Sampled in Pairs and Driven by Text: A New Graph Embedding Framework

Liheng Chen
Shanghai Jiao Tong University
lhchen@apex.sjtu.edu.cn

Yanru Qu
Shanghai Jiao Tong University
kevinqu@apex.sjtu.edu.cn

Zhenghui Wang
Shanghai Jiao Tong University
felixwzh@apex.sjtu.edu.cn

Weinan Zhang
Shanghai Jiao Tong University
wnzhang@apex.sjtu.edu.cn

Ken Chen
Synyi LLC.
chen.ken@synyi.com

Shaodian Zhang
Synyi LLC.
shaodian@apex.sjtu.edu.cn

Yong Yu
Shanghai Jiao Tong University
yyu@apex.sjtu.edu.cn

ABSTRACT

In graphs with rich texts, incorporating textual information with structural information would benefit constructing expressive graph embeddings. Among various graph embedding models, random walk (RW)-based is one of the most popular and successful groups. However, it is challenged by two issues when applied on graphs with rich texts: (i) *sampling efficiency*: deriving from the training objective of RW-based models (e.g., DeepWalk and node2vec), we show that RW-based models are likely to generate large amounts of redundant training samples due to three main drawbacks. (ii) *text utilization*: these models have difficulty in dealing with zero-shot scenarios where graph embedding models have to infer graph structures directly from texts. To solve these problems, we propose a novel framework, namely Text-driven Graph Embedding with Pairs Sampling (TGE-PS). TGE-PS uses Pairs Sampling (PS) to improve the sampling strategy of RW, being able to reduce ∼99% training samples while preserving competitive performance. TGE-PS uses Text-driven Graph Embedding (TGE), an inductive graph embedding approach, to generate node embeddings from texts. Since each node contains rich texts, TGE is able to generate high-quality embeddings and provide reasonable predictions on existence of links to unseen nodes. We evaluate TGE-PS on several real-world datasets, and experiment results demonstrate that TGE-PS produces state-of-the-art results on both traditional and zero-shot link prediction tasks.

CCS CONCEPTS

• **Computing methodologies** → *Learning latent representations*;
• **Information systems** → *Information retrieval.*

KEYWORDS

Graph Embedding, Data Mining, Link Prediction, Zero-shot

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313520

ACM Reference Format:
Liheng Chen, Yanru Qu, Zhenghui Wang, Weinan Zhang, Ken Chen, Shaodian Zhang, and Yong Yu. 2019. Sampled in Pairs and Driven by Text: A New Graph Embedding Framework. In *Proceedings of the 2019 World Wide Web Conference (WWW '19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3308558.3313520

1 INTRODUCTION

Graph provides a fundamental tool to represent interconnected entities (e.g., articles, diseases) and their attributes (e.g., entity description). Graphs with rich text information are ubiquitous in many fields [18], and there is often a strong dependency between graph structure and text structure in these graphs. Textural information may also expose structural information [56]. Hence, it is promising to better utilize textual information in graphs.

Graph embedding is famous for its efficient representations for entities in graphs [16, 38]. A series of models are proposed to maximize edge reconstruction probability with different proximities [5], e.g., LINE [45], DeepWalk [42] and node2vec [17]. Although these models are widely used in mapping nodes to low-dimensional dense vectors, they are not well designed for graphs with rich text information. To solve this problem, models like DDRW [26], GENE [8], PPNE [25] and Tri-DNR [40] focus on preserving vertex labels. Especially, TADW [53], CANE [47] and Paper2Vec [14] are proposed to utilize textural information with effectiveness in many scenarios.

However, these graph embedding models still need to resolve two issues. The first issue is *sampling efficiency*. The optimization goals of these models can be summarized as maximizing pairwise node similarity, thus the number of training node pairs is critical to training time and can be used as a metric of sampling efficiency. These models usually use edges as training node pairs directly like LINE, or sample training sequences using random walk (RW) and extract node pairs within a specific shortest distance like DeepWalk and node2vec. RW generalizes the idea of directly sampling edges and shows better experimental performance under similar settings. However, our theoretical and empirical analyses show that RW samples redundant node pairs which severely lags efficiency.

The second issue is *text utilization*. Text utilization models like TADW and CANE mainly rely on training node embeddings (NE)

and text embeddings (TE) together (denoted as NE+TE). In zero-shot settings, where link existence with previously unseen nodes is to be predicted and the only reliable information is texts attributed to nodes , humans can make inference by assuming textual connections even if he has little prior knowledge about this field. However, NE+TE is incapable of dealing with this scenario due to two drawbacks: (i) it requires both NE and TE to generate high-quality representations, but NE is missing for unseen nodes, and (ii) even if it uses only TE part to predict, empirical results show that it may be no better than simple text matching. This reveals that NE+TE does not utilize text information sufficiently.

