Content to Node: Self-Translation Network Embedding

Jie Liu
Nankai University
Tianjin, China
jliu@nankai.edu.cn

Zhicheng He
Nankai University
Tianjin, China
hezhicheng@mail.nankai.edu.cn

Lai Wei
Nankai University
Tianjin, China
future@mail.nankai.edu.cn

Yalou Huang
Nankai University
Tianjin, China
ylhuang@nankai.edu.cn

ABSTRACT

This paper concerns the problem of network embedding (NE), whose aim is to learn low-dimensional representations for nodes in networks. Such dense vector representations offer great promises for many network analysis problems. However, existing NE approaches are still faced with challenges posed by the characteristics of complex networks in real-world applications. First, for many real-world networks associated with rich content information, previous NE methods tend to learn separated content and structure representations for each node, which requires a post-processing of combination. The empirical and simple combination strategies often make the final vector suboptimal. Second, the existing NE methods preserve the structure information by considering short and fixed neighborhood scope, such as the first- and/or the second-order proximities. However, it is hard to decide the scope of the neighborhood when facing a complex problem. To this end, we propose a novel sequence-to-sequence model based NE framework which is referred to as Self-Translation Network Embedding (STNE) model. With the sequences generated by random walks on a network, STNE learns the mapping that translates each sequence itself from the content sequence to the node sequence. On the one hand, the bi-directional LSTM encoder of STNE fuses the content and structure information seamlessly from the raw input. On the other hand, high-order proximity can be flexibly learned with the memories of LSTM to capture long-range structural information. By such self-translation from content to node, the learned hidden representations can be adopted as node embeddings. Extensive experimental results based on three real-world datasets demonstrate that the proposed STNE outperforms the state-of-the-art NE approaches. To facilitate reproduction and further study, we provide Internet access to the code and datasets[1].

[1]http://dm.nankai.edu.cn/code/STNE.rar

KDD '18, August 19–23, 2018, London, United Kingdom

ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219988

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; *Natural language processing*; *Unsupervised learning*; *Learning latent representations*;

KEYWORDS

Network Embedding; Feature Learning; Sequence to Sequence

ACM Reference Format:
Jie Liu, Zhicheng He, Lai Wei, and Yalou Huang. 2018. Content to Node: Self-Translation Network Embedding. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219988

1 INTRODUCTION

Network data is ubiquitous in many fields. Complex networks, such as social networks and citation networks, often involve complex structure and attribute information, which makes network mining a challenging problem to deal with. Furthermore, large-scale networks cause issues of high dimensional representation and computational complexity for conventional network analysis approaches. Hence, finding effective node representations plays a critical role in many network analysis tasks, such as node classification, clustering, and link prediction.

Recently, network embedding (NE) methods have been widely recognized as effective network representation learning approaches. NE aims to learn low-dimensional vectors for nodes in a network by preserving the structural information. Perozzi et al. proposed DeepWalk [19] to learn node representations based on local network information. It first conducts random walks to obtain node sequences. Then it employs the skip-gram model [17] to learn node representations by treating node sequences as sentences. Inspired by the success of DeepWalk, a number of NE approaches, such as LINE [26], PTE [25], and node2vec [9] are proposed. It has been theoretically proved that these approaches are closely related or equivalent [20]. These approaches only focus on the structure of the network. However, there is rich content information associated with each node. In order to incorporate the content or attribute information, TADW [32] is proposed to extend DeepWalk by taking into consideration the text content associated with nodes, which proves that DeepWalk is equivalent to factorizing a matrix derived from node adjacency. Similarly, [28] extends LINE to incorporate text information as well as the network structure.

However, existing NE approaches are still faced with challenges posed by the characteristics of the complex networks in real-world applications. First, for many real-world networks, the nodes are associated with rich content information, in addition to the network structure information. Existing NE methods tend to learn node representation with separated structure and content embedding vectors, which requires post-processing of the combination of these two kinds of vectors. The empirical combination strategies often make the final vector suboptimal. Second, the existing NE methods preserve the structure information by considering the short, fixed, and handcrafted neighborhood information, such as the first and the second order proximities, which cannot capture long-range structure but local neighborhood information. Besides, it is hard to decide the appropriate scope of neighborhood information when facing a complex problem. Although there are a variety of approaches for NE, these approaches do not have a focus on dealing with these challenges.

To address these issues, in this paper, we present a Self-Translation Network Embedding (STNE) model to learn network embedding with flexible neighborhood scope for capturing more meaningful proximity. We cast the network embedding problem as a machine translation problem and determine the mapping from content sequence to node identity sequence. Specifically, given a mass of sampled sequences using random walk, we devise an end-to-end network embedding to encode each content sequence as a compressed vector, and then decode it to generate the corresponding node sequence. We exploit sequence to sequence model (seq2seq) [24] to encode and decode the sampled sequences. The seq2seq models have been successfully applied to machine translation [1, 24], and other natural language processing (NLP) problems[8]. The idea of seq2seq is to use a long short-term memory (LSTM) to read the input sequence, one step at a time, to obtain an overall sequence vector representation, and then use another LSTM to learn the output sequence from that vector. Since the structure information is preserved by sequence context vector learned in a data-driven manner via LSTM, it avoids the rigid assumption of the scope of neighborhood.

Compared with the traditional NE approaches, our STNE directly models the generation process of node sequences, and the generation function can be automatically learned from a large number of text sequences, which provides an end-to-end solution. The key to incorporate and learn the content information from scratch is that we couple the seq2seq network model with additional text embedding layers. Through learning the mapping from content sequences to node sequences, the content information and structure information are seamlessly fused into the latent vectors of hidden layers, which can be effectively used as the representations of nodes.

In addition, STNE assigns context-aware embeddings to a node according to different contexts it interacts with. Most existing NE models assign each node a static embedding vector, which results into context-free embeddings. Context-aware embeddings are ideal because it is intuitive that one node may demonstrate various aspects when interacting with different neighboring nodes. Very recently, CANE [28] is proposed to learn context-aware embedding vectors from text contents of neighboring nodes. Instead of leaning context-aware embeddings mainly from texts of neighboring nodes, our STNE learns dynamic embeddings of a node when confronting different sequences, which could take longer range and more flexible context into account.

