CaRe: Open Knowledge Graph Embeddings

Swapnil Gupta
Indian Institute of Science
Bangalore, India
swapnilgupta.229@gmail.com

Sreyash Kenkre
Walmart Labs
Bangalore, India
sreyash@gmail.com

Partha Talukdar
Indian Institute of Science
Bangalore, India
ppt@iisc.ac.in

Abstract

Open Information Extraction (OpenIE) methods are effective at extracting (noun phrase, relation phrase, noun phrase) triples from text, e.g., *(Barack Obama, took birth in, Honolulu)*. Organization of such triples in the form of a graph with noun phrases (NPs) as nodes and relation phrases (RPs) as edges results in the construction of Open Knowledge Graphs (OpenKGs). In order to use such OpenKGs in downstream tasks, it is often desirable to learn embeddings of the NPs and RPs present in the graph. Even though several Knowledge Graph (KG) embedding methods have been recently proposed, all of those methods have targeted *Ontological KGs*, as opposed to OpenKGs. Straightforward application of existing Ontological KG embedding methods to OpenKGs is challenging, as unlike Ontological KGs, OpenKGs are not *canonicalized*, i.e., a real-world entity may be represented using multiple nodes in the OpenKG, with each node corresponding to a different NP referring to the entity. For example, nodes with labels *Barack Obama*, *Obama*, and *President Obama* may refer to the same real-world entity *Barack Obama*. Even though canonicalization of OpenKGs has received some attention lately, output of such methods has not been used to improve OpenKG embeddings. We fill this gap in the paper and propose Canonicalization-infused Representations (CaRe) for OpenKGs. Through extensive experiments, we observe that CaRe enables existing models to adapt to the challenges in OpenKGs and achieve substantial improvements for the link prediction task.

1 Introduction

Open Information Extraction (OpenIE) methods such as ReVerb (Fader et al., 2011), OLLIE (Mausam et al., 2012), BONIE (Saha et al., 2017) and CALMIE (Saha and Mausam, 2018) can automatically extract *(noun phrase, relation phrase, noun phrase)* triples from text, e.g., *(Barack Obama, took birth in, Honolulu), (Michelle Obama, wife of, Barack), etc*. An *Open Knowledge Graph* (*OpenKG*) can be constructed out of such triples by representing noun phrases (NPs) as nodes and relation phrases (RPs) as edges connecting them. Example of an OpenKG is shown in Figure 1. OpenKGs do not require pre-specified ontology, making them highly adaptable. In order to use OpenKGs in downstream tasks such as Question Answering, Document Classification, etc., it is often necessary to learn embeddings of NPs and RPs present as nodes and edges in an OpenKG.

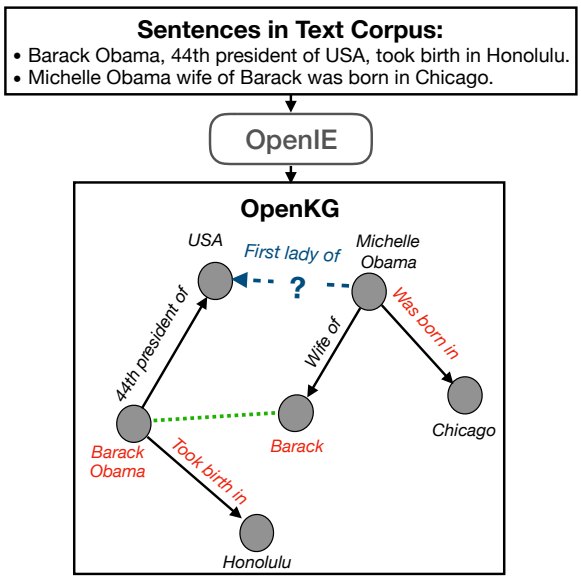


Figure 1: **Challenges in OpenKG**. KG embedding methods are effective in downstream tasks such as Link Prediction. Existing KG embedding methods are ineffective on OpenKGs as they are not canonicalized. CaRe can effectively utilize canonicalization information to learn better embeddings in OpenKGs. Green dotted line represents the missing NP canonicalization information. Please see Section 1 for more details.

Even though Knowledge Graph (KG) embed-

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 378–388, Hong Kong, China, November 3–7, 2019.

ding has been an active area of research (Bordes et al., 2013; Yang et al., 2014), all the proposed KG embedding methods have focused on embedding *Ontological KGs*, such as WikiData (Vrandečić and Krötzsch, 2014), DBpedia (Auer et al., 2007), YAGO (Suchanek et al., 2007), NELL (Mitchell et al., 2018), and Freebase (Bollacker et al., 2008). Existing KG embedding models train representation of each node and edge label based on the context of triples they are present in. Doing this is suitable for ontological KGs as they are *canonicalized*. However, in OpenKGs, the same latent entity may be represented in different nodes labelled with different NPs. Similarly, the same latent relation can be represented with different RPs. For example, in Figure 1, *Barack Obama* the entity is represented using two NP nodes: *Barack Obama* and *Barack*. Similarly, the two RPs – *took birth in* and *was born in* – refer to the same underlying relation. Hence, the paradigm of learning embeddings for each node and edge label only from the context of the triples they appear in is ineffective for OpenKGs.

