Improved Semantic-Aware Network Embedding with Fine-Grained Word Alignment

Dinghan Shen, Xinyuan Zhang, Ricardo Henao, Lawrence Carin
Department of Electrical and Computer Engineering
Duke University, Durham, NC, USA
{dinghan.shen, xy.zhang, r.henao, lcarin}@duke.edu

Abstract

Network embeddings, which learn low-dimensional representations for each vertex in a large-scale network, have received considerable attention in recent years. For a wide range of applications, vertices in a network are typically accompanied by rich textual information such as user profiles, paper abstracts, *etc*. We propose to incorporate semantic features into network embeddings by matching important words between text sequences for all pairs of vertices. We introduce a *word-by-word* alignment framework that measures the compatibility of embeddings between word pairs, and then adaptively accumulates these alignment features with a simple yet effective aggregation function. In experiments, we evaluate the proposed framework on three real-world benchmarks for downstream tasks, including link prediction and multi-label vertex classification. Results demonstrate that our model outperforms state-of-the-art network embedding methods by a large margin.

1 Introduction

Networks are ubiquitous, with prominent examples including social networks (*e.g.*, Facebook, Twitter) or citation networks of research papers (*e.g.*, arXiv). When analyzing data from these real-world networks, traditional methods often represent vertices (nodes) as one-hot representations (containing the connectivity information of each vertex with respect to all other vertices), usually suffering from issues related to the inherent sparsity of large-scale networks. This results in models that are not able to fully capture the relationships between vertices of the network (Perozzi et al., 2014; Tu et al., 2016). Alternatively, network embedding (*i.e.*, network representation learning) has been considered, representing each vertex of a network with a *low-dimensional vector* that preserves information on its similarity relative to other vertices. This approach has attracted considerable attention in recent years (Tang and Liu, 2009; Perozzi et al., 2014; Tang et al., 2015; Grover and Leskovec, 2016; Wang et al., 2016; Chen et al., 2016; Wang et al., 2017a; Zhang et al., 2018).

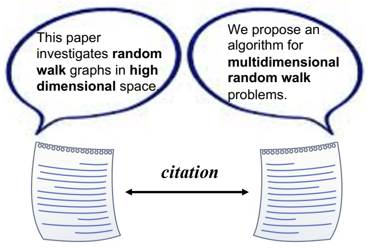

This paper investigates random walk graphs in high dimensional space
We propose an algorithm for multidimensional random walk problems.
citation

Figure 1: Example of the text information (abstracts) associated to two papers in a citation network. Key words for matching are highlighted.

Traditional network embedding approaches focus primarily on learning representations of vertices that preserve local structure, as well as internal structural properties of the network. For instance, Isomap (Tenenbaum et al., 2000), LINE (Tang et al., 2015), and Grarep (Cao et al., 2015) were proposed to preserve first-, second-, and higher-order proximity between nodes, respectively. DeepWalk (Perozzi et al., 2014), which learns vertex representations from random-walk sequences, similarly, only takes into account structural information of the network. However, in real-world networks, vertices usually contain rich textual information (*e.g.*, user profiles in Facebook, paper abstracts in arXiv, user-generated content on Twitter, *etc*.), which may be leveraged effectively for learning more informative embeddings.

To address this opportunity, Yang et al. (2015) proposed text-associated DeepWalk, to incorporate textual information into the vectorial representations of vertices (embeddings). Sun et al. (2016) employed deep recurrent neural networks to integrate the information from vertex-associated text into network representations. Further, Tu et al. (2017) proposed to more effectively model the semantic relationships between vertices using a mutual attention mechanism.

Although these methods have demonstrated performance gains over structure-only network embeddings, the relationship between text sequences for a pair of vertices is accounted for solely by comparing their sentence embeddings. However, as shown in Figure 1, to assess the similarity between two research papers, a more effective strategy would compare and align (via local-weighting) individual important words (keywords) within a pair of abstracts, while information from other words (*e.g.*, stop words) that tend to be less relevant can be effectively ignored (down-weighted). This alignment mechanism is difficult to accomplish in models where text sequences are first embedded into a common space and then compared in pairs (He and Lin, 2016; Parikh et al., 2016; Wang and Jiang, 2017; Wang et al., 2017b; Shen et al., 2018a).

We propose to learn a semantic-aware Network Embedding (NE) that incorporates word-level alignment features abstracted from text sequences associated with vertex pairs. Given a pair of sentences, our model first aligns each word within one sentence with keywords from the other sentence (adaptively up-weighted via an attention mechanism), producing a set of fine-grained matching vectors. These features are then accumulated via a simple but efficient aggregation function, obtaining the final representation for the sentence. As a result, the word-by-word alignment features (as illustrated in Figure 1) are explicitly and effectively captured by our model. Further, the learned network embeddings under our framework are adaptive to the specific (local) vertices that are considered, and thus are context-aware and especially suitable for downstream tasks, such as link prediction. Moreover, since the word-by-word matching procedure introduced here is highly parallelizable and does not require any complex encoding networks, such as Long Short-Term Memory (LSTM) or Convolutional Neural Networks (CNNs), our framework requires significantly less time for training, which is attractive for large-scale network applications.

