SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

November 2021, Vol. 64 212102:1–212102:10
https://doi.org/10.1007/s11432-020-2934-6

Representation learning on textual network with personalized PageRank

Teng LI* & Yong DOU

National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, Changsha 410073, China

Received 23 March 2020/Revised 20 April 2020/Accepted 18 May 2020/Published online 18 May 2021

Abstract Representation learning on textual network or textual network embedding, which leverages rich textual information associated with the network structure to learn low-dimensional embedding of vertices, has been useful in a variety of tasks. However, most approaches learn textual network embedding by using direct neighbors. In this paper, we employ a powerful and spatially localized operation: personalized PageRank (PPR) to eliminate the restriction of using only the direct connection relationship. Also, we analyze the relationship between PPR and spectral-domain theory, which provides insight into the empirical performance boost. From the experiment, we discovered that the proposed method provides a great improvement in link-prediction tasks, when compared to existing methods, achieving a new state-of-the-art on several real-world benchmark datasets.

Keywords representation learning, network embedding, PageRank, textual network, personalized PageRank

Citation Li T, Dou Y. Representation learning on textual network with personalized PageRank. Sci China Inf Sci, 2021, 64(11): 212102, https://doi.org/10.1007/s11432-020-2934-6

1 Introduction

The aim of the representation learning on a textual network or textual network embedding is to leverage dimensionality reduction techniques, and also to extract the high-dimensional information about a vertex's structural information and associated textual information into a dense feature vector embedding. The following downstream network data mining task utilizes these network embeddings: link prediction, vertex classification, and clustering [1].

Traditional network embedding methods for dimensional reduction [2, 3] have good performance on small network datasets. However, the complexity of these embedding methods is quadratic in the size of network vertices. Thus, this makes it impossible to run on large-scale networks. To address the scalability limitation, Perozzi et al. [4] proposed DeepWalk to learn a low-dimensional vector by introducing deep learning techniques into network embedding. Several researches have followed suit [5, 6]. These neural network-based methods demonstrate both high scalability and performance. The methods achieve excellent results on link prediction and vertex classification tasks in large-scale network datasets. Moreover, NetMF [7] provides theoretical analysis to show that these methods are closely equivalent after transformation. Meanwhile, these methods are designed for structure-only network datasets despite their progress. Network vertices such as social networks and citation networks contain rich textual information. TADW [8] proposed DeepWalk to incorporate textual information into the matrix factorization model. CENE [9] formulated a novel network embedding method that leverages both structure and textual content information in a network by considering contexts as a special kind of vertices. CANE [10] extended LINE to incorporate textual information to learn context-aware embedding for a vertex embedding according to the direct neighbor it interacts with.

We identify two major limitations of existing textual network embedding methods despite their success. Besides, TADW [8] successfully incorporates textual information into the embedding learning process.

* Corresponding author (email: liteng09@nudt.edu.cn)

However, a major limitation of this method is that it is inherently flat. This is because it optimizes the problem using a time-consuming matrix-factorization objective function and is difficult to generalize for a large dataset. Meanwhile, CANE [10] performs better than CENE [9] by modeling relationships between direct connect vertices more precisely and also overcomes the scalability problem since it utilizing the neural network-based technique. Moreover, CANE [10] depends on the first-hop neighbor's information without considering a larger neighborhood. Thus, such a strategy denies the desirable feature, thereby reducing the overall performance.

In this paper, we present a representation learning on textual networks with the personalized PageRank (PPR) method to address the aforementioned issues. Instead of aggregating textual information only from the direct neighbors, the proposed model aggregates textual information from a larger neighborhood. We provide a theoretical analysis from the spectral domain theory perspective. To demonstrate the effectiveness of the proposed, we conduct extensive experiments on several benchmark datasets, achieving a new state-of-the-art when compared to the existing methods.

2 Related work

Classical network embedding methods mainly focus on the structural information of the network. These network embedding methods learn low-dimensional embeddings by utilizing random walk statistics and matrix-factorization-based learning objective function. For instance, Perozzi et al. [4] proposed DeepWalk to learn a low-dimensional feature vector by performing random walks over networks in an efficient similar skip-gram learning fashion as word2vec [11]. Tang et al. [5] proposed LINE to explicitly capture first-order and second-order proximity information from the vertices of the large-scale network. Grover et al. [6] modified the random walk into a biased random walk to make the network embedding more efficient. However, these methods only consider the topology information of the network, without considering the associated textual information.

Furthermore, researches were carried out to integrate the associated textual information into the network embedding learning process. Regarding this, Yang et al. [8] proposed a text-associated DeepWalk (TADW) to integrate textual features into network representation learning under the model of matrix factorization. CENE [9] considered the text information as a special kind of vertex and use a bidirectional recurrent neural network to abstract the semantic information for network embedding learning. CANE [10] learns the context-aware embeddings of vertices through mutual attention mechanisms, and expected to recognize the semantic relationship between vertices more accurately.