A possible solution is to canonicalize the OpenKGs. This involves identifying NPs and RPs that refer to the same entity and relation, and assigning them unique IDs. Nodes in the OpenKG having the same ID are *merged*, leading to a clean and canonicalized graph. Recent works that automatically canonicalize OpenKGs, including (Galárraga et al., 2014) and CESI (Vashishth et al., 2018), pose canonicalization as a clustering task of the NPs and RPs. However, due to automatic generation, the output clusters often contain some incorrectly canonicalized elements. Thus, directly merging nodes or relations with the same IDs would result in the propagation of errors in the canonicalization step to down-stream tasks.

Our premise is that the output of these automatic canonicalization models can be utilized to improve OpenKG embedding. Instead of explicitly merging nodes with common IDs, KG embedding models can be designed to judiciously account for mistakes during the canonicalization step. Towards establishing this premise, we propose a flexible OpenKG embedding approach to integrate and utilize the output of a canonicalization model in an error-conscious manner. Our contributions are as follows:

- We draw attention to an important but relatively unexplored problem of learning representations for OpenKGs.
- We propose Canonicalization-infused Representations (CaRe) for Open KGs - a novel approach to enrich OpenKG embedding models with the output of a canonicalization model. To the best of our knowledge, this is the first model of its kind.
- Through extensive experiments on real-world datasets, we establish CaRe's effectiveness in embedding OpenKGs.

CaRe source code is available at `https://github.com/malllabiisc/CaRE`.

2 Related Work

OpenKG extraction and canonicalization: Multiple OpenIE systems have been developed over the years. These include TextRunner (Yates et al., 2007), (Angeli et al., 2015), ReVerb (Fader et al., 2011), OLLIE (Mausam et al., 2012), SRLIE (Christensen et al., 2011), RelNoun (Pal and Mausam, 2016) and ClauseIE (Del Corro and Gemulla, 2013). A recent survey on the progress in OpenIE systems is presented in (Mausam, 2016).

To perform automatic canonicalization of OpenKGs, (Galárraga et al., 2014) first cluster NPs over manually defined feature spaces. These are then passed to AMIE (Galárraga et al., 2013) for RP clustering. Whereas, CESI (Vashishth et al., 2018) jointly learns vector representations of NPs and RPs by infusing side information in KG embedding models which are then used to cluster NPs and RPs. In this work, we use CESI to generate canonicalization clusters for our datasets.

KG Embedding Methods: KG embedding methods aim to learn low dimensional vector representations for the nodes and edge labels encoding the graph topology. All these methods train the embeddings by optimizing a link prediction based objective. They primarily differ in their way of mathematically modelling the likelihood of a triple being true. Translation-based hypothesis which regards relation vector for any *(subject, relation, object)* triple as a translation from the subject vector to the object vector is used in methods like TransE (Bordes et al., 2013) and TransH (Wang et al., 2014). Semantic matching models, such as DistMult (Yang et al., 2014), ComplEx (Trouillon et al., 2016) and HolE (Nickel et al., 2016), use

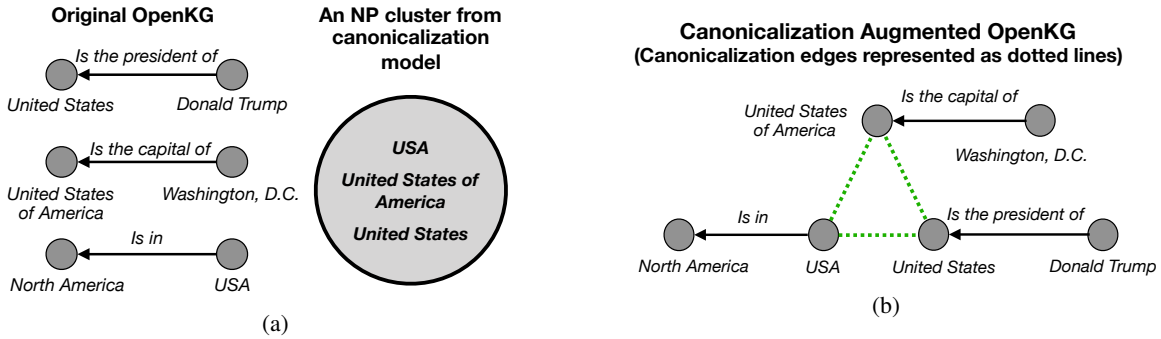


Figure 2: **CaRe Step 1:** First CaRe augments the original OpenKG with the output of a canonicalization model. (a) OpenKG and NP clusters from a canonicalization model. (b) Augmented OpenKG by adding undirected edges between canonical NPs (represented as dotted lines). Please refer to Section 4.2 for more details.

similarity-based scoring functions to measure the likelihood of a fact. Multi-Layer neural network models, ConvE (Dettmers et al., 2018) and R-GCN (Schlichtkrull et al., 2018) have shown better expressive strength. R-GCN adapts graph convolutional network (GCN) (Kipf and Welling, 2016) to a relational graph proposing an auto-encoder model for the link prediction task. Implicit in all these approaches is the assumption that each node in the graph is a different entity, and distinct edge labels refer to distinct relations. This assumption does not hold in OpenKGs. Existing KG embeddings are thus unsuitable for a task like link prediction on OpenKGs. CaRe addresses these limitations by infusing the output of a canonicalization model with the KG embedding models.