In this paper, we propose a novel Text-driven Graph Embedding with Pairs Sampling (TGE-PS) framework. TGE-PS uses Pairs Sampling (PS) to efficiently generate training samples, and uses Text-driven Graph Embedding (TGE) to produce final node representations. We theoretically analyze the redundant sample phenomenon of RW from three perspectives , and propose PS to improve sampling efficiency. PS samples center-neighbor node pairs directly from central node's neighborhood. Our experiment results on 6 datasets show that PS produces competitive or even better results in link prediction task with much fewer training samples (saving ~99% samples) compared with RW. In embedding stage, we propose inductive TGE method. TGE incorporates text information with structure information and encodes character- and word-level embeddings into node embeddings while following graph structure. Since node embeddings are generated from text embedding in TGE, they can be applied to zero-shot scenarios. The comparison between TGE-PS and other strong baseline models shows that TGE-PS produces remarkably good results in traditional and zero-shot link prediction tasks.

2 RELATED WORKS

Recent years have witnessed various graph embedding models with applications in link prediction[2, 27, 30], node classification[21, 44, 46, 54], clustering[3, 12], recommendation [50, 51, 55], knowledge graph [4], etc. These graph embedding models can be categorized into three classes, factorization-based, random walk (RW)-based, and deep learning-based[5, 16]. Factorization-based models focus on the connections among nodes and using matrix factorization (MF) to learn the low-rank representations of nodes [1, 6, 39, 53]. TADW [53] uses MF to decompose the transition matrix with textual information incorporated. RW-based models explore the neighborhood of each node through sampling paths, and thus can maintain local structural information in node embeddings. Among RW-based models, DeepWalk [42] and node2vec [17] are the most representative, and DeepWalk can be regarded as a special case of node2vec. Deep learning-based models mainly use deep representation learning techniques to improve the quality of node embeddings [7, 20, 47, 48].

There are also models that do not belong to these three classes. LINE [45] proposes to train graph embeddings through first-order and second-order proximities and is efficient in large-scale networks. Besides, LINE claims to preserve both local and global network structures. GraphGAN [49] adopts GAN [15] framework to learn the underlying true connectivity distribution implicitly. The generator produces "fake edges" while the discriminator tries to tell generated node pairs from ground truth. It also proposes Graph Softmax to boost its efficiency in training.

Among graph embedding models, RW-based models present robust and remarkable performance on various datasets. The training of RW-based models have two phases, the sampling phase and the optimization phase. Different sampling policies are proposed to generate high-quality node sequences to explore the neighborhoods of certain nodes. Especially, DeepWalk adopts the simplest policy where each node is generated only depending on its predecessor, while node2vec adopts a biased policy to trade off between Breadth-first Sampling (BFS) and Depth-first Sampling (DFS). After obtaining sufficient node sequences, RW-based models use Skip-Gram [33, 35] model to maximize the log-probability of observing a network neighborhood for a node conditioned on its feature representation. However, we concern the sampling strategies of RW-based models have intrinsic drawbacks, which we discuss in Section 3.2.2.

For graphs with rich texts, many models are proposed to incorporate textual information with structural information. TADW [53] incorporates text features under the framework of matrix factorization. CANE [47] uses convolutional neural networks and mutual attention mechanism to learn text embeddings, which are interacted with node embeddings via vector inner product. These models encode structural information and textual information into two separate embedding spaces and generate final node representations from the interactions between these two spaces. Paper2Vec [14] pre-trains node embeddings with text embeddings in Skip-gram model, and then the node embeddings are trained with node2vec. STNE [29] "self-translates" sequences of text embeddings into sequences of node embeddings. All the above models rely on known connections to generate graph embeddings, thus are not applicable to zero-shot scenarios. Although some previous works [13, 22, 52] discuss graph representation learning in zero-shot setting, they do not solve the same problem as we do.

When processing texts in zero-shot scenarios, it is common to encounter out-of-vocabulary words. In NLP field, character-level embeddings have proven success like Neural Machine Translation [9, 23, 28] and Named Entity Recognition [31, 43]. These models are based on a hybrid of character- and word-level embeddings, thus can automatically capture patterns in the sub-word level and present advantages in dealing with unseen words.

3 TEXT-DRIVEN GRAPH EMBEDDING WITH PAIRS SAMPLING FRAMEWORK

In this section, we firstly give definitions of notions, and then introduce PS and TGE separately. We show the architecture of TGE-PS in Fig. 1, with fully-connected and lookup layers omitted.

3.1 Definitions

Let $G = (V, E)$ be the given graph and $f : V \to \mathbb{R}^n$ be the mapping function from the node set to the n-dimensional embedding space. We denote the central and context embeddings of node v_i as $\mathbf{e}_i = f(v_i)$ and $\mathbf{e}'_i = f'(v_i)$, and the trainable parameters as θ. We define the distance $\mathcal{L}(u, v)$ between u and v as the length of the shortest path between them. And we define node v_i's o-neighborhood $\mathcal{N}_i^o$ as a set of nodes within a given distance o from v_i, $\mathcal{N}^o(v_i) = \{v_j \in$

