PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks

基于大规模异构文本网络的预测性文本嵌入

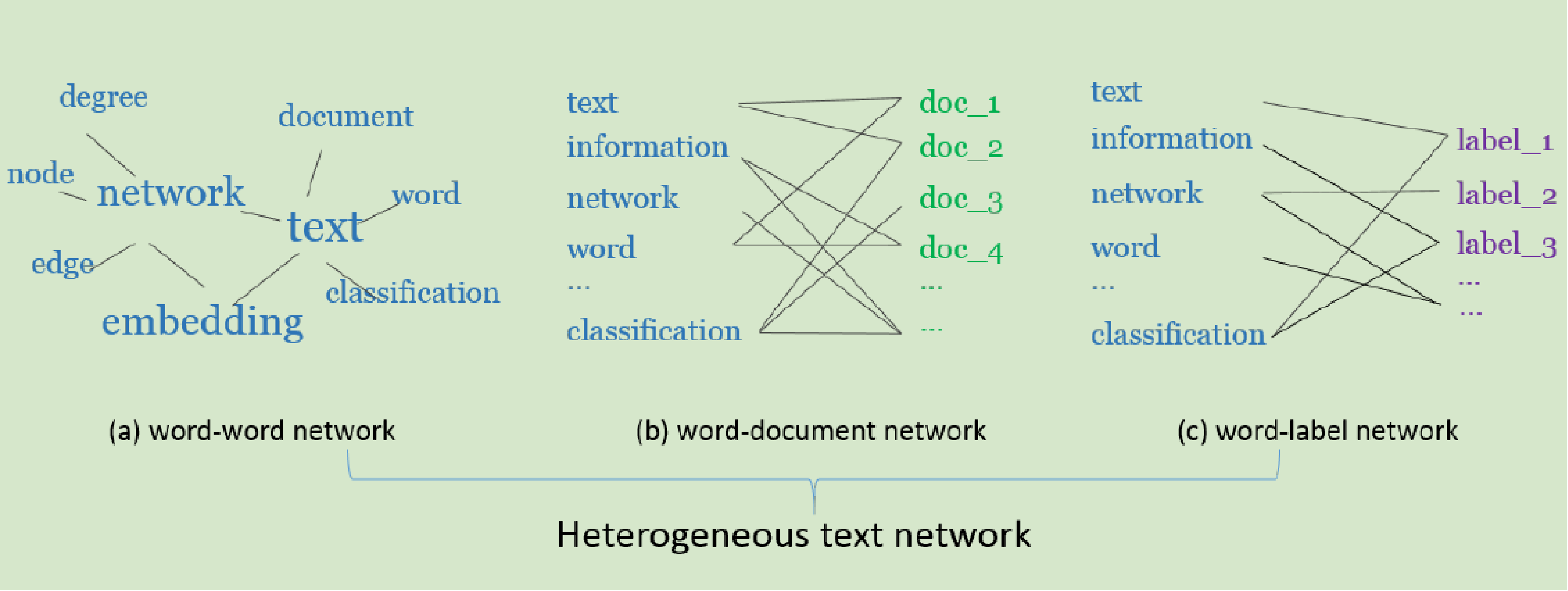
KDD 2015

Jian Tang
Microsoft Research Asia
jiatang@microsoft.com

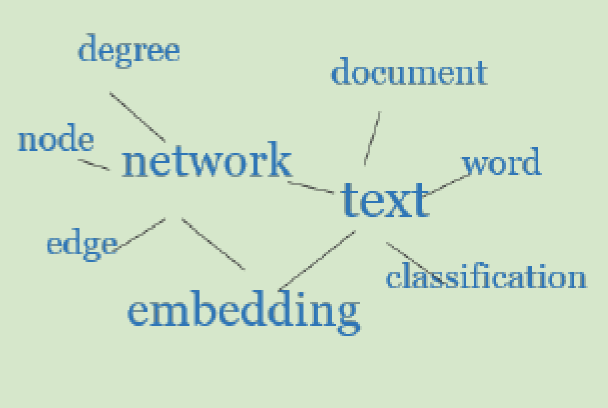
Meng Qu∗
Peking University
mnqu@pku.edu.cn

Qiaozhu Mei
University of Michigan
qmei@umich.edu

Heterogeneous Text Networks



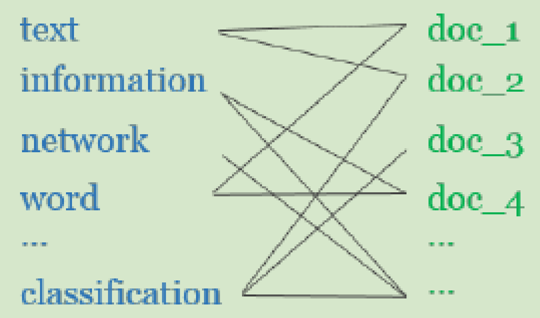
What Is Word-Word Network？



(a) word-word network

DEFINITION 1. ***(Word-Word Network) Word-word*** *co-occurrence network, denoted as $G_{ww} = (\mathcal{V}, E_{ww})$, captures the word co-occurrence information in local contexts of the unlabeled data. $\mathcal{V}$ is a vocabulary of words and E_{ww} is the set of edges between words. The weight w_{ij} of the edge between word v_i and v_j is defined as the number of times that the two words co-occur in the context windows of a given window size.*