We conduct extensive experiments on multiple real-world datasets from different areas. Experimental results demonstrate the effectiveness of our proposed STNE model as compared to other state-of-the-art methods.

Our contributions are as follows:

- We propose to cast network embedding problem as a seq2seq task. This advances the mainstream NE methods, such as DeepWalk and its extensions, from local structure modeling to global structure modeling of a sequence, which enables capturing more semantic information and provides more meaningful network presentation.
- We devise a heterogeneous seq2seq model architecture which embeds raw input text and then learns the mapping from content sequence to node sequence in an end-to-end manner.
- Extensive experiments are conducted on multiple real-world network datasets. The results demonstrate the effectiveness of our approach.

The rest of the paper is organized as follows. In Section 2, we briefly survey related work in network embedding and sequence learning. Section 3 presents the formal definitions of our problem. We give the technical details for representation learning using STNE in Section 4. In Section 5, we empirically evaluate STNE on various real-world networks and assess the parameter sensitivity of our algorithm. We conclude our work in section 6.

2 RELATED WORK

Network embedding, whose aim is to learn low-dimensional node representations, is an emerging network analysis paradigm. Traditionally, a network is represented as a graph. And then the affinity graph [21] is constructed using the feature vectors of the data points, e.g., the K-nearest neighbor graph of data. In this way, the affinity graph can be embedded into a low dimensional space. Extensive graph embedding approaches have been proposed, such as multidimensional scaling (MDS) [6], IsoMap [27], LLE [21] and Laplacian Eigenmaps [2]. Due to the relying on solving the leading eigenvectors of the affinity matrices, the computational complexity is a critical bottleneck and makes them inefficient in real-world applications.

Recently, network embedding or network representation learning has become an active research problem. DeepWalk [19] performs random walks over networks and introduces an efficient word representation learning model, Skip-Gram [17], to learn network embeddings. LINE [26] optimizes the joint and conditional probabilities of edges in large-scale networks to learn node representations. Node2vec [9] modifies the random walk strategy in DeepWalk into biased random walks to explore the network structure more efficiently. All of these approaches only consider the first- and/or second-order proximities which preserve the microscopic and local structure. Cao et al. [3] proposed to capture higher-order proximity. Wang et al. [31] introduced task-specific structure, i.e., community module, for higher order proximity. Essentially, these previous approaches mainly focus on the pairwise relation or other task-specific local structures. Instead, our proposed approach can

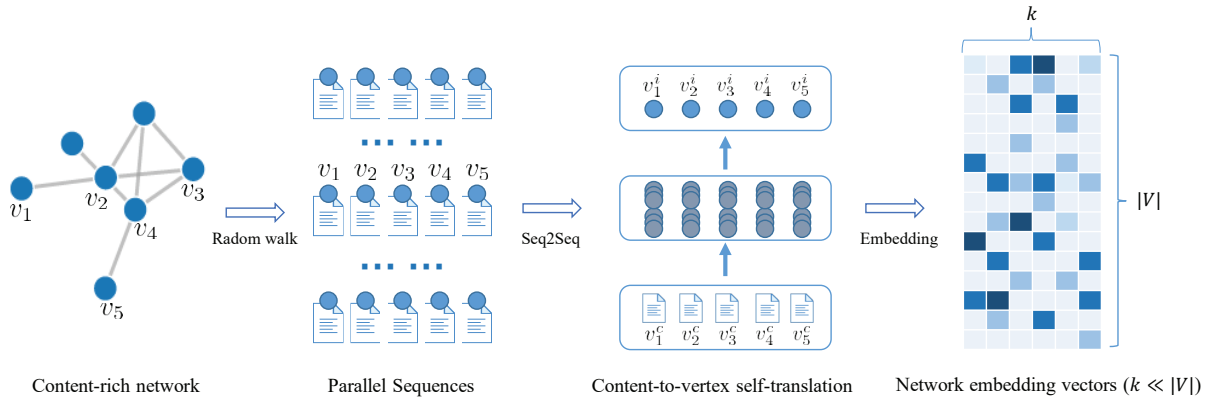


Figure 1: The framework of Self-Translation Network Embedding.

flexibly capture global proximity in an data-driven manner without making any task-specific assumption.

In addition to the network topological information, the nodes are often associated with rich attributes, such as text content, labels, etc. In order to take content into account, Yang et al. presented text-associated DeepWalk (TADW) [32] to improve matrix factorization based DeepWalk with text information. Tu et al. proposed max-margin DeepWalk (MMDW) [29] to learn discriminative network representations by utilizing labeling information of nodes. Chen et al. introduced group enhanced network embedding (GENE) [4] to integrate existing group information in NE. Sun et al. regarded text content as a special kind of nodes and proposed context-enhanced network embedding (CENE) [23] through leveraging both structural and textural information to learn network embeddings.

Another line of related work is sequence modeling which we exploit to build our network embedding model. Recently, the broad adoption of deep learning methods in NLP has given rise to the prevalent use of the recurrent neural network (RNN) [7]. Long short-term memory (LSTM) [10], a particular variant of RNN, have become particularly popular, and been successfully applied to a large number of tasks: speech recognition [8], sequence tagging [12, 15], document categorization [33]. Moreover, in machine translation [1, 5, 24], an LSTM is used to encode a source sequence and then another LSTM is adopted to decode a target sequence, which is called Seq2Seq. Seq2Seq has also received significant research attention in other NLP tasks such as parsing [13, 30], text summarization [18] and multi-task learning [14]. In this work, we develop an end-to-end Seq2Seq model to learn the mapping from content sequence to node sequence.

3 PROBLEM FORMULATION

We consider the problem of learning low-dimensional representation for the content-rich network. In addition to the network structure, there is often heterogeneous information accompanied with nodes in real-world networks which are referred to as content-rich networks.