We evaluate our approach on three real-world datasets spanning distinct network-embedding-based applications: link prediction, vertex classification and visualization. We show that the proposed word-by-word alignment mechanism efficiently incorporates textual information into the network embedding, and consistently exhibits superior performance relative to several competitive baselines. Analyses considering the extracted word-by-word pairs further validate the effectiveness of the proposed framework.

2 Proposed Methods

2.1 Problem Definition

A network (graph) is defined as $\boldsymbol{G} = \{\boldsymbol{V}, \boldsymbol{E}\}$, where $\boldsymbol{V}$ and $\boldsymbol{E}$ denote the set of N vertices (nodes) and edges, respectively, where elements of $\boldsymbol{E}$ are two-element subsets of $\boldsymbol{V}$. Here we only consider undirected networks, however, our approach (introduced below) can be readily extended to the directed case. We also define $\boldsymbol{W}$, the symmetric $\mathbb{R}^{N\times N}$ matrix whose elements, w_{ij}, denote the weights associated with edges in $\boldsymbol{V}$, and $\boldsymbol{T}$, the set of text sequences assigned to each vertex. Edges and weights contain the structural information of the network, while the text can be used to characterize the semantic properties of each vertex. Given network $\boldsymbol{G}$, with the *network embedding* we seek to encode each vertex into a low-dimensional vector $\boldsymbol{h}$ (with dimension much smaller than N), while preserving structural and semantic features of $\boldsymbol{G}$.

2.2 Framework Overview

To incorporate both structural and semantic information into the network embeddings, we specify two types of (latent) embeddings: (i) $\boldsymbol{h}_s$, the *structural embedding*; and (ii) $\boldsymbol{h}_t$, the *textual embedding*. Specifically, each vertex in $\boldsymbol{G}$ is encoded into a low-dimensional embedding $\boldsymbol{h} = [\boldsymbol{h}_s; \boldsymbol{h}_t]$. To learn these embeddings, we specify an objective that leverages the information from both $\boldsymbol{W}$ and $\boldsymbol{T}$, denoted as

$$\mathcal{L} = \sum_{\boldsymbol{e}\in\boldsymbol{E}} \mathcal{L}_{\text{struct}}(\boldsymbol{e}) + \mathcal{L}_{\text{text}}(\boldsymbol{e}) + \mathcal{L}_{\text{joint}}(\boldsymbol{e}) \,, \quad (1)$$

where $\mathcal{L}_{\text{struct}}$, $\mathcal{L}_{\text{text}}$ and $\mathcal{L}_{\text{joint}}$ denote structure, text, and joint structure-text training losses, respectively. For a vertex pair $\{v_i, v_j\}$ weighted by w_{ij}, $\mathcal{L}_{\text{struct}}(v_i, v_j)$ in (1) is defined as (Tang et al., 2015)

$$\mathcal{L}_{\text{struct}}(v_i, v_j) = w_{ij} \log p(\boldsymbol{h}_s^i | \boldsymbol{h}_s^j) \,, \qquad (2)$$

where $p(\boldsymbol{h}_s^i|\boldsymbol{h}_s^j)$ denotes the conditional probability between structural embeddings for vertices $\{v_i, v_j\}$. To leverage the textual information in $\boldsymbol{T}$, similar text-specific and joint structure-text training objectives are also defined

$$\mathcal{L}_{\text{text}}(v_i, v_j) = w_{ij}\alpha_1 \log p(\boldsymbol{h}_t^i|\boldsymbol{h}_t^j)\,, \quad (3)$$

$$\mathcal{L}_{\text{joint}}(v_i, v_j) = w_{ij}\alpha_2 \log p(\boldsymbol{h}_t^i|\boldsymbol{h}_s^j) \quad (4)$$

$$+ w_{ij}\alpha_3 \log p(\boldsymbol{h}_s^i|\boldsymbol{h}_t^j)\,, \quad (5)$$

where $p(\boldsymbol{h}_t^i|\boldsymbol{h}_t^j)$ and $p(\boldsymbol{h}_t^i|\boldsymbol{h}_s^j)$ (or $p(\boldsymbol{h}_s^i|\boldsymbol{h}_t^j)$) denote the conditional probability for a pair of text embeddings and text embedding given structure embedding (or *vice versa*), respectively, for vertices $\{v_i, v_j\}$. Further, α_1, α_2 and α_3 are hyperparameters that balance the impact of the different training-loss components. Note that structural embeddings, $\boldsymbol{h}_s$, are treated directly as parameters, while the text embeddings $\boldsymbol{h}_t$ are learned based on the text sequences associated with vertices.

For all conditional probability terms, we follow Tang et al. (2015) and consider the second-order proximity between vertex pairs. Thus, for vertices $\{v_i, v_j\}$, the probability of generating $\boldsymbol{h}_i$ conditioned on $\boldsymbol{h}_j$ may be written as

$$p(\boldsymbol{h}^i|\boldsymbol{h}^j) = \frac{\exp\left(\boldsymbol{h}^{j^T}\boldsymbol{h}^i\right)}{\sum_{k=1}^{N}\exp\left(\boldsymbol{h}^{j^T}\boldsymbol{h}^k\right)}\,. \quad (6)$$

Note that (6) can be applied to both structural and text embeddings in (2) and (3).