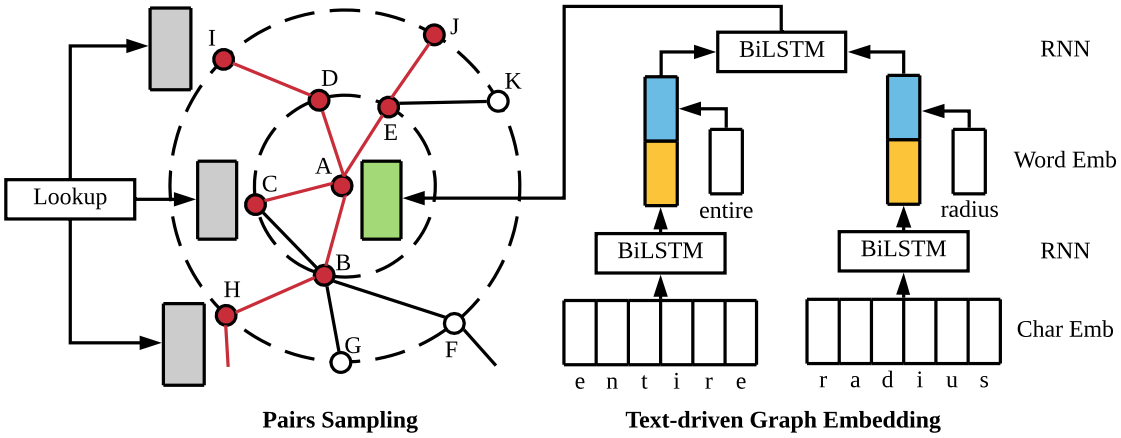

Figure 1: Model architecture of TGE-PS. *Note:* Circles represent nodes and colored boxes represent embeddings. Blue boxes represent word-level embeddings, yellow boxes represent character-based word embeddings, green boxes represent central embeddings and gray boxes represent neighbor embeddings. The sampled nodes and paths are colored as red.

$V|\text{dist}(v_i, v_j) \leq o\}$. For every node v_i, we call one node v_j^o as an o-th order neighbor of v_i when the distance between v_i and v_j is o.

3.2 Pairs Sampling Method

3.2.1 Revisit Random Walk. Since DeepWalk can be regarded as a special case of node2vec [17, 42], our discussion mainly focuses on node2vec. The objective of node2vec is to maximize the log-probability of observing the o-neighborhood $\mathcal{N}_i^o$ of a node v_i as in Eq. (1), and following conditional independence assumption of SkipGram, node2vec changes its objective to maximize the log-probability of all center-neighbor node pairs as in Eq. (2). Following symmetry in feature space assumption, the conditional probability of a node pair is defined as in Eq. (4), where $\tau_{i,j}$ is the abbreviation of the score function $\tau(v_i, v_j)$. It is worth noting that, the score function $\tau_{i,j}$ is asymmetric since the input v_j is conditioned on the other input v_i. With Eq. (2) and Eq. (4), the objective becomes Eq. (3), where Z_i denotes the normalizing term, and each window has size $2k$. Z_i is usually approximated by hierarchical softmax or negative sampling in training. And we mainly focus on the scoring terms $\sum_i \sum_j \tau_{i,j}$.

$$\max_\theta \sum_{v_i \in V} \log Pr(\mathcal{N}_i^o | v_i) \tag{1}$$

$$= \max_\theta \sum_{v_i \in V} \sum_{v_j \in \mathcal{N}_i^k} \log Pr(v_j | v_i) \tag{2}$$

$$= \max_\theta \sum_{v_i \in V} \Big(\sum_{v_j \in \mathcal{N}_i^k} \tau_{i,j} \Big) - |\mathcal{N}_i^k| \log Z_i \tag{3}$$

$$Pr(v_j | v_i) = \frac{\exp(\tau_{i,j})}{\sum_{v_k \in V} \exp(\tau_{i,k})} \tag{4}$$

From the training perspective, node2vec defines the inner product of a central embedding and a context embedding to represent the score function, i.e., $\mathbf{e}_j'^\top \mathbf{e}_i$. The training objective of the scoring terms becomes Eq. (5), where s denote a node sequence, S is the set of all sequences, ω_i^s is the abbreviation of the window function $\omega(v_i, s)$ which denotes the nodes within the window of v_i when v_i appears in s and j' denotes the position of v_j in s. We transform Eq. (5) to Eq. (6) and Eq. (7), where T denotes the sampling time of RW starting from v_i, T_i denotes the amount of sequences where v_i is not the starting point, $T_{j|i}$ denotes the amount of v_j appearing in N_i^k when v_i appears in sequence s. It is worth noting that, each node serves as the starting point in T sequences, and each node also appears in the sequences starting from the other nodes. Thus, there are $T + T_i$ windows centering at v_i to be optimized in the sequences. α_i is the ratio of v_i being more "important" than the other nodes, e.g., bridge nodes are likely to be sampled more frequently, and these nodes have larger α. $\beta_{j|i}$ is the probability of v_j sampled in the neighborhood $\mathcal{N}_i^o$ of v_i. Since there are $T + T_i$ windows centering at v_i, there are $2k(T + T_i)$ center-neighbor node pairs. Thus $2k(T + T_i) = \sum_{v_j \in N_i^k} T_{j|i}$, and $\beta_{\cdot|i}$ reflects the distribution of the neighbor nodes appearing in training samples. Till now, we get the ideal training objective of RW regardless of the sampling strategies. In another word, any RW policies will converge to Eq. (7) when the amount of sampled sequences approaches infinity.