Definition 1: Content-rich Network. Suppose there is a network $G = (V, E)$, where V is the set of all vertices, and E is the set of all connections between these vertices, i.e., $E \subset V \times V$. For each vertex v, v^i is the identity of the vertex v, and v^c is the content associated with v. Each edge $e_{u,v} \in E$ represents the relation between two vertices (u, v).

Network embedding for content-rich network aims to learn a low-dimensional representation $x_v \in \mathbb{R}^k$ for each vertex $v \in V$ where k is the the dimension of representation space and expected much smaller than $|V|$. The learned representations encode semantic information of nodes in the network, which can be used to do further network analysis, such as node classification.

Conventional network embedding approaches mainly focus on short-range proximity to preserve the network structure semantic information. One of the main representative network embeddings is DeepWalk [19], which has inspired a number of extensions, including TADW [32] for content-rich network embedding.

Compared with previous approaches, the key to our method is that we formulate this task as a sequence to sequence problem. It makes an analogy with machine translation tasks [5]. Using random walk to generate truncated sequences from network, a set of parallel sequences can be obtained.

Definition 2: Parallel Sequences. Let $S = \{v_1, v_2, \cdots, v_T\}$ be a sequence of vertices sampled from a network using random walk, the vertex identity sequence $S^i = \{v_1^i, v_2^i, \cdots, v_T^i\}$ and the corresponding content sequence $S^c = \{v_1^c, v_2^c, \cdots, v_T^c\}$ are a pair of parallel sequences.

In order to capture long-range proximity, we propose to learn a specific seq2seq model. With the set of parallel sequences, we cast the network embedding task as a machine translation problem. Specifically, as is shown in Figure 1, it is a heterogeneous self-translation of vertices from content to vertex.

Definition 3: Content-to-node Self-translation. Given a set of parallel sequences $S = \{(S_n^i, S_n^c)_1^N\}$, content-to-node self-translation is to learn a mapping function $f_\theta : S_n^c \mapsto S_n^i$ for each $S_n \in S$.

Figure 1 shows the overview of our proposed method. Given a content-rich network, the random walk is employed to generate a "corpus" that consists of parallel sequences. Then a specific seq2seq model will be learned on the parallel sequences "corpus". Finally the latent representations of the intermediate layers are regarded as the embedding vectors which can be used for various network analysis tasks.

4 METHODOLOGY

In this paper, we propose a Self-Translation Network Embedding model to learn node representations on the basis of both content and structure information. The critical point of STNE is the content-to-node self-translation process that maps content sequences into corresponding node sequences. Figure 2 illustrates the overall framework of STNE which we will explain in details in this section.

4.1 Content Embedding

STNE is a hierarchical seq2seq model that works on the parallel sequences and learns the mapping relation from content to nodes. Compared with conventional seq2seq models, we couple a hierarchical text embedding layer before sequence encoder to encode the input content sequences into latent semantic space.

Formally, for a given parallel sequence (S_n^i, S_n^c), content sequence $S_n^c = \{v_1^c, v_2^c, \ldots, v_T^c\}$ and identity sequence $S_n^i = \{v_1^i, v_2^i, \ldots, v_T^i\}$, STNE first reads S_n^c and encodes it into a context-aware vector representation $\mathbf{w}$. Since v_t^c is the raw content of node v_i, e.g. text, it should be preprocessed into a vector. For end-to-end learning purpose, an embedding layer is adopted to reduce manual intervention:

$$\mathbf{v}_t^c = \text{Emb}(v_t^c). \tag{1}$$

The embedding function $\text{Emb}(\cdot)$ could be any combination of feature learning neural networks layers, such as the fully connected layer, convolution layer, etc. Furthermore, In this paper, to make a fair comparison with other approaches, we use the raw TFIDF representations and fully connected layer for semantic feature learning.

4.2 Content Sequence Encoder

In order to capture the global semantic information over sequences, we employ LSTM [24] to encode the sequence of embedding vectors. $\{\mathbf{v}_1^c, \ldots, \mathbf{v}_T^c\}$

$$\mathbf{h}_t = \mathcal{H}(\mathbf{v}_t^c, \mathbf{h}_{t-1}) \tag{2}$$

and

$$\mathbf{w} = \mathcal{Q}(\{\mathbf{h}_1, \ldots, \mathbf{h}_T\}), \tag{3}$$

where $\mathbf{h}_t \in \mathbb{R}^k$ is the hidden state vector at time step t and context vector $\mathbf{w}$ is calculated from the sequence of $\mathbf{h}_t$s. $\mathcal{H}(\cdot,\ \cdot)$ and $\mathcal{Q}(\cdot)$ are some nonlinear functions.

LSTM uses purpose-built memory cells to store information which makes it better at considering long-range context information. At the t-th time step, $\mathcal{H}(\cdot,\ \cdot)$ is implemented by the following composite functions:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{vi}\mathbf{v}_t^c + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i), \tag{4}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{vf}\mathbf{v}_t^c + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f), \tag{5}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{vo}\mathbf{v}_t^c + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_{t-1} + \mathbf{b}_o), \tag{6}$$

$$\mathbf{c}_t = \mathbf{f}_t \otimes \mathbf{c}_{t-1} + \mathbf{i}_t \otimes \tanh(\mathbf{W}_{vc}\mathbf{v}_t^c + \mathbf{W}_{hc}h_{t-1+\mathbf{b}_c}), \tag{7}$$

$$\mathbf{h}_t = \mathbf{o}_t \otimes \tanh(\mathbf{c}_t), \tag{8}$$

where $\sigma(\cdot)$ is the logistic sigmoid function, $\otimes$ is the point-wise product of vectors, $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget, and output gate vectors respectively, and $\mathbf{c}_t$ is the cell memory vector. To model both the forward and backward context information along random walks, we adopt a bi-directional LSTM (Bi-LSTM) encoder layer:

$$\overrightarrow{\mathbf{h}_t} = \mathcal{H}^{fw}(v_t^c, \overrightarrow{\mathbf{h}_{t-1}}), \qquad \overleftarrow{\mathbf{h}_t} = \mathcal{H}^{bw}(v_t^c, \overleftarrow{\mathbf{h}_{t+1}}). \tag{9}$$

