the metagraphs. When we generate the validation and test sets for a metagraph, we exclude triplets that already appear in the train set. For the test set, we also exclude triplets that appear in the validation set.

In our metagraph-based knowledge graph embedding method, we can apply any knowledge graph embedding models to learn the representations on the metagraph as well as on the original knowledge graph. We use three knowledge graph embedding methods: TransE [2], DistMult [31], and RotatE [23] which are implemented in OpenKE [11] where TransE [2] and DistMult [31] are initialized by Glorot initialization [9] and RotatE [23] is initialized by the uniform distribution. We compare the performance of these models with our metagraph-based pre-training methods which are denoted by meta-TransE, meta-DistMult, and meta-RotatE, respectively. Table 3 shows the best hyperparameters with respect to the validation set where *model*@mgraph indicates the hyperparameters of the corresponding model on the metagraph. In Table 3, α indicates a learning rate, β indicates a regularization rate, γ indicates a margin, and s indicates the temperature of sampling.

We compare the performance of the methods on the link prediction task. By following the conventional setting, we report the 'filtered' scores. Also, we use the standard evaluation metrics: Mean Rank (MR), Mean Reciprocal Rank (MRR), and Hit@10. Details about these settings and metrics can be found in [26]. A lower MR, a higher MRR, and a higher Hit@10 score indicates a better result. For each metric, we compute the gain as follows:

$$\text{Gain}_{\text{metric}} = sign(\text{metric}) \frac{(Score_{\text{model}} - Score_{\text{meta-model}})}{Score_{\text{model}}} \times 100\%$$

where $sign(\text{metric})$ is positive for MR and negative for MRR and Hit@10, and $Score_{\text{model}}$ is the score of the original knowledge graph embedding method whereas $Score_{\text{meta-model}}$ is the score of our method. A positive gain indicates our method performs better than the baseline method. Since each of these three metrics focuses on a different perspective of the results, a method can get a good score in terms of one metric but get a bad score in terms of another metric. To get a holistic interpretation of the experiments, we compute the total gain by summing all the gains of the three metrics. If the total gain is positive and large, it indicates our method substantially outperforms the baseline method. Table 4 shows the results where we repeat experiments of each method for five times and report the average scores. Using three different metrics, three different models, and three different datasets, we see that our method (meta-*model*) shows a positive gain for most of the cases. Even though our method gets some negative gains on some datasets with some models in terms of some metrics, the total gains are always positive. Therefore, our experimental results show that our metagraph-based pre-training is effective in improving the performance of the knowledge graph embedding methods.

Table 4: Link Prediction Results. A positive gain indicates our method (meta-*model*) outperforms the baseline.

		MR (↓)	MRR (↑)	Hit@10 (↑)	Total Gain (↑)
FB15K	TransE	89.0	**0.596**	0.733	
	meta-TransE	**75.0**	0.551	**0.798**	
	Gain (↑)	**15.8%**	-7.5%	**8.8%**	**17.1%**
	DistMult	**106.4**	0.414	0.644	
	meta-DistMult	143.7	**0.541**	**0.786**	
	Gain (↑)	-35.0%	**30.8%**	**21.9%**	**17.7%**
	RotatE	34.4	**0.691**	0.869	
	meta-RotatE	**33.6**	0.690	**0.871**	
	Gain (↑)	**2.1%**	-0.1%	**0.2%**	**2.2%**
NELL-995	TransE	7202.4	0.278	**0.477**	
	meta-TransE	**6507.5**	**0.287**	0.434	
	Gain (↑)	**9.6%**	**3.3%**	-9.1%	**3.8%**
	DistMult	10312.7	**0.298**	0.388	
	meta-DistMult	**8046.0**	0.288	**0.397**	
	Gain (↑)	**22.0%**	-3.6%	**2.3%**	**20.7%**
	RotatE	9243.9	0.350	0.428	
	meta-RotatE	**8618.7**	**0.352**	**0.435**	
	Gain (↑)	**6.8%**	**0.7%**	**1.8%**	**9.3%**
WN18	TransE	210.4	0.521	0.943	
	meta-TransE	**185.9**	**0.535**	**0.949**	
	Gain (↑)	**11.6%**	**2.7%**	**0.7%**	**15.0%**
	DistMult	301.1	0.320	0.550	
	meta-DistMult	**289.1**	**0.463**	**0.732**	
	Gain (↑)	**4.0%**	**44.7%**	**33.2%**	**81.9%**
	RotatE	76.681	**0.661**	0.882	
	meta-RotatE	**73.718**	0.655	**0.884**	
	Gain (↑)	**3.9%**	-0.9%	**0.2%**	**3.2%**

5 CONCLUSION & FUTURE WORK

We propose the metagraph-based pre-training method for knowledge graph embedding. By proposing a new affinity metric that measures the structural similarity between entities, we cluster entities without any predefined entity types. We define the metagraph of a knowledge graph by extracting representative between-cluster triplets. The pre-trained representations on the metagraph effectively initialize the entities and relations in the original knowledge graph leading to improving the link prediction performances. Our affinity measures can be plugged into any kind of clustering process besides the hierarchical agglomerative clustering. We plan to extend our method to generate overlapping clusters [28] of entities so that we can incorporate a more flexible and natural clustering structure into knowledge graph embedding models.

