EDGE: Enriching Knowledge Graph Embeddings with External Text

Saed Rezayi[1], **Handong Zhao**[2], **Sungchul Kim**[2], **Ryan A. Rossi**[2],
Nedim Lipka[2], and **Sheng Li**[1]
[1]Department of Computer Science, University of Georgia, Athens, GA, USA
[2]Adobe Research, San Jose, CA, USA
{saedr,sheng.li}@uga.edu
{hazhao,sukim,ryrossi,lipka}@adobe.com

Abstract

Knowledge graphs suffer from sparsity which degrades the quality of representations generated by various methods. While there is an abundance of textual information throughout the web and many existing knowledge bases, aligning information across these diverse data sources remains a challenge in the literature. Previous work has partially addressed this issue by enriching knowledge graph entities based on *"hard"* co-occurrence of words present in the entities of the knowledge graphs and external text, while we achieve *"soft"* augmentation by proposing a knowledge graph enrichment and embedding framework named EDGE. Given an original knowledge graph, we first generate a rich but noisy augmented graph using external texts in semantic and structural level. To distill the relevant knowledge and suppress the introduced noise, we design a graph alignment term in a shared embedding space between the original and augmented graph. To enhance the embedding learning on the augmented graph, we further regularize the locality relationship of target entity based on negative sampling. Experimental results on four benchmark datasets demonstrate the robustness and effectiveness of EDGE in link prediction and node classification.

1 Introduction

Knowledge Graph (KG)[1] embedding learning has been an emerging research topic in natural language processing, which aims to learn a low dimensional latent vector for every node. One major challenge is sparsity. Knowledge graphs are often incomplete, and it is a challenge to generate low-dimensional representations from a graph with many missing edges. To mitigate this issue, auxiliary texts that are easily accessible have been popularly exploited for enhancing the KG (as illustrated in Figure 1). More specifically, given that KG entities contain textual features, we can link them to an auxiliary source of knowledge, *e.g.*, WordNet, and therefore enhance the existing feature space. With notable exceptions, the use of external textual properties for KG embedding has not been extensively explored before. Recently, (Kartsaklis et al., 2018) used entities of the KG to query BabelNet (Navigli and Ponzetto, 2012), added new nodes to the original KG based on co-occurrence of entities, and produced more meaningful embeddings using the enriched graph. However, this hard-coded, co-occurrence based KG enrichment strategy fails to make connections to other semantically related entities. As motivated in Figure 1, the newly added entities "*wound*", "*arthropod*" and "*protective body*", are semantically close to some input KG entity nodes (marked in red). However, they cannot be directly retrieved from BabelNet using co-occurrence matching.

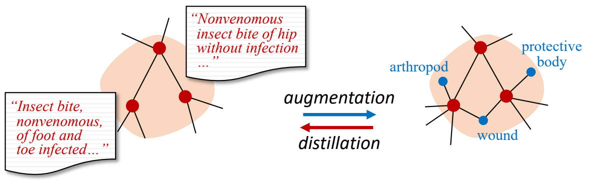

"Nonvenomous insect bite of hip without infection ..."
"Insect bite, nonvenomous, of foot and toe infected..."
augmentation
distillation
arthropod
protective body
wound

Figure 1: An example illustrating the original (left) and augmented knowledge graphs (right). Red nodes are knowledge graph entities and small blue nodes are textual nodes obtained from the external text. In augmentation process, a new set of keywords are discovered and attached to the original entities. To keep the augmented graph semantically close to the original graph, a backward pass of knowledge distillation is achieved by the proposed graph alignment.

In this paper, we aim to address the sparsity issue by integrating a learning component into the process. We propose a novel framework, EDGE, for KG enrichment and embedding. EDGE first constructs a graph using the external text based on

[1]Knowledge graph usually represents a heterogeneous multigraph whose nodes and relations can have different types. However in the work, we follow (Kartsaklis et al., 2018), consider knowledge graph enrichment problem where only one relation type (connected or not) appears.

similarity and aligns the enriched graph with the original KG in the same embedding space. It infuses learning in the knowledge distillation process by graph alignment, ensuring that similar entities remain close, and dissimilar entities get as far from each other. Consuming information from an auxiliary textual source helps improve the quality of final products, *i.e.*, low dimensional embeddings, by introducing new features. This new feature space is effective because it is obtained from a distinct knowledge source and established based on affinity captured by the learning component of our model.

More specifically, our framework takes $\mathcal{KG}$, and an external source of texts, $\mathcal{T}$, as inputs, and generates an augmented knowledge graph, $a\mathcal{KG}$. in generating $a\mathcal{KG}$ we are mindful of semantic and structural similarities among $\mathcal{KG}$ entities, and we make sure it contains all the original entities of $\mathcal{KG}$. This ensures that there are common nodes in two graphs which facilitates the alignment process. To align $\mathcal{KG}$ and $a\mathcal{KG}$ in the embedding space, a novel multi-criteria objective function is devised. In particular, we design a cost function that minimizes the distance between the embeddings of the two graphs. As a result, textual nodes (*e.g.*, blue nodes in Figure 1) related to each target entity are rewarded while unrelated ones get penalized in a negative sampling setting.