Inspired by Tu et al. (2017), we further assume that vertices in the network play different roles depending on the vertex with which they interact. Thus, for a given vertex, the text embedding, $\boldsymbol{h}_t$, is adaptive (specific) to the vertex it is being conditioned on. This type of context-aware textual embedding has demonstrated superior performance relative to context-free embeddings (Tu et al., 2017). In the following two sections, we describe our strategy for encoding the text sequence associated with an edge into its adaptive textual embedding, via word-by-context and word-by-word alignments.

2.3 Word-by-Context Alignment

We first introduce our base model, which re-weights the importance of individual words within a text sequence in the context of the edge being considered. Consider text sequences associated with two vertices connected by an edge, denoted $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ and contained in $\boldsymbol{T}$. Text sequences $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ are of lengths M_a and M_b, respectively, and are represented by $\mathbf{X}_a \in \mathbb{R}^{d\times M_a}$ and $\mathbf{X}_b \in \mathbb{R}^{d\times M_b}$, respectively, where d is the dimension of the word embedding. Further, $\boldsymbol{x}_a^{(i)}$ denotes the embedding of the i-th word in sequence $\boldsymbol{t}_a$.

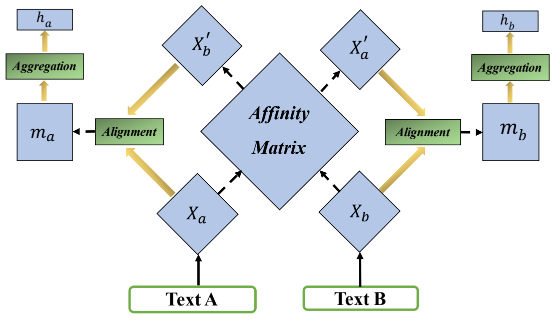


Figure 2: Schematic of the proposed *fine-grained word alignment module* for incorporating textual information into a network embedding. In this setup, word-by-word matching features are explicitly abstracted to infer the relationship between vertices.

Our goal is to encode text sequences $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$ into *counterpart-aware* vectorial representations $\boldsymbol{h}_a$ and $\boldsymbol{h}_b$. Thus, while inferring the adaptive textual embedding for sentence $\boldsymbol{t}_a$, we propose re-weighting the importance of each word in $\boldsymbol{t}_a$ to explicitly account for its alignment with sentence $\boldsymbol{t}_b$. The weight α_i, corresponding to the i-th word in $\boldsymbol{t}_a$, is generated as:

$$\alpha_i = \frac{\exp(\tanh(\mathbf{W}_1\boldsymbol{c}_b + \mathbf{W}_2\boldsymbol{x}_a^{(i)}))}{\sum_{j=1}^{M_a}\exp(\tanh(\mathbf{W}_1\boldsymbol{c}_b + \mathbf{W}_2\boldsymbol{x}_a^{(j)}))}\,, \quad (7)$$

where $\mathbf{W}_1$ and $\mathbf{W}_2$ are model parameters and $\boldsymbol{c}_b = \sum_{i=1}^{M_b}\boldsymbol{x}_i^b$ is the context vector of sequence $\boldsymbol{t}_b$, obtained by simply averaging over all the word embeddings in the sequence, similar to fastText (Joulin et al., 2016). Further, the *word-by-context* embedding for sequence $\boldsymbol{t}_a$ is obtained by taking the weighted average over all word embeddings

$$\boldsymbol{h}_a = \textstyle\sum_{i=1}^{M_a}\alpha_i\boldsymbol{x}_a^{(i)}\,. \quad (8)$$

Intuitively, α_i may be understood as the relevance score between the ith word in $\boldsymbol{t}_a$ and sequence $\boldsymbol{t}_b$. Specifically, keywords within $\boldsymbol{t}_a$, in the context of $\boldsymbol{t}_b$, should be assigned larger weights, while less important words will be correspondingly down-weighted. Similarly, $\boldsymbol{h}_b$ is encoded as a weighted embedding using (7) and (8).

2.4 Fine-Grained Word-by-Word Alignment

With the alignment in the previous section, word-by-context matching features α_i are modeled; however, the *word-by-word* alignment information (*fine-grained*), which is key to characterize the relationship between two vertices (as discussed in the above), is not explicitly captured. So motivated, we further propose an architecture to explicitly abstract word-by-word alignment information from $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$, to learn the relationship between the two vertices. This is inspired by the recent success of Relation Networks (RNs) for relational reasoning (Santoro et al., 2017).

As illustrated in Figure 2, given two input embedding matrices $\mathbf{X}_a$ and $\mathbf{X}_b$, we first compute the affinity matrix $\mathbf{A} \in \mathbb{R}^{M_b \times M_a}$, whose elements represent the affinity scores corresponding to all word pairs between sequences $\boldsymbol{t}_a$ and $\boldsymbol{t}_b$

$$\mathbf{A} = \mathbf{X}_b^T \mathbf{X}_a \,. \tag{9}$$

Subsequently, we compute the context-aware matrix for sequence $\boldsymbol{t}_b$ as

$$\mathbf{A}_b = \text{softmax}(\mathbf{A}) \,, \qquad \widetilde{\mathbf{X}}_b = \mathbf{X}_b \mathbf{A}_b \,, \tag{10}$$

where the softmax$(\cdot)$ function is applied column-wise to $\mathbf{A}$, and thus $\mathbf{A}_b$ contains the attention weights (importance scores) across sequence $\boldsymbol{t}_b$ (columns), which account for each word in sequence $\boldsymbol{t}_a$ (rows). Thus, matrix $\widetilde{\mathbf{X}}_b \in \mathbb{R}^{d \times M_a}$ in (10) constitutes an attention-weighted embedding for $\mathbf{X}_b$. Specifically, the i-th column of $\widetilde{\mathbf{X}}_b$, denoted as $\widetilde{\boldsymbol{x}}_b^{(i)}$, can be understood as a weighted average over all the words in $\boldsymbol{t}_b$, where higher attention weights indicate better alignment (match) with the i-th word in $\boldsymbol{t}_a$.