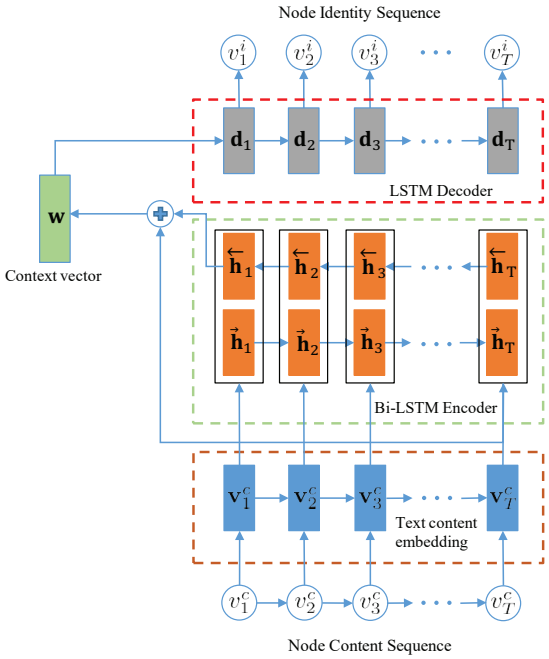


Figure 2: STNE for network embedding.

And the $\mathcal{Q}(\cdot)$ function concatenates the last hidden state vectors of the forward and backward LSTM:

$$\mathbf{w} = \mathcal{Q}(\{\overrightarrow{\mathbf{h}_1}, \ldots, \overrightarrow{\mathbf{h}_T}, \overleftarrow{\mathbf{h}_1}, \ldots, \overleftarrow{\mathbf{h}_T}\}) = [\overrightarrow{\mathbf{h}_T};\ \overleftarrow{\mathbf{h}_1}]. \tag{10}$$

4.3 Node Sequence Generation

Now that the content sequence S_n^c has been compressed into the context vector representation $\mathbf{w}$. The context vector $\mathbf{w}$ seamlessly fuses the content information in $\mathbf{v}_t^c$s and the structure information implied by the sequence itself. Before translating $\mathbf{w}$ into the node identity sequence S_n^i, a decoder layer is employed to decode $\mathbf{w}$ into a sequence of high-level hidden representation vectors so that sequences can be mapped from the content semantic space into the identity semantic space.

Formally, the decoder layer takes the compressed context vector $\mathbf{w}$ as input and generates a sequence of representations $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_T\}$ for the translation layer. Because LSTM has been proved to perform well in various sequence generating tasks, we use an LSTM decoder function $\mathcal{D}(\cdot,\ \cdot)$ to generate $\mathbf{D}$:

$$\mathbf{d}_t = \mathcal{D}(\mathbf{w},\ \mathbf{d}_{t-1}). \tag{11}$$

Compared with natural word space, the semantics of the identity space is relatively concise, and a simplified LSTM is enough to decode $\mathbf{w}$:

$$\mathbf{d}_t = \mathcal{D}(\mathbf{w},\ \mathbf{d}_{t-1}) = \begin{cases} \mathcal{H}(\mathbf{0},\ \mathbf{w}) & t = 1 \\ \mathcal{H}(\mathbf{0},\ \mathbf{d}_{t-1}) & t > 1 \end{cases}, \tag{12}$$

where $\mathbf{0}$ is an all-zero vector.

After $\mathbf{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_T\}$ is decoded from the context vector $\mathbf{w}$, the final step is translating $\mathbf{D}$ into the identity sequence S_n^i with a translation layer. Specifically, the translation is a mapping function

from the identity semantic vector $\mathbf{d}_t$ to the corresponding identity v_t^i. For such purpose, a fully connected layer is first utilized to transform $\mathbf{d}_t$ into a $|V|$-dimension vector, where each dimension corresponds to an identity of a node,

$$\mathbf{g}_t = \sigma(\mathbf{W}_{fc}\mathbf{d}_t + \mathbf{b}_{fc}). \tag{13}$$

Then a softmax layer transforms $\mathbf{g}_t$ into the probabilities,

$$p_t(j) = \text{softmax}(\mathbf{g}_t)_j = \frac{\exp(\mathbf{g}_t(j))}{\sum_{j'} \exp(\mathbf{g}_t(j'))}. \tag{14}$$

4.4 Optimization

After the prediction layer, a cross-entropy loss is adopted to measure the correctness of the translation,

$$L = -\sum_{n=1}^{N} \sum_{v_t \in S_n} \sum_{j}^{|V|} \delta(v_t^i, j) p_t(j), \tag{15}$$

where $\delta(\cdot,\cdot)$ is a binary function that outputs 1 if v_t^i equals j, otherwise 0. To make the predicted identity sequence continuous, the predicted node v_t^i should be a neighbor of the previous node v_{t-1}^i, thus the loss function can be further improved to concentrate only on the neighborhood nodes of v_{t-1}^i:

$$L_t = -\sum_{n=1}^{N} \sum_{v_t \in S_n} \sum_{j \in N(v_{t-1}^i)} \delta(v_t^i, j) p_t(j), \tag{16}$$

where $N(v_{t-1}^i)$ denotes the neighborhood set of v_{t-1}^i.

The model parameter set is $\theta = \{\mathbf{W}_* s,\ \mathbf{b}_* s\}$ where the size of each is $O(|V|k)$. The RMSProp algorithm [22] is used to optimize these parameters (Line 6 and 7 in Algorithm 1), which is very effective for optimizing RNNs. The derivatives are solved by using the chain rules in the back-propagation process. The learning rate η for RMSProp is initially set to 0.001, and the decay rate α is 0.9.

Algorithm 1 STNE Optimization Algorithm.