$$\max_\theta \sum_{v_i \in V} \sum_{s \in S} \sum_{v_{j'} \in \omega_i^s} \mathbf{e}_{j'}'^\top \mathbf{e}_i \tag{5}$$

$$= \max_\theta \sum_{v_i \in V} (T + T_i) \Big(\sum_{v_j \in \mathcal{N}_i^k} \frac{T_{j|i}}{2k(T + T_i)} \mathbf{e}_{j'}'^\top \mathbf{e}_i \Big) \tag{6}$$

$$= \max_\theta T \sum_{v_i \in V} (1 + \alpha_i) \Big(\sum_{v_j \in \mathcal{N}_i^k} \beta_{j|i} \mathbf{e}_{j'}'^\top \mathbf{e}_i \Big) \tag{7}$$

3.2.2 Inefficiency of Random Walk. We introduce the sampling strategy of node2vec at first. For every node v_i in the graph, node2vec simulates a random walk with length L. During the sampling process, given previous node t and current node v, the next node x is sampled from the following distribution:

$$Pr(x|v) = \begin{cases} \frac{p_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

where p_{vx} is the unnormalized transition probability from nodes v to x, and Z is the normalizing term. The unnormalized transition probability p_{vx} is set to be $\alpha(t, x; p, q) \cdot w_{vx}$, where $\alpha(t, x; p, q)$ is a function of $\text{dist}(t, x)$, w_{vx} denotes the edge weight of (v, x),

$\text{dist}(t, x)$ denotes the shortest path length between t and x, p and q control the searching strategy of random walks. Without loss of generality, we take $w_{vx} = 1$.

DeepWalk can be regarded as a special case of node2vec when $p = q = 1$. TADW has proved that DeepWalk is reconstructing the transition matrix M. Induced from this conclusion, we view node2vec as reconstructing a biased transition matrix M'. Thus in DeepWalk, $\beta_{j|i}$ in Eq. (7) are actually the non-zero elements of the i-th row of M in DeepWalk, or the i-th row of M' in node2vec. The sampling strategy of node2vec is an exploration of neighborhoods with second-order Markov Property. We analyze the inefficiency of RW in 3 aspects.

Biased Objective. Comparing Eq. (3) and (7), we find node2vec is biased from its log-probability objective since the neighborhoods N_i^k are weighted by different α_i. α_i can be regarded as an "importance ratio", given the previously mentioned bridge nodes example. Hence node2vec introduces a prior distribution implicitly, making its training objective biased from its proposal. If we follow Eq. (1) strictly, α should be all zero and we should only sample each neighborhood T time, which can reduce a lot of training samples.

Interconnected Neighbors. The target of node embedding is to encode structural information in an embedding space. It is a free lunch to assume when optimize a central node v_i, its neighbor nodes $v_j \in N_i^k$ have already been trained and the context embedding $\mathbf{e}'_j$ have already been encoded with the structural information of v_j respectively after training RW for some time, otherwise, the target of node embedding becomes ill-posed and can never be achieved. Thus center-neighbor connections are much more important than neighbor-neighbor connections, and the neighbor-neighbor connections within one neighborhood will become the center-neighbor connections of other neighborhoods. With these observations, we concern RW is less efficient since it results in lots of neighbor-neighbor connections when training central nodes. Therefore, adopting a stronger assumption that alleviates variance of distribution $\beta_{\cdot|i}$ is still possible to produce good results while decreases sampling complexity.

Revisiting Nodes. DeepWalk has the first-order Markov property since it only remembers the current node. node2vec is second-order Markov since it remembers the current node and the previous node. When the window size is larger than the Markov order, RW cannot prevent from revisiting a previously visited neighbor node. The worst case is revisiting the central node itself, because that introduces self-loops and doubles the training time of the central node. In ideal cases, revisiting is equivalent to approximate the transition matrix, where $\beta_{j|i} = M_{i,j}$, $\tau_{i,j} = \beta_{j|i}\mathbf{e}'^{\top}_j \mathbf{e}_i$. Since $\beta_{j|i}$ is a constant determined by the graph structure, the node embeddings $\mathbf{e}_i$ and $\mathbf{e}'_j$ are approximating $\tau_{i,j}/\beta_{j|i}$, where $\tau_{i,j}$ is the unormalized probability. The ideal sample complexity of a neighborhood N_i^k is determined by the minimal sampling probability $\beta_{j*|i}$, by assuming every node in the neighborhood N_i^k gets sufficient training samples. For simplicity, we say the sample complexity of N_i^k is $O(1/\beta_{j*|i})$. And this complexity must be greater than $O(|N_i^k|)$ since $\beta_{j*|i} \le 1/|N_i^k|$. This means any distribution β over N_i^k results in a higher sample complexity than uniform distribution $\beta_{\cdot|i} = 1/|N_i^k|$.

3.2.3 Method Introduction. The intuition of Pairs Sampling (PS) has three points: as for the **biased objective** problem, it is desirable to sample each neighborhood with same time; as for the **interconnected neighbors** problem, it is desirable to sample center-neighbor pairs directly from a neighborhood; as for the **revisiting nodes** problem, it is promising to introduce higher-order Markov property.