To abstract the word-by-word alignments, we compare $\boldsymbol{x}_a^{(i)}$ with $\widetilde{\boldsymbol{x}}_b^{(i)}$, for $i = 1, 2, ..., M_a$, to obtain the corresponding matching vector

$$\boldsymbol{m}_a^{(i)} = f_{\text{align}}\left(\boldsymbol{x}_a^{(i)}, \widetilde{\boldsymbol{x}}_b^{(i)}\right) \,, \tag{11}$$

where $f_{\text{align}}(\cdot)$ represents the alignment function. Inspired by the observation in Wang and Jiang (2017) that simple comparison/alignment functions based on element-wise operations exhibit excellent performance in matching text sequences, here we use a combination of element-wise subtraction and multiplication as

$$f_{\text{align}}(\boldsymbol{x}_a^{(i)}, \widetilde{\boldsymbol{x}}_a^{(i)}) = [\boldsymbol{x}_a^{(i)} - \widetilde{\boldsymbol{x}}_a^{(i)}; \boldsymbol{x}_a^{(i)} \odot \widetilde{\boldsymbol{x}}_a^{(i)}] \,,$$

where $\odot$ denotes the element-wise Hadamard product, then these two operations are concatenated to produce the matching vector $\boldsymbol{m}_a^{(i)}$. Note these operators may be used individually or combined as we will investigate in our experiments.

Subsequently, matching vectors from (11) are aggregated to produce the final textual embedding $\boldsymbol{h}_t^a$ for sequence $\boldsymbol{t}_a$ as

$$\boldsymbol{h}_t^a = f_{\text{aggregate}}\left(\boldsymbol{m}_a^{(1)}, \boldsymbol{m}_a^{(2)}, ..., \boldsymbol{m}_a^{(M_a)}\right) \,, \tag{12}$$

where $f_{\text{aggregate}}$ denotes the aggregation function, which we specify as the max-pooling pooling operation. Notably, other commutative operators, such as summation or average pooling, can be otherwise employed. Although these aggregation functions are simple and invariant to the order of words in input sentences, they have been demonstrated to be highly effective in relational reasoning (Parikh et al., 2016; Santoro et al., 2017). To further explore this, in Section 5.3, we conduct an ablation study comparing different choices of alignment and aggregation functions.

The representation $\boldsymbol{h}_b$ can be obtained in a similar manner through (9), (10), (11) and (12), but replacing (9) with $\mathbf{A} = \mathbf{X}_a^T \mathbf{X}_b$ (its transpose). Note that this word-by-word alignment is more computationally involved than word-by-context; however, the former has substantially fewer parameters to learn, provided we no longer have to estimate the parameters in (7).

2.5 Training and Inference

For large-scale networks, computing and optimizing the conditional probabilities in (1) using (6) is computationally prohibitive, since it requires the summation over all vertices $\boldsymbol{V}$ in $\boldsymbol{G}$. To address this limitation, we leverage the negative sampling strategy introduced by Mikolov et al. (2013), *i.e.*, we perform computations by sampling a subset of negative edges. As a result, the conditional in (6) can be rewritten as:

$$p(\boldsymbol{h}^i|\boldsymbol{h}^j) = \log \sigma\left(\boldsymbol{h}^{j^T} \boldsymbol{h}^i\right) + \sum_{i=1}^{K} \mathbb{E}_{\boldsymbol{h}^i \sim P(v)} \left[\log \sigma\left(-\boldsymbol{h}^{j^T} \boldsymbol{h}^i\right)\right] \,,$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. Following Mikolov et al. (2013), we set the noise distribution $P(v) \propto d_v^{3/4}$, where d_v is the out-degree of vertex $v \in \boldsymbol{V}$. The number of negative samples K is treated as a hyperparameter. We use Adam (Kingma and Ba, 2014) to update the model parameters while minimizing the objective in (1).

3 Related Work

Network embedding methods can be divided into two categories: (*i*) methods that solely rely on the structure, *e.g.*, vertex information; and (*ii*) methods that leverage both the structure the network and the information associated with its vertices.

For the first type of models, DeepWalk (Perozzi et al., 2014) has been proposed to learn node representations by generating node contexts via truncated random walks; it is similar to the concept of Skip-Gram (Mikolov et al., 2013), originally introduced for learning word embeddings. LINE (Tang et al., 2015) proposed a principled objective to explicitly capture first-order and second-order proximity information from the vertices of a network. Further, Grover and Leskovec (2016) introduced a biased random walk procedure to generate the neighborhood for a vertex, which infers the node representations by maximizing the likelihood of preserving the local context information of vertices. However, these algorithms generally ignore rich heterogeneous information associated with vertices. Here, we focus on incorporating *textual information* into network embeddings.