3 Background

In this section, we present a quick overview of a few basic methods which are useful for understanding the rest of the paper, especially the experiments section. We first start with the notations.

Notations: OpenKG is denoted as $\mathrm{G} = (\mathrm{N}, \mathrm{R}, \mathrm{T}^{+})$, where N and R are the set of NPs and RPs, respectively, and $\mathrm{T}^{+} = \{(s, r, o) | s \in \mathrm{N}, r \in \mathrm{R}, o \in \mathrm{N}\}$ is the set of observed triples. Here s, o are the subject and object phrase, respectively.

TransE: Given a triple (s, r, o), consider $e_s, r_r, e_o \in \mathbb{R}^d$ as the d-dimensional vector representations of subject (s), relation (r) and object (o) respectively. TransE follows a translation based triple scoring function $\psi(.)$:

$$\psi(s, r, o) = -\left\| e_s + r_r - e_o \right\|_p .$$

Here, $\|.\|$ denotes norm and p is either 1 or 2. For training parameters, TransE uses margin-based pairwise ranking loss with negative sampling.

ConvE: ConvE (Dettmers et al., 2018) first reshapes e_s and r_r to $\bar{e_s}$ and $\bar{r_r}$, respectively, and passes them through a 2D convolution layer to compute the score corresponding to a triple:

$$\psi(s, r, o) = f(vec(f([\bar{e_s}; \bar{r_r}] * w))W)e_o.$$

Here, $*$ and w denote the convolution operator and convolution filters, f represents an activation function and W, the weights of the final linear layer. For training, ConvE uses binary cross-entropy loss with correct samples considered as positive instances while negative instances are generated through negative sampling.

Graph Neural Networks (GNN): GNNs were introduced in (Gori et al., 2005) and (Scarselli et al., 2009) as a generalization of recursive neural networks. Later, the generalization of CNN to graph-structured data, popularly known as Graph Convolution Network (GCN), was proposed in (Bruna et al., 2014). A first-order formulation of GCN as proposed in (Kipf and Welling, 2016). Under GCN formulation, the representation of a node n after l^{th} layer is defined as follows.

$$e_n^{l+1} = f\left(\sum_{i \in \mathcal{N}(n)} \left(W^l e_i^l + b^l\right)\right), \forall n \in \mathrm{N} \quad (1)$$

Here, W^l, b^l are layer parameters, $\mathcal{N}(n)$ corresponds to the immediate neighborhood of n and f denotes an activation function.

GAT: GAT uses the attention mechanism to determine the weights of a node's neighbors (Veličković et al., 2018). Further, it uses multi-head attentions and defines the representation of

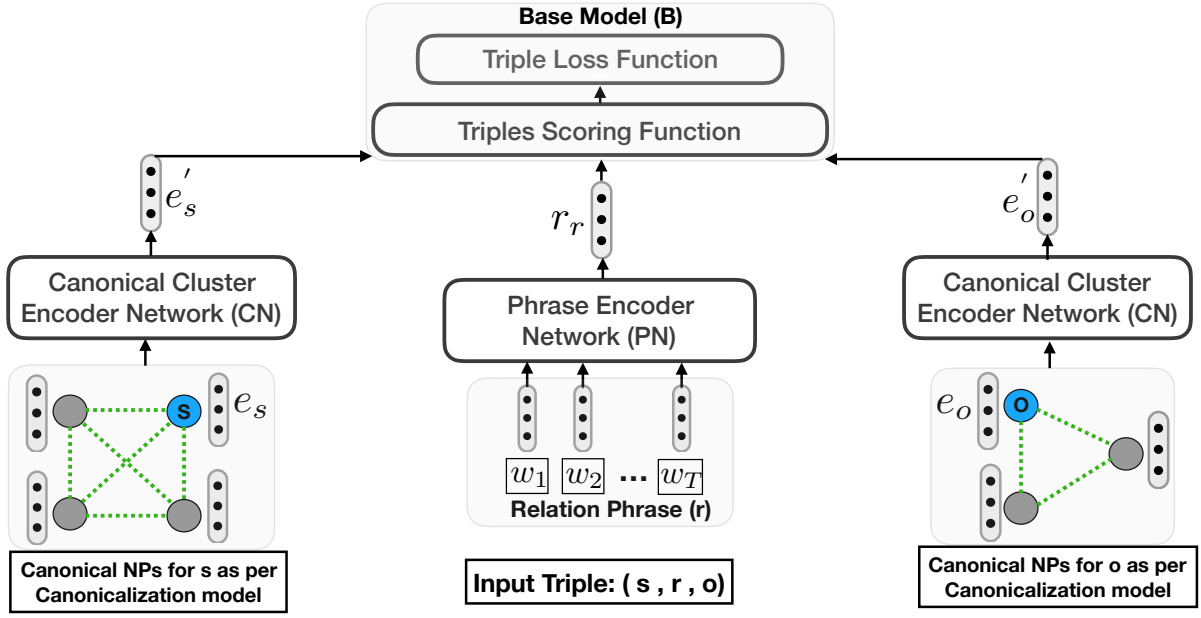


Figure 3: **CaRe Step 2:** In this step, CaRe learns KG embeddings from the augmented OpenKG (Figure 2). Base model can be any existing KG embedding model (e.g., TransE, ConvE). RP embeddings are parameterized by encoding vector representations of the word sequence composing them. This enables CaRe to capture semantic similarity of RPs. Embeddings of NPs are made more context rich by updating them with the represenations of canonical NPs (connected with dotted lines). Please see Section 3 and 4 for details.

n^{th} the node as follows

$$e_n^{l+1} = ||_{k=1}^{K} f\left(\sum_{i \in \mathcal{N}(n)} \left(\alpha_k(e_n^l, e_i^l) W_k^l e_i^l\right)\right). \quad (2)$$

Here, $\alpha(.)$ denotes attention weights and $||$ denotes concatenation.