To obtain the training node pairs set $\mathcal{P}$, we sample the neighbor nodes in the neighborhood of v_i in the following process:

1. Add all first-order neighbor nodes of v_i into N_i^O.
2. If $O > 1$, for every v_j^o $(1 \le o \le O-1)$, sample a next-order neighbor node v_k^{o+1} from the following distribution:

$$Pr(v_k^{o+1}|v_j^o) = \begin{cases} \frac{1}{Z_j} & \text{if } (v_j^o, v_k^{o+1}) \in E \text{ and } \mathcal{L}(v_i, v_k^{o+1}) = o+1 \\ 0 & \text{otherwise.} \end{cases}$$

   where Z_j is the number of $o+1$-order neighbors connected to v_j^o. The sampled v_k^{o+1} is added into N_i^O.
3. For each neighbor node v_j^o, add node pair (v_i, v_j^o) into pairs set $\mathcal{P}$.
4. Repeat for N times.

We take the graph in Fig. 1 as an example to illustrate how PS works. In this graph, A is the central node, with dashed circles indicating the neighbors of the same order. We color sampled neighbors as red circles and leave the others as white ones. Following the aforementioned process, first-order neighbors $\{B, C, D, E\}$ are all sampled. H is sampled from $\{F, G, H\}$ as the successor of B. I is sampled as the successor of D. J is sampled from $\{J, K\}$ as the successor of E. Unsampled nodes are ignored in the next iteration of sampling, only sampled nodes in this order can be used to generate the next-order samples, e.g., H continues searching while F stops. By now PS samples a set of node pairs $\{(X, A)\}|X \in \{B, C, D, E, H, I, J\}\}$.

By restricting the max order of neighbors and the number of successors for each node to be at most 1, we successfully set an upper bound for the total number of node pairs. In fact, under the same training objective, pairs sampled by different sampling strategies will converge to different distributions of α and β. Theoretical optimal distributions are so far too complex to compute, but we show empirically that pairs sampled by Pairs Sampling can yield competitive results as RW-based does.

3.3 Text-driven Embedding Model

3.3.1 Intuition. Previous graph embedding models focus on generating graph embeddings from only structural information (denoted as NE) or incorporating text attributes with structural information (denoted as NE+TE). NE+TE models can be regarded as encoding structural information into an NE space and encoding textual information into a TE space. Even though complicated interactions are explored between these two spaces, we concern the textual information has not been fully utilized, and propose to generate graph embeddings from text embeddings (denoted as TE2NE). TE2NE is more suitable for large graphs with strong text dependency. A large-scale graph with rich texts may contain million- to billion-level nodes, where the number of nodes is much larger than the number of words. Besides, TE2NE can also apply to zero-shot scenarios, since no explicit NE is required in inference. Therefore,

we propose the Text-driven Graph Embedding (TGE) method that makes the most of textual information by projecting textual information into the NE space. To model the text, we adopt bidirectional LSTM (BiLSTM) [31] for its success in Nature Language Processing field [34, 36]. To deal with out-of-vocabulary words in unseen nodes, we adopt character-level embeddings. Character-level embeddings have proven success in NLP tasks [9, 31]. The advantages of character-level embeddings over word-level embeddings are summarized by [9]. Hence, we adopt character-level embeddings in addition to word-level embeddings.

3.3.2 Generating Embeddings. We denote the set of words as $\mathcal{D}^w$, the set of characters as $\mathcal{D}^c$ and text of node v_i to be $t_i = \{w_{ij}, 1 \le j \le |t_i|\}$, where $w_{ij} \in \mathcal{D}^w$ and $|t_i|$ is the length of t_i. Each word w_{ij} contains characters as $w_{ij} = \{c_{ijk}, 1 \le k \le |w_{ij}|\}$, where $c_{ijk} \in \mathcal{D}^c$ and $|w_{ij}|$ is the length of w_{ij}. We denote character- and word-level embedding vectors as $\mathbf{e}^c$ and $\mathbf{e}^w$ respectively.

We start by generating embedding of node v_i. We feed the sequence of character embeddings $\mathbf{e}^c_{ij,1:|w_{ij}|}$ into character-level BiLSTM and obtain a character-based word embedding $\mathbf{e}^{w'}_{ij}$ as in Eq. (8). The character-based word embedding is concatenated with the corresponding word embedding $\mathbf{e}^w_{ij}$ and fed into the word-level BiLSTM layer as in Eq. (9).

$$\mathbf{e}^{w'}_{ij} = \text{BiLSTM}^c(\mathbf{e}^c_{ij,1:|w_{ij}|}) \quad (8)$$

$$\mathbf{e}_i = \tanh(W\text{BiLSTM}^w([\mathbf{e}^w_{i,1:|T_i|}; \mathbf{e}^{w'}_{i,1:|T_i|}]) + b) \quad (9)$$

Now we obtain the text-based node embedding $\mathbf{e}_i$ of node v_i. We also set up a lookup layer which embeds v_i into dim-dimensional structure-based vector $\mathbf{e}'_i$ that is used to help train $\mathbf{e}_i$, and outputs $\mathbf{e}_i$ as the embedding vector of v_i.

