Disentangle-based Continual Graph Representation Learning

Xiaoyu Kou\textsuperscript{1}; Yankai Lin\textsuperscript{2}, Shaobo Liu\textsuperscript{2}, Peng Li\textsuperscript{2}, Jie Zhou\textsuperscript{2}, Yan Zhang\textsuperscript{1}

\textsuperscript{1}Key Laboratory of Machine Perception (MOE), Department of Machine Intelligence, Peking University, Beijing, China
\textsuperscript{2}Pattern Recognition Center, WeChat AI, Tencent Inc., China
\{kouxiaoyu,zhyzhy001\}@pku.edu.cn
\{yankailin,patrickpli,withtomzhou\}@tencent.com

Abstract

Graph embedding (GE) methods embed nodes (and/or edges) in graph into a low-dimensional semantic space, and have shown its effectiveness in modeling multi-relational data. However, existing GE models are not practical in real-world applications since it overlooked the streaming nature of incoming data. To address this issue, we study the problem of continual graph representation learning which aims to continually train a graph embedding model on new data to learn incessantly emerging multi-relational data while avoiding catastrophically forgetting old learned knowledge. Moreover, we propose a disentangle-based continual graph representation learning (DiC-GRL) framework inspired by the human’s ability to learn procedural knowledge. The experimental results show that DiCGRL could effectively alleviate the catastrophic forgetting problem and outperform state-of-the-art continual learning models.

1 Introduction

Multi-relational data represents relationships between entities in the world, which is usually denoted as a multi-relational graph with nodes and edges connecting them. It is widely used in real-world NLP applications such as knowledge graphs (KGs) (e.g., Freebase (Bollacker et al., 2008) and DBpedia (Lehmann et al., 2015)) and information networks (e.g., social network and citation network). Therefore, modeling multi-relational graph with graph embeddings (Bordes et al., 2013; Tang et al., 2015a; Sun et al., 2019; Bruna et al., 2014) has been attracting intensive attentions in both academia and industry. Graph embedding (GE), aiming to embed nodes and/or edges in the graph into a low-dimensional semantic space to enable neural models to effectively and efficiently utilize multi-relational data, has demonstrated remarkable effectiveness in various downstream NLP tasks such as question answering (Bordes et al., 2014) and dialogue system (Moon et al., 2019).

Nevertheless, most existing graph embedding works overlook the streaming nature of the incoming data in real-world scenarios. In consequence, these models have to be retrained from scratch to reflect the data change, which is computationally expensive. To tackle this issue, we propose to study the problem of continual graph representation learning (CGRL) in this work.

The goal of continual learning is to alleviate catastrophically forgetting old data while learning new data. There are two mainstream continual learning methods in NLP: (1) consolidation-based methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Liu et al., 2018; Ritter et al., 2018) which consolidate the important model parameters of old data when learning new data; and (2) memory-based methods (Lopez-Paz and Ranzato, 2017; Shin et al., 2017; Chaudhry et al., 2019) which remember a few old examples and learn them with new data jointly. Despite the promising results these methods have achieved on classification tasks, their effectiveness has not been validated on graph representation learning. Unlike the classification problem where instances are generally independent
and can be operated individually, nodes and edges in multi-relational data are correlated, making it sub-optimal to directly deploy existing continuous learning methods on the multi-relational data.

In cognitive psychology (Solso et al., 2005), procedural knowledge refers to a set of operational steps. Its smallest unit is production, where multiple productions can complete a series of cognitive activities. When learning new procedural knowledge, humans would update cognitive results by only updating a few related productions and leave the rest intact. Intuitively, such a process can be mimicked to learn constantly growing multi-relational data by regarding each new data as a new procedural knowledge. For example, as illustrated in Figure 1, the relational triplets of Barack Obama and Michelle Obama are related to three concepts: “family”, “occupation” and “location”. When a new relational triplet (Michelle Obama, Daughter, Malia Ann Obama) appears, we only need to update the “family”-related information in Barack Obama. Consequently, we can further infer that the triplet (Barack Obama, Daughter, Malia Ann Obama) also holds.

Inspired by procedural knowledge learning, we propose a disentangle-based continual graph representation learning framework DiCGRL in this work. Our proposed DiCGRL consists of two modules: (1) **Disentangle module.** It decouples the relational triplets in the graph into multiple independent components according to their semantic aspects, and leverages two typical GE methods including Knowledge Graph Embedding (KGE) and Network Embedding (NE) to learn disentangled graph embeddings; (2) **Updating module.** When new relational triplets arrive, it selects the relevant old relational triplets and only updates the corresponding components of their graph embeddings. Compared with memory-based continual learning methods which save a fixed set of old data, DiCGRL could dynamically select important old data according to new data to fine-tune the model, which makes DiCGRL better model the complex multi-relational data stream.