4 CaRe: Proposed Method

4.1 Overview

CaRe consists of two steps. We first give an overview of the two steps below and then provide a detailed description of each step.

- **Step 1**: In this step, the given OpenKG is augmented by adding edges from a canonicalization model. This step is outlined in Figure 2 and described in detail in Section 4.2.
- **Step 2**: In this step, embeddings of nodes and edges present in the augmented OpenKG obtained in Step 1 are learned. This step is outlined in Figure 3. The system consists of three components: Base Model, Phrase Encoder Network, and Canonical Cluster Encoder Network. We describe these components in Section 4.3, Section 4.4, and Section 4.5, respectively.

CaRe architecture and its components are depicted in Figure 3.

4.2 Step 1: Canonicalization Augmented OpenKG

As discussed in Section 1, based on the automatic clusters generated by a canonicalization model, merging NPs with a single ID can lead to the propagation of errors in the graph. Hence, we propose a soft integration scheme by adding an unlabeled and undirected edge between any two NPs which are canonical as per the canonicalization model as shown in Figure 2. More formally, suppose n_1, n_2 are two NPs that have been identified to be the same in the canonicalization step. We add an undirected and unlabelled edge (n_1, n_2) between the nodes n_1 and n_2, and call them canonicalization edges. If the canonicalization step produces a confidence score for n_1 and n_2 being the same, then this score can be incorporated as a weight on the canonicalization edge. If no such score is produced, then the edges are kept unweighted. We then collect all these canonicalization edges into the set C. Finally, these edges are added to the original OpenKG G to get a *canonicalization augmented OpenKG*, $\mathrm{G}' = (\mathrm{N}, \mathrm{R}, \mathrm{T}^{+} \cup \mathrm{C})$

4.3 Step 2: Base Model (B)

This component decides the way parameters are trained in CaRe based on the relational edges T^+. While CaRe is flexible to accommodate any KG embedding model as the base model, for the experiments in paper, we used TransE and ConvE (please see Section 3).

4.4 Step 2: Phrase Encoder Network (PN)

We generate embeddings of RPs by encoding the vector representations of the words composing them. Consider an RP as a sequence of T words $(w_1, w_2, ..., w_T)$ and their corresponding word vectors as $(x_1, x_2, ..., x_T)$. They are passed through a Phrase Encoder Network module (Figure 3) for which we use a bidirectional GRU (Cho et al., 2014) model with last pooling as described below.

$$(\overrightarrow{h_1}, \overrightarrow{h_2}, ..., \overrightarrow{h_T}) = \overrightarrow{GRU}(x_1, x_2, ..., x_T)$$
$$(\overleftarrow{h_1}, \overleftarrow{h_2}, ..., \overleftarrow{h_T}) = \overleftarrow{GRU}(x_1, x_2, ..., x_T).$$

Finally, $r_r = [\overrightarrow{h_T} : \overleftarrow{h_1}]$ is concatenation of final hidden states in both the directions. This approach allows parameter sharing across RPs with word overlaps while leveraging the rich semantic information from pre-trained word embeddings.

4.5 Step 2: Canonical Cluster Encoder Network (CN)

The canonicalization information is incorporated in the NP embeddings by utilizing the canonicalization-induced edges C.

Each NP $n \in$ N is assigned a vector e_n. As NP canonicalization is expressed as a clustering step, for each NP, its canonical NPs are a single edge away in C (see Figure 3). Hence, a single layer of network aggregation is sufficient. We propose a non-parametric message passing and update network which works in the following two steps:

Context vector: First for each NP, $n \in$ N a context vector e_n^c is generated by the following message passing scheme from its canonical neighbors.

$$e_n^c = \left(\sum_{i \in \mathcal{N}(n)} \frac{1}{|\mathcal{N}(n)|} e_i \right), \forall n \in \text{N}.$$

Here, $\mathcal{N}(n) = \{i | i \in \text{N}, (n, i) \in \text{C}\}$, is the canonical neighborhood of n.

Updating NP embeddings: The updated embeddings for each NP is computed as below:

$$e_n^{'} = \frac{e_n}{2} + \frac{e_n^c}{2}$$

which are passed to the decoding stage. For the case of weighted canonicalization edges, the weights can be used while generating the context vector of an NP, by making the contribution of its canonical NPs proportional to the edge weights joining them. The proposed approach can be interpreted as a *local embedding smoothening* of canonical NPs and we call this network as Local Averaging Network **(LAN)**. We also experimented with more sophisticated attention-based context vector generation and gating mechanism for embedding update stage, but they resulted in a performance drop.