3.3.3 Training. The training objective is Eq. (1). To reduce computation complexity, we define the loss function to be

$$\mathcal{L}_{\text{sim}}(v_i, v_j) = -\log(\sigma(\mathbf{e}'^{\top}_j \mathbf{e}_i)) - \sum_{v_k \sim P(v)}^{N_{\text{neg}}} \log(\sigma(-\mathbf{e}'^{\top}_k \mathbf{e}_i))$$

where $\mathbf{e}_i$, $\mathbf{e}'_j$ and $\mathbf{e}'_k$ denotes embeddings of the central node, the neighbor node in a pair, and the randomly sampled negative node, N_{neg} denotes the number of negative samples [35], σ denotes the sigmoid function and $P(v) \propto d_v^{3/4}$ denotes the distribution of nodes when sampling negative samples, where d_v is the degree of v.

We also apply L_2-regularization on parameters and embeddings We use AdaGrad optimizer [11] to minimize the loss.

4 EXPERIMENTS

4.1 Datasets

To verify the effectiveness and efficiency of PS, we conduct a series of experiments over 6 datasets. We list their details in Tab. 1, where $|V|$ and $|E|$ refer to number of nodes and edges, respectively. In practice, we keep all nodes and $p\%$ edges of the dataset $Data$ for training and denote this setting as $Data@p\%$. Before conducting our experiments, we pre-process texts including lowering all characters, removing stop words and discarding punctuations.

Table 1: Details of Datasets

Datasets	$\|V\|$	$\|E\|$	Type
Cora [32]	2,211	5,214	Citation Graph
Facebook [24]	4,039	88,234	Social Network
Zhihu [47]	10,000	43,894	Q&A Datasets
AstroPh [24]	18,772	198,110	Co-work Network
HepTh [24]	27,400	352,542	Citation Graph
SNOMED [10, 37]	391,892	2,047,749	Health Terminology

4.2 Baselines

We evaluate our model against several graph embedding models: **LINE** [45], **DeepWalk** [42] **node2vec** [17], **CANE** [47], **Paper2Vec** [14] and **STNE** [29]. Since all listed models are not capable of fitting in zero-shot scenarios. Therefore, we design a rule-based embedding method, **Text Matching**, which represents each node as the average vector of pre-trained word embeddings from Glove [41] for every word in the text. The performance of Text Matching reflects the dependency between graph and text structure.

4.3 Evaluation Metrics

In evaluation, we sample an unobserved link (v_c, v_p) and a non-exist link (v_c, v_n) as the positive and negative samples for each node v_c, respectively. we adopt AUC (Area Under Curve) [19] as the metric in two different ways: we use the a portion of pairs to train a logistic regression classifier, use the trained classifier to infer the connectivity of another set of pairs and compute the final AUC score as $\mathbf{AUC_{LR}}$; we directly compute the times of $\mathbf{e}_c^{\top}\mathbf{e}_p > \mathbf{e}_c^{\top}\mathbf{e}_n$ as $\mathbf{AUC_{pair}}$, similar to that in [30]. We report the results of all experiments in two columns, which represent AUC_{LR} and AUC_{pair} respectively.

4.4 Experiments of Pairs Sampling

4.4.1 Theoretical Analysis. Firstly, we compute the sample complexity of each method. Given the number of nodes $|V|$, average degree $\bar{d}$, walk length L, window size W and walk time T in RW, and max order O and sampling time N in PS, the number of sample pairs under RW and PS are $(2L - W - 1)WT|V|$ and $NO\bar{d}|V|$, and the ratio of them r is $\frac{(2L-W-1)WT}{NO\bar{d}}$. We compute the ratio of node pairs with fine-tuned parameters in Tab. 2. Note that real ratios will be larger for that the number of pairs sampled by PS is actually no greater than $NO\bar{d}|V|$. From results in Tab. 2, we can see that PS can significantly reduce the training samples (reducing ~99% samples) compared with RW. Note that r is often much larger in sparse networks like Cora, Zhihu and SNOMED. In graphs with small average degree, RW often conducts DFS-like walks and encounter leaf nodes, which results in frequent revisiting behaviors and hence more redundant samples.

Table 2: Ratios of Different Datasets

Datasets	$\|V\|$	$\|E\|$	$\bar{d}$	r
Cora@50%	2,211	2,607	2.33	183.60
Cora@100%	2,211	5,214	4.42	140.46
Facebook@50%	4,039	44,117	21.84	36.17
Zhihu@50%	10,000	21,947	4.28	369.16
AstroPh@50%	18,772	99,054	10.56	55.40
HepTh@50%	27,400	176,271	12.86	99.79
SNOMED@20%	391,892	409,550	2.09	409.36

4.4.2 Link Prediction. We evaluate PS against RW on 6 datasets listed in Section 4.1. We use PS and RW to generate training sets, and train node embeddings respectively. Results in Tab. 3 show that PS outperforms RW on almost all datasets. This indicate that even if PS adopts stronger assumptions for efficiency consideration, it still proves to be a competitive alternative to RW.