We conduct extensive experiments on both KGE and NE settings based on the real-world scenarios, and the experimental results show that DiCGRL effectively alleviates the catastrophic forgetting problem and significantly outperforms existing continual learning models while remaining efficient.

2 Related Work

2.1 Graph Embedding

Graph embedding (GE) methods are critical techniques to obtain a good representation of multi-relational data. There are mainly two categories of typical multi-relational data in the real-world, knowledge graphs (KGs) and information networks. GE handles them via Knowledge Graph Embedding (KGE) and Network Embedding (NE) respectively, and our DiCGRL framework can adapt to the above two typical GE methods, which demonstrates the generalization ability of our model.

KGE is an active research area recently, which can be mainly divided into two categories to tackle link prediction task (Ji et al., 2020). One line of work is reconstruction-based models, which reconstruct the head/tail entity’s embedding of a triplet using the relation and tail/head embeddings, such as TransE (Bordes et al., 2013), RotatE (Sun et al., 2019), and ConvE (Dettmers et al., 2018). Another line of work is bilinear-based models, which consider link prediction as a semantic matching problem. They take head, tail and relation’s embeddings as inputs, and measure a semantic matching score for each triplet using bi-linear transformation (e.g., DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016), ConvKB (Nguyen et al., 2018)). Besides KGE, NE is also widely explored in both academia and industry. Early works (Perozzi et al., 2014; Tu et al., 2016; Tang et al., 2015b) focus on learning static node embeddings on information graphs. More recently, graph neural networks (Bruna et al., 2014; Henaff et al., 2015; Veličković et al., 2018) have been attracting considerable attention and achieved remarkable success in learning network embeddings. However, most existing GE models assume the training data is static, i.e., do not change over time, which makes them impractical in real-world applications.

2.2 Continual Learning

Continual learning, also known as life-long learning, helps alleviate catastrophic forgetting and enables incremental training for stream data. Methods for continual learning in natural language processing (NLP) field can mainly be divided into two categories: (1) consolidation-based methods (Kirkpatrick et al., 2017; Zenke et al., 2017), which slow down parameter updating to preserve old knowledge, and (2) memory-based methods (Lopez-Paz and Ranzato, 2017; Shin et al., 2017; Chaudhry
et al., 2019; Wang et al., 2019), which retain examples from old data for re-play upon learning the new data. Although continual learning has been widely studied in NLP (Sun et al., 2020) and computer vision (Kirkpatrick et al., 2017), its exploration on graph embedding is relatively rare. Sankar et al. (2018) seek to train graph embedding on constantly evolving data. However, it assumes the timestamp information is known beforehand, which hinders its application to other tasks. Song and Park (2018) extend the idea of regulation-based methods to continually learn graph embeddings which straightforwardly limits parameter updating on only the embedding layer. It is therefore hard to generalize to more complex multi-relational data. Our proposed DiCGRL model is distinct from previous works in two aspects: (1) Our method does not require pre-annotated timestamps, which make it more feasible in various types of multi-relational data; (2) Inspired by procedural knowledge learning, we exploit disentanglement to conduct continual learning and achieve promising results.

3 Methodology

3.1 Task Formulation and Overall Framework

We represent multi-relational data as a multi-relational graph \( G = (V, E) \), where \( V, E \) denote the node set and the edge set within a graph \( G \), and \( G \) can be formalized as a set of relational triplets \( \{(u, r, v)\} \subseteq V \times E \times V \). Given a relational triplet \((u, r, v) \in G\), we denote the embeddings of them as \( u, v \in \mathbb{R}^d \) and \( r \in \mathbb{R}^l \), where \( d \) and \( l \) indicate the vector dimension.