3 Problem definition

3.1 Textual network

We consider a textual network $G = (V, E)$, where V denotes the vertex set and E denotes the edge set, respectively. Also, we define $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ as an adjacency matrix encoding the connection situation between two vertices and $\boldsymbol{T}$, the textual information associated with vertices. The textual information associated a specific vertex $\boldsymbol{v} \in V$ is represented as a token sequence $\boldsymbol{S}_v = (\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{n_v})$, where $n_v = |\boldsymbol{S}_v|$. We pre-embedded all the tokens in a fixed dimensional feature vector. The goal of textual network embedding is to learn a low-dimensional embedding for each vertex by utilizing both network structural information and associated textual information simultaneously.

3.2 From PageRank to personalized PageRank

The original PageRank [12] started with a summation equation, which originates from bibliometrics study and the analysis of the citation structure in academic papers. The PageRank of a page P_i denoted as $r(P_i)$, is the sum of the PageRanks of all pages pointing into P_i.

$$r(P_i) = \sum_{P_j \in B_{P_i}} \frac{r(P_j)}{|P_j|}, \tag{1}$$

where B_{P_i} is the set of pages pointing into P_i and $|P_j|$ is the number of out-links from page P_j.

To write Eq. (1) using matrices, we consider a directed graph $G = (V, E)$, let $\boldsymbol{A}$ be the adjacent matrix of the directed graph, and define $\boldsymbol{A}$ as follows:

$$\boldsymbol{A}(\boldsymbol{v}_i, \boldsymbol{v}_j) = 1, \quad \text{if}(\boldsymbol{v}_i, \boldsymbol{v}_j) \subseteq E. \tag{2}$$

Now, let d be the number of edges leaving a vertex $\boldsymbol{v}_i$ and $\boldsymbol{D}$ be the diagonal matrix with the out-links of nodes on the diagonal. Then we have

$$\boldsymbol{A}_{\text{rw}} = \boldsymbol{A}\boldsymbol{D}^{-1}. \tag{3}$$

Then, we introduce a $1 \times n$ vector $\boldsymbol{\pi}$, which represents the PageRank value for all pages in the index, and finally, the aforementioned Eq. (1) is re-formulated:

$$\boldsymbol{\pi}^{(k+1)T} = \boldsymbol{\pi}^{(k)T}\boldsymbol{A}_{\text{rw}}, \tag{4}$$

where k denotes that the PageRank vector at the kth iteration.

The PPR vector $\boldsymbol{\pi}$ for vertex $\boldsymbol{v}_i$ is the stationary distribution of the following random walk starting from $\boldsymbol{v}_i$: at each step, return to $\boldsymbol{v}_i$ with probability α, and otherwise move to a random neighbor of the current vertex. The PPR vector of vertex $\boldsymbol{v}_i$ is given by

$$\boldsymbol{\pi}_{\boldsymbol{v}_i}^{(k+1)T} = (1-\alpha)\boldsymbol{\pi}_{\boldsymbol{v}_i}^{(k)T}\boldsymbol{A}_{\text{rw}} + \alpha\boldsymbol{e}_{\boldsymbol{v}_i}^{T}, \tag{5}$$

where $\boldsymbol{e}_{\boldsymbol{v}_i}^{T}$ is identity vector of $\boldsymbol{v}_i$ and the parameter $\alpha \in (0, 1]$.

4 Method

4.1 Model framework overview

To efficiently encode both the structural information and the associated textual information in the textual network embedding, we state two types of embedding for each vertex $\boldsymbol{v} \in V$, namely structural information embedding $\boldsymbol{v}^s$ and textual information embedding $\boldsymbol{v}^t$. The final embedding of vertex $\boldsymbol{v}$ is constructed by concatenating the two types of the embedding: $\boldsymbol{v} = \boldsymbol{v}^s \oplus \boldsymbol{v}^t$.

To leverage both the structural information and textual information into the final embedding process, we define the objective function of the proposed model as

$$\mathcal{L} = \sum L_{\text{structure}}(e) + L_{\text{text}}(e) + L_{\text{joint}}(e), \tag{6}$$

where $L_{\text{structure}}(e)$ denotes the structural information objective, $L_{\text{text}}(e)$ denotes the textual information objective, and $L_{\text{joint}}(e)$ denotes joint learning loss for structural and textual information, respectively. In general, the structural information objective is represented as

$$L_{\text{structure}}(e) = \omega_{ij} \log p(\boldsymbol{v}_i^s | \boldsymbol{v}_j^s), \tag{7}$$

where $p(\boldsymbol{v}_i^s | \boldsymbol{v}_j^s)$ denotes the conditional probability between structural information embedding vertex pair $\boldsymbol{v}_i^s$, $\boldsymbol{v}_j^s$ with weight parameter ω_{ij}.