Note that, our approach to embed NPs is inspired by the auto-encoder framework used in R-GCN (Schlichtkrull et al., 2018) but with a key difference. In R-GCN, the same set of edges are utilized at both the encoding and decoding stages. Whereas in CaRe, the canonicalization edges C are used at the encoder (Canonical Cluster Encoder Network) while the decoder (Base Model) operates on the relational edges T^+.

CaRe nomenclature: We introduce a generic nomenclature for the possible variants of the CaRe framework based on the choice of the Base Model (B), Phrase Encoder Network (PN) and Canonical Cluster Encoder Network (CN) as **CaRe(B, PN, CN)**. For e.g., CaRe(B=ConvE, PN=Bi-GRU, CN=LAN) corresponds to a CaRe model with ConvE as base model, Bi-GRU network with last pooling (Section 4.4) as the Phrase Encoder Network and Local Averaging Network (Section 4.5) as the Canonical Cluster Encoder Network. We define Bi-GRU and LAN as default values for the PN and CN arguments respectively. In case, values for these arguments are not explicitly mentioned, they take the default values. Thus, CaRe(B=ConvE) represents the same network as CaRe(B=ConvE, PN=Bi-GRU, CN=LAN).

5 Experiments

5.1 Datasets

Statistics of the two datasets used in the experiments of this paper are summarized in Table 1. In order to build these datasets, we first obtained the three graph datasets – Base, Ambiguous and ReVerb45K. While Base and Ambiguous datasets are created in (Galárraga et al., 2014), ReVerb45K is

Datasets	ReVerb45K	ReVerb20K
# NPs	27K	11.1K
# RPs	21.6K	11.1K
# Gold NP Clusters	18.6K	10.8K
# Train Triples	36K	15.5K
# Test Triples	5.4K	2.4K
# Validation Triples	3.6K	1.6K

Table 1: Details of datasets used. Please see Section 5.1 for details.

introduced in (Vashishth et al., 2018). ReVerb20K is created by combining the two smaller datasets Base and Ambiguous. All the datasets are constructed through ReVerb Open KB (Fader et al., 2011). We refer the readers to the respective papers for the construction details.

To generate the train, validation and test triples the following procedure is adopted. First, the entire set of triples is divided in $80 : 20$ ratio ensuring that each NP and RP in the triples of the smaller set is at least present once in the triples of the bigger set. This consideration is essential because of the transductive nature of existing KG embedding models. The bigger set is considered as the train dataset and the smaller set is further randomly divided in $30 : 70$ to get the validation and test datasets respectively. Both datasets contain gold canonicalization clusters for the NPs extracted through the Freebase entity linking information (Gabrilovich et al., 2013) which enables automatic evaluation in the canonicalization task.

5.2 Open KG Link Prediction Evaluation

In the typical link prediction evaluation, an unseen triple (s, r, o) is taken and partial triples $(s, r, ?)$ and $(?, r, o)$ are shown to the model. It ranks all the entities in the graph for their likelihood to be the missing entity and the rank assigned to the true missing entity is considered.

However, while this is suitable for ontological KGs, it is not valid for our setting. In OpenKGs, instead of entities, NPs are present and several of them can refer to the same entity. This means that, even when predicting correct entity, a model will be unfairly penalized if the prediction is a different canonical form of the entity than the one present in the considered triple. We, therefore propose to rank gold NP clusters, available in the dataset, instead of ranking each NP. We do this in the following manner:

- List all the NPs in decreasing order of their likelihood to be the missing part of the triple.
- Prune this list by keeping the best ranked NPs for each gold cluster. This gives us the ranked list of gold clusters.
- Consider the rank of the cluster to which the true missing NP belongs.

We provide results using three commonly used evaluation metrics: *mean rank* (MR), *mean reciprocal rank* (MRR) and *Hits@n* with $n = \{10, 30, 50\}$. Filtered setting as introduced in (Bordes et al., 2013) is followed.

5.3 Experimental Setup

We run CESI on both the datasets to generate the NP canonical clusters. Note that, none of the existing automatic canonicalization models output edge weights. Hence, for all the experiments, the canonicalization-induced edges are kept unweighted. For Phrase Encoder Network, we found single layer in the Bi-GRU model works best. CaRe allows the use of any pre-trained embeddings. However, (Vashishth et al., 2018) demonstrates the effectiveness of pre-trained GloVE (Pennington et al., 2014) vectors for initializing representations for the same datasets. Hence, the word vectors are initialized with 300-dimensional pre-trained GloVE embeddings and are kept trainable. We use PyTorch Geometric library (Fey and Lenssen, 2019) for the Canonical Cluster Encoder Network module.

The choice of optimizer and regularization based hyper-parameters is directly adopted from the ones proposed in the original work of the base models. Both the NP and RP embedding size is kept fixed at 300, while the learning rate is selected through a grid search over $\{0.1, 0.01, 0.001\}$.

Models like ConvE introduce inverse relations. For our experiments we generate inverse phrase for an RP by adding a phrase *"inverse of"* to that RP. In ComplEx the embeddings have both real and imaginary parts. For this, we use two separate Phrase Encoder Network and Graph Neural Network modules for the real and imaginary parts of RP and NP embeddings respectively.