Continual graph representation learning trains graph embedding (GE) models on constantly growing multi-relational data, where the \( i \)-th part of multi-relational data has its own training set \( \mathcal{T}_i \), validation set \( \mathcal{V}_i \), and query set \( \mathcal{Q}_i \). The \( i \)-th training set is defined as a set of relational triplets, i.e., \( \mathcal{T}_i = \{(u_1^{T_i}, r_1^{T_i}, v_1^{T_i}), \ldots, (u_N^{T_i}, r_N^{T_i}, v_N^{T_i})\} \), where \( N \) is the instance number of \( \mathcal{T}_i \). The \( i \)-th validation and query sets are defined similarly. As new relational triplets emerges, continual graph representation learning requires GE models to achieve good results on all previous query sets. Therefore, after training on the \( i \)-th training set \( \mathcal{T}_i \), GE models will be evaluated on \( \mathcal{Q}_i = \bigcup_{j=1}^{i} \mathcal{Q}_j \) to measure whether they could well model both new and old multi-relational data. The evaluation protocol indicates that it will be more and more difficult for the model to achieve high performance as the emerging of new relational triplets.

In general, our model continually learns on the streaming data. Whenever there comes a new part of multi-relational data, DiCGRL will learn the new graph embeddings and meanwhile prevent catastrophically forgetting old learned knowledge through two procedures: (1) **Disentangle module.** It decouples the relational triplets in the graph into multiple components according to their semantic aspects, and learns disentangled graph embeddings that divide node embeddings into multiple independent components where each component describes a semantic aspect of node; (2) **Updating module.** When new relational triplets arrive, DiCGRL first activates the old relational triplets from previous graphs which have relevant semantic aspects with the new ones, and only updates the corresponding components of their graph embeddings.

3.2 Disentangle Module

When the \( i \)-th training set \( \mathcal{T}_i \) becomes available, DiCGRL needs to update the graph embeddings according to these new relational triplets. To this end, for each node \( u \in V \), we want to learn a disentangled node embedding \( u \), which is composed of \( K \) independent components, i.e., \( u = [u^1, u^2, \ldots, u^K] \), where \((0 \leq k \leq K)\) and \( u^k \in \mathbb{R}^d \). The component \( u^k \) is used to represent the \( k \)-th semantic aspect of node \( u \). As shown in Figure 2, the key challenge of the disentangle module is how to decouple the relational triplets into multiple components according to their semantic aspects, and learn the disentangled graph embeddings in different components independently.

Formally, given a relational triplet \((u, r, v) \in \mathcal{T}_i \), we aim to extract the most related semantic components of \( u \) and \( v \) with respective to the relation \( r \). Specifically, we model this process with an attention mechanism, where \((u, r, v) \) is associated with \( K \) attention weight \((\alpha_1^{u, r}, \alpha_2^{u, r}, \ldots, \alpha_K^{u, r})\), which respectively represent the probability being assigned
to the $k$-th semantic component. After that, we select the top-$n$ related components of $u$ and $v$ with the highest attention weight. Then we leverage exiting GE methods to extract different features in the selected top-$n$ related components, and we denote the feature extraction operation as $f$. Here, $f$ could be any graph embedding operation that can incorporate the features of node $u$ and $v$ in the selected top-$n$ related components. In this work, we adapt our DiCGRL model in two typical graph embeddings including:

**Knowledge Graph Embeddings (KGEs)** Intuitively, the most related semantic components of a relational triplet in KG are related to their relation $r$. Therefore, we can directly set $K$ attention values for each explicit relation $r$, and the $k$-th attention value $\alpha_r^k (0 \leq k \leq K)$ is a trainable parameter which indicates how related this edge is to the $k$-th component. The normalized attention weight $\alpha_r^k$ is computed as:

$$\alpha_r^k = \frac{\exp(\alpha_r^k)}{\sum_{j=1}^{K} \exp(\alpha_r^j)}, \quad (1)$$

As described in related work, KGE models can mainly be divided into two categories: reconstruction-based and bilinear-based models. We explore the effectiveness of both two lines of works to extract features in our framework. Specifically, we leverage two classic KGE models as $f$ to extract latent features in our experiment including TransE (reconstruction-based):

$$f = ||\hat{u} + r - \hat{v}||_p, \quad (2)$$

and ConvKB (bilinear-based):

$$f = W_1(\text{ReLU} (\text{Conv}([\hat{u}; r; \hat{v}]))), \quad (3)$$

where $\hat{u}, \hat{v}$ are the concatenation of top-$n$ component embeddings of node $u$ and node $v$ respectively; $|| \cdot ||_p$ denotes the $p$-norm operation; $[\cdot; \cdot]$ denotes the concatenate operation; $\text{Conv} (\cdot)$ indicates the convolutional layer with $M$ filters, and $W_1 \in \mathbb{R}^{1 \times M d_2}$ is a trainable matrix. In total, $f$ is expected to give higher scores for valid triplets than invalid ones.