Based on the method by [5], we define the conditional probability as

$$p(\boldsymbol{v}_i^s | \boldsymbol{v}_j^s) = \frac{\exp(\boldsymbol{v}_i^s \cdot \boldsymbol{v}_j^s)}{\sum_{\boldsymbol{v}_k^s \in V} \exp(\boldsymbol{v}_k^s \cdot \boldsymbol{v}_j^s)}. \tag{8}$$

Also, the textual information objective is represented as

$$L_{\text{text}}(e) = \omega_{ij} \log p(\boldsymbol{v}_i^t | \boldsymbol{v}_j^t), \tag{9}$$

where

$$p(\boldsymbol{v}_i^t | \boldsymbol{v}_j^t) = \frac{\exp(\boldsymbol{v}_i^t \cdot \boldsymbol{v}_j^t)}{\sum_{\boldsymbol{v}_k^t \in V} \exp(\boldsymbol{v}_k^t \cdot \boldsymbol{v}_j^t)}. \tag{10}$$

Furthermore, we also introduce the joint learning loss for structural and textual information simultaneously. Consequently, the joint structural-textual training objective is given by

$$L_{\text{joint}}(e) = \beta_1 \omega_{ij} \log p(\boldsymbol{v}_i^t | \boldsymbol{v}_j^s) + \beta_2 \omega_{ij} \log p(\boldsymbol{v}_i^s | \boldsymbol{v}_j^t), \tag{11}$$

where

$$p(\boldsymbol{v}_i^t|\boldsymbol{v}_j^s) = \frac{\exp(\boldsymbol{v}_i^t \cdot \boldsymbol{v}_j^s)}{\sum_{\boldsymbol{v}_k^t \in V} \exp(\boldsymbol{v}_k^t \cdot \boldsymbol{v}_j^s)}, \tag{12}$$

$$p(\boldsymbol{v}_i^s|\boldsymbol{v}_j^t) = \frac{\exp(\boldsymbol{v}_i^s \cdot \boldsymbol{v}_j^t)}{\sum_{\boldsymbol{v}_k^s \in V} \exp(\boldsymbol{v}_k^s \cdot \boldsymbol{v}_j^t)}, \tag{13}$$

and β_1 and β_2 are hyper-parameters, which measure the impact of the different components.

4.2 Textual embedding

Here, we now introduce the textual embedding component to convert the original input text content of each vertex into the latent representation. For each vertex $\boldsymbol{v}$ with associated textual information $\boldsymbol{v}^t$, we obtain the textual embedding using an end-to-end CNN [13] method. For any vertex $\boldsymbol{v}$ with associated textual information $\boldsymbol{v}^t = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_{|\boldsymbol{v}^t|}\}$, the model first leverages LookingUp layer to transform each $\boldsymbol{p}_i$ into their corresponding word embedding $\boldsymbol{p}_i \in \mathbb{R}^l$, where l indicates the dimension of the word embedding and $\oplus$ is the concatenation operator. Therefore, the textual information representation of vertex $\boldsymbol{v}^t$ is given by

$$\boldsymbol{P} = \text{LookingUp}(\boldsymbol{v}^t) = \boldsymbol{p}_1 \oplus \boldsymbol{p}_2 \oplus \cdots \oplus \boldsymbol{p}_{|v^t|}. \tag{14}$$

We then apply a convolution operation that involves the filter $\boldsymbol{w}$, to a window of m words for generating a new feature. For instance, the feature x_i is generated from the window of words $\boldsymbol{p}_{i:i+m}$:

$$x_i = f(\boldsymbol{w} \cdot \boldsymbol{p}_{i:i+m} + b), \tag{15}$$

where b is a bias term and f is a non-linear function such as hyperbolic tangent. Therefore we can obtain the feature map using

$$\boldsymbol{x} = [x_1, x_2, \ldots, x_{|v^t|-m+1}]. \tag{16}$$

We conduct a max-over-time pooling operation $\hat{x} = \max\{\boldsymbol{x}\}$ on x to extract the most relevant feature corresponding to this particular filter. Finally, we encode the textual information of vertex $\boldsymbol{v}^t$ and obtain the textual embedding representation as

$$\boldsymbol{v}^t = [x_1, x_2, \ldots, x_n]^{\mathrm{T}}, \tag{17}$$

where n is the number of filters.

4.3 Personalized PageRank integrating

The PPR is a variation of PageRank which is biased towards a set of root vertices. This variant of PageRank increases the chance of teleporting back to the root vertex. This ensures that the PageRank score encodes the local neighborhood information of each root vertex. We calculate the original PageRank using the following equation: $\boldsymbol{\pi}_{pr}^{(k+1)T} = \boldsymbol{\pi}_{pr}^{(k)T} \boldsymbol{A}_{\text{rw}}$ with $\boldsymbol{A}_{\text{rw}} = \boldsymbol{A}\boldsymbol{D}^{-1}$, $\boldsymbol{D}$ is the diagonal matrix with the out-links of nodes on the diagonal. Considering the root vertex, we formulate the PPR vector of vertex $\boldsymbol{v}_i$ as

$$\boldsymbol{\pi}_{\boldsymbol{v}_i}^{(k+1)T} = (1-\alpha)\boldsymbol{\pi}_{\boldsymbol{v}_i}^{(k)T} \boldsymbol{A}_{\text{rw}} + \alpha \boldsymbol{e}_{\boldsymbol{v}_i}^T, \tag{18}$$

where $\boldsymbol{e}_{\boldsymbol{v}_i}^T$ is identity vector of $\boldsymbol{v}_i$ and the parameter $\alpha \in (0, 1]$ (Figure 1).