To learn semantic-aware network embeddings, Text-Associated DeepWalk (TADW) (Yang et al., 2015) proposed to integrate textual features into network representations with matrix factorization, by leveraging the equivalence between DeepWalk and matrix factorization. CENE (Content-Enhanced Network Embedding) (Sun et al., 2016) used bidirectional recurrent neural networks to abstract the semantic information associated with vertices, which further demonstrated the advantages of employing textual information. To capture the interaction between sentences of vertex pairs, Tu et al. (2017) further proposed Context-Aware Network Embedding (CANE), that employs a mutual attention mechanism to adaptively account for the textual information from neighboring vertices. Despite showing improvement over structure-only models, these semantic-aware methods cannot capture word-level alignment information, which is important for inferring the relationship between node pairs, as previously discussed. In this work, we introduce a Word-Alignment-based Network Embedding (WANE) framework, which aligns and aggregates word-by-word matching features in an explicit manner, to obtain more informative network representations.

4 Experimental Setup

Datasets We investigate the effectiveness of the proposed WANE model on two standard network-embedding-based tasks, *i.e.*, link prediction and multi-label vertex classification. The following three real-world datasets are employed for quantitative evaluation: (*i*) *Cora*, a standard paper citation network that contains 2,277 machine learning papers (vertices) grouped into 7 categories and connected by 5,214 citations (edges) (*ii*) *HepTh*, another citation network of 1,038 papers with abstract information and 1,990 citations; (*iii*) *Zhihu*, a network of 10,000 active users from Zhihu, the largest Q&A website in China, where 43,894 vertices and descriptions of the Q&A topics are available. The average lengths of the text in the three datasets are 90, 54, and 190, respectively. To make direct comparison with existing work, we employed the same preprocessing procedure[1] of Tu et al. (2017).

Training Details For fair comparison with CANE (Tu et al., 2017), we set the dimension of network embedding for our model to 200. The number of negative samples K is selected from $\{1, 3, 5\}$ according to performance on the validation set. We set the batch size as 128, and the model is trained using Adam (Kingma and Ba, 2014), with a learning rate of 1×10^{-3} for all parameters. Dropout regularization is employed on the word embedding layer, with rate selected from $\{0.5, 0.7, 0.9\}$, also on the validation set. Our code will be released to encourage future research.

Baselines To evaluate the effectiveness of our framework, we consider several strong baseline methods for comparisons, which can be categorized into two types: (*i*) models that only exploit *structural* information: MMB (Airoldi et al., 2008), DeepWalk (Perozzi et al., 2014), LINE (Tang et al., 2015), and node2vec (Grover and Leskovec, 2016). (*ii*) Models that take both *structural* and *textual* information into account: Naive combination (which simply concatenates the structure-based embedding with CNN-based text embeddings, as explored in (Tu et al., 2017), TADW (Yang et al., 2015), CENE (Sun et al.,

[1]https://github.com/thunlp/CANE

%Training Edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	54.7	57.1	59.5	61.9	64.9	67.8	71.1	72.6	75.9
DeepWalk	56.0	63.0	70.2	75.5	80.1	85.2	85.3	87.8	90.3
LINE	55.0	58.6	66.4	73.0	77.6	82.8	85.6	88.4	89.3
node2vec	55.9	62.4	66.1	75.0	78.7	81.6	85.9	87.3	88.2
Naive combination	72.7	82.0	84.9	87.0	88.7	91.9	92.4	93.9	94.0
TADW	86.6	88.2	90.2	90.8	90.0	93.0	91.0	93.4	92.7
CENE	72.1	86.5	84.6	88.1	89.4	89.2	93.9	95.0	95.9
CANE	86.8	91.5	92.2	93.9	94.6	94.9	95.6	96.6	97.7
WANE	86.1	90.9	92.3	93.1	93.4	94.5	95.1	95.4	95.9
WANE-*wc*	88.7	92.1	92.9	94.4	94.8	95.1	95.7	96.5	97.4
WANE-*ww*	**91.7**	**93.3**	**94.1**	**95.7**	**96.2**	**96.9**	**97.5**	**98.2**	**99.1**

Table 1: AUC scores for link prediction on the *Cora* dataset.

%Training Edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	54.6	57.9	57.3	61.6	66.2	68.4	73.6	76.0	80.3
DeepWalk	55.2	66.0	70.0	75.7	81.3	83.3	87.6	88.9	88.0
LINE	53.7	60.4	66.5	73.9	78.5	83.8	87.5	87.7	87.6
node2vec	57.1	63.6	69.9	76.2	84.3	87.3	88.4	89.2	89.2
Naive combination	78.7	82.1	84.7	88.7	88.7	91.8	92.1	92.0	92.7
TADW	87.0	89.5	91.8	90.8	91.1	92.6	93.5	91.9	91.7
CENE	86.2	84.6	89.8	91.2	92.3	91.8	93.2	92.9	93.2
CANE	90.0	91.2	92.0	93.0	94.2	94.6	95.4	95.7	96.3
WANE	88.5	90.7	91.1	92.6	93.5	94.2	94.9	95.3	95.8
WANE-*wc*	90.1	91.4	91.9	94.1	95.3	95.9	96.5	96.9	97.2
WANE-*ww*	**92.3**	**94.1**	**95.7**	**96.7**	**97.5**	**97.5**	**97.7**	**98.2**	**98.7**

Table 2: AUC scores for link prediction on the *HepTh* dataset.