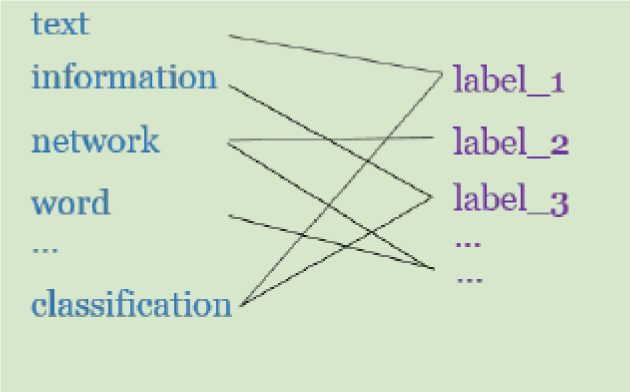
What Is Word-Document Network？



(b) word-document network

Definition 2. ***(Word-Document Network) Word-document*** *network, denoted as* $G_{wd} = (\mathcal{V} \cup \mathcal{D}, E_{wd})$*, is a bipartite network where* $\mathcal{D}$ *is a set of documents and* $\mathcal{V}$ *is a set of words.* E_{wd} *is the set of edges between words and documents. The weight* w_{ij} *between word* v_i *and document* d_j *is simply defined as the number of times* v_i *appears in document* d_j*.*

What Is Word-Label Network？



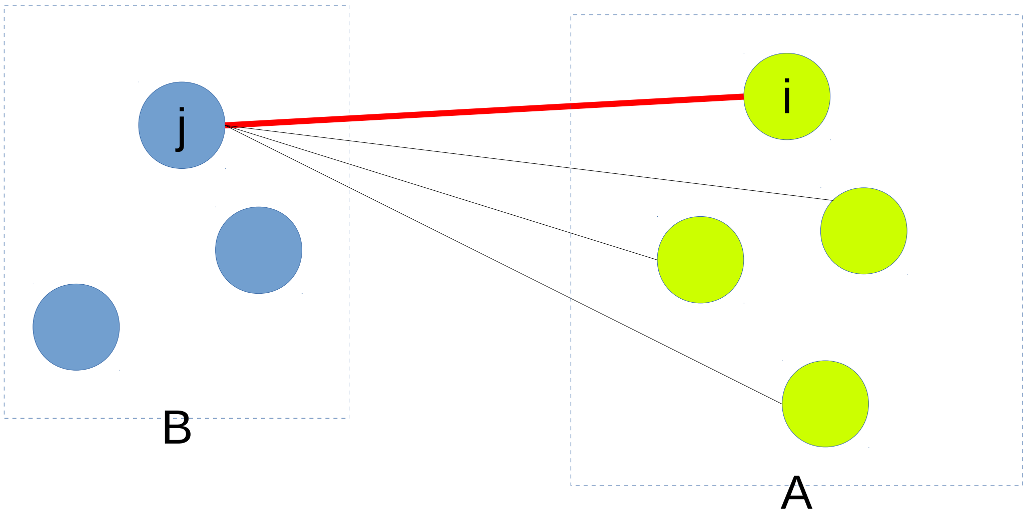
(c) word-label network

DEFINITION 3. ***(Word-Label Network) Word-label*** network, denoted as $G_{wl} = (\mathcal{V} \cup \mathcal{L}, E_{wl})$, is a bipartite network that captures category-level word co-occurrences. $\mathcal{L}$ is a set of class labels and $\mathcal{V}$ a set of words. E_{wl} is a set of edges between words and classes. The weight w_{ij} of the edge between word v_i and class c_j is defined as: $w_{ij} = \sum_{(d:l_d=j)} n_{di}$, where n_{di} is the term frequency of word v_i in document d, and l_d is the class label of document d.