Instead of using the random walk adjacency matrix $\boldsymbol{A}_{\text{rw}}$ directly, here we adopt the symmetrically normalized adjacency matrix with self-loop $\hat{\boldsymbol{A}}$, which proved to be more effective by [14]. Therefore, we formulate the PPR vector of vertex $\boldsymbol{v}_i$ as follows:

$$\boldsymbol{\pi}_{\boldsymbol{v}_i}^{(k+1)T} = (1-\alpha)\boldsymbol{\pi}_{\boldsymbol{v}_i}^{(k)T} \hat{\boldsymbol{A}} + \alpha \boldsymbol{e}_{\boldsymbol{v}_i}^T, \tag{19}$$

where $\hat{\boldsymbol{A}} = \hat{\boldsymbol{D}}^{-1/2}(\boldsymbol{A} + \boldsymbol{I}_n)\hat{\boldsymbol{D}}^{-1/2}$ is the symmetrically normalized adjacency matrix with self-loop, with the diagonal degree matrix $\hat{\boldsymbol{D}}_{ij} = \sum_p (\boldsymbol{A} + \boldsymbol{I})_{ip}$. And k denotes that the PPR vector at the kth iteration.

After several iterations, we obtain that the above Eq. (19) converges to the equilibrium state, we have

$$\boldsymbol{\pi}_{\boldsymbol{v}_i} = \alpha(\boldsymbol{I}_n - (1-\alpha)\hat{\boldsymbol{A}})^{-1} \boldsymbol{e}_{\boldsymbol{v}_i}. \tag{20}$$

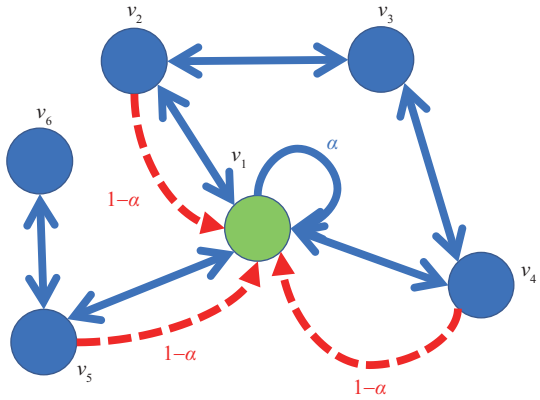

Figure 1 (Color online) Illustration of the PPR. At the iteration k, the root vertex v_1 aggregates textual information from its own and its neighbors, the parameter α controls the priority given to the textual information aggregation from itself as opposed to its large neighbors.

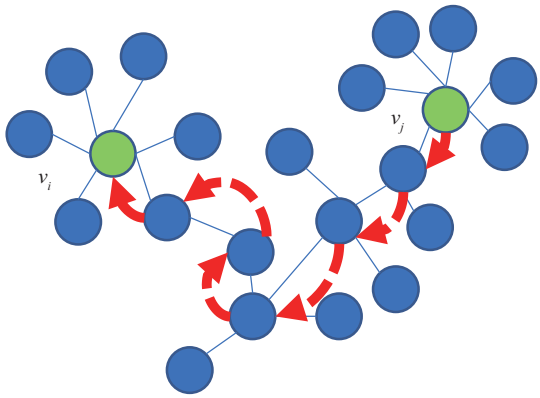

Figure 2 (Color online) High-level illustration of our proposed method. The textual information propagates from the right vertex v_j to the left root vertex v_i via the PPR.

The influence score of root vertex $\boldsymbol{v}_i$ to vertex $\boldsymbol{v}_j$ is proportional to the jth element of $\boldsymbol{\pi}_{\boldsymbol{v}_i}$. While the propagate speed from root vertex $\boldsymbol{v}_i$ to vertex $\boldsymbol{v}_j$ is controlled by parameter α. Replacing the identity vector $\boldsymbol{e}_{\boldsymbol{v}_i}$ with unit matrix $\boldsymbol{I}_n$, we obtain the fully PPR matrix as

$$\boldsymbol{\Pi} = \alpha(\boldsymbol{I}_n - (1-\alpha)\hat{\boldsymbol{A}})^{-1}. \tag{21}$$

We note that $\Pi_{ij} = \Pi_{ji}$, since the influence score from vertex i to vertex j is equal to the influence score from vertex j to vertex i.

Combining the textual embedding of each vertex $\boldsymbol{v}^t$ and the PPR scheme, we obtain our proposed model equation as

$$\widetilde{\boldsymbol{V}} = \alpha(\boldsymbol{I}_n - (1-\alpha)\hat{\boldsymbol{A}})^{-1}\boldsymbol{V}, \quad \boldsymbol{V}_{i,:} = \boldsymbol{v}_i^t. \tag{22}$$

Consequently, the textual information of each vertex is generated from itself as well as from other vertex's textual information propagated through the PPR scheme (Figure 2).

4.4 Spectral analysis of PPR

By studying and analyzing the PPR from a graph spectral domain perspective, we obtain that PPR corresponds to an equivalent polynomial filter on the graph spectral domain.