**Network Embeddings (NEs)** We first determine $\alpha_r^k$ according to the representations of node $u$ and node $v$ since NE usually does not provide explicit relations. Hence, $\alpha_r^k$ is calculated by performing a non-linearity transformation over the concatenation of $u$ and $v$:

$$\alpha_r^k = \frac{\exp(\text{ReLU}(W_2[u^k; v^k]))}{\sum_{j=1}^{K} \exp(\text{ReLU}(W_2[u^j; v^j]))}, \quad (4)$$

where $W_2 \in \mathbb{R}^{1 \times 2d}$ is a trainable matrix.$^1$

Graph attention networks (GATs) (Veličković et al., 2018) gather information from the node’s neighborhood and assign varying levels of importance to neighborhoods, which is a widely used and powerful way to learn embeddings for information networks. Thus we leverage GATs as $f$ to extract latent features for NE. Given a target node $u$ and its neighbors $\{v \mid v \in N_u\}$, we first determine the top-$n$ related components for each pair of nodes $(u, v)$ according to the attention weights $\alpha_r^k$. When updating the $k$-th component of $u$, a neighbor $v$ is considered if and only if the $k$-th component is in the the top-$n$ related components for the node pair $(u, v)$. In this way, we can thoroughly disentangle the neighbors of the target node into different components to play their roles separately. GATs are used to update each component as follows:

$$u^k = \sum_{v \in N_u} \sigma(W_3v^k)W_4^k u^k, \quad (5)$$

where $W_3 \in \mathbb{R}^{1 \times d}$ and $W_4 \in \mathbb{R}^{h \times d}$ are two trainable matrices, $h$ is hidden size within GATs, and $\sigma$ is the softmax function which is used to calculate the neighbor’s relative attention value in the $k$-th component.

### 3.3 Updating Module

Now, the remaining problem is how to update the disentangled graph embedding when new relation triplets appear while preventing catastrophic forgetting. As shown in Figure 3, this process mainly includes two steps:

1. **Neighbor activation**: DiCGRL needs to identify which relational triplets from $T_1, \ldots, T_{i-1}$ need to be updated. Since in the multi-relational data, nodes are not independent, and therefore a new relational triplet may have influence on the embeddings of nodes that not directly connect to it. Inspired by that, for each relational triplet $(u, r, v)$, we activate both their direct and indirect neighbor triplets.$^2$

$^1$Note that we also evaluate this definition on KG data, and the experimental results are presented in Appendix A.

$^2$1-order and 2-order neighbors are both considered in our experiments.
As mentioned before, for newly arrived multi-relational data \( T_i \), we iteratively train our model on \( T_i \) and its activated neighbor relational triplets. We denote loss functions of these two parts as \( L_{\text{new}} \) and \( L_{\text{old}} \) respectively. For KGE, we utilize soft-max loss. For NE, we leverage a standard cross-entropy loss according to GATs and train our model on node classification task. \( L_{\text{new}} \) can be formulated as follows:

\[
L_{\text{new}} = - \sum_{(u,r,v) \in T_i} \frac{1}{|C|} \sum_{c=1}^{|C|} y(c) \cdot \log (\sigma(W_5 u)),
\]

where \( c \) is node’s class and \( y(c) = 1 \) if the node label is \( c \), otherwise \( y(c) = 0 \); \( N(T_i) \) is the node set of \( T_i \). \(|C|\) indicates the number of class, and \( W_5 \in \mathbb{R}^{C \times d} \) is a trainable matrix. For KGE and NE, \( L_{\text{old}} \) can be defined in the same way with \( L_{\text{new}} \) on the selected old relational triplets set.

Intuitively, the less components a relation focuses on, the better the disentanglement is. Therefore, we add a constraint loss terms \( L_{\text{norm}} \) for \( T_i \) to encourage the sum of the attention weights of the top-\( n \) selected components to reach 1, i.e.,

\[
L_{\text{norm}} = \sum_{(u,r,v) \in T_i} (1 - \sum_{k} a_r^k),
\]

where \( n \) indicates the number of selected components.

The overall loss function \( L \) of our proposed model is defined as follows:

\[
L = L_{\text{old}} + L_{\text{new}} + \beta \cdot L_{\text{norm}},
\]

where \( \beta \) is a hyper-parameter.

4 Experiments

In this section, we evaluate our model on two popular tasks: link prediction for knowledge graph and node classification for information network.