Table 3: Link Prediction Results of RW and PS

Datasets	Random Walk		Pairs Sampling	
	AUC_{LR}	AUC_{pair}	AUC_{LR}	AUC_{pair}
Cora@50%	0.9200	0.9293	**0.9272**	**0.9394**
Facebook@50%	0.9921	0.9892	**0.9922**	**0.9911**
Zhihu@50%	0.8659	**0.9144**	**0.8673**	0.9136
AstroPh@50%	0.9788	0.9768	**0.9795**	**0.9789**
HepTh@50%	0.9741	0.9648	**0.9743**	**0.9730**
SNOMED@20%	0.9350	0.9396	**0.9359**	**0.9402**

4.5 Experiments of TGE-PS

4.5.1 Link Prediction. We evaluate TGE-PS against all baseline models on SNOMED and HepTh respectively. The results are shown in Tab. 4, where "-" refers to results of failed experiments. From Tab. 4, we have following observations:

- TGE-PS outperforms all baseline models on both datasets. Incorporating textual information with structural information helps constructing expressive graph embeddings. Improvements of TGE-PS over PS and Paper2Vec over node2vec strongly support this point. Only preserving structural information like LINE or textual information like Text Matching both presents limitations.
- The improvements of TGE-PS are quite different on the two graphs. Recall that we use the text description of concepts in SNOMED and the whole abstract in HepTh, we owe this to that SNOMED has stronger text dependency than HepTh. The performance of Text Matching in SNOMED supports this point. Similar difference of improvements also appears between Paper2Vec and node2vec.

Table 4: Link Prediction Results of Different Models

Model	**SNOMED**@20%		**HepTh**@50%	
	AUC_{LR}	AUC_{pair}	AUC_{LR}	AUC_{pair}
LINE	0.6461	0.6985	0.7883	0.7520
DeepWalk	0.9164	0.9258	0.9711	0.9573
node2vec	0.9350	0.9396	0.9741	0.9648
Text Matching	0.8954	0.8717	0.7057	0.5700
TADW	-	-	0.8866	0.8977
CANE	0.9613	0.9544	0.9785	0.9388
Paper2Vec	0.9581	0.9604	0.9745	0.9748
STNE	-	-	0.9651	0.9572
PS	0.9359	0.9402	0.9758	0.9716
TGE-PS	**0.9721**	**0.9621**	**0.9793**	**0.9752**

It is also worth noting that not all models are capable of dealing with large graphs: TADW requires pre-loading the whole adjacent matrix with $O(|V|^2)$ complexity; the training speed of STNE is extremely slow on astronomical number of walks generated in SNOMED. We also try to conduct experiments on GraphGAN in link prediction task, but they fail on both datasets due to OOM. On the contrary, our TGE-PS performs well in handling large graphs.

4.5.2 Zero-Shot Experiments. Unlike common link prediction scenarios, where each node embedding is trained at least once before inference, zero-shot scenarios require prediction on link existence with unseen nodes. Zero-shot scenarios are common in real-world applications. For example, when updating medical terminology graphs like SNOMED, terms of newly found diseases will be added into existing databases, and it is possible for a human to infer link existence by text information with high accuracy even if he is not an expert in this field. Therefore, zero-shot scenario in fact reflects whether the model incorporates text information by capturing important message and potential connections or simply matching literal similarities.

We conduct zero-shot experiments on SNOMED and HepTh where 0.5% of the nodes and related edges are removed from the graph. Embeddings of these nodes will be generated by trained models directly and used for evaluation. As we stated in previous sections, zero-shot scenarios are not widely studied and most current models are incapable of generating embeddings for unseen nodes. However, to better study this problem, we manage to conduct experiments on a variant of CANE with only averaged TE part , which we denote as CANE (TE): We also conduct ablation experiments on zero-shot scenarios to analyze the influence of character- and word-level embeddings. In ablation experiments, we either replace $[\mathbf{e}^w; \mathbf{e}^{w'}]$ with $\mathbf{e}^{w'}$ (w/o word), or replace $[\mathbf{e}^w; \mathbf{e}^{w'}]$ with $\mathbf{e}^w$ (w/o char). High AUC scores in Tab. 5 indicate TGE-PS is practical and reliable in zero-shot scenarios. Besides, we have following observations:

- Word-level embeddings are essential in embedding nodes. This is not surprising given that there are much more words (100,471 in SNOMED and 72,083 in HepTh) than characters (88 in SNOMED and 59 in HepTh, mostly English characters and numbers). Besides, when words are often composed of sub-words as in SNOMED, the improvements of character-level embeddings are apparently higher.
- Using Text Matching as a baseline, TGE-PS presents larger promotion on HepTh than SNOMED. Unlike terms in SNOMED, texts in HepTh are abstracts of papers, which have longer average length. In this case, BiLSTM shows its advantages over naive matching methods in processing long texts.

Table 5: Zero-Shot Experiment Results

Model	SNOMED		HepTh	
	AUC_{LR}	AUC_{pair}	AUC_{LR}	AUC_{pair}
Text Matching	0.9059	0.8813	0.5934	0.6250
CANE (TE)	0.5271	0.5344	0.6036	0.5288
TGE-PS (w/o word)	0.5000	0.5003	0.5000	0.5037
TGE-PS (w/o char)	0.9701	0.9786	0.8979	**0.9485**
TGE-PS	**0.9760**	**0.9811**	**0.8990**	**0.9485**

ACKNOWLEDGMENTS

The work is supported by National Natural Science Foundation of China (61702327, 61772333, 81771937), Shanghai Sailing Program (17YF1428200). Weinan Zhang and Yong Yu are the corresponding authors.

REFERENCES

[1] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 37–48.