Traditional spectral theory [15] to graph has three different Laplace operators, namely, the unnormalized Laplacian $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, the random-walk normalized Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{A}_{\mathrm{rw}}$ and the symmetric normalized graph Laplacian $\boldsymbol{L} = \boldsymbol{I} - \boldsymbol{A}_{\mathrm{sym}}$. The Laplacian is indeed diagonalized using the Fourier basis $\boldsymbol{U} = [\boldsymbol{u}_0, \ldots, \boldsymbol{u}_{n-1}]$ such that $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$ where $\boldsymbol{\Lambda} = \mathrm{diag}[\lambda_0, \ldots, \lambda_{n-1}]$.

The graph Fourier transform of a signal $\boldsymbol{x} \in \mathbb{R}^n$ is defined as $\hat{\boldsymbol{x}} = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}$ and its inverse as $\boldsymbol{x} = \boldsymbol{U}\hat{\boldsymbol{x}}$. The convolution operator on graph $\mathcal{G}$ is define as

$$\boldsymbol{x} *_{\mathcal{G}} \boldsymbol{y} = \boldsymbol{U}((\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}) \odot (\boldsymbol{U}^{\mathrm{T}}\boldsymbol{y})), \tag{23}$$

where $\odot$ is the element-wire Hadamard product. A signal $\boldsymbol{x}$ is filter by g_θ can be formulated as

$$y = g_\theta(\boldsymbol{L})\boldsymbol{x} = g_\theta(\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}})\boldsymbol{x} = \boldsymbol{U}g_\theta(\boldsymbol{\Lambda})\boldsymbol{U}^{\mathrm{T}}\boldsymbol{x}, \tag{24}$$

where $g_\theta(\boldsymbol{\Lambda}) = \mathrm{diag}(\boldsymbol{\theta})$ and $\boldsymbol{\theta} \in \mathbb{R}^n$. One common choice of g_θ is a polynomial filter of order J to reduce the learning complexity and capture the localized information simultaneously [16].

$$g_\theta(\boldsymbol{L}) = \sum_{j=0}^{J} \theta_j \boldsymbol{L}^j = \boldsymbol{U}\left(\sum_{j=0}^{J} \theta_j \boldsymbol{\Lambda}^j\right)\boldsymbol{U}^{\mathrm{T}}, \tag{25}$$

where $\boldsymbol{\theta} \in \mathbb{R}^J$ is a vector of polynomial coefficients.

As shown in [17], the PPR model can be approximately reformulated as

$$f_\alpha(\boldsymbol{A}) = \alpha \sum_{k=0}^{K} (1-\alpha)^k (\hat{\boldsymbol{A}})^k. \tag{26}$$

To obtain the relationship between Eqs. (25) and (26), we choose the Laplacian corresponding to $\boldsymbol{L} = \boldsymbol{I}_n - \hat{\boldsymbol{A}}$, and define $\eta_k = \alpha\,(1-\alpha)^k$. Using the binomial theorem, Eq. (26) can be represented as

$$\begin{aligned}
\sum_{k=0}^{K} \eta_k \left(\boldsymbol{I}_n - \boldsymbol{L}\right)^k &= \sum_{k=0}^{K} \eta_k \sum_{j=0}^{k} \binom{k}{j} \boldsymbol{I}_n^{k-j} (-1)^j \boldsymbol{L}^j \\
&= \sum_{k=0}^{K} \sum_{j=0}^{k} \binom{k}{j} \eta_k \boldsymbol{I}_n^{k-j} (-1)^j \boldsymbol{L}^j \\
&= \sum_{k=0}^{K} \sum_{j=0}^{k} \binom{k}{j} \eta_k (-1)^j \boldsymbol{L}^j \\
&= \sum_{j=0}^{K} \sum_{k=j}^{K} \binom{k}{j} \eta_k (-1)^j \boldsymbol{L}^j.
\end{aligned} \tag{27}$$

Comparing Eqs. (25) and (27), we can conclude $\theta_j = \sum_{k=j}^{K} \binom{k}{j} \eta_k (-1)^j$. Now, replacing $\eta_k = \alpha(1-\alpha)^k$ and setting $K = +\infty$, we obtain

$$\begin{aligned}
\theta_j &= \sum_{k=j}^{+\infty} \binom{k}{j} \eta_k (-1)^j = \sum_{l=0}^{+\infty} \binom{l+j}{l} \eta_{l+j} (-1)^j \\
&= \sum_{l=0}^{+\infty} \binom{l+j}{l} \alpha (1-\alpha)^{l+j} (-1)^j \\
&= \alpha (1-\alpha)^j \sum_{l=0}^{+\infty} \binom{l+j}{l} (1-\alpha)^j (-1)^j \\
&= \alpha (1-\alpha)^j \frac{1}{\alpha^{j+1}} = \left(\frac{1-\alpha}{\alpha}\right)^j.
\end{aligned} \tag{28}$$

This shows that the PPR can be expressed as polynomial filters on the graph spectral domain.