Input: network $G = (V, E)$
Output: $\{\mathbf{h}(v_1), \ldots, \mathbf{h}(v_{|V|})\}$

1: Generate parallel sequence set $S = \{(S_n^i, S_n^c)_1^N\}$ with random walk
2: Initialize model parameters θ
3: **for each** (S_n^i, S_n^c) **do**
4: **for** $v_t \in (S_n^i, S_n^c)$ **do**
5: $L_t = -v_t^i \mathbf{g}_t(v_t^i)$
6: $\eta = \text{decay}(\eta, \alpha, \frac{\partial L_t}{\partial \theta})$
7: $\theta = \theta - \eta \frac{\partial L_t}{\partial \theta}$
8: **end for**
9: **end for**

4.5 Node Embedding

After the training process converges, the outputs of the encoder layers are taken as the node representations. It is worth noting that the node representations learned by STNE are context-aware. That is, one node appears in multiple sequences and has multiple hidden representations. This characteristic is appealing that it can capture different semantic aspects of a node when interacting with different neighbors. For example, a paper can be cited by other papers from different research sub-fields. They might be related due to similar application tasks, same theoretics, or something else. Suppose that node v_i appears $|v_i|$ times in different sequences, it would have $|v_i|$ representations $\overrightarrow{\mathbf{H}(v_i)} = \{\overrightarrow{\mathbf{h}_1(v_i)}, \ldots, \overrightarrow{\mathbf{h}_{|v_i|}(v_i)}\}$ from the forward encoder layer and $|v_i|$ representations $\overleftarrow{\mathbf{H}(v_i)} = \{\overleftarrow{\mathbf{h}_1(v_i)}, \ldots, \overleftarrow{\mathbf{h}_{|v_i|}(v_i)}\}$ from the backward encoder layer. The final representation of v_i is calculated as the average of the concatenation of $\overrightarrow{\mathbf{H}(v_i)}$ and $\overleftarrow{\mathbf{H}(v_i)}$:

$$\mathbf{h}(v_i) = \frac{1}{|v_i|} \sum_{j=1}^{|v_i|} [\overrightarrow{\mathbf{h}_j(v_i)};\ \overleftarrow{\mathbf{h}_j(v_i)}]. \tag{17}$$

The overall end-to-end learning process of our STNE model is summarized in Figure 1.

5 EXPERIMENTS

To investigate the effectiveness of STNE in the joint modeling of content and structure information, we evaluate our proposed method on several real-world datasets. The experimental results prove our points.

5.1 Datasets

We conduct experiments of node classification on three publicly available real-world datasets[2], including two citation networks and one web page network.

- Cora is a citation network dataset which contains 2708 machine learning papers from seven research categories. There are 5429 citation relations between all these papers. Each document is described by its title and abstract. After removing stop words and low-frequency words, we have a vocabulary of 1433 words. Samples are represented as TFIDF vectors, and each sample has 18 words on average.
- Citeseer is also a citation network dataset. It contains 3312 research papers from six categories, and there are 4732 citation links between them. Each paper is also described by its title and abstract. Stop words and low-frequency words are also removed, and the final vocabulary contains 3703 terms. Each document has 32 words on average and is represented as a TFIDF vector too.
- Wiki dataset contains 2405 web pages from 17 categories. The pages are long texts and there are 17981 hyper links among them. After removing stop words, low-frequency words, and documents that have no links to the others, vocabulary size is reduced to 4973. Each document has 640 words on average and is represented as a TFIDF vector as well.

To facilitate random walks on the edges, all three networks are treated as undirected graphs. Table 1 summarizes the statistics of datasets.

5.2 Comparison Models

As a hot research topic, various models have been developed to learn node representations in content-rich networks. To achieve comprehensive and comparative analysis of STNE, we compared it with three kinds of representative models: content-only models,

[2] http://linqs.cs.umd.edu/projects//projects/lbc/index.html

Table 1: Statistics of Datasets.

Datasets	Cora	Citeseer	Wiki
# Nodes	2708	3312	2405
# Edges	5429	4732	17981
Edge Density	0.074%	0.043%	31.1%
# Words	1433	3703	4973
# Avg. Words / Doc.	18	32	640
# Labels	7	6	17
Max. class size	818	701	406
Min. class size	180	248	9
Avg. class size	387	552	141

structure-only models, and models that combine both content and structure information.

- DeepWalk. DeepWalk [19] is a structure-only NE model. We follow the parameter settings in the original paper, where the number of walks started at each node is set to 80, the length of each walk is set to 40, and the window size is 10. After grid search from 50 to 200, the dimension of learned representations is 100 for Cora and Citeseer datasets, and it is 200 for Wiki dataset.
- MMDW. MMDW [29] is an extension of DeepWalk, which incorporates the semi-supervised information. Hence, MMDW is still a structure-only baseline. The pairwise structural relations between nodes are summarized into a matrix which is latter factorized with the max-margin loss for discrimination purpose. Following the original paper, the maximum length of random walks is set to $t = 2$. And the balancing parameter η is set to 0.01.
- SVD. In SVD, documents are first formalized into a TFIDF matrix whose rows and columns correspond to documents and words respectively. After that, singular value decomposition is performed on the TFIDF matrix, and the left singular matrix $T \in \mathbb{R}^{200\times|V|}$ is treated as document representations. Therefore, SVD is a content-only baseline.
- PLSA. Similar to SVD, PLSA [11] is also a content-only baseline. Instead of directly decomposing the TFIDF matrix, PLSA learns the topic distributions of documents and words. And the topic distributions are regarded as node representations.
- Naive Combination. As a simple baseline that considers both structure and content information, we concatenate node representations learned by SVD and DeepWalk. Due to the above settings, the dimension of learned representations is 300 for Cora and Citeseer datasets, and it is 400 for Wiki dataset.
- NetPLSA. NetPLSA [16] is another baseline method that considers both structure and text information. According to the assumption that linked documents should share similar semantics, it learns network enhanced topic distributions with a link based regularization term. After grid search, we set the number of topics to 160 for Cora and Citeseer datasets, and 200 for Wiki dataset.
- TADW. TADW [32] is another extension of DeepWalk which considers both structure and content information of nodes. Relations between node pairs are summarized into a matrix and then factorized with the assistance of content information. Following the original paper, the maximum random walk steps is set to $t = 2$. Node content features are obtained the same as the SVD method. The representation dimension k is set to 80 for Cora and Citeseer datasets, and 200 for Wiki dataset. And the balancing parameter is $\lambda = 0.2$.