6 Results

In this section we evaluate the following questions:

Method	ReVerb45K					ReVerb20K				
	MR	MRR	Hits@10	Hits@30	Hits@50	MR	MRR	Hits@10	Hits@30	Hits@50
TransE	2955.8	.193	.361	.446	.478	1425.8	.126	.299	.411	.468
TransH	2998.2	.194	.362	.442	.478	1464.4	.129	.303	.409	.467
DistMult	8988.8	.051	.051	.052	.065	6260.0	.033	.044	.055	.060
ComlEx	7786.5	.047	.047	.048	.073	5502.2	.037	.058	.075	.085
R-GCN	2866.8	.042	.046	.091	.113	1204.3	.122	.187	.263	.305
ConvE	2650.8	.233	.338	.401	.429	1014.5	.294	.402	.491	.541
CaRe(B=ConvE)	**1308.0**	**.324**	**.456**	**.543**	**.579**	**973.2**	**.318**	**.439**	**.525**	**.566**

Table 2: **Link Prediction results**. CaRe(B=ConvE) substantially outperforms all the existing KG embedding models. For all the experiments, evaluation strategy described in Section 5.2 is followed. B, CN, PN are the arguments of CaRe framework (Section 4 and Figure 3). For the reported results: PN=Bi-GRU and CN=LAN.

Method	ReVerb45K					ReVerb20K				
	MR	MRR	Hits@10	Hits@30	Hits@50	MR	MRR	Hits@10	Hits@30	Hits@50
TransE	2955.8	.193	.361	.446	.478	1425.8	.126	.299	.411	.468
CaRe(B=TransE, CN=ϕ)	2522.3	.195	.378	.457	.488	978.2	.286	.411	.515	.565
ComplEx	7786.5	.047	.047	.048	.073	5502.2	.037	.058	.075	.085
CaRe(B=ComplEx, CN=ϕ)	5205.4	.185	.225	.235	.325	3010.0	.216	.288	.345	.376
R-GCN	2866.8	.042	.046	.091	.113	1204.3	.122	.187	.263	.305
CaRe(B=R-GCN, CN=ϕ)	2508.1	.145	.203	.260	.305	1210.3	.195	.275	.340	.370
ConvE	2650.8	.233	.338	.401	.429	1014.5	.294	.402	.491	.541
CaRe(B=ConvE, CN=ϕ)	1656.1	.293	.401	.477	.509	966.9	.307	.419	.514	.556

Table 3: Impact of parameterizing RP embeddings. B, CN, PN are the arguments of CaRe framework (Section 4 and Figure 3). ϕ value for CN argument implies that Canonical Cluster Encoder Network module is not used in these experiments. PN=Bi-GRU in all these experiments. Please refer to Section 6.2.

Q1. Is CaRe effective for the link prediction task in OpenKGs? (Section 6.1)

Q2. What is the quantitave and qualitative impact of parameterizing RP embeddings in CaRe? (Section 6.2 and Section 6.4)

Q3. How does the local embedding smoothening approach adopted in CaRe compare against other competitive baselines? (Section 6.3)

6.1 Overall Performance

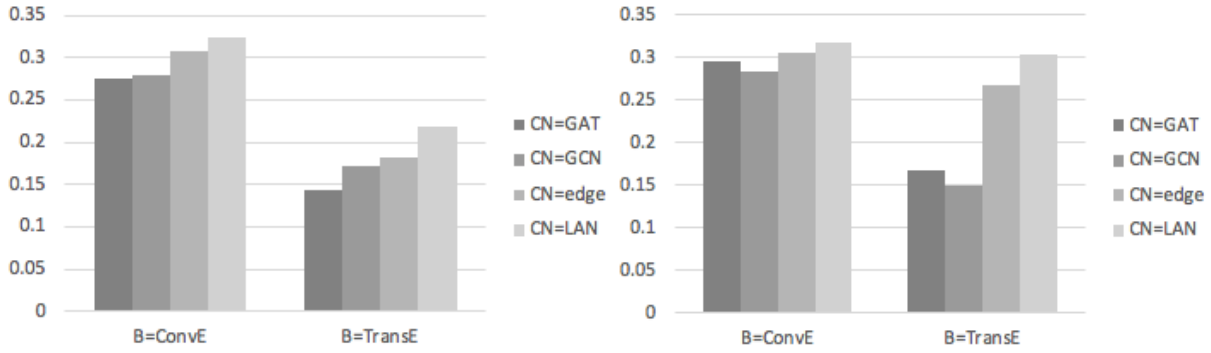
As shown in Figure 3, any existing KG embedding model can be used as the base model in the CaRe framework. We experimented with several KG embedding models and found the CaRe achieved substantial improvements in comparison to standalone use of the KG embedding model in each case. We achieved overall best performance working with ConvE as the base model, CaRe(B=ConvE). The experimental results are presented in Table 2.

Table 1 shows that in both the datasets, on an average, the number of train triples for each NP and RP is less than 2. In contrast, FB15k (Bordes et al., 2013), an ontological KG, has on an average 32 triples for each entity and 360 triples for each relation. This highlights the extremely sparse and fragmented nature of OpenKGs. Hence, the superior performance of CaRe supports the hypothesis that the flow of information while learning the representations of canonical NPs and RPs in OpenKGs is beneficial.