Extensive experimental results on four benchmark datasets demonstrate that EDGE outperforms state-of-the-art models in different tasks and scenarios, including link prediction and node classification. Evaluation results also confirm the generalizability of our model. We summarize our contributions as follows: (i) We propose EDGE, a general framework to enrich knowledge graphs and node embeddings by exploiting auxiliary knowledge sources. (ii) We introduce a procedure to generate an augmented knowledge graph from external texts, which is linked with the original knowledge graph. (iii) We propose a novel knowledge graph embedding approach that optimizes a multi-criteria objective function in an end-to-end fashion and aligns two knowledge graphs in a joint embedding space. (iv) We demonstrate the effectiveness and generalizability of EDGE by evaluating it on two tasks, namely link prediction and node classification, on four graph datasets.

The rest of the paper is organized as follows. In the next section, we try to identify the gap in the existing literature and motivate our work. Next, in Section 3, we set up the problem definition and describe how we approach the problem by in-depth explanation of our model. We evaluate our proposed model by experimenting link prediction and node classification on four benchmark datasets and present the results and ablation study in Section 4. Finally, we conclude our work and give the future direction in Section 5.

2 Related Work

Knowledge graph embedding learning has been studied extensively in the literature (Bordes et al., 2013; Wang et al., 2014; Yang et al., 2015; Sun et al., 2019; Zhang et al., 2019; Xian et al., 2020; Yan et al., 2020; Sheu and Li, 2020). A large number of them deal with the heterogeneous knowledge graph, where it appears different types of edges. While in this work we consider the type of knowledge graph with only one type (i.e. connected or not) of relation, and only focus on entity embedding learning. Our work is related to graph neural networks, such as the graph convolutional networks (GCN) (Kipf and Welling, 2017) and its variants (Wu et al., 2020; Jiang et al., 2019, 2020), which learn node embeddings by feature propagation. In the following, we mainly review the most relevant works in two aspects, i.e., graph embedding learning with external text and knowledge graph construction.

2.1 Graph Embedding with External Text

The most similar line of work to ours is where an external textual source is considered to enrich the graph and learn low dimensional graph embeddings using the enriched version of the knowledge graph. For instance, (Wang and Li, 2016) annotates the KG entities in text, creates a network based on entity-word co-occurrences, and then learns the enhanced KG. Similarly, (Kartsaklis et al., 2018) adds an edge (e, t) to KG per entity e based on co-occurrence and finds graph embeddings using random walks. However, there is no learning component in these approaches in constructing the new knowledge graph. And the enrichment procedure is solely based on occurrences (“hard" matching) of entities in the external text.

For graph completion task, (Malaviya et al., 2020) uses pre-trained language models to improve the representations and for Question Answering task, (Sun et al., 2018) extracts a sub-graph $\mathcal{G}_q$ from KG and Wikipedia, which contains the an-

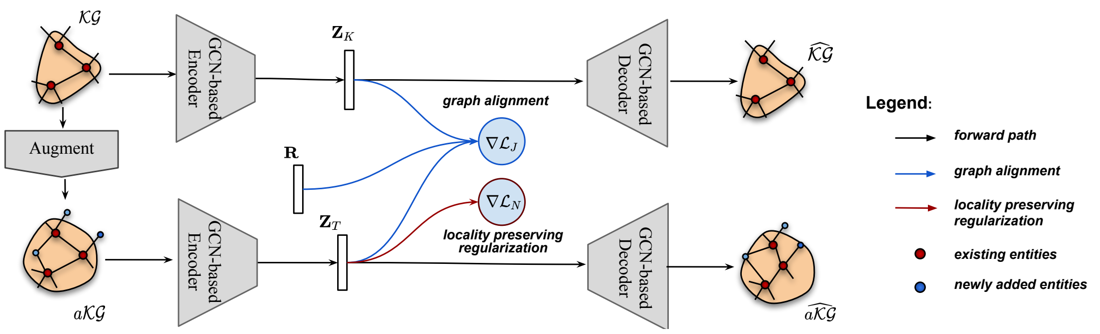


Figure 2: Our proposed framework for aligning two graphs in the embedding space. The graph alignment component, $\mathcal{L}_J$, requires an additional matrix, $\mathbf{R}$, that selects embeddings of $\mathcal{KG}$ entities from $\mathbf{Z}_T$, so the resulting matrix, $\mathbf{RZ}_T$, would have the same dimension as $\mathbf{Z}_K$. Furthermore, $\mathcal{L}_N$ penalizes additional entities that are unrelated to the target entity, while rewards the related ones. We omit the graph reconstruction loss for simplicity.

swer to the question with a high probability and apply GCN on $\mathcal{G}_q$ which is limited to a specific task. We emphasize that the main difference between our model and previous work is that we first create an augmented knowledge graph from an external source, and improve the quality of node representation by jointly mapping two graphs to an embedding space. To the best of our knowledge, this is the first time that a learning component is incorporated to enriching knowledge graphs.