4.1 Datasets

We conduct experiments on several continual learning datasets adapted from existing graph embedding benchmarks:

KGE datasets. We considered two link prediction benchmark datasets, namely FB15K-237 (Toutanova and Chen, 2015) and FB15k-237 (Toutanova and Chen, 2015) and
WN18RR (Dettmers et al., 2018). We randomly split each benchmark dataset into five parts to simulate the real-world scenarios, with each part having the ratio of 0.8 : 0.05 : 0.05 : 0.05 : 0.05 respectively 5. We further divide each part into training set, validation set, and query set. The statistics of Cora, CiteSeer and PubMed are presented in Appendix C.

**NE datasets.** We conduct our experiments on three real-world information networks for node classification task: Cora, CiteSeer and PubMed (Sen et al., 2008). The nodes, edges and labels in these three citation datasets represent articles, citations and research areas respectively, and their nodes are provided with rich features. Like KGE datasets, we split each dataset into four parts and the partition ratio is 0.7 : 0.1 : 0.1 : 0.1. We further split train/validation/query set for each part. The statistics of Cora, CiteSeer and PubMed are presented in Appendix C.

### 4.2 Experimental Settings

We use Adam (Kingma and Ba, 2015) as the optimizer and fine-tune the hyper-parameters on the validation set for each task. We perform a grid search for the hyper-parameters specified as follows: the number of components $K \in \{2, 4, 6, 8, 10\}$, the number of top components $n \in \{2, 4\}$, node embedding dimension $d \in \{100, 200\}$ (note that relation embedding dimension in KG is $l = \frac{d \times n}{K}$, and $d$ is fixed to feature length in information network), initial learning rate $lr \in \{1e-3, 1e-4\}$, and the weight of regulation loss $\ell_{\text{norm}} \beta \in \{0.1, 0.3\}$. The optimal hyper-parameters on FB15k-237 dataset are: $K = 8$, $n = 4$, $d = 200$, $lr = 1e-4$, $\beta = 0.3$; and the optimal hyper-parameters on WN18RR dataset are: $K = 4$, $n = 2$, $d = 200$, $lr = 1e-4$, $\beta = 0.1$. For the NE datasets, the optimal hyper-parameters are: $K = 8$, $n = 4$, $lr = 1e-3$, $\beta = 0.1$.

For a fair comparison, we implement the baseline models (TransE, ConvKB, and GATs) by ourselves based on released codes respectively, and use the same hyper-parameters as DiCGRL. For example, the embedding dimension for TransE and ConvKB are both 200; the number of filters for ConvKB is 50, and the number of heads in GATs is 8. We implement the baseline models of continual learning based on the toolkit 6 released by Han et al. (2020). For fair comparison, we use the same memory size for both our model and baselines. For other hyper-parameters, we also follow the settings in Han et al. (2020).

Following existing works (Bordes et al., 2013), we use the “Filtered” setting protocol, i.e., not taking any corrupted triples that appear in the KG into accounts, and the evaluation metrics of link prediction task include mean reciprocal rank (MRR) and proportion of valid test triplets in top-10 ranks (H@10). For node classification task, we use accuracy as our evaluation metric. We use two settings to evaluate the overall performance of our DiCGRL after learning on all graphs: (1) **whole performance** calculates the evaluation metrics on the whole test set of all data; (2) **average performance** averages the evaluation metrics on all test sets. As average performance highlights the performance of handling catastrophic forgetting problem, thus it is the main metric to evaluate models.

### 4.3 Baselines

We compare our model with several baselines including two theoretical models to measure the lower and upper bounds of continual learning:

1. **Lower Bound**, which continually fine-tunes models on the new multi-relational dataset without memorizing any historical instances;
2. **Upper Bound**, which continually re-train models with all historical and new incoming instances. In fact, this model serves as the ideal upper bound for the performance of continual learning; and several typical continual learning models:
3. **EWC** (Kirkpatrick et al., 2017), which continually fine-tunes and several typical continual learning models: parameters when learning new data;
4. **EMR** (Parisi et al., 2019), a basic memory-based method, which memorizes a few historical instances and conducts memory replay. Each time when new data comes in, EMR mixes memorized instances with new instances to fine-tune models;
5. **GEM** (Lopez-Paz and Ranzato, 2017), which memorizes a few historical instances and adds a constraint on directions of new gradients to make sure that there is no conflict of optimization directions with gradients on old data.