How to Embed？

Given a bipartite network $G = (\mathcal{V}_A \cup \mathcal{V}_B, E)$, where $\mathcal{V}_A$ and $\mathcal{V}_B$ are two disjoint sets of vertices of different types, and E is the set of edges between them. We first define the conditional probability of vertex v_i in set $\mathcal{V}_A$ generated by vertex v_j in set $\mathcal{V}_B$ as:

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}, \tag{1}$$

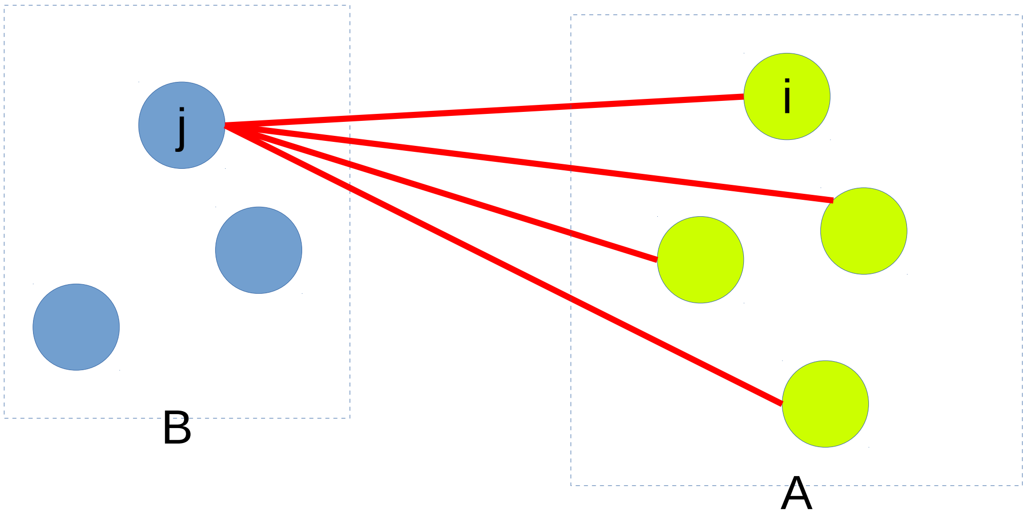


How to Embed？

Given a bipartite network $G = (\mathcal{V}_A \cup \mathcal{V}_B, E)$, where $\mathcal{V}_A$ and $\mathcal{V}_B$ are two disjoint sets of vertices of different types, and E is the set of edges between them. We first define the conditional probability of vertex v_i in set $\mathcal{V}_A$ generated by vertex v_j in set $\mathcal{V}_B$ as:

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}, \tag{1}$$

$p(\cdot|v_j)$



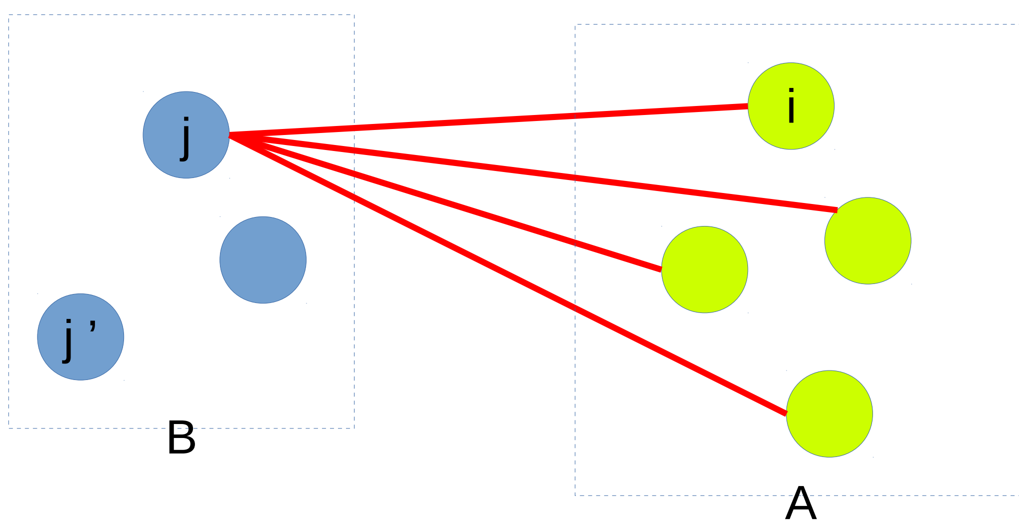
How to Embed？

Given a bipartite network $G = (\mathcal{V}_A \cup \mathcal{V}_B, E)$, where $\mathcal{V}_A$ and $\mathcal{V}_B$ are two disjoint sets of vertices of different types, and E is the set of edges between them. We first define the conditional probability of vertex v_i in set $\mathcal{V}_A$ generated by vertex v_j in set $\mathcal{V}_B$ as:

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}, \tag{1}$$

$$p(\cdot|v_j)$$

$$p(\cdot|v_{j'})$$



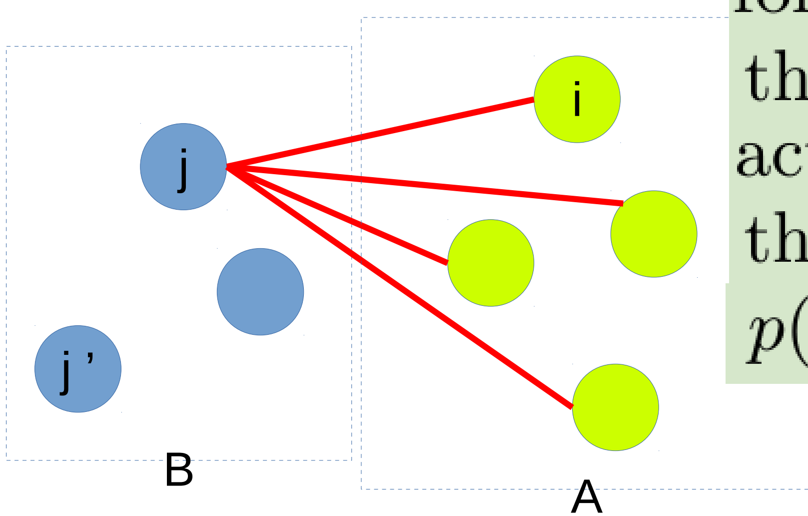
How to Embed？

Given a bipartite network $G = (\mathcal{V}_A \cup \mathcal{V}_B, E)$, where $\mathcal{V}_A$ and $\mathcal{V}_B$ are two disjoint sets of vertices of different types, and E is the set of edges between them. We first define the conditional probability of vertex v_i in set $\mathcal{V}_A$ generated by vertex v_j in set $\mathcal{V}_B$ as:

$$p(v_i|v_j) = \frac{\exp(\vec{u}_i^T \cdot \vec{u}_j)}{\sum_{i' \in A} \exp(\vec{u}_{i'}^T \cdot \vec{u}_j)}, \tag{1}$$

$p(\cdot|v_j)$

$p(\cdot|v_{j'})$



for each pair of vertices $v_j, v_{j'}$, the second-order proximity can actually be determined by their conditional distributions $p(\cdot|v_j), p(\cdot|v_{j'})$.

How to Embed？
Cost Function

$$O = -\sum_{(i,j)\in E} w_{ij} \log p(v_j|v_i). \tag{3}$$

SGD

How to Embed？

- By this way, we can define the conditional probabilities and learn the embeddings by optimizing objective function(3)
  - word-word network $\longrightarrow$ $p(v_i|v_j)$
  - word-document network $\longrightarrow$ $p(v_i|d_j)$
  - work-label network $\longrightarrow$ $p(v_i|l_j)$

$$O_{ww} = -\sum_{(i,j)\in E_{ww}} w_{ij} \log p(v_i|v_j)$$

$$O_{wd} = -\sum_{(i,j)\in E_{wd}} w_{ij} \log p(v_i|d_j)$$

$$O_{wl} = -\sum_{(i,j)\in E_{wl}} w_{ij} \log p(v_i|l_j)$$

Heterogeneous Text Network Embedding

objective function:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}, \tag{4}$$

Heterogeneous Text Network Embedding

objective function:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}, \tag{4}$$

One solution is to train the model with the unlabeled data (the word-word and word-document networks) and the labeled data simultaneously.

We call this approach *joint training*.

Joint training

Algorithm 1: Joint training.

Data: G_{ww}, G_{wd}, G_{wl}, number of samples T, number of negative samples K.

Result: word embeddings $\vec{w}$.

while $iter \leq T$ **do**

- sample an edge from E_{ww} and draw K negative edges, and update the word embeddings;
- sample an edge from E_{wd} and draw K negative edges, and update the word and document embeddings;
- sample an edge from E_{wl} and draw K negative edges, and update the word and label embeddings;

end

Heterogeneous Text Network Embedding

objective function:

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}, \tag{4}$$

An alternative solution is to learn the embeddings with unlabeled data first, and then fine-tune the embeddings with the word-label network.

This is inspired by the idea of *pre-training and fine-tuning*

Pre-training + Fine-tuning

Algorithm 2: Pre-training + Fine-tuning.

Data: G_{ww}, G_{wd}, G_{wl}, number of samples T, number of negative samples K.

Result: word embeddings $\vec{w}$.

while $iter \leq T$ **do**

- sample an edge from E_{ww} and draw K negative edges, and update the word embeddings;
- sample an edge from E_{wd} and draw K negative edges, and update the word and document embeddings;

end

while $iter \leq T$ **do**

- sample an edge from E_{wl} and draw K negative edges, and update the word and label embeddings;

end

Text Embedding

Once the word vectors are learned, the representation of an arbitrary piece of text can be obtained by simply averaging the vectors of the words in that piece of text.