2.2 Knowledge Graph Construction

Knowledge graph construction methods are broadly classified into two main groups: 1) Curated approaches where facts are generated manually by experts, *e.g.*, WordNet (Fellbaum, 1998) and UMLS (Bodenreider, 2004), or volunteers such as Wikipedia, and 2) Automated approaches where facts are extracted from semi-structured text like DBpedia (Auer et al., 2007), or unstructured text (Carlson et al., 2010). The latter approach can be defined as extracting structured information from unstructured text. In this work, we do not intend to construct a knowledge base from scratch, instead we aim to generate an augmented knowledge graph using side information. Hence, we employ existing tools to acquire a set of new facts from external text and link them to an existing KG.

3 Proposed Model

3.1 Problem Statement

We formulate the knowledge graph enrichment and embedding problem as follows: given a knowledge graph $\mathcal{KG} = (\mathcal{E}, \mathcal{R}, X)$ with $|\mathcal{E}|$ nodes (or entities), $|\mathcal{R}|$ edges (or relations) and $X \in \mathbb{R}^{|\mathcal{E}| \times D}$ as feature matrix, where D is the number of features per entity, also given an external textual source, $\mathcal{T}$, the goal is to generate an augmented knowledge graph and jointly learn d ($d << |\mathcal{E}|$) dimensional embeddings for knowledge graph entities, which preserve structural and semantic properties of the knowledge graph. The learned representations are then used for the tasks of link prediction and node classification. Link prediction is defined as a binary classification whose goal is to predict whether or not an edge exists in KG, and node classification is the task of determining node labels in labelled graphs.

To address the problem of knowledge graph enrichment and embedding, we propose EDGE, a framework that contains two major components, *i.e.*, augmented knowledge graph construction, and knowledge graph alignment in a joint embedding space.

3.2 Augmented Knowledge Graph Construction

Given the entities of $\mathcal{KG}$ and an external source of textual data, $\mathcal{T}$, we aim to generate an augmented graph, $a\mathcal{KG}$, which is a supergraph of $\mathcal{KG}$ (*i.e.*, $\mathcal{KG}$ is a subgraph of $a\mathcal{KG}$). Augmentation is the process of adding new entities to $\mathcal{KG}$. These newly added entities are called *textual entities* or *textual nodes*. A crucial property of $a\mathcal{KG}$ is that it contains entities of $\mathcal{KG}$. The presence of these entities establishes a relationship between the two graphs, and such a relationship will be leveraged to learn the shared graph embeddings. To construct $a\mathcal{KG}$, we need to find a set of keywords to query an external source, To obtain high quality keywords and acquire new textual entities, we design the following procedure per target entity e_t (For every step of this process refer to Table 1 for a real example from SNOMED dataset).

First, we find a set of semantically and struc-

Table 1: We employ representation learning algorithms to find a set of semantically and structurally similar entities to each target entity (column 2). We then find a set of keywords, K, that are representative of the target entity (column 3) and use them to query an external text and obtain a set of sentences, S (column 4). Finally, we extract textual entities (column 5), and connect them to the target entity.

Target Entity		semantically and structurally Similar Entities	Most definitive keywords	Sentences obtained from auxiliary text	Entities obtained from information extraction
Nonvenomous insect bite of hip without infection	semantic	1. Nonvenomous insect bite of foot with infection 2. Crushing injury of hip and/or thigh 3. Superficial injury of lip with infection 4. Infected insect bite of hand	1. bite 2. insect 3. nonvenomous 4. infect	1. a wound resulting from biting by an animal or a person 2. small air-breathing arthropod 3. not producing or resulting from poison 4. contaminate with a disease or microorganism	1. wound 2. arthropod 3. poison 4. microorganism
	structural	1. Insect bite, nonvenomous, of back 2. Tick bite 3. Animal bite of calf 4. Inset bite, nonvenomous, of foot and toe			
Insect bite, nonvenomous, of foot and toe infected	semantic	1. Insect bite, nonvenomous, of lower limb, infected 2. Infected insect bite of hand 3. Insect bite, nonvenomous, of hip 4. Insect bite granuloma	1. bite 2. insect 3. lower 4. skin	1. a wound resulting from biting by an animal or a person 2. small air-breathing arthropod 3. move something or somebody to a lower position 4. a natural protective body	1. wound 2. arthropod 3. position 4. protective body
	structural	1. Nonvenomous insect bite of hip without infection 2. Insect bite, nonvenomous, of back 3. Recurrent infection of skin 4. Skin structure of lower leg			

turally similar entities to e_t denoted by $\mathcal{E}_{e_t}$. This set creates a textual context around e_t which we use to find keywords to query an external text, *e.g.*, WordNet or Wikipedia. Here by *query* we mean using the API of the external text to find related sentences, S (for instance for a given keyword "bite" we can capture several sentences from the wikipedia page for the entry "biting" or find several Synsets[2] from WordNet when we search for "bite").

Finally, we extract entities from S and attach them to e_t. We call these new entities, *textual entities* or *textual features*. By connecting these newly found textual entities to the e_t, we enhance $\mathcal{KG}$ and generate the augmented knowledge graph, $a\mathcal{KG}$. We observed that the new textual entities are different from our initial feature space. Also, it is possible that two different target entities share one or more textual nodes, hence the distance between them in $a\mathcal{KG}$ would decrease. The implementation details of this process is provided in Supplementary materials.

Querying an external text allows us to extend the feature space beyond the context around e_t. By finding other entities in $\mathcal{KG}$ that are similar to the target entity and extracting keywords from the collection of them to query the external text, distant entities that are related but not connected would become closer to each other owing to the shared keywords.