Table 2: Configurations of STNE.

Datasets	Cora	Citeseer	Wiki
TFIDF embedding dimension	1433	3703	4973
Forward encoder dimension	500	500	500
# Forward encoder cells	1	2	1
Backward encoder dimension	500	500	500
# Backward encoder cells	1	2	1
Encoder context dimension	1000	2000	1000
Decoder layer dimension	1000	2000	1000
Prediction layer dimension	2708	3312	2405
# Total layers	7	9	7

5.3 STNE Settings

The architectures of STNE for different datasets are listed in Table 2. The neural networks have 7 layers for Cora and Wiki datasets, and 9 layers for Citeseer dataset. If deeper models are adopted, performances almost remain unchanged or even deteriorate.

For all three datasets, we generate 10 random walks that start at each node, and the length of the walks is set to 10. Detailed analysis of these two parameters will be provided later. For encoder and decoder layers, we apply dropout with probability $p = 0.2$, the batch size is set to 128 during training. The optimization process converges within 2, 1 and 7 epochs on Cora, Citeseer, and Wiki datasets respectively.

For all compared algorithms, to eliminate the classifier's impact on performances, we simply apply the Logistic Regression classifier for classification after node representations are learned. Classification results are evaluated with the F1-score metric. In detail, for a label A, we denote TP(A), FP(A) and FN(A) as the number of true positives, false positives and false negatives in the instances which are predicted as A, respectively. Suppose C is the overall label set. F1-score is defined as:

$$Pr = \frac{\sum_{A\in C} TP(A)}{\sum_{A\in C} TP(A) + FP(A)},\ R = \frac{\sum_{A\in C} TP(A)}{\sum_{A\in C} TP(A) + FN(A)}, \quad (18)$$

$$F1 - score = \frac{2 \times Pr \times R}{Pr + R}. \quad (19)$$

5.4 Classification Results

Table 3, Table 4 and Table 5 show the classification results on Cora, Citeseer and Wiki datasets respectively. The percentage of labeled nodes varies from 10% to 50%. And the best results are boldfaced. From these results, we have the following observations and analysis:

- Structure-only baselines (DeepWalk and MMDW) perform better than content-only baselines (PLSA and SVD) on Cora dataset, while content-only baselines perform better on the other two datasets. This phenomenon can be explained by the characteristics of datasets. First, the Cora documents

Table 3: F1-score on Cora dataset with the percentage of labeled nodes varies from 10% to 50%.

% Labeled Nodes	10%	20%	30%	40%	50%
DeepWalk	76.4	78.0	79.5	80.5	81.0
MMDW	74.9	80.8	82.8	83.7	84.7
SVD	58.3	67.4	71.1	73.3	74.0
PLSA	57.0	63.1	65.1	66.6	67.6
Naive Combination	76.5	80.4	82.3	83.3	84.1
NetPLSA	80.2	83.0	84.0	84.9	85.4
TADW	82.4	85.0	85.6	86.0	86.7
STNE	**84.2**	**86.5**	**87.0**	**86.9**	**88.2**

Table 4: F1-score on Citeseer dataset with the percentage of labeled nodes varies from 10% to 50%.

% Labeled Nodes	10%	20%	30%	40%	50%
DeepWalk	52.4	54.7	56.0	56.5	57.3
MMDW	55.6	60.1	63.2	65.1	66.9
SVD	58.3	66.4	69.2	71.2	72.2
PLSA	54.1	58.3	60.9	62.1	62.6
Naive Combination	61.0	66.7	69.1	70.8	72.0
NetPLSA	58.7	61.6	63.3	64.0	64.7
TADW	**70.6**	**71.9**	**73.3**	73.7	74.2
STNE	69.6	71.2	72.2	**74.3**	**74.8**

Table 5: F1-score on Wiki dataset with the percentage of labeled nodes varies from 10% to 50%.

% Labeled Nodes	10%	20%	30%	40%	50%
DeepWalk	59.3	64.3	66.2	68.1	68.8
MMDW	57.8	62.3	65.8	67.3	67.3
SVD	65.1	72.9	75.6	77.1	77.4
PLSA	69.0	72.5	74.7	75.5	76.0
Naive Combination	66.3	73.0	75.2	77.1	78.6
NetPLSA	67.2	70.6	71.7	71.9	72.3
TADW	72.6	77.3	79.2	79.9	80.3
STNE	**73.9**	**78.0**	**80.6**	**81.5**	**82.7**

contain fewer words than the other two, which makes the content information relatively sparse. Second, documents in Citeseer datasets contain more words than those in Cora dataset, but there are fewer edges among nodes. So the content information contributes relatively more to performances on Citeseer dataset. Third, although the Wiki dataset contains far more edges than the other two, the long documents seem to contribute more to the classification. However, by simultaneously modeling content and structure information, Naive Combination, NetPLSA, TADW, and STNE can all outperform the content-only or structure-only baselines.

- On Cora and Wiki datasets, STNE outperforms all compared baselines. On Citeseer dataset, STNE achieves comparable results to TADW, but still outperforms other baselines. These observations demonstrate the effectiveness of STNE. As shown in Table 1, the Citeseer dataset has a lower edge density. Because our STNE utilizes higher order proximity information than TADW, it suffers more from the insufficient links.
- On Cora and Wiki datasets, STNE can outperform compared baselines even if it uses fewer labeled nodes. The F1-scores of most compared baselines would drop when the classifier is trained with fewer labeled nodes. The reason is that they cannot effectively combine the content and structure information, and inconsistencies exist between the representations of training and testing samples. Because STNE can jointly learn from content and structure information, and take advantage of long node sequences to smooth nodes' representations, the training and testing samples would be more consistent.