ACKNOWLEDGMENTS

This work was supported by NRF of Korea (2019R1C1C1008956, 2018R1A5A1059921).

REFERENCES

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. 1247–1250.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-Relational Data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. 2787–2795.

[3] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. 2010. Toward an Architecture for Never-Ending Language Learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. 1306–1313.

[4] Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. 2018. HARP: Hierarchical Representation Learning for Networks. In *Proceedings of the 32rd AAAI Conference on Artificial Intelligence, the 30th innovative Applications of Artificial Intelligence and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence*. 2127–2134.

[5] Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. 2020. GraphZoom: A Multi-level Spectral Approach for Accurate and Scalable Graph Embedding. In *Proceedings of the 8th International Conference on Learning Representations*.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 4171–4186.

[7] Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. 2017. Metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 135–144.

[8] Yang Fang, Xiang Zhao, Peixin Huang, Weidong Xiao, and Maarten de Rijke. 2019. M-HIN: Complex Embeddings for Heterogeneous Information Networks via Metagraphs. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 913–916.

[9] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 249–256.

[10] Shu Guo, Quan Wang, Bin Wang, Lihong Wang, and Li Guo. 2015. Semantically Smooth Knowledge Graph Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computatinal Linguistics and the 7th International Joint Conference on Natural Language Processing - Volume 1*. 84–94.

[11] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li. 2018. OpenKE: An Open Toolkit for Knowledge Embedding. 139–144.

[12] Denis Krompaβ, Stephan Baier, and Volker Tresp. 2015. Type-Constrained Representation Learning in Knowledge Graphs. In *Proceedings of the 14th International Conference on The Semantic Web*. 640–655.

[13] Hanxiao Liu, Yuexin Wu, and Yiming Yang. 2017. Analogical Inference for Multi-Relational Embeddings. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. 2168—-2178.

[14] Teng Long, Ryan Lowe, Jackie Chi Kit Cheung, and Doina Precup. 2016. Leveraging Lexical Resources for Learning Entity Embeddings in Multi-Relational Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 112–117.

[15] George A. Miller. 1995. WordNet: A Lexical Database for English. *Commun. ACM* 38, 11 (1995), 39–41.

[16] Changsung Moon, Paul Jones, and Nagiza F. Samatova. 2017. Learning Entity Type Embeddings for Knowledge Graph Completion. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. 2215—-2218.

[17] Benjamin Moseley and Joshua R. Wang. 2017. Approximation Bounds for Hierarchical Clustering: Average Linkage, Bisecting K-Means, and Local Search. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 3097–3106.

[18] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 1135—-1145.

[19] Andrea Giovanni Nuzzolese, Aldo Gangemi, Valentina Presutti, Francesco Draicchio, Alberto Musetti, and Paolo Ciancarini. 2013. Tipalo: A Tool for Automatic Typing of DBpedia Entities. In *The Semantic Web: ESWC 2013 Satellite Events*. 253–257.

[20] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830.

[21] Jianbo Shi and Jitendra Malik. 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22, 8 (2000), 888–905.

[22] Richard Socher, Danqi Chen, Christopher D. Manning, and Andrew Y. Ng. 2013. Reasoning with Neural Tensor Networks for Knowledge Base Completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. 926–934.

[23] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *Proceedings of the 7th International Conference on Learning Representations*.

[24] Théo Trouillon, Christopher R Dance, Éric Gaussier, Johannes Welbl, Sebastian Riedel, and Guillaume Bouchard. 2017. Knowledge graph completion via complex tensor factorization. *Journal of Machine Learning Research (JMLR)* 18, 130 (2017), 1–38.

[25] Chun-Chih Wang and Pu-Jen Cheng. 2018. Translating Representations of Knowledge Graphs with Neighbors. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 917–920.

[26] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743.

[27] Joyce Jiyoung Whang, Rundong Du, Sangwon Jung, Geon Lee, Barry Drake, Qingqing Liu, Seonggoo Kang, and Haesun Park. 2020. MEGA: Multi-View Semi-Supervised Clustering of Hypergraphs. *Proceedings of the VLDB Endowment* 13, 5 (2020), 698–711.

[28] Joyce Jiyoung Whang, Yangyang Hou, David F. Gleich, and Inderjit S. Dhillon. 2019. Non-exhaustive, Overlapping Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 11 (2019), 2644–2659.

[29] Ruobing Xie, Zhiyuan Liu, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Hierarchical Types. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*. 2965–2971.

[30] Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. DeepPath: A Reinforcement Learning Method for Knowledge Graph Reasoning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 564–573.

[31] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the 3rd International Conference on Learning Representations*.

[32] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. MetaGraph2Vec: Complex Semantic Path Augmented Heterogeneous Network Embedding. In *PAKDD 2018: Advances in Knowledge Discovery and Data Mining*. 196–208.

[33] Zhiyuan Zhang, Xiaoqian Liu, Yi Zhang, Qi Su, Xu Sun, and Bin He. 2020. Pretrain-KGE: Learning Knowledge Representation from Pretrained Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 259–266.

[34] Yu Zhao, Anxiang Zhang, Ruobing Xie, Kang Liu, and Xiaojie Wang. 2020. Connecting Embeddings for Knowledge Graph Entity Typing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 6419–6428.

[35] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with Hypergraphs: Clustering, Classification, and Embedding. In *Proceedings of 19th Advances in Neural Information Processing Systems*. 1601–1608.