2016), and CANE (Tu et al., 2017). It is worth noting that unlike all these baselines, WANE explicitly captures word-by-word interactions between text sequence pairs.

Evaluation Metrics We employ AUC (Hanley and McNeil, 1982) as the evaluation metric for link prediction, which measures the probability that vertices within an existing edge, randomly sampled from the test set, are more similar than those from a random pair of non-existing vertices, in terms of the inner product between their corresponding embeddings.

For multi-label vertex classification and to ensure fair comparison, we follow Yang et al. (2015) and employ a linear SVM on top of the learned network representations, and evaluate classification accuracy with different training ratios (varying from 10% to 50%). The experiments for each setting are repeated 10 times and the average test accuracy is reported.

5 Experimental Results

We experiment with three variants for our WANE model: (*i*) WANE: where the word embeddings of each text sequence are simply average to obtain the sentence representations, similar to (Joulin et al., 2016; Shen et al., 2018c). (*ii*) WANE-*wc*: where the textual embeddings are inferred with word-by-context alignment. (*iii*) WANE-*ww*: where the word-by-word alignment mechanism is leveraged to capture word-by-word matching features between available sequence pairs.

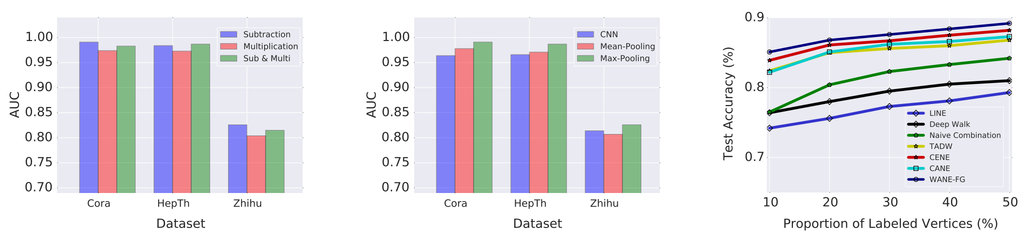
5.1 Link Prediction

Table 1 presents link prediction results for all models on Cora dataset, where different ratios of edges are used for training. It can be observed that when only a small number of edges are available, *e.g.*, 15%, the performances of structure-only methods is much worse than semantic-aware models that have taken textual information into consideration The perfromance gap tends to be smaller when a larger proportion of edges are employed for training. This highlights the importance of incorporating associated text sequences into network embeddings, especially in the case of representing a relatively sparse network. More importantly, the proposed WANE-*ww* model consistently outperforms other semantic-aware NE models by a substantial margin, indicating that our model better abstracts word-by-word alignment features from the text sequences available, thus yields more informative network representations.

Further, WANE-*ww* also outperforms WANE or WANE-*wc* on a wide range of edge training pro-

%Training Edges	15%	25%	35%	45%	55%	65%	75%	85%	95%
MMB	51.0	51.5	53.7	58.6	61.6	66.1	68.8	68.9	72.4
DeepWalk	56.6	58.1	60.1	60.0	61.8	61.9	63.3	63.7	67.8
LINE	52.3	55.9	59.9	60.9	64.3	66.0	67.7	69.3	71.1
node2vec	54.2	57.1	57.3	58.3	58.7	62.5	66.2	67.6	68.5
Naive combination	55.1	56.7	58.9	62.6	64.4	68.7	68.9	69.0	71.5
TADW	52.3	54.2	55.6	57.3	60.8	62.4	65.2	63.8	69.0
CENE	56.2	57.4	60.3	63.0	66.3	66.0	70.2	69.8	73.8
CANE	56.8	59.3	62.9	64.5	68.9	70.4	71.4	73.6	75.4
WANE	52.1	56.6	60.7	64.2	67.5	69.1	71.3	72.8	73.9
WANE-*wc*	55.2	59.9	64.2	68.1	71.3	73.4	75.6	76.3	78.8
WANE-*ww*	**58.7**	**63.5**	**68.3**	**71.9**	**74.9**	**77.0**	**79.7**	**80.0**	**82.6**

Table 3: AUC scores for link prediction on the *Zhihu* dataset.



(a) f_{align} (b) $f_{\text{aggregate}}$ (c) vertex classification

Figure 3: (a, b) Ablation study on the choice of different alignment and aggregation functions. (c) Test accuracy of *supervised* vertex classification on the *Cora* dataset.

portions. This suggests that: (*i*) adaptively assigning different weights to each word within a text sequence (according to its paired sequence) tends to be a better strategy than treating each word equally (as in WANE). (*ii*) Solely considering the context-by-word alignment features (as in WANE-*wc*) is not as efficient as abstracting word-by-word matching information from text sequences. We observe the same trend and the superiority of our WANE-*ww* models on the other two datasets, HepTh and Zhihu datasets, as shown in Table 2 and 3, respectively.