5.5 Parameter Analysis

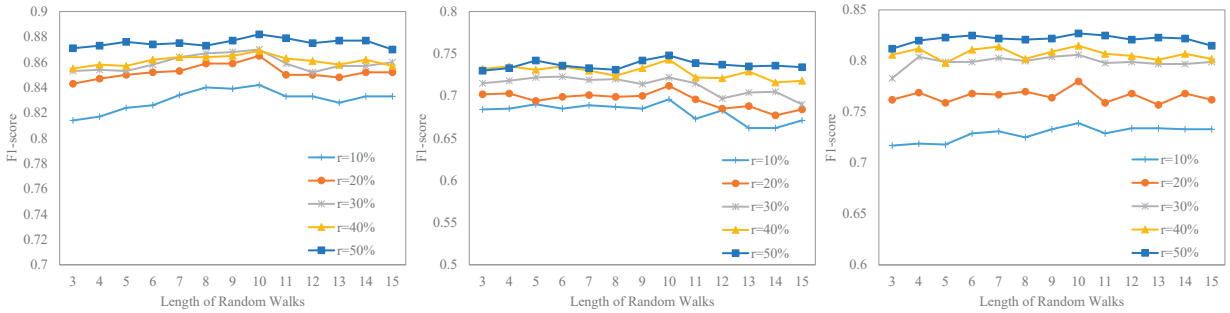
There are two important parameters in STNE, the length of random walks and the number of walks started at each node. In this subsection, we evaluate how different values of walk length and walk number can affect the results.

5.5.1 Length of Random Walks. We first show how the length of random walks affects the performance in Figure 3. The length varies from 3 to 15, and the F1-scores are shown. We can see that the performance initially raises a little when the length increases. This is intuitive as longer sequences can encode higher-order proximity information. However, when the length of walks continuously increases after it reaches 10, the performance starts to drop slowly. The reason is that too long sequences may introduce noises and deteriorate performances. Compared with the Figure 5 in node2vec [9], STNE is less sensitive to the length of random walks. Thus jointly modeling both content and structure information can reduce the dependence on high order proximity information. In addition, STNE only requires length 10 walks to achieve best performances, while DeepWalk and node2vec require 40.

5.5.2 Number of Random Walks per Node. Then we show how the number of random walks started at each node affects the performance in Figure 4, where it varies from 1 to 15. The number of random walks decides how many times we traverse the inner sequential structure information contained in graphs. As observed in Figure 4, a small number of random walks is incapable of fully exploiting the structure information, and performance would be improved when it increases. However, when the number of walks continuously increases after it reaches 10, the performance starts to drop slowly. The reason is that too many random walks may introduce noises and redundancy and deteriorate performance. In addition, compared with the Figure 5 in DeepWalk [19] and the Figure 5 in node2vec [9], STNE is less sensitive to the number of random walks, as the content information can help reduce the dependence on structure information.

5.6 Node Representation Analysis

Besides the encoder output $\mathbf{h}(v_i)$, the decoder output $\mathbf{d}(v_i)$ can also be similarly calculated as in Equation (17). Both $\mathbf{h}$s and $\mathbf{d}$s can be taken as node representations in subsequent tasks, so as the concatenation of them. It is interesting to investigate the performances of different node representations in classification. The F1-scores are plotted out in Figure 5, where the percent of labeled nodes r varies from 10% to 50%. The overall trend is similar on all three datasets, where encoder output $\mathbf{h}$ achieves the best F1-scores,

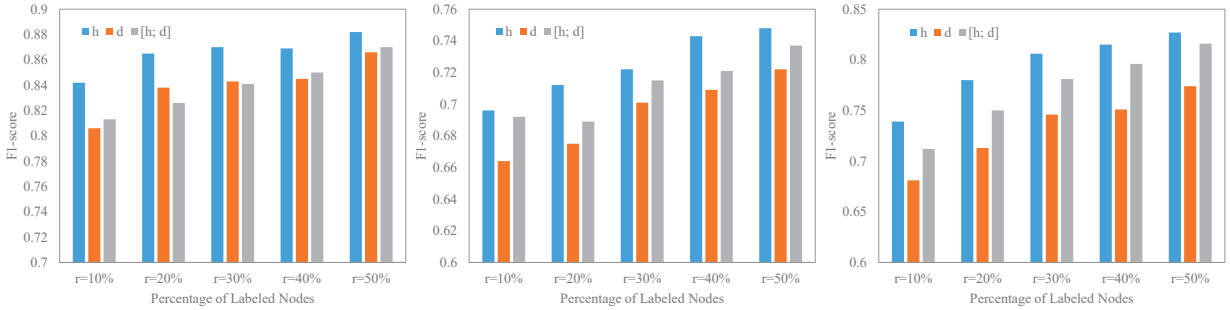


(a) Cora (b) Citeseer (c) Wiki

Figure 3: Analysis of the length of random walks with the percentage of labeled nodes r varying from 10% to 50%.

(a) Cora (b) Citeseer (c) Wiki

Figure 4: Analysis of the number of random walks started at each node with the percentage of labeled nodes r varying from 10% to 50%.



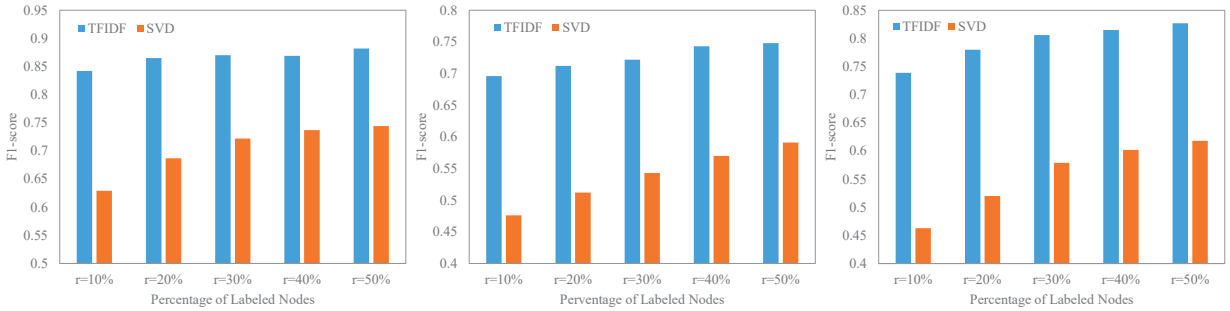
(a) Cora (b) Citeseer (c) Wiki

Figure 5: Analysis of different node representations with the percentage of labeled nodes r varying from 10% to 50%.

decoder output **d** performs worst, and the concatenation of them [**h**; **d**] gets a compromise between them. The possible reason is that the encoder layer learns node representations directly from the input sequence with content and structure, while the decoder layer learns on the basis of the compressed sequence encoding vector **w**, which leads to more information loss in the outputs of decoder layer. The results suggest that the **h** is the ideal choice for node representation.