Figure 1 illustrates a subset of SNOMED graph and its augmented counterpart by following the above procedure. As this figure reveals, the structure of $a\mathcal{KG}$ is different from $\mathcal{KG}$, and as a result of added textual nodes, distant but similar entities would become closer. Therefore, augmenting knowledge graphs would alleviate the KG sparsity issue. Although we may introduce noise by adding new entities but later in the alignment process we address this issue.

Remarks. In the above procedure, we need to obtain similar entities before looking for textual entities, and the rationality of such a strategy is discussed as follows. One naive approach is to simply use keywords included in the target entity to find new textual features. In this way, we would end up with textual features that are related to that target entity, but we cannot extend the feature space to capture similarity (*i.e.*, dependency) among entities.

3.3 Knowledge Graph Alignment in Joint Embedding Space

With the help of augmented knowledge graph $a\mathcal{KG}$, we aim to enrich the graph embeddings of $\mathcal{KG}$. However, inevitably, a portion of newly added entities are noisy, and even potentially wrong. To mitigate this issue, we are inspired by Hinton et al. (Hinton et al., 2015), and propose a graph alignment process for knowledge distillation. In fact, $a\mathcal{KG}$ and $\mathcal{KG}$ share some common entities, which makes it possible to map two knowledge graphs into a joint embedding space. In particular, we propose to extract low-dimensional node embeddings of two knowledge graphs using graph auto-encoders (Kipf and Welling, 2016), and design novel constraints to align two graphs in the embedding space. The architecture of our approach is illustrated in Figure 2.

Let $\mathbf{A}_K$ and $\mathbf{A}_T$ denote the adjacency matrices of $\mathcal{KG}$ and $a\mathcal{KG}$, respectively. The loss functions

[2]Synset is the fundamental building block of WordNet which is accompanied by a definition, example(s), etc.

of graph auto-encoders that reconstruct knowledge graphs are defined as:

$$\mathcal{L}_K = \min_{\mathbf{Z}_K} ||\mathbf{A}_K - \hat{\mathbf{A}}_K||_2, \quad (1)$$

$$\mathcal{L}_T = \min_{\mathbf{Z}_T} ||\mathbf{A}_T - \hat{\mathbf{A}}_T||_2, \quad (2)$$

where $\hat{\mathbf{A}}_K = \sigma(\mathbf{Z}_K \mathbf{Z}_K^\top)$ is the reconstructed graph using node embeddings $\mathbf{Z}_K$. And $\mathbf{Z}_K$ is the output of graph encoder that is implemented by a two-layer GCN (Kipf and Welling, 2016):

$$\mathbf{Z}_K = \mathrm{GCN}(\mathbf{A}_K, \mathbf{X}_K) = \tilde{\mathbf{A}_K} \tanh(\tilde{\mathbf{A}}_K \mathbf{X}_K \mathbf{W}_0)\mathbf{W}_1, \quad (3)$$

where $\tilde{\mathbf{A}}_K = \mathbf{D}_K^{-\frac{1}{2}} \mathbf{A}_K \mathbf{D}_K^{-\frac{1}{2}}$. $\mathbf{D}_K$ is the degree matrix, $\tanh(.)$ is the Hyperbolic Tangent function that acts as the activation function of the neurons, $\mathbf{W}_i$ are the model parameters, and $\mathbf{X}_K$ is the feature matrix.[3] Similarly, $\hat{\mathbf{A}}_T = \sigma(\mathbf{Z}_T \mathbf{Z}_T^\top)$, and $\mathbf{Z}_T$ is learned by another two-layer GCN. Equations (1) and (2) are l_2-*norm* based loss functions that aim to minimize the distance between original graphs and the reconstructed graphs.

Furthermore, to map $\mathcal{KG}$ and $a\mathcal{KG}$ to a joint embedding space and align their embeddings through common entities, we define the following graph alignment loss function:

$$\mathcal{L}_J = ||\mathbf{Z}_K - \mathbf{R}\mathbf{Z}_T||_2, \quad (4)$$

where $\mathbf{R}$ is a transform matrix that selects common entities that exist in $\mathcal{KG}$ and $a\mathcal{KG}$. Note that the two terms $\mathbf{Z}_K$ and $\mathbf{R}\mathbf{Z}_T$ should be of the same size in the L_2 norm equation. Our motivation is to align the embeddings of common entities across two knowledge graphs. By using $\mathbf{R}$, the node embeddings of common entities can be selected from $\mathbf{Z}_T$. Note that $\mathbf{Z}_T$ is always larger than $\mathbf{Z}_K$, as $\mathcal{KG}$ is a subgraph of $a\mathcal{KG}$. Equation (4) also helps preserve local structures of the original knowledge graph $\mathcal{KG}$ in the graph embedding space. In other words, nodes that are close to each other in the original knowledge graph will be neighbors in the augmented graph as well.