---

5The characteristics of multi-relational graphs in real world scenarios are: 1) large-scale 2) the new multi-relational data is coming every day and small in scale proportional to the original size of these graphs.

6https://github.com/thunlp/ContinualRE
### Dataset

<table>
<thead>
<tr>
<th>Dataset</th>
<th>FB15k-237</th>
<th>WN18RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>KGE Model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower Bound</td>
<td>24.1</td>
<td>29.8</td>
</tr>
<tr>
<td>TransE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EWC</td>
<td>29.8</td>
<td>32.2</td>
</tr>
<tr>
<td>EMR</td>
<td>37.4</td>
<td>45.9</td>
</tr>
<tr>
<td>GEM</td>
<td>38.3</td>
<td>46.1</td>
</tr>
<tr>
<td>DiCGRL</td>
<td>40.9</td>
<td>47.7</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>43.5</td>
<td>49.7</td>
</tr>
<tr>
<td>ConvKB</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower Bound</td>
<td>13.2</td>
<td>19.3</td>
</tr>
<tr>
<td>EWC</td>
<td>18.8</td>
<td>22.7</td>
</tr>
<tr>
<td>EMR</td>
<td>29.5</td>
<td>33.8</td>
</tr>
<tr>
<td>GEM</td>
<td>28.0</td>
<td>35.0</td>
</tr>
<tr>
<td>DiCGRL</td>
<td>31.7</td>
<td>40.2</td>
</tr>
<tr>
<td>Upper Bound</td>
<td>33.9</td>
<td>42.4</td>
</tr>
</tbody>
</table>

Table 1: H@10 (%) results of models on two KG benchmarks. “W” stands for the Whole performance, and “A” stands for the Average performance.

### Overall Results

Table 1 and Table 2 show the overall performance on both KGE and NE benchmarks under two evaluation settings. From the tables, we can see that:

1. Our proposed DiCGRL model significantly outperforms other baselines and achieves state-of-the-art performance almost in all settings and datasets. It verifies the effectiveness of our disentangled approach in continual learning, which decouples the node embeddings into multiple components with respect to the semantic aspects, and only updates the corresponding components of graph embedding for new relational triplets.

2. There is still a gap between our model and the upper bound. It indicates although we have proposed an effective approach for continual graph representation learning, it still remains an open problem deserving further exploration.

3. Although DiCGRL outperforms other baselines in almost all settings in three information network benchmarks, the performance gain is not as high as it is on the KG datasets. The reason is that these three citation benchmarks are provided with rich node features, which would reduce the impact of topology changes. As can be seen, even the weakest Lower Bound achieves relatively high results close to Upper Bound.

To further investigate how evaluation metrics change while learning new relational triplets, we show the average performance on the KG and NE datasets at each part in Figure 4 and Figure 5. From the figures, we observe that:

1. With increasing numbers of new relational triplets, the performance of all the models in almost all the datasets decreases to some degree (CiteSeer may introduce some instability in random data splitting since this dataset is small). This indicates that catastrophically forgetting old data is inevitable, and it is indeed one of the major difficulties for continual graph representation learning.

2. The memory-based methods outperforms the consolidation-based methods, which demonstrates the memory-based methods may be more suitable for alleviating catastrophic forgetting in
Our proposed DiCGRL model achieves significantly better results compared to other baseline models. It indicates that disentangling relational triplets and updating dynamically selected components of relational triplets are more useful and reasonable than rote memorization of static examples from old multi-relational data.

4.5 Hyper-Parameter Sensitivity

In this section, we investigate the effect of the number of components $K$ and the top selected component number $n$, which are important hyperparameters of our DiCGRL. These experiments are only performed on NE datasets, since the node embedding dimension $d$ can be affected by $K$ and $n$ in KG as introduced in Section 4.2, which would make it difficult to make a fair comparison.

**Component Number $K$:** We use $n = 2, \beta = 0.1$ to run DiCGRL on the Cora dataset with five different $K$ settings. The results are illustrated in Figure 6(a). From the figure, we find that overall the average accuracy raises when $K$ increases from 2 to 8, which suggests the importance of disentangling components. However, when $K$ grows larger than 8, the performance starts to decline. One possible reason is that the number of components is already larger than that of semantics aspects, making it harder to achieve a good disentanglement. Therefore, we select the component number $K$ for each dataset on the development set, and for most dataset, we select $K = 4$ or $K = 8$.