5.7 Node Content Embedding Analysis

Here we investigate the effectiveness of the content embedding in STNE. An STNE model with content embedding from raw TFIDF input is compared with another STNE without content embedding. For the latter one, we use SVD to extract features as in TADW [32]. Figure 6 shows the F1-scores of node classification, where the percentage of labeled nodes r varies from 10% to 50%. It is obvious that content embedding on the raw TFIDF features achieves significantly better performances than SVD features. This phenomenon illustrates that there exists a lot of information loss in the non-trainable SVD process. And it is better to directly learn from the raw inputs with end-to-end methods, which is exactly what we want to do with the STNE model.



(a) Cora (b) Citeseer (c) Wiki

Figure 6: Analysis of content embedding with the percentage of labeled nodes r varying from 10% to 50%.

6 CONCLUSION

In this paper, we explored the task of content-rich network embedding. We cast this problem as a neural machine translation task, and proposed STNE to learn node representation with a content-to-node seq2seq model. To the best of our knowledge, this is the first time that the network embedding problem is cast as a machine translation task. The merits of STNE come from three aspects. First, it overcomes the restrictions of fixed and low-order proximity. Second, an integrated embedding vector that seamlessly fuses content and structure information can be learned for each node automatically, which avoids handcrafted combinations of separated content and structure vectors. Third, the representation is context-aware, which is appealing in many real-world applications. Extensive experiments demonstrated that our proposed end-to-end content-to-node translation framework substantially outperforms the state-of-the-art approaches.

ACKNOWLEDGMENTS

This research is supported by the National Natural Science Foundation of China under the grant No. U1633103, the Key Projects in Tianjin Science and Technology Pillar Program under the grant No. 17YFZCGX00610, and the Open Project Foundation of Information Technology Research Base of Civil Aviation Administration of China under the grant No. CAAC-ITRB-201601.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[2] Mikhail Belkin and Partha Niyogi. 2001. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. *Advances in Neural Information Processing Systems* 14, 6 (2001).
[3] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. GraRep: Learning Graph Representations with Global Structural Information. In *ACM International on Conference on Information and Knowledge Management*. 891–900.
[4] Jifan Chen, Qi Zhang, and Xuanjing Huang. 2016. Incorporate Group Information to Enhance Network Embedding. In *Proceedings of the 25th CIKM (CIKM '16)*. ACM, New York, NY, USA, 1901–1904. https://doi.org/10.1145/2983323.2983869
[5] Kyunghyun Cho, Bart Van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Computer Science* (2014).
[6] Michael A. A. Cox and Trevor F. Cox. 2001. Multidimensional Scaling. *Journal of the Royal Statistical Society* 46, 2 (2001), 1050âĂŞ1057.
[7] Jeffrey L. Elman. 1990. Finding Structure in Time. *Cognitive Science* 14, 2 (1990), 179–211.
[8] Alex Graves. 2013. Generating Sequences With Recurrent Neural Networks. *CoRR* abs/1308.0850 (2013). arXiv:1308.0850 http://arxiv.org/abs/1308.0850
[9] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. 2016 (2016), 855–864.
[10] Sepp Hochreiter and JÃijrgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9, 8 (1997), 1735–1780.
[11] Thomas Hofmann. 1999. Probabilistic Latent Semantic Indexing. In *IGIR*. 50–57. https://doi.org/10.1145/312624.312649
[12] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. *arXiv: Computation and Language* (2015).
[13] Dong Li and Mirella Lapata. 2016. Language to Logical Form with Neural Attention. In *Meeting of the Association for Computational Linguistics*.
[14] Minh Thang Luong, Quoc V Le, Ilya Sutskever, Oriol Vinyals, and Lukasz Kaiser. 2015. Multi-task Sequence to Sequence Learning. *Computer Science* (2015).
[15] Xuezhe Ma and Eduard H Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. *ACL* (2016), 1064–1074.
[16] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. 2008. Topic modeling with network regularization. In *WWW*. 101–110.
[17] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Computer Science* (2013).
[18] Ramesh Nallapati, Bing Xiang, and Bowen Zhou. 2016. Sequence-to-Sequence RNNs for Text Summarization. *CoRR* abs/1602.06023 (2016). arXiv:1602.06023
[19] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *SIGKDD*. 701–710.
[20] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2017. Network Embedding as Matrix Factorization: Unifying DeepWalk, LINE, PTE, and node2vec. (2017).
[21] Sam T. Roweis and Lawrence K. Saul. 2000. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science* 290, 5500 (2000), 2323–6.
[22] Sebastian Ruder. 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).
[23] Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A General Framework for Content-enhanced Network Representation Learning. *arXiv preprint arXiv:1610.02906* (2016).
[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*. Curran Associates, Inc., 3104–3112.
[25] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks.
[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. 2 (2015), 1067–1077.
[27] J. B. Tenenbaum, Silva V De, and J. C. Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science* 290, 5500 (2000), 2319.
[28] Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. CANE: Context-Aware Network Embedding for Relation Modeling. In *Meeting of the Association for Computational Linguistics*. 1722–1731.
[29] Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-margin deepwalk: discriminative learning of network representation. In *IJCAI*. 3889–3895.
[30] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2014. Grammar as a foreign language. *Eprint Arxiv* (2014), 2773–2781.
[31] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community Preserving Network Embedding. In *AAAI*.
[32] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. 2015. Network representation learning with rich text information. In *IJCAI*.
[33] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. 2016. Hierarchical Attention Networks for Document Classification. (2016), 1480–1489.