[2] Lars Backstrom and Jure Leskovec. 2011. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 635–644.

[3] Mikhail Belkin and Partha Niyogi. 2002. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*. 585–591.

[4] Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[5] Hongyun Cai, Vincent W Zheng, and Kevin Chang. 2018. A comprehensive survey of graph embedding: problems, techniques and applications. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[6] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 891–900.

[7] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep neural networks for learning graph representations. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, 1145–1152.

[8] Jifan Chen, Qi Zhang, and Xuanjing Huang. 2016. Incorporate group information to enhance network embedding. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1901–1904.

[9] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. 2016. A Character-level Decoder without Explicit Segmentation for Neural Machine Translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1693–1703.

[10] Kevin Donnelly. 2006. SNOMED-CT: The advanced terminology and coding system for eHealth. *Studies in health technology and informatics* 121 (2006), 279.

[11] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[12] Santo Fortunato. 2010. Community detection in graphs. *Physics reports* 486, 3-5 (2010), 75–174.

[13] Yanwei Fu, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. 2015. Transductive multi-view zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence* 37, 11 (2015), 2332–2345.

[14] Soumyajit Ganguly and Vikram Pudi. 2017. Paper2vec: Combining graph and text information for scientific paper representation. In *European Conference on Information Retrieval*. Springer, 383–395.

[15] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[16] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 855–864.

[18] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584* (2017).

[19] James A Hanley and Barbara J McNeil. 1982. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology* 143, 1 (1982), 29–36.

[20] Yuting Jia, Qinqin Zhang, Weinan Zhang, and Xinbing Wang. 2019. CommunityGAN: Community Detection with Generative Adversarial Nets. *arXiv preprint arXiv:1901.06631* (2019).

[21] Przemyslaw Kazienko and Tomasz Kajdanowicz. 2012. Label-dependent node classification in the network. *Neurocomputing* 75, 1 (2012), 199–209.

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

[23] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. 2017. Fully character-level neural machine translation without explicit segmentation. *Transactions of the Association for Computational Linguistics* 5 (2017), 365–378.

[24] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[25] Chaozhuo Li, Senzhang Wang, Dejian Yang, Zhoujun Li, Yang Yang, Xiaoming Zhang, and Jianshe Zhou. 2017. PPNE: property preserving network embedding. In *International Conference on Database Systems for Advanced Applications*. Springer, 163–179.

[26] Juzheng Li, Jun Zhu, and Bo Zhang. 2016. Discriminative deep random walk for network classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1004–1013.

[27] David Liben-Nowell and Jon Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58, 7 (2007), 1019–1031.

[28] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. 2015. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586* (2015).

[29] Jie Liu, Zhicheng He, Lai Wei, and Yalou Huang. 2018. Content to node: Self-translation network embedding. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1794–1802.

[30] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[31] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1064–1074.

[32] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[34] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[35] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[36] Tomas Mikolov and Geoffrey Zweig. 2012. Context dependent recurrent neural network language model. In *2012 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 234–239.

[37] Jane Millar. 2016. The Need for a Global Language-SNOMED CT Introduction. *Studies in health technology and informatics* 225 (2016), 683–685.

[38] SS Nishana and Subu Surendran. 2013. Graph embedding and dimensionality reduction-a survey. *International Journal of Computer Science & Engineering Technology (IJCSET)* 4, 1 (2013), 29–34.

[39] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1105–1114.

[40] S Pan, J Wu, X Zhu, C Zhang, and Y Wang. 2016. Tri-party deep network representation. In *IJCAI International Joint Conference on Artificial Intelligence*.

[41] 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)*. 1532–1543.

[42] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 701–710.

[43] Cicero Nogueira dos Santos and Victor Guimaraes. 2015. Boosting named entity recognition with neural character embeddings. *arXiv preprint arXiv:1505.05008* (2015).

[44] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI magazine* 29, 3 (2008), 93–93.

[45] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1067–1077.

[46] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)* 3, 3 (2007), 1–13.

[47] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 1722–1731.

[48] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[49] Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. GraphGAN: graph representation learning with generative adversarial nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[50] Pengyang Wang, Jiawei Zhang, Guannan Liu, Yanjie Fu, and Charu Aggarwal. 2018. Ensemble-Spotting: Ranking Urban Vibrancy via POI Embedding with Multi-view Spatial Graphs. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 351–359.

[51] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning graph-based POI embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 15–24.

[52] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Thirtieth*

AAAI Conference on Artificial Intelligence.

[53] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

[54] Shuang-Hong Yang, Bo Long, Alex Smola, Narayanan Sadagopan, Zhaohui Zheng, and Hongyuan Zha. 2011. Like like alike: joint friendship and interest propagation in social networks. In *Proceedings of the 20th international conference on World wide web*. ACM, 537–546.

[55] Chao Zhang, Keyang Zhang, Quan Yuan, Haoruo Peng, Yu Zheng, Tim Hanratty, Shaowen Wang, and Jiawei Han. 2017. Regions, periods, activities: Uncovering urban dynamics via cross-modal representation learning. In *Proceedings of the 26th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 361–370.

[56] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. 2018. Network representation learning: A survey. *IEEE transactions on Big Data* (2018).