Moreover, we notice that the proposed augmented knowledge graph $a\mathcal{KG}$ involves more complicated structures than the original knowledge graph $\mathcal{KG}$, due to the newly added textual nodes for each target entity in $\mathcal{KG}$. In $a\mathcal{KG}$, one target entity is closely connected to its textual nodes, and their embeddings should be very close to each other in the graph embedding space. However, such local structures might be distorted in the graph embedding space. Without proper constraints, it is possible that one target entity is close to textual entities of other target entities in the embedding space, which is undesired for downstream applications. To address this issue, we design a margin-based loss function with negative sampling to preserve the locality relationship as follows:

[3]In case of a featureless graph, an identity matrix, $\mathbf{I}$, replaces $\mathbf{X}_K$.

Algorithm 1 Training process of EDGE

```
Input: $\mathbf{A}_K, \mathbf{X}_K, \mathbf{A}_T, \mathbf{X}_T$, POS, NEG,
Input: $\mathbf{R} \in \mathbb{R}^{|\mathcal{E}_K| \times (|\mathcal{E}_T| - |\mathcal{E}_K|)}$
 1: for each epoch do
 2:     $\hat{\mathbf{A}}_K = \sigma(\mathbf{Z}_K \mathbf{Z}_K^\top)$
 3:     $\mathbf{Z}_K = \tilde{\mathbf{A}}_K \tanh(\tilde{\mathbf{A}}_K \mathbf{X}_K \mathbf{W}_0^K)\mathbf{W}_1^K$
 4:     $\hat{\mathbf{A}}_T = \sigma(\mathbf{Z}_T \mathbf{Z}_T^\top)$
 5:     $\mathbf{Z}_T = \tilde{\mathbf{A}}_T \tanh(\tilde{\mathbf{A}}_T \mathbf{X}_T \mathbf{W}_0^T)\mathbf{W}_1^T$
 6:     Calculate $\mathcal{L}_K$ and $\mathcal{L}_T$ using Equations (1) and (2).
 7:     Compute $\mathcal{L}_J$ using Equation (4)
 8:     Find negative and positive samples and calculate $\mathcal{L}_N$
        using Equation (5)
 9:     Sum up all losses with their corresponding ratios
        using Equation (6)
10:     Run Adam optimizer to minimize $\mathcal{L}$
11:     Update model parameters $\mathbf{W}_i^K$ and $\mathbf{W}_i^T$
12: end for
Output: $Z_K$
```

$$\mathcal{L}_N = -\log(\sigma(\mathbf{z}_e^\top \mathbf{z}_t)) - \log(\sigma(-\mathbf{z}_e^\top \mathbf{z}_{t'})), \quad (5)$$

where $\mathbf{z}_t$ are the embeddings of the related textual nodes, $\mathbf{z}_t'$ are the embeddings of textual nodes that are not related to the target entity, and σ is the *sigmoid* function.

Finally, the overall loss function is defined as:

$$\mathcal{L} = \min_{\mathbf{Z}_K, \mathbf{Z}_T} \underbrace{\mathcal{L}_K + \alpha\mathcal{L}_T}_{\text{reconstruction loss}} + \underbrace{\beta\mathcal{L}_J}_{\text{graph alignment}} + \underbrace{\gamma\mathcal{L}_N}_{\text{locality preserving}}, \quad (6)$$

where α, β, and γ are hyper-parameters. We perform full-batch gradient descent using the Adam optimizer to learn all the model parameters in an end-to-end fashion. The whole training process of our approach is summarized in Algorithm 1.

The learned low-dimensional node embeddings $\mathbf{Z}_K$ could benefit a number of unsupervised and supervised downstream applications, such as link prediction and node classification. Link prediction is the task of inferring missing links in a graph, and node classification is the task of predicting labels to vertices of a (partially) labeled graph. Extensive evaluations on both tasks will be provided in the experiment section.

Table 2: Link prediction results for SNOMED and three citation networks. Numbers for SNOMED are obtained from rerunning their code on the dataset. The rest of the results are reported from corresponding papers.

Model	SNOMED		Cora		Citeseer		PubMed	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
GAE (Kipf and Welling, 2016)	0.773	0.844	0.914	0.926	0.908	0.920	0.964	0.965
LoNGAE (Tran, 2018)	0.890	0.910	0.954	0.963	0.953	0.961	0.960	0.963
ARVGE (Pan et al., 2018)	0.805	0.864	0.924	0.926	0.924	0.930	0.968	0.971
SCAT (Zou and Lerman, 2019)	0.902	0.918	0.945	0.946	0.973	**0.976**	**0.975**	**0.972**
GIC (Mavromatis and Karypis, 2020)	-	-	0.935	0.933	0.970	0.968	0.937	0.935
EDGE (This work)	**0.916**	**0.944**	**0.973**	**0.975**	**0.974**	**0.976**	0.969	0.968

3.4 Model Discussions

We have proposed a general framework for graph enrichment and embedding by exploiting auxiliary knowledge sources. What we consider as a source of knowledge is a textual knowledge base that can provide additional information about the entities of the original knowledge graph. It is a secondary source of knowledge that supplies new sets of features outside of the existing feature space, which improves the quality of representations.

The proposed graph alignment approach can fully exploit augmented knowledge graph and thus improve the graph embeddings. Although $a\mathcal{KG}$ is a supergraph of $\mathcal{KG}$, its connectivity pattern is different. With the help of our customized loss function for graph alignment, both graphs contribute in the quality of derived embeddings. We will also demonstrate the superiority of our joint embedding approach over the independent graph embedding approach (with only $a\mathcal{KG}$) in the experiments, and we investigate which component of our model contributes more in the final performance in the ablation study in Subsection 4.4.