4.5 Optimization

Direct optimization of the objective function (6) requires considering all the vertex information, which is computationally demanding for many large-scale network data sets. To remedy this issue, we leverage the negative sampling method introduced by Mikolov et al. [11]. We formulate the objective function into the following form:

$$\log \sigma(\boldsymbol{v}_i \boldsymbol{v}_j) + \sum_{k=1}^{K} E_{v_k \sim P(v)} [\log \sigma(-\boldsymbol{v}_k \boldsymbol{v}_i)], \tag{29}$$

where $\sigma(\boldsymbol{x}) = 1/(1+\exp(-\boldsymbol{x}))$ is the sigmoid function, K is the number of negative samples. The first term maximizes the probability of occurrence for vertices that lie in the context window. While the second term iterates over some random vertex k that do not lie in the window and minimizes their probability of co-occurrence. We set the noisy distribution $P(v) \propto d_v^{3/4}$, where d_v is the out-degree of vertex $\boldsymbol{v}$. Thereafter, we employ Adam [18] to optimize the objective function.

5 Experiments

In this section, we demonstrate the proposed method through a series of experiments.

Table 1 Datasets statistics

Dataset	Type	Vertex	Edge	Label
Cora	Citation network	2708	5429	7
Hepth	Citation network	1039	1990	–
Zhihu	Social network	10000	43894	–

5.1 Datasets

We evaluate the link prediction performance of the proposed method on three widely studied real-world textual network datasets: Cora, Hepth, and Zhihu.

Cora is a typical paper citation network constructed by [10], vertices are documents and edges are citation links. After filtering out papers without text information, there are 2277 papers in this citation network.

Hepth is another paper citation network from Arxiv website[1)] on high energy physics theory released by Stanford. There are 1038 papers after filtering out the paper without text information.

Zhihu is a social network dataset constructed by Tu et al. [10], which extracts 10000 active users as network vertex from Zhihu website[2)] and takes the descriptions of concerned topics as text information. Dataset statistics are summarized in Table 1.

5.2 Baselines

In the performance comparison, we consider baselines based on structure-only methods as well as the structure-text fusion method.

DeepWalk [4]**.** Learning low-dimensional vector by performing random walks over networks in an efficient similar skip-gram learning fashion as Word2vec.

LINE [5]**.** Capturing first-order and second-order proximity information from the vertices of the large-scale network.

Node2vec [6]**.** Modifying the random walk into a biased random walk to infer the network embedding more efficiently.

Concatenate. Simply concatenate the structural feature and textual feature into network embedding.

TADW [8]**.** Integrating textual features into network representation learning under the framework of matrix factorization.

CENE [9]**.** Utilizing bidirectional recurrent neural network to abstract the semantic information to learn network embedding.

CANE [10]**.** Learning the context-aware embeddings of vertices through mutual attention mechanisms and is expected to capture the semantic relationship between vertices more accurately.

5.3 Experiment setup

For a fair comparison, we adopt experimental setup in Tu et al. [10] to prevent unnecessary bias in our experiment. Precisely, we set the embedding dimension into 200 and use Adam with a learning rate $1e^{-3}$ to train our model. Also, we use a grid search to set the hyper-parameter on the split validation set. Meanwhile, we tune the number of epochs based on both convergence behavior and validation accuracy on all datasets. We apply a standard evaluation metric area under the curve (AUC) for link prediction.

5.4 Experimental results

Based on the results in Tables 2–4, we conclude that the proposed method performs better and is very competitive.

(1) Tables 2 and 3 show that the performance of the proposed method matches the performance of CANE in most cases. Remarkably, under 55% of edges setting, our method is about 2% worse than CANE on Cora, this is attributed to overfitting since the training set is small to fit our model.

(2) On Zhihu, Table 4 shows that the proposed method achieves significant improvement when compared to all the baseline methods. In particular, the proposed method obtains over 10% performance

1) http://arxiv.org.
2) http://zhihu.com.

Table 2 AUC scores for link prediction on Cora

	Percentage of edges				
	55%	65%	75%	85%	95%
DeepWalk	80.1	85.2	85.3	87.8	90.3
LINE	77.6	82.8	85.6	88.4	89.3
Node2vec	78.7	81.6	85.9	87.3	88.2
Concatenate	88.7	91.9	92.4	93.9	94.0
TADW	90.0	93.0	91.0	93.4	92.7
CENE	89.4	89.2	93.9	95.0	95.9
CANE	**94.6**	94.9	95.6	96.6	97.7
PPR	92.4	**95.0**	**95.8**	**96.9**	**98.1**

Table 3 AUC scores for link prediction on Hepth

	Percentage of edges				
	55%	65%	75%	85%	95%
DeepWalk	81.3	83.3	87.6	88.9	88.0
LINE	78.5	83.8	87.5	87.7	87.6
Node2vec	84.3	87.3	88.4	89.2	89.2
Concatenate	88.7	91.8	92.1	92.0	92.7
TADW	91.1	92.6	93.5	91.9	91.7
CENE	92.3	91.8	93.2	92.9	93.2
CANE	94.2	94.6	95.4	95.7	96.3
PPR	**94.7**	**95.9**	**96.8**	**97.5**	**98.7**

Table 4 AUC scores for link prediction on Zhihu

	Percentage of edges				
	55%	65%	75%	85%	95%
DeepWalk	61.8	61.9	63.3	63.7	67.8
LINE	64.3	66.0	67.7	69.3	71.1
Node2vec	58.7	62.5	66.2	67.6	68.5
Concatenate	64.4	68.7	68.9	69.0	71.5
TADW	60.8	62.4	65.2	63.8	69.0
CENE	66.3	66.0	70.2	69.8	73.8
CANE	68.9	70.4	71.4	73.6	75.4
PPR	**78.7**	**81.1**	**83.9**	**85.7**	**87.2**

boost than the CANE method, verifying that our model can be fit more precisely on the large dataset by aggregating more textual information from a larger neighborhood.