Also, a comparison of the number of unique NPs and gold NP clusters for the two datasets (Table 1) shows that the number of NPs canonical to each other is more significant in ReVerb45K than in ReVerb20K. Hence, a more prominent improvement due to CaRe in ReVerb45K as compared to ReVerb20K proves the effectiveness of CaRe in utilizing the canonicalization information.

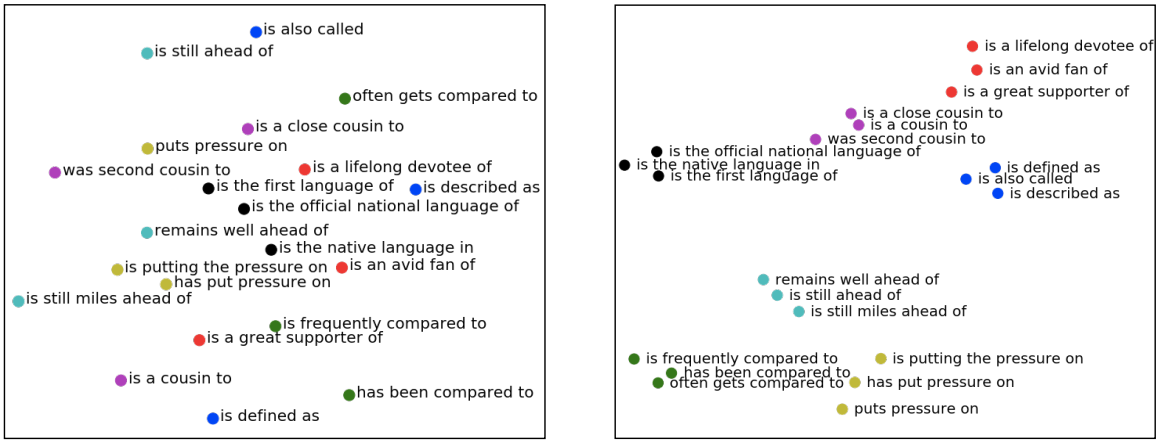
6.2 Impact of parameterizing RP embeddings

As described in Section 4.4, a Phrase Encoder Network is used to parametrize RP embeddings, which allows parameter sharing while learning embeddings. Table 3 provides a quantitative anal-



(a) ReVerb45K (b) ReVerb20K

Figure 4: Performance comparison of CaRe framework for diffrent values of the CN (as described in Section 6.3). B=ConvE (left group) and B=TransE (right group) in both the plots. PN=Bi-GRU. **Mean Reciprocal Rank (MRR)** is plotted on the y axis (higher is better).



(a) ConvE (b) CaRe(B=ConvE)

Figure 5: **t-SNE visualization of RP embeddings**. RP embeddings learned by CaRe(B=ConvE) are able to capture the semantic similarity of the RPs whereas in ConvE this information is lost. Please refer to Section 6.4.

ysis of the impact of parameterizing RP embeddings across several KG embedding models. In these experiments, the CN argument of CaReis kept ϕ, which implies no canonicalization information is used. The NP embeddings are trained in the same manner as the base KG embedding model.

Comparing the performance of CaRe(B=ConvE,CN=ϕ) in Table 3 with the performance of CaRe(B=ConvE) in Table 2, it can be noticed that a major part of the performance boost of CaRe can be attributed to parameterization of RP embeddings.

6.3 Different ways to utilize Canonicalization edges

There can be several methods through which the canonicalization edges integrated into the graph (as described in Figure 2), can be utilized in the model. In Section 4.5, we described a local embedding smoothening method adopted in CaRe. In this section, we present a comparative analysis with the following competitive baselines:

- **CaRe(CN=GCN)**: A single layer of GCN (Kipf and Welling, 2016) for the Canonical Cluster Encoder Network. Refer to Equation 1.
- **CaRe(CN=GAT)**: A single layer of GAT (Veličković et al., 2018) for the Canonical Cluster Encoder Network. Refer to Equation 2.
- **CaRe(CN=edge)**: Here the canonicalization edges in G' are labelled with a symmetric RP *"is canonical to"*. This adds a new edge type in the graph and is treated by KG embedding models like any other edge type.

We show these comparisons using both TransE and ConvE as base models in Figure 4. The GCN and GAT architectures have an adverse effect on the performance. We believe this can be attributed to the complex nature of these architectures in an already over-parameterized and noisy setting. In CaRe(CN=edge), the model is expected to learn the meaning of two NPs being canonical and encode it in the vector embedding of the new edge type. The local averaging network CaRe(CN=LAN) used in CaRe is a GNN architecture with fixed weights. The choice of weights follows from the prior belief that canonical NP embeddings should be close in the vector space, inducing a useful inductive bias. The above results indicate that this leads to better performance. Additionally, unlike the GNN based methods, CaRe(CN=edge) has a limitation in its modelling capacity as it provides no way to handle the case where canonicalization edges are weighted.