That is, the vector representation of a piece of text $d = w_1 w_2 \cdots, w_n$ can be computed as

$$\vec{d} = \frac{1}{n} \sum_{i=1}^{n} \vec{u}_i, \tag{8}$$

Revisiting Semi-Supervised Learning with Graph Embeddings

基于图嵌入的半监督学习

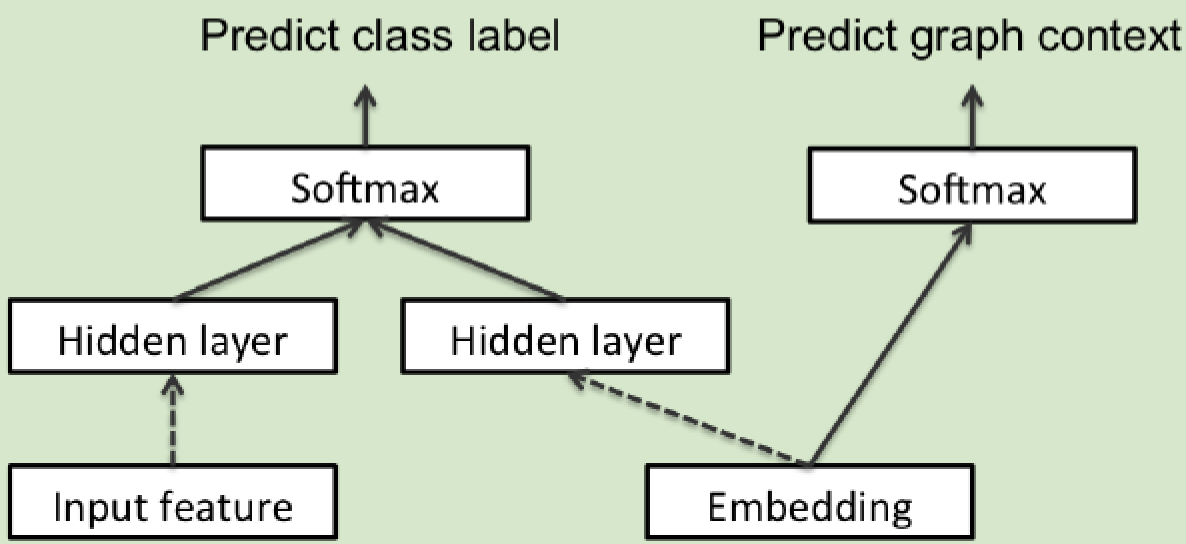
ICML 2016

Zhilin Yang ZHILINY@CS.CMU.EDU
William W. Cohen WCOHEN@CS.CMU.EDU
Ruslan Salakhutdinov RSALAKHU@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University

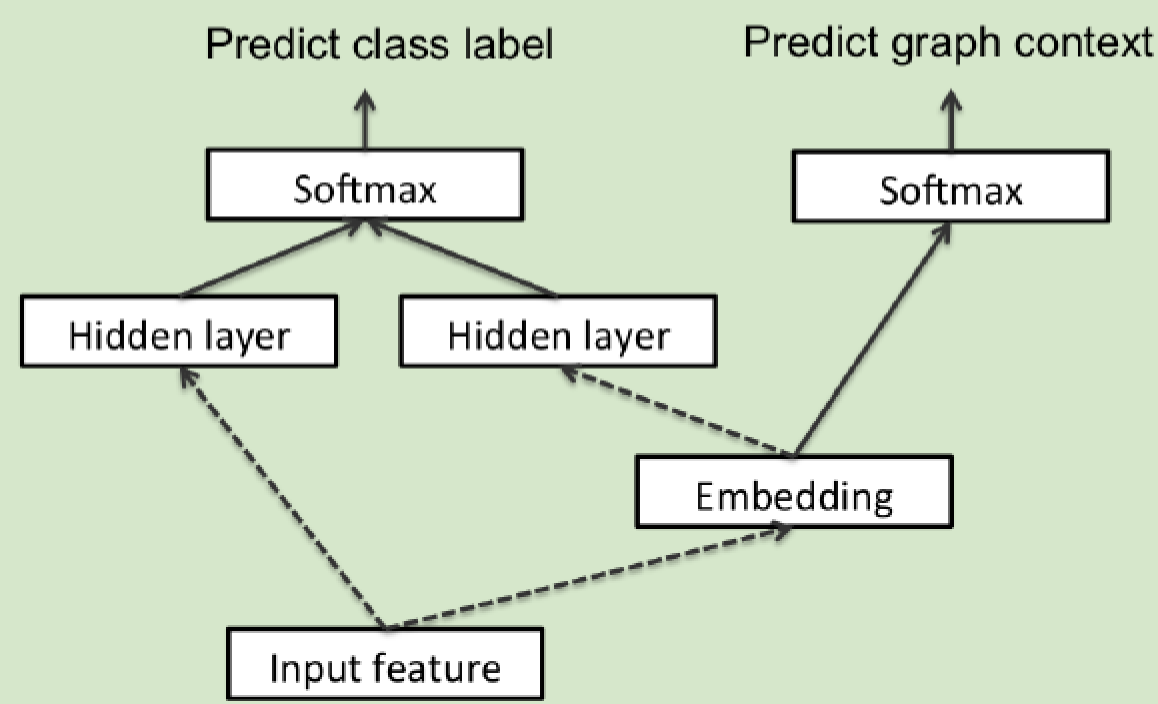
Main Work

The main highlight of our work is to incorporate embedding techniques into the graph-based semi-supervised learning setting.