4 Experiment

We design our experiments to investigate effectiveness of different components of EDGE as well as its overall performance. To this end, we aim to answer the following three questions[4].

$Q1$ How well does EDGE perform compared to state-of-the-art in the task of link prediction? (Section 4.1)

$Q2$ How is the quality of embeddings generated by EDGE compared to similar methods? (Sections 4.2 and 4.3)

$Q3$ What is the contribution of each component (augmentation and alignment) in the overall performance? (Section 4.4)

[4]We plan to release our code upon publication.

4.1 Task 1: Link Prediction

To investigate Q1 we perform link prediction on four benchmark datasets, and compare the performance of our model with five relevant baselines. For this task we consider SNOMED and three citation networks. For SNOMED, similar to (Kartsaklis et al., 2018), we select 21K medical concepts from the original dataset. Each entity in SNOMED is a text description of a medical concept, *e.g.*, *Nonvenomous insect bite of hip without infection*. According to the procedure explained in subsection 3.2, we construct an augmented knowledge graph, $a\mathcal{KG}$. Additionally, we consider three other datasets, namely Cora, Citeseer, and PubMed, which are citation networks consisting of 2,708, 3,312, and 19,717 papers, respectively. In all three datasets, a short text accompanies each node which is extracted from the title or abstract of the paper. For these networks, *relation* is defined as citation and the textual content of the nodes enables us to obtain $a\mathcal{KG}$. Cora and Citeseer datasets come with a set of default features. We defer the detailed description of datasets in the supplementary.

In this experiment, for each dataset, we train the model on 85% of the input graph. Other 15% of the data is split into 5% validation set and 10% as part of the test set (positive samples only). An additional set of edges are produced, equal to the number of positive samples, which does not exist in the graph, as negative samples. The union of positive and negative samples are used as the test set. In all baselines, we test the model on $\mathcal{KG}$. We obtain the following values for loss ratios after hyper-parameter tuning: $\alpha = 0.001$, $\beta = 10$, $\gamma = 1$. We discuss parameter tuning and explain the small value of α in Section 4.5.

We provide comparison against VGAE (Kipf and Welling, 2016) and its adversarial variant ARVGE (Pan et al., 2018). Also we consider LoNGAE (Tran, 2018), SCAT (Zou and Lerman, 2019) and

Table 3: Node classification results in terms of accuracy for citation networks. TR stands for training ratio and *un.* and *semi.* are short for unsupervised and semi-supervised.

Model	Approach	Cora TR=0.5	Citeseer TR=0.03	PubMed TR=0.003
DeepWalk	*un.*	0.67	0.43	0.65
GCN	*semi.*	0.81	0.70	0.79
GAT	*semi.*	**0.83**	**0.72**	0.79
LoNGAE	*semi.*	0.78	0.71	0.79
MixHop	*semi.*	0.82	0.71	**0.81**
EDGE	*un.*	0.81	0.66	0.76

GIC (Mavromatis and Karypis, 2020) which are designed for link prediction task on graphs, hence they make strong baselines. Table 2 presents the Area Under the ROC Curve (AUC) and average precision (AP) scores for five baselines and our methods across all datasets. We observe that EDGE outperforms all baselines in three out of four datasets and produces comparable results for PubMed dataset.

4.2 Task 2: Node Classification on Citation Networks

To evaluate the quality of embeddings (Q2) we design a node classification task based on the final product of our model. For this task, we use Cora, Citeseer and PubMed datasets, and follow the same procedure explained in 3.2 to generate $a\mathcal{KG}$ and jointly map the two graphs into an embedding space. All the settings are identical to Task 1. To perform node classification, we use the final product of our model, which is a 160 dimensional vector per node. We train a linear SVM classifier and obtain the accuracy measure to compare the performance of our model with state-of-the-art methods. Training ratio varies across different datasets, and we consider several baselines to compare our results against.

We compare our approach with state-of-the-art semi-supervised models for node classification, including GCN (Kipf and Welling, 2017), GAT (Veličković et al., 2018), LoNGAE (Tran, 2018), and MixHop (Abu-El-Haija et al., 2019). These models are semi-supervised, thus they were exposed to node labels during training while our approach is completely unsupervised. We also include DeepWalk, an unsupervised approach, to have a more complete view for our comparison. Table 3 reveals that our model achieves reasonable performance compared with semi-supervised models in two out of three datasets. Since EDGE

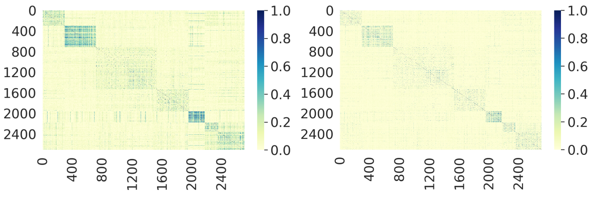


Figure 3: Pair-wise similarity comparison between GAE and EDGE.