**Top Selected Component Number $n$:** For a fair comparison, we set $K = 8, \beta = 0.1$ and vary $n$ on the Cora dataset. As shown in Figure 6(b), except for the case of $n = 1$, the other settings have comparable performance. However, it can be seen that when $n = 4$, the average accuracy on the last task is the highest, which indicates that the model has the strongest ability to avoid catastrophic forgetting problem when $n = 4$.

4.6 Efficiency Analysis

We show the training time of different continual learning methods on the biggest benchmark FB15k-237, so as to highlight the efficiency gap in different methods. For a fair comparison, all algorithms use the same KGE method TransE. As shown in Figure 7, GEM takes much longer training time compared with other baselines, which is pretty close to Upper Bound. Although our model also requires some previous data, it takes less time than GEM and EMR, which verifies the efficiency of our disentangled approach in continual learning.

4.7 Case Study

In this section, we visualize an example case from FB15k-237 dataset (more readable than other datasets) to show that the activated neighbors in our updating module are in line with human commonsense. For example, as shown in Figure 8(a) newly arrived relational triplets such as (Robert Clohessy, award_nominee, Jack Huston) and (Robert Clohessy, award_winner, Dominic Chianese), both related to “award” semantic aspects. Therefore,
only “award”-related neighbors of new triplets are updated, like \((\text{Jack Huston}, \text{nominated_for}, \text{Boardwalk Empire})\). since \text{Robert Clohessy} is also very likely to be related to the movie of \text{Boardwalk Empire}. Meanwhile, relational triples of \text{place_of_birth} and \text{gender}, which are not related to “award”, will not be updated.

Moreover, to verify the learned representation satisfies the intuition that different relations focus on different components of entities, we plot the attention values on the components of the entity \text{Britain} in Figure 8(b), where the y-coordinate is sampled relations that appear in the same triplets with “Britain”. We observe that semantically similar relations have similar attention value distributions. For example, relations “\text{gdp_nominal}”, “\text{gdp_real}”, “\text{dated_money}”, “\text{ppp_dollars}”, are all related to economics, relations “\text{olympic_medal}”, “\text{olympics}”, “\text{medal_won}” are all related to Olympics competitions. These results demonstrate that the disentangled representations learned by our DiCGRL are semantically meaningful.

5 Conclusion and Future Work

In this paper, we propose to study the problem of continual graph representation learning, aiming to handle the streaming nature of the emerging multi-relational data. To this end, we propose a disentangled-based continual graph representation learning (DiCGRL) framework, inspired by human’s ability to learn procedural knowledge. Extensive experiments on several typical KGE and NE datasets show that DiCGRL achieves consistent and significant improvement compared to existing continual learning models, which verifies the effectiveness of our model on alleviating the catastrophic forgetting problem. In the future, we will explore to extend the idea of disentanglement in the continual learning of other NLP tasks.

Acknowledgments

This work is supported by NSFC under Grant No. 61532001, National Key Research and Development Program of China under Grant No. 2018AAA0101902, and MOE-ChinaMobile Program under Grant No. MCM20170503.

References


James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu,


A Results under Non-Continual Learning Settings

Results under non-continual learning settings, i.e., using the entire training sets to train the models, are presented in Table 3 and Table 4. For a fair comparison, we also reproduce the results of baselines, and using the same optimal hyper-parameters as in Section Experimental Settings in the main paper. DiCGRL (T) and DiCGRL (C) indicate using ConvKB and TransE as feature extraction method for DiCGRL respectively. DiCGRL (·),αk represents that αk is calculated by performing a non-linearity transformation over the concatenation of u and v as doing in Network Embedding, and DiCGRL (·),αk represents that αk is calculated by performing a non-linearity transformation over the concatenation of u, r and v, since relations are an integral part of KGs.

From the tables, we can see that:

(1) DiCGRL is comparable with our reproduced baselines, especially on the WN18RR with ConvKB as feature extraction method and FB15k-237 with TransE as feature extraction method, our DiCGRL performs better than vanilla GE methods. This phenomenon shows the effectiveness of our disentangled approach by decoupling the relational triplets in the graph into multiple independent components according to their semantic aspects.