There are
two learning paradigms, transductive learning and inductive learning.



(a) Transductive Formulation



(b) Inductive Formulation

Semi-Supervised Learning with Graph Embeddings

We formulate our framework based on feed-forward neural networks.

Given the input feature vector $\mathbf{x}$, the k-th hidden layer of the network is denoted as $\mathbf{h}^k$, which is a nonlinear function of the previous hidden layer $\mathbf{h}^{k-1}$ defined as: $\mathbf{h}^k(\mathbf{x}) = \text{ReLU}(\mathbf{W}^k\mathbf{h}^{k-1}(\mathbf{x}) + b^k)$, where $\mathbf{W}^k$ and b^k are parameters of the k-th layer, and $\mathbf{h}^0(\mathbf{x}) = \mathbf{x}$. We adopt rectified linear unit $\text{ReLU}(x) = \max(0, x)$ as the nonlinear function in this work.

Cost Function

The loss function of our framework can be expressed as

$$\mathcal{L}_s + \lambda \mathcal{L}_u,$$

where $\mathcal{L}_s$ is a supervised loss of predicting the labels, and $\mathcal{L}_u$ is an unsupervised loss of predicting the graph context.

Unsupervised Loss

the unsuper-
vised loss with negative sampling can be written as

$$\mathcal{L}_u = -\mathbb{E}_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i) \tag{3}$$

Label Loss

$$p(y|\mathbf{x},\mathbf{e}) = \frac{\exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_y}{\sum_{y'} \exp[\mathbf{h}^k(\mathbf{x})^T, \mathbf{h}^l(\mathbf{e})^T]\mathbf{w}_{y'}}, \qquad (4)$$

$[\cdot,\cdot]$ denotes concatenation of two row vectors and $\mathbf{w}$ represents the model parameter.

Cost Function

Combined with Eq. (3), the loss function of transductive learning is defined as:

$$-\frac{1}{L}\sum_{i=1}^{L}\log p(y_i|\mathbf{x}_i, \mathbf{e}_i) - \lambda\mathbb{E}_{(i,c,\gamma)}\log\sigma(\gamma\mathbf{w}_c^T\mathbf{e}_i),$$

Model Training

Algorithm 2 Model Training (Transductive)

Input: A, $\mathbf{x}_{1:L+U}$, $y_{1:L}$, λ, batch iterations T_1, T_2 and sizes N_1, N_2

repeat

  for $t \leftarrow 1$ **to** T_1 **do**

    Sample a batch of labeled instances i of size N_1

    $\mathcal{L}_s = -\frac{1}{N_1} \sum_i p(y_i | \mathbf{x}_i, \mathbf{e}_i)$

    Take a gradient step for $\mathcal{L}_s$

  end for

  for $t \leftarrow 1$ **to** T_2 **do**

    Sample a batch of context from $p(i, c, \gamma)$ of size N_2

    $\mathcal{L}_u = -\frac{1}{N_2} \sum_{(i,c,\gamma)} \log \sigma(\gamma \mathbf{w}_c^T \mathbf{e}_i)$

    Take a gradient step for $\mathcal{L}_u$

  end for

until stopping

Supervised Training on label

1.2"

Unsupervised Training on context

1.2"

SGD with mini-batch mode

TransNet: Translation-Based Network Representation Learning for Social Relation Extraction

用于社交关系提取的基于转移的网络表示学习

AAAI

Cunchao Tu[1,2], **Zhengyan Zhang**[1], **Zhiyuan Liu**[1,2*], **Maosong Sun**[1,2]

[1]Department of Computer Science and Technology, State Key Lab on Intelligent Technology and Systems, National Lab for Information Science and Technology, Tsinghua University, Beijing, China

[2]Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou, China

What Does “Translation” Mean?

“Translation” here means the movement that changes the position of a vector in representation space.

Translation Mechanism

Specifically, for each edge $e = (u, v)$ and its corresponding label set l, the representation of vertex v is expected to be close to the representation of vertex u plus the representation of edge e.

the translation mechanism among u, v and e can be formalized as

$$\mathbf{u} + \mathbf{l} \approx \mathbf{v}'. \tag{1}$$

head vertex

tail vertex

Translation Mechanism Cost

Specifically, for each edge $e = (u, v)$ and its corresponding label set l, the representation of vertex v is expected to be close to the representation of vertex u plus the representation of edge e.

We employ a distance function $d(\mathbf{u} + \mathbf{l}, \mathbf{v}')$ to estimate the degree of (u, v, l) that matches the Eq. (1). In practice, we simply adopt L_1-norm.