5.2 Multi-label Vertex Classification

We further evaluate the effectiveness of proposed framework on vertex classification tasks with the Cora dataset. Similar to Tu et al. (2017), we generate the global embedding for each vertex by taking the average over its *context-aware* embeddings with all other connected vertices. As shown in Figure 3(c), semantic-aware NE methods (including naive combination, TADW, CENE, CANE) exhibit higher test accuracies than semantic-agnostic models, demonstrating the advantages of incorporating textual information. Moreover, WANE-*ww* consistently outperforms other competitive semantic-aware models on a wide range of labeled proportions, suggesting that explicitly capturing word-by-word alignment features is not only useful for vertex-pair-based tasks, such as link prediction, but also results in better global embeddings which are required for vertex classification tasks. These observations further demonstrate that WANE-*ww* is an effective and robust framework to extract informative network representations.

Semi-supervised classification We further consider the case where the training ratio is less than 10%, and evaluate the learned network embedding with a semi-supervised classifier. Following Yang et al. (2015), we employ a Transductive SVM (TSVM) classifier with a linear kernel (Joachims, 1998) for fairness. As illustrated in Table 4, the proposed WANE-*ww* model exhibits superior performances in most cases. This may be due to the fact that WANE-*ww* extracts information from the vertices and text sequences jointly, thus the obtained vertex embeddings are less noisy and perform more consistently with relatively small training ratios (Yang et al., 2015).

5.3 Ablation Study

Motivated by the observation in Wang and Jiang (2017) that the advantages of different functions to match two vectors vary from task to task, we further explore the choice of alignment and aggregation functions in our WANE-*ww* model. To match the word pairs between two sequences, we experimented with three types of operations: *sub-*

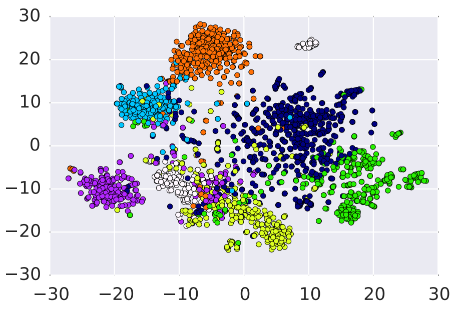


Figure 4: *t*-SNE visualization of the learned network embeddings on the Cora dataset.

traction, *multiplication*, and *Sub & Multi* (the concatenation of both approaches). As shown in Figure 3(a) and 3(b), element-wise *subtraction* tends to be the most effective operation performance-wise on both Cora and Zhihu datasets, and performs comparably to *Sub & Multi* on the HepTh dataset. This finding is consistent with the results in Wang and Jiang (2017), where they found that simple comparison functions based on *element-wise* operations work very well on matching text sequences.

In terms of the aggregation functions, we compare (one-layer) *CNN*, *mean-pooling*, and *max-pooling* operations to accumulate the matching vectors. As shown in Figure 3(b), *max-pooling* has the best empirical results on all three datasets. This may be attributed to the fact that the *max-pooling* operation is better at selecting important word-by-word alignment features, among all matching vectors available, to infer the relationship between vertices.

5.4 Qualitative Analysis

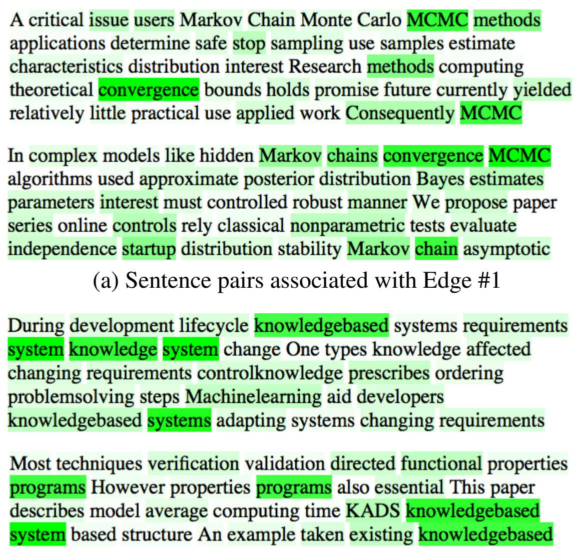
Embedding visualization To visualize the learned network representations, we further employ *t*-SNE to map the low-dimensional vectors of the vertices to a 2-D embedding space. We use the *Cora* dataset because there are labels associated with each vertex and WANE-*ww* to obtain the network embeddings.

As shown in Figure 4 where each point indicates one paper (vertex), and the color of each point indicates the *category* it belongs to, the embeddings of the same label are indeed very close in the 2-D plot, while those with different labels are relatively farther from each other. Note that the model is not trained with any label information, indicating that WANE-*ww* has extracted meaningful patterns from the text and vertex information available.

Baseline Models	1%	3%	7%	10%
Text Only	33.0	43.0	57.1	62.8
Naive Combination	67.4	70.6	75.1	77.4
TADW	72.1	77.0	79.1	81.3
CENE	**73.8**	79.1	81.5	84.5
CANE	72.6	78.2	80.4	83.4
WANE-*ww* (ours)	73.4	**79.6**	**82.7**	**85.1**

Table 4: *Semi-supervised* vertex classification results on the *Cora* dataset.