6.4 Qualitative Analysis

Figure 5 demonstrates a qualitative comparison of RP embeddings between ConvE and CaRe(B=ConvE). For this experiment, we selected seven RPs, and for each RP, through human judgement, we selected two more RPs with similar meaning. Thus, there are seven different clusters. The figure shows t-SNE (van der Maaten and Hinton, 2008) visualization of the embeddings learnt for these RPs by CaRe(B=ConvE) and ConvE. t-SNE is a non-linear transformation which tries to map points in high dimensional space to a lower dimension preserving the local relationships between the points. The figure verifies the hypothesis that due to the parameterization of RP embeddings and utilizing pre-trained word embeddings, CaRe is able to better capture the semantic similarity of the RPs in comparison to the base models. Due to the explicit integration of NP canonicalization in CaRe, we observed a similar desirable impact on the embeddings learned for NPs as well.

7 Conclusion

Open Information Extraction (OpenIE) provides an effective way to bootstrap Open Knowledge Graphs (OpenKGs) from text corpus. OpenKGs consist of noun phrases (NPs) as nodes and relations phrases (RPs) as edges. In spite of this advantage, OpenKGs are often sparse and non-canonicalized, i.e., the same entity could be expressed using multiple nodes in the graph (and similarly for relations). This renders existing Ontological KG embedding methods ineffective in learning embeddings of NPs and RPs in OpenKGs. In spite of this limitation, there has been no prior work which has focused on OpenKG embedding. We fill this gap in the paper and propose CaRe. CaRe infuses canonicalization information combined with the neighborhood graph structure to learn rich representations of NPs. Further, it captures the semantic similarity of RPs by utilizing the word sequence information in these relation phrases to parameterize the RP embeddings. Through extensive experiments on real-world datasets, we demonstrate the effectiveness of embeddings learned by CaRe.

As part of future work, we hope to extend CaRe to also utilize RP canonicalization information. Utilizing OpenKG embeddings in tasks beyond link prediction is another avenue of further work.

Acknowledgments

We thank the anonymous reviewers for their constructive comments. This work is supported by the Ministry of Human Resource Development (MHRD), Government of India. Finally, we thank all the members of MALL Lab, IISc for their invaluable suggestions.

References

Gabor Angeli, Melvin Jose Johnson Premkumar, and Christopher D. Manning. 2015. Leveraging linguistic structure for open domain information extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 344–354, Beijing, China. Association for Computational Linguistics.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The Semantic Web*, pages 722–735, Berlin, Heidelberg. Springer Berlin Heidelberg.

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*, SIGMOD '08, pages 1247–1250, New York, NY, USA. ACM.

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*, NIPS'13, pages 2787–2795, USA. Curran Associates Inc.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann Lecun. 2014. Spectral networks and locally connected networks on graphs. In *International Conference on Learning Representations (ICLR2014), CBLS, April 2014*.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Janara Christensen, Mausam, Stephen Soderland, and Oren Etzioni. 2011. An analysis of open information extraction based on semantic role labeling. In *Proceedings of the Sixth International Conference on Knowledge Capture*, K-CAP '11, pages 113–120, New York, NY, USA. ACM.

Luciano Del Corro and Rainer Gemulla. 2013. Clausie: Clause-based open information extraction. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 355–366, New York, NY, USA. ACM.

Tim Dettmers, Minervini Pasquale, Stenetorp Pontus, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Anthony Fader, Stephen Soderland, and Oren Etzioni. 2011. Identifying relations for open information extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1535–1545, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

Evgeniy Gabrilovich, Michael Ringgaard, and Amarnag Subramanya. 2013. Facc1: Freebase annotation of clueweb corpora, version 1 (release date 2013-06-26, format version 1, correction level 0). *Note: http://lemurproject.org/clueweb09/FACC1/Cited by*, 5.

Luis Galárraga, Geremy Heitz, Kevin Murphy, and Fabian M Suchanek. 2014. Canonicalizing open knowledge bases. In *Proceedings of the 23rd acm international conference on conference on information and knowledge management*, pages 1679–1688. ACM.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 413–422, New York, NY, USA. ACM.

Marco Gori, Gabriele Monfardini, and Franco Scarselli. 2005. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. 2012. Open language learning for information extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 523–534, Jeju Island, Korea. Association for Computational Linguistics.

Mausam Mausam. 2016. Open information extraction systems and downstream applications. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 4074–4077. AAAI Press.

T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, and J. Welling. 2018. Never-ending learning. *Commun. ACM*, 61(5):103–115.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1955–1961. AAAI Press.

Harinder Pal and Mausam. 2016. Demonyms and compound relational nouns in nominal open IE. In *Proceedings of the 5th Workshop on Automated Knowledge Base Construction*, pages 35–39, San Diego, CA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language*

Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

Michael Sejr Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling relational data with graph convolutional networks. In *ESWC*, volume 10843 of *Lecture Notes in Computer Science*, pages 593–607. Springer.

Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. In *International Conference on Machine Learning*, pages 2071–2080.

Shikhar Vashishth, Prince Jain, and Partha Talukdar. 2018. Cesi: Canonicalizing open knowledge bases using embeddings and side information. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1317–1327. International World Wide Web Conferences Steering Committee.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. *International Conference on Learning Representations*. Accepted as poster.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Twenty-Eighth AAAI conference on artificial intelligence*.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575*.

Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. 2007. Textrunner: open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, pages 25–26. Association for Computational Linguistics.