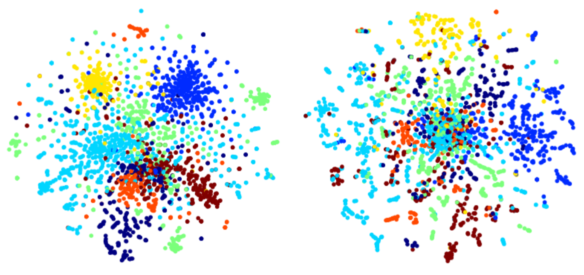


Figure 4: Visualization of embedding vectors on Cora: a) with and b) without features to study the effect of features on quality of embeddings in node classification.

is fully unsupervised, it is fair to declare that its performance is comparable as other methods are exposed to more information (*i.e.*, node labels).

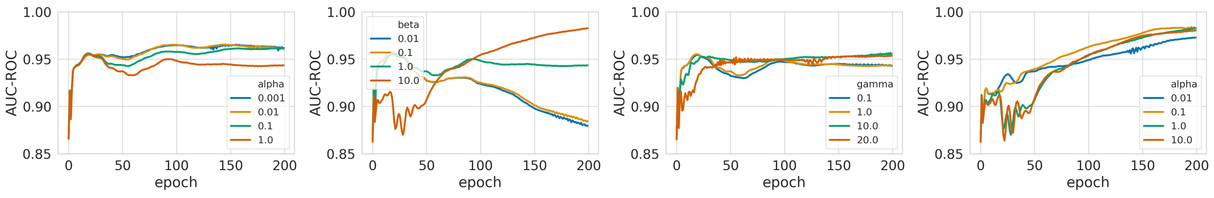
4.3 Embedding Effectiveness

Further, to measure the quality of embeddings produced by our model and compare it against the baseline, we visualize the similarity matrix of node embeddings for two scenarios on the Cora dataset: 1) GAE on $\mathcal{KG}$, and 2) EDGE on $\mathcal{KG}$ and $a\mathcal{KG}$. The results are illustrated in Figure 3. In this heatmap, elements are pair-wise similarity values sorted by different labels (7 classes). We can observe that the block-diagonal structure learned by our approach is clearer than that of GAE, indicating enhanced separability between different classes.

Next, we examine our model in more details and study how different parameters affect its performance.

4.4 Ablation Study

To investigate the effectiveness of different modules of our model (Q3), we consider two scenarios. First we use a single graph to train our model. Note that when we use a single graph, the graph alignment and locality preserving losses are discarded and our model is reduced to GAE. In single graph scenario we consider two versions of augmented graph, $a\mathcal{KG}$ that was explained in subsection 3.2



(a) Cora $\beta = 1, \gamma = 1$ (b) Cora $\alpha = 1, \gamma = 1$ (c) Cora $\alpha = 1, \beta = 1$ (d) Cora $\beta = 10, \gamma = 1$

Figure 5: Effect of parameterization on link prediction performance

Table 4: Link prediction results for SNOMED dataset. In this table $a\mathcal{KG}^*$ is the augmented knowledge graph generated using the method explained in (Kartsaklis et al., 2018)

Input	Model	AUC	AP
$\mathcal{KG}$	GAE (Kipf and Welling, 2016)	0.77	0.84
$a\mathcal{KG}^*$	GAE (Kipf and Welling, 2016)	0.85	0.88
$a\mathcal{KG}$	GAE (Kipf and Welling, 2016)	0.86	0.90
$\mathcal{KG} + a\mathcal{KG}^*$	EDGE (This work)	0.90	0.93
$\mathcal{KG} + a\mathcal{KG}$	EDGE (This work)	**0.91**	**0.94**

Table 5: Node classification results in terms of accuracy for citation networks. TR stands for training ratio and $a\mathcal{KG}^*$ is an augmented knowledge graph produced by the method proposed in (Kartsaklis et al., 2018)

Input	Model	Cora TR=0.5	Citeseer TR=0.03	PubMed TR=0.003
$\mathcal{KG}$	GAE	0.62	0.51	0.60
$a\mathcal{KG}^*$	GAE	0.70	0.57	0.65
$a\mathcal{KG}$	GAE	0.75	0.61	0.67
$\mathcal{KG} + a\mathcal{KG}^*$	EDGE	0.80	0.64	0.73
$\mathcal{KG} + a\mathcal{KG}$	EDGE	**0.81**	**0.66**	**0.76**

and $a\mathcal{KG}^*$ that was created based on co-occurrence proposed by (Kartsaklis et al., 2018). In the second scenario, we use two graphs to jointly train EDGE, and we feed our model with $\mathcal{KG} + a\mathcal{KG}^*$ and $\mathcal{KG} + a\mathcal{KG}$ to show the effect of augmentation.

For link prediction we only consider SNOMED dataset which is the largest dataset, and as Table 4 presents we observe that our augmentation process is slightly more effective than co-occurrence based augmentation. More importantly, by comparing second two rows with first two rows we realize that alignment module improves the performance more than augmentation process which highlights the importance of our proposed joint learning method. Moreover, we repeat this exercise for node classification (see Table 5) which results in a similar trend across all datasets.

Finally, we plot the t-SNE visualization of embedding vectors of our model with and without features. Figure 4 clearly illustrates the distinction between quality of the clusters for the two approaches. This implies that knowledge graph text carries useful information. When the text is incorporated into the model, it can help improve the model performance.