With the above definitions, for each (u, v, l) and its negative sample $(\hat{u}, \hat{v}, \hat{l})$, the translation part of TransNet aims to minimize the hinge-loss as follows:

$$\mathcal{L}_{trans} = \max(\gamma + d(\mathbf{u} + \mathbf{l}, \mathbf{v}') - d(\hat{\mathbf{u}} + \hat{\mathbf{l}}, \hat{\mathbf{v}}'), 0), \tag{2}$$

where $\gamma > 0$ is a margin hyper-parameter

What Is Social Relation Extraction？

Formally, we define the problem of SRE as follows. Suppose there is a social network $G = (V, E)$, where V is the set of vertices, and $E \subseteq (V \times V)$ are edges between vertices. Besides, the edges in E are partially labeled, denoted as E_L.

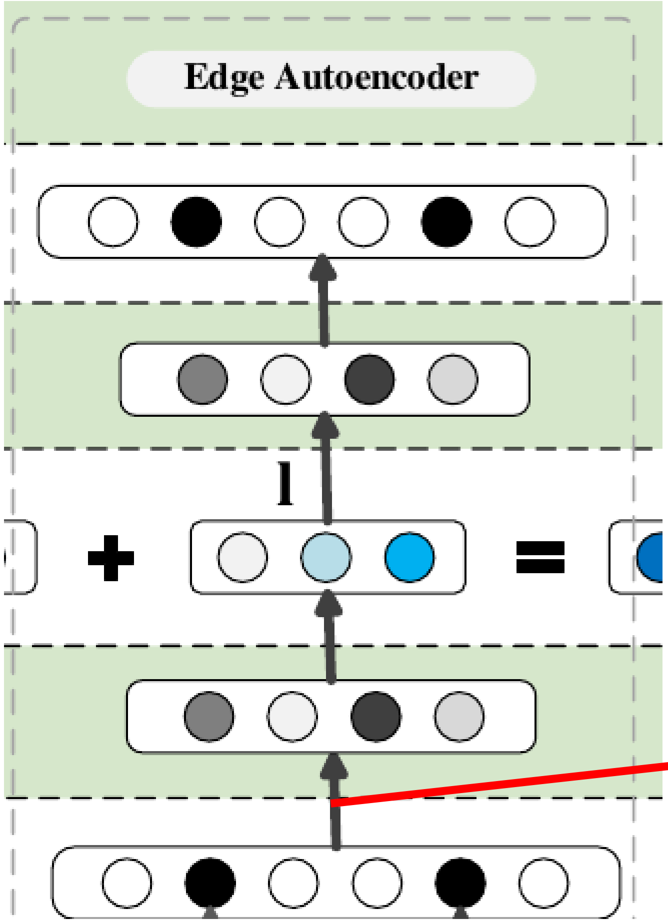
Finally, given the overall network structure and the labeled edges in E_L, SRE aims to predict the labels over unlabeled edges in E_U, where $E_U = E - E_L$ represents the unlabeled

Main Work

In this work, we focus on the problem of incorporating rich relation information on edges into NRL.

Edge Representation Construction

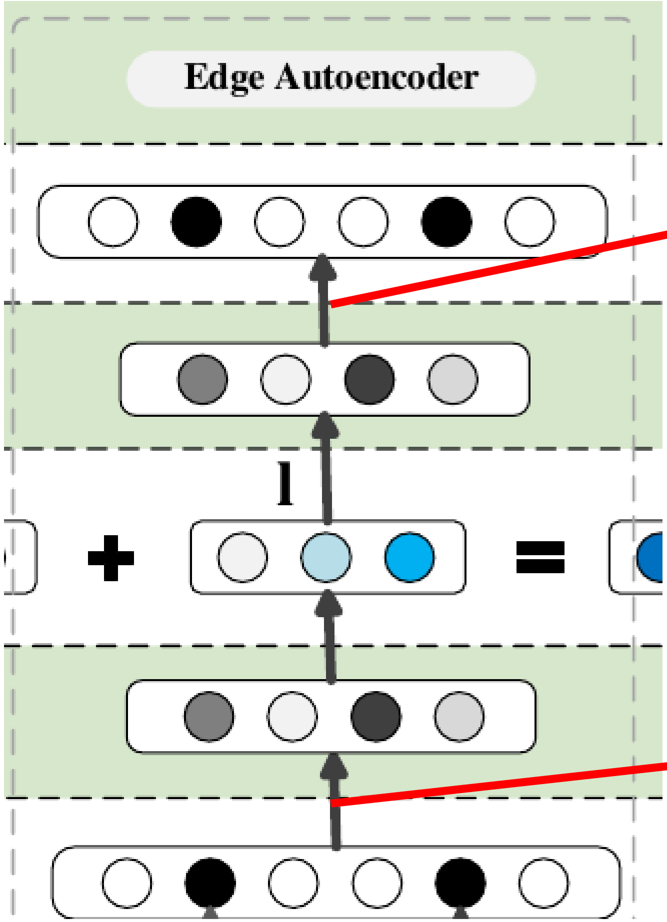
we employ a deep autoencoder to construct the edge representations.



$$\mathbf{h}^{(1)} = f(\mathbf{W}^{(1)}\mathbf{s} + \mathbf{b}^{(1)}),$$
$$\mathbf{h}^{(i)} = f(\mathbf{W}^{(1)}\mathbf{h}^{(i-1)} + \mathbf{b}^{(i)}), i = 2, \dots, K.$$

Edge Representation Construction

we employ a deep autoencoder to construct the edge representations.