Case study The proposed word-by-word alignment mechanism can be used to highlight the most informative words (and the corresponding matching features) wrt the relationship between vertices. We visualize the norm of matching vector obtained in (11) in Figure 5 for the Cora dataset. It can be observed that matched key words, *e.g.*, 'MCMC', 'convergence', between the text sequences are indeed assigned higher values in the matching vectors. These words would be selected preferentially by the final *max-pooling* aggregation operation. This indicates that WANE-*ww* is able to abstract important word-by-word alignment features from paired text sequences.


A critical issue users Markov Chain Monte Carlo MCMC methods applications determine safe stop sampling use samples estimate characteristics distribution interest Research methods computing theoretical convergence bounds holds promise future currently yielded relatively little practical use applied work Consequently MCMC

In complex models like hidden Markov chains convergence MCMC algorithms used approximate posterior distribution Bayes estimates parameters interest must controlled robust manner We propose paper series online controls rely classical nonparametric tests evaluate independence startup distribution stability Markov chain asymptotic

(a) Sentence pairs associated with Edge #1

During development lifecycle knowledgebased systems requirements system knowledge system change One types knowledge affected changing requirements controlknowledge prescribes ordering problemsolving steps Machinelearning aid developers knowledgebased systems adapting systems changing requirements

Most techniques verification validation directed functional properties programs However properties programs also essential This paper describes model average computing time KADS knowledgebased system based structure An example taken existing knowledgebased

(b) Sentence pairs associated with Edge #2

Figure 5: Visualization of the word-level *matching vectors*. Darker shades represent larger values of the norm of $\boldsymbol{m}^{(i)}$ at each word position.

6 Conclusions

We have presented a novel framework to incorporate the semantic information from vertex-associated text sequences into network embeddings. An *align-aggregate* framework is introduced, which first aligns a sentence pair by capturing the word-by-word matching features, and then adaptively aggregating these word-level alignment information with an efficient *max-pooling* function. The semantic features abstracted are further encoded, along with the structural information, into a shared space to obtain the final network embedding. Compelling experimental results on several tasks demonstrated the advantages of our approach. In future work, we aim to leverage abundant unlabeled text data to abstract more informative sentence representations (Dai and Le, 2015; Zhang et al., 2017; Shen et al., 2017; Tang and de Sa, 2018) . Another interesting direction is to learn *binary* and compact network embedding, which could be more efficient in terms of both computation and memory, relative to its continuous counterpart (Shen et al., 2018b).

Acknowledgments This research was supported in part by DARPA, DOE, NIH, ONR and NSF.

References

Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed membership stochastic blockmodels. *JMLR* 9:1981–2014.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *CIKM*. ACM, pages 891–900.

Jifan Chen, Qi Zhang, and Xuanjing Huang. 2016. Incorporate group information to enhance network embedding. In *CIKM*. ACM, pages 1901–1904.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*. pages 3079–3087.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. ACM, pages 855–864.

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):29–36.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *NAACL*. pages 937–948.

Thorsten Joachims. 1998. Making large-scale svm learning practical. Technical report, Technical report, SFB 475: Komplexitätsreduktion in Multivariaten Datenstrukturen, Universität Dortmund.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759* .

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* .

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*. pages 3111–3119.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *EMNLP* .

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. ACM, pages 701–710.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. 2017. A simple neural network module for relational reasoning. In *NIPS*. pages 4974–4983.

Dinghan Shen, Renqiang Min Martin, Yitong Li, and Lawrence Carin. 2018a. Learning context-sensitive convolutional filters for text processing. In *EMNLP*.

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018b. Nash: Toward end-to-end neural architecture for generative semantic hashing. *arXiv preprint arXiv:1805.05361* .

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018c. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In *ACL*.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017. Deconvolutional latent-variable model for text sequence matching. *arXiv preprint arXiv:1709.07109* .

Xiaofei Sun, Jiang Guo, Xiao Ding, and Ting Liu. 2016. A general framework for content-enhanced network representation learning. *arXiv preprint arXiv:1610.02906* .

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. International World Wide Web Conferences Steering Committee, pages 1067–1077.

Lei Tang and Huan Liu. 2009. Relational learning via latent social dimensions. In *SIGKDD*. ACM, pages 817–826.

Shuai Tang and Virginia R de Sa. 2018. Multi-view sentence representation learning. *arXiv preprint arXiv:1805.07443* .

Joshua B Tenenbaum, Vin De Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290(5500):2319–2323.

Cunchao Tu, Han Liu, Zhiyuan Liu, and Maosong Sun. 2017. Cane: Context-aware network embedding for relation modeling. In *ACL*. volume 1, pages 1722–1731.

Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. 2016. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*. pages 3889–3895.

Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *SIGKDD*. ACM, pages 1225–1234.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. In *ICLR*.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017a. Community preserving network embedding. In *AAAI*. pages 203–209.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017b. Bilateral multi-perspective matching for natural language sentences. In *IJCAI*.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network representation learning with rich text information. In *IJCAI*. pages 2111–2117.

Xinyuan Zhang, Yitong Li, Dinghan Shen, and Lawrence Carin. 2018. Diffusion maps for textual network embedding. *arXiv preprint arXiv:1805.09906* .

Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*. pages 4169–4179.