4.5 Parameter Sensitivity

We evaluate the parameterization of EDGE, and specifically we examine how changes to hyper parameters of our loss function (*i.e.*, α, β and γ) could affect the model performance in the task of link prediction on Cora dataset. In each analysis, we fix the values of two out of three parameters and study the effect of the variation of the third parameter on evaluating AUC scores across 200 epochs. The detailed results are shown in Figure 5.

Figure 5a shows the effect of varying α, when $\beta = 1$ and $\gamma = 1$ are fixed. We observe a somewhat consistent trend across performance for different values of α. It is evident that decreasing α improves the performance. α is the coefficient of $\mathcal{L}_T$ (see Equation 2). This examination suggests that the effect of this loss function is less significant, because we re-address it in the $\mathcal{L}_N$ part of the loss function, where we consider the same graph ($a\mathcal{KG}$) and try to optimize distance between its nodes but with more constraints.

Figure 5b illustrates the effect of varying β, while $\alpha = 1$ and $\gamma = 1$ are fixed. Tuning β results in more radical changes in the model performance, which is again consistent between the two datasets. Small values for β degrades performance remarkably, and we observe a much more improved AUC score for larger values of β. This implies the dominant effect of the joint loss function, $\mathcal{L}_J$, which is defined as the distance between corresponding entities of $\mathcal{KG}$ and $a\mathcal{KG}$.

Next, we fix $\alpha = 1$ and $\beta = 1$ and tweak γ from 0.1 to 10. As Figure 5c reveals, the variation in performance is very small. Finally, as we obtained the best results when $\beta = 10$, we set $\gamma = 1$ and once again tune α. Figure 5d shows the results for this updated setting. These experiments confirm the insignificance of parameter α. In practice, we obtained the best results by setting α to 0.001.

5 Conclusion

Sparsity is a major challenge in KG embedding, and many studies failed to properly address this issue. We proposed EDGE, a novel framework to enrich KG and align the enriched version with the original one with the help of auxiliary text. Using external source of information introduces new sets of features that enhance the quality of embeddings. We applied our model on three citation networks and one large scale medical knowledge graph. Experimental results show that our approach outperforms existing graph embedding methods on link prediction and node classification.

Acknowledgment

This research is supported in part by the U.S. Army Research Office Award under Grant Number W911NF-21-1-0109, and a gift funding from Adobe Research.

References

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *Proceedings of the 36th International Conference on Machine Learning*, pages 21–29.

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. Springer, Berlin, Heidelberg.

Olivier Bodenreider. 2004. The unified medical language system (umls): integrating biomedical terminology. *Nucleic acids research*, 32(suppl_1):D267–D270.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems*, pages 2787–2795.

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 *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Xiaodong Jiang, Pengsheng Ji, and Sheng Li. 2019. Censnet: Convolution with edge-node switching in graph neural networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 2656–2662.

Xiaodong Jiang, Ronghang Zhu, Pengsheng Ji, and Sheng Li. 2020. Co-embedding of nodes and edges with graph neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Dimitri Kartsaklis, Mohammad Taher Pilehvar, and Nigel Collier. 2018. Mapping text to knowledge graph entities using multi-sense lstms. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1959–1970.

Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. In *Bayesian Deep Learning Workshop (NIPS 2016)*.

Thomas N Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*.

Chaitanya Malaviya, Chandra Bhagavatula, Antoine Bosselut, and Yejin Choi. 2020. Commonsense knowledge base completion with structural and semantic context. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*.

Costas Mavromatis and George Karypis. 2020. Graph infoclust: Leveraging cluster-level node information for unsupervised graph representation learning. *arXiv preprint arXiv:2009.06946*.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, Lina Yao, and Chengqi Zhang. 2018. Adversarially regularized graph autoencoder for graph embedding. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 2609–2615. AAAI Press.

Heng-Shiou Sheu and Sheng Li. 2020. Context-aware graph embedding for session-based news recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 657–662.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4231–4242.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*.

Phi Vu Tran. 2018. Learning to make predictions on graphs with autoencoders. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 237–245. IEEE.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *International Conference on Learning Representations*.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119. AAAI Press.

Zhigang Wang and Juanzi Li. 2016. Text-enhanced representation learning for knowledge graph. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 1293–1299. AAAI Press.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*.

Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard de Melo, S. Muthukrishnan, and Yongfeng Zhang. 2020. CAFE: coarse-to-fine neural symbolic reasoning for explainable recommendation. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1645–1654. ACM.

Jun Yan, Mrigank Raman, Tianyu Zhang, Ryan A. Rossi, Handong Zhao, Sungchul Kim, Nedim Lipka, and Xiang Ren. 2020. Learning contextualized knowledge structures for commonsense reasoning. *CoRR*, abs/2010.12873.

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 *International Conference on Learning Representations*.

Wen Zhang, Bibek Paudel, Liang Wang, Jiaoyan Chen, Hai Zhu, Wei Zhang, Abraham Bernstein, and Huajun Chen. 2019. Iteratively learning embeddings and rules for knowledge graph reasoning. In *The World Wide Web Conference*, pages 2366–2377. ACM.

Dongmian Zou and Gilad Lerman. 2019. Encoding robust representation for graph generation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.