<table>
<thead>
<tr>
<th>Model</th>
<th>WN18RR</th>
<th>FB15k-237</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>H@10</td>
</tr>
<tr>
<td>TransE</td>
<td>22.6</td>
<td>50.1</td>
</tr>
<tr>
<td>TransE*</td>
<td>24.0</td>
<td>51.1</td>
</tr>
<tr>
<td>DiCGRL (T),α1</td>
<td>17.3</td>
<td>40.8</td>
</tr>
<tr>
<td>DiCGRL (T),α2</td>
<td>23.2</td>
<td>50.2</td>
</tr>
<tr>
<td>DiCGRL (T)</td>
<td>24.2</td>
<td>50.4</td>
</tr>
<tr>
<td>ConvKB</td>
<td>24.9</td>
<td>52.4</td>
</tr>
<tr>
<td>ConvKB*</td>
<td>40.4</td>
<td>53.6</td>
</tr>
<tr>
<td>DiCGRL (C),α1</td>
<td>38.6</td>
<td>50.6</td>
</tr>
<tr>
<td>DiCGRL (C),α2</td>
<td>39.2</td>
<td>49.4</td>
</tr>
<tr>
<td>DiCGRL (C)</td>
<td>45.2</td>
<td>52.9</td>
</tr>
</tbody>
</table>

Table 3: Link prediction results on whole WN18RR and FB15k-237. The best score is in bold. Results of TransE and ConvKB are taken from (Nguyen et al., 2018) and (Sun et al., 2019) respectively. * indicates reproduced by us.

<table>
<thead>
<tr>
<th></th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GATs</td>
<td>87.0</td>
<td>74.7</td>
<td>85.6</td>
</tr>
<tr>
<td>DiCGRL</td>
<td>88.1</td>
<td>75.1</td>
<td>85.9</td>
</tr>
</tbody>
</table>

Table 4: Node classification results on three whole information networks. The best score is in bold.

(2) As shown in Table 3, the performance of DiCGRL (·),α1 and DiCGRL (·),α2 are worse than DiCGRL, even worse than original baseline in some settings. This indicates that assigning global attention values for each relation as done in DiCGRL is an optimal option for KG datasets.

B PubMed Result

The results of DiCGRL on the PubMed data is shown in Figure 9.

![Figure 9: Changes in accuracy with increasing PubMed data through the continual learning process.](image)

C Dataset Statistics

The statistics of FB15k-237 and WN18RR are presented in Table 5, where “P_i” denotes the i-th part, “# Accumulated Entities” and “# Accumulated Relations” represent the cumulative entities and relations...
<table>
<thead>
<tr>
<th>Datasets</th>
<th>FB15k-237</th>
<th>WN18RR</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>P2</td>
<td>P3</td>
</tr>
<tr>
<td># Entities</td>
<td>11,632</td>
<td>727</td>
</tr>
<tr>
<td># Relations</td>
<td>236</td>
<td>236</td>
</tr>
<tr>
<td># Train Set</td>
<td>178,274</td>
<td>20,347</td>
</tr>
<tr>
<td># Validation Set</td>
<td>11,726</td>
<td>1,263</td>
</tr>
<tr>
<td># Test Set</td>
<td>13,681</td>
<td>1,556</td>
</tr>
<tr>
<td># Accumulated Entities</td>
<td>11,632</td>
<td>12,359</td>
</tr>
<tr>
<td># Accumulated Relations</td>
<td>236</td>
<td>236</td>
</tr>
</tbody>
</table>

Table 5: Statistics of knowledge graph datasets.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Cora (# Class = 7)</th>
<th>Citeseer (# Class = 6)</th>
<th>PubMed (# Class = 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>P2</td>
<td>P3</td>
<td>P4</td>
</tr>
<tr>
<td># Nodes</td>
<td>1,895</td>
<td>271</td>
<td>271</td>
</tr>
<tr>
<td># Edges</td>
<td>2,475</td>
<td>764</td>
<td>1,046</td>
</tr>
<tr>
<td># Train Set</td>
<td>568</td>
<td>81</td>
<td>81</td>
</tr>
<tr>
<td># Val Set</td>
<td>379</td>
<td>54</td>
<td>54</td>
</tr>
<tr>
<td># Test Set</td>
<td>948</td>
<td>136</td>
<td>136</td>
</tr>
<tr>
<td># Acc_Nodes</td>
<td>1,895</td>
<td>2,166</td>
<td>2,437</td>
</tr>
<tr>
<td># Acc_Edges</td>
<td>2,475</td>
<td>3,239</td>
<td>4,285</td>
</tr>
</tbody>
</table>

Table 6: Statistics of three citation datasets.

After each new part of multi-relational data is generated. Statistics of Cora, Citeseer, and PubMed are presented in Table 6, where “# Acc_Nodes” and “# Acc_Edges” represent the cumulative nodes and edges after each new part of multi-relational data is generated.