5.5 Parameter analysis

As shown in Eq. (19), the PPR introduces the hyper-parameter α to control the priority given to the textual information aggregation from itself as opposed to its large neighbors. Therefore, we conduct experiments to show the effect of the AUC scores on all datasets. Figure 3 shows the effect of PPR by varying α from 0.01 to 1.0. From these figures, we observe that the optimal choice for different datasets differs slightly. For small datasets such as Cora and Hepth, the AUC score increases to a slight variation of high value with the increasing value of α. This means that under a small dataset setting, the AUC score is not very sensitive to the variation of the hyper-parameter. For large dataset Zhihu, we obtain that $\alpha \in [0.05, 0.1]$ performs better, indicating that aggregate the large neighbor's textual information plays an important role in the performance improvement.

5.6 Embedding visualization

Figure 4 demonstrates the 2D visualization of the textual network embedding on the Cora dataset by utilizing the t-SNE [19] toolbox. We obtain that the data of different classes are distributed more clearly in our method when compared to the method without PPR. However, this shows the effectiveness and advantages of incorporating PPR into the textual network learning process.

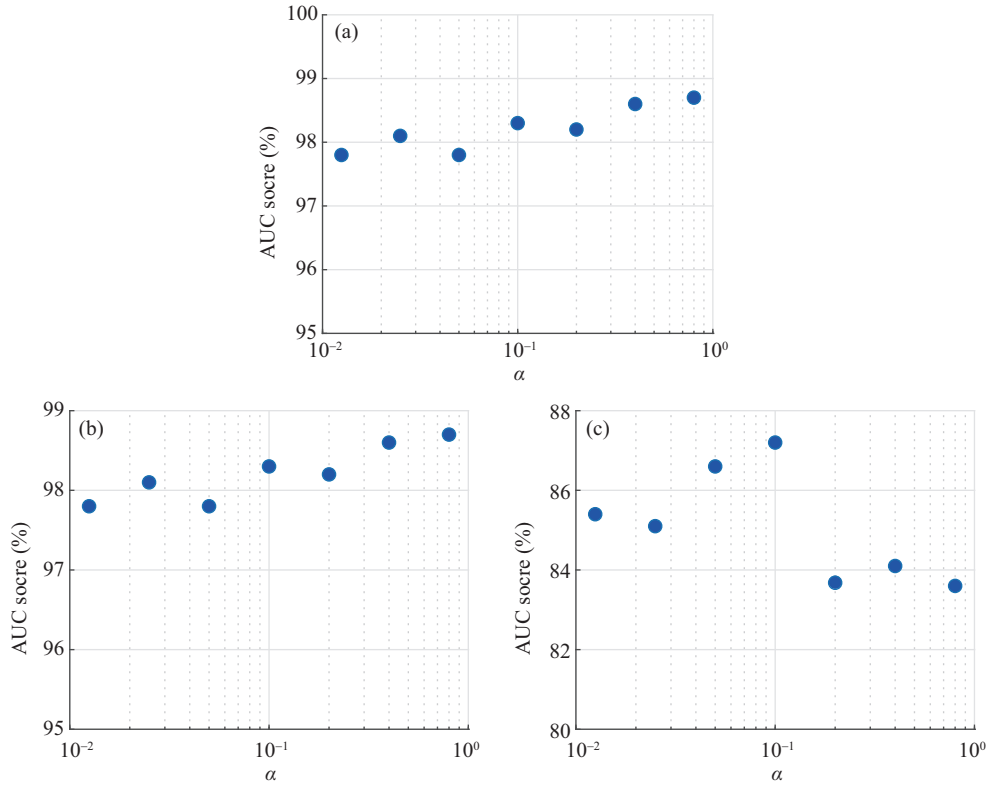

Figure 3 (Color online) AUC scores depending on hyperparameter α. (a) Cora; (b) Hepth; (c) Zhihu.

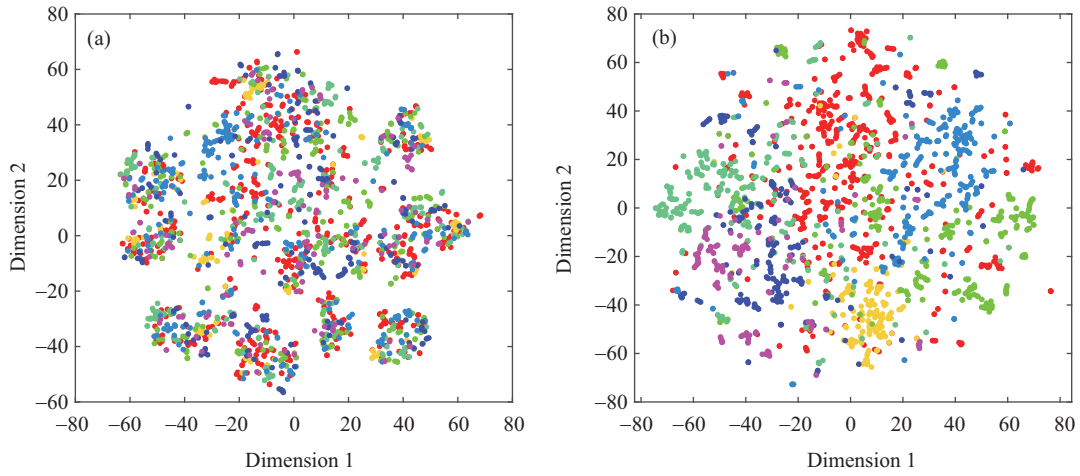

Figure 4 (Color online) t-SNE visualization on the Cora dataset. Different classes are marked by different colors. (a) w/o PPR; (b) w/ PPR.

6 Conclusion

In this paper, we introduced a PPR method for textual network embedding that can aggregate more information from a larger neighborhood. Using the proposed method, we achieve new state-of-the-art results on several real-world link prediction benchmarks.

In addition to our empirical analysis, we analyze our method from a spectral domain theory perspective and manifest PPR as polynomial filters on the graph spectral domain. This provides theoretical evidence for performance improvement.

Given its empirical performance and theoretical interpretability, we argue that the proposed method would be highly beneficial to the research community from a different perspective.

Furthermore, an important future direction is to explore some approximate algorithms to compute

the inversion operation more efficiently, e.g., methods presented by [20, 21]. Also, we suggest the design of neural networks that can learn general textual network embedding paradigms and to explore neural architectural search space of the algorithm structures.

Acknowledgements This work was supported by National Science and Technology Major Projects on Core Electronic Devices, High-End Generic Chips and Basic Software (Grant No. 2018ZX01028101) and National Natural Science Foundation of China (Grant No. 61732018). The authors acknowledge the anonymous reviewers for their valuable comments, which improve the quality of this paper.

References

1 Xu R F, Du J C, Zhao Z S, et al. Inferring user profiles in social media by joint modeling of text and networks. Sci China Inf Sci, 2019, 62: 219104
2 Ng A Y, Jordan M I, Weiss Y. On spectral clustering: analysis and an algorithm. In: Proceedings of Advances in Neural Information Processing Systems 14, Vancouver, 2001. 849–856
3 Zhang Q, Li R, Chu T G. Kernel semi-supervised graph embedding model for multimodal and mixmodal data. Sci China Inf Sci, 2020, 63: 119204
4 Perozzi B, Al-Rfou R, Skiena S. Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, 2014. 701–710
5 Tang J, Qu M, Wang M Z, et al. LINE: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, Florence, 2015. 1067–1077
6 Grover A, Leskovec J. Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 2016. 855–864
7 Qiu J Z, Dong Y X, Ma H, et al. Network embedding as matrix factorization: unifying deepwalk, line, PTE, and node2vec. In: Proceedings of the 11th ACM International Conference on Web Search and Data Mining, Marina Del Rey, 2018. 459–467
8 Yang C, Liu Z Y, Zhao D L, et al. Network representation learning with rich text information. In: Proceedings of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, 2015. 2111–2117
9 Sun X F, Guo J, Ding X, et al. A general framework for content-enhanced network representation learning. 2016. ArXiv:1610.02906
10 Tu C C, Liu H, Liu Z Y, et al. CANE: context-aware network embedding for relation modeling. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, Vancouver, 2017. 1722–1731
11 Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. In: Proceedings of Advances in Neural Information Processing Systems 26, Lake Tahoe, 2013. 3111–3119
12 Page L, Brin S, Motwani R, et al. The pagerank citation ranking: bringing order to the web. 1999. http://courses.washington.edu/ir2010/readings/page.pdf
13 Kim Y. Convolutional neural networks for sentence classification. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, Doha, 2014. 1746–1751
14 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations, Toulon, 2017
15 von Luxburg U. A tutorial on spectral clustering. Stat Comput, 2007, 17: 395–416
16 Defferrard M, Bresson X, Vandergheynst P. Convolutional neural networks on graphs with fast localized spectral filtering. In: Proceedings of Advances in Neural Information Processing Systems 29, Barcelona, 2016. 3837–3845
17 Chung F. The heat kernel as the pagerank of a graph. Proc Natl Acad Sci USA, 2007, 104: 19735–19740
18 Kingma D P, Ba J. Adam: a method for stochastic optimization. In: Proceedings of the 3rd International Conference on Learning Representations, San Diego, 2015
19 van der Maaten L, Hinton G. Visualizing data using t-SNE. J Mach Learn Res, 2008, 9: 2579–2605
20 Wang S B, Yang R C, Xiao X K, et al. FORA: simple and effective approximate single-source personalized pagerank. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, 2017. 505–514
21 Wei Z W, He X D, Xiao X K, et al. TopPPR: top-k personalized pagerank queries with precision guarantees on large graphs. In: Proceedings of International Conference on Management of Data, Houston, 2018. 441–456