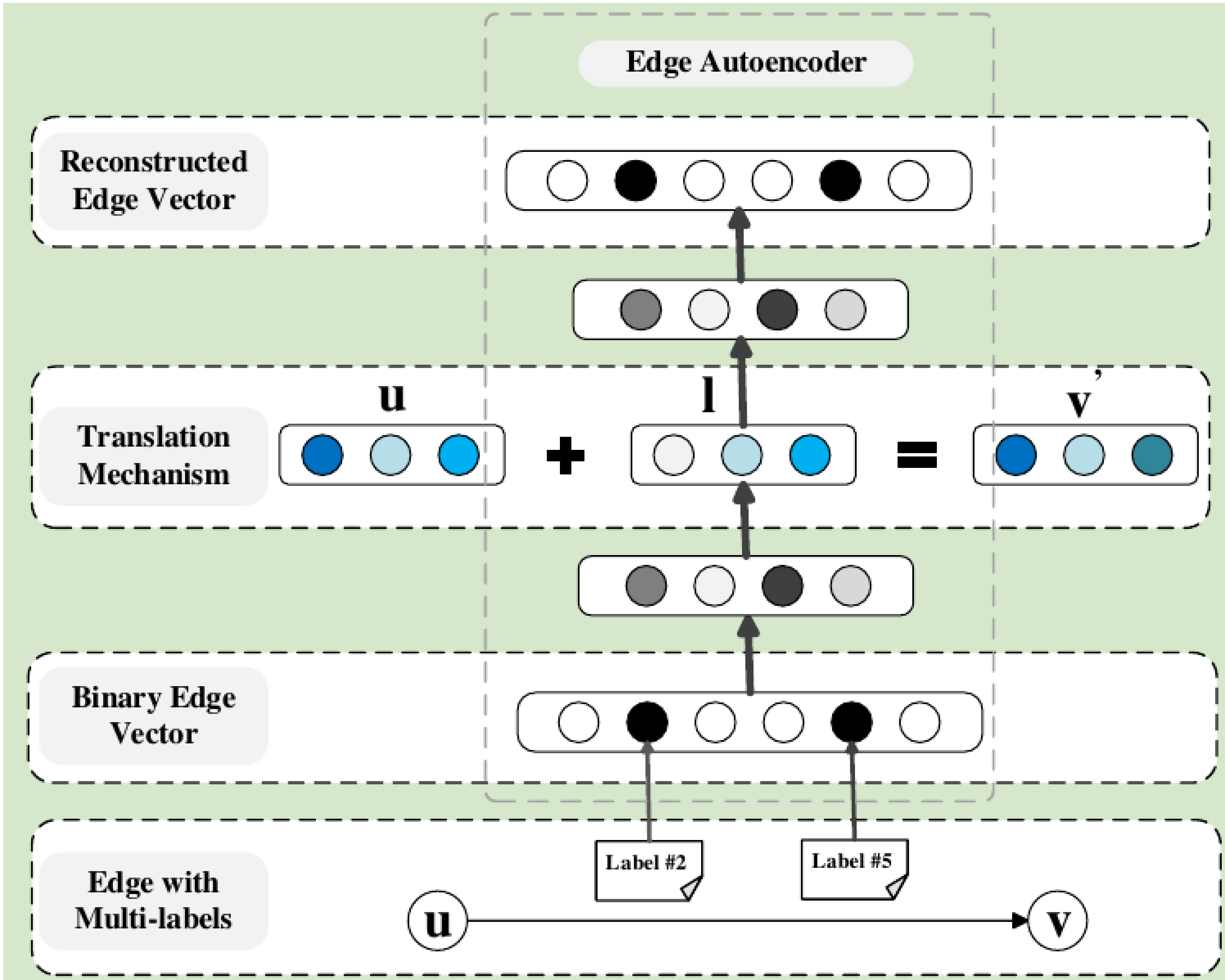


sigmoid activation function

$$\mathbf{h}^{(1)} = f(\mathbf{W}^{(1)}\mathbf{s} + \mathbf{b}^{(1)}),$$
$$\mathbf{h}^{(i)} = f(\mathbf{W}^{(1)}\mathbf{h}^{(i-1)} + \mathbf{b}^{(i)}), i = 2, \ldots, K.$$

Translation Mechanism
&
Edge Representation
?

&



Edge Autoencoder
Reconstructed Edge Vector
Translation Mechanism
u
l
v'
+
=
Binary Edge Vector
Edge with Multi-labels
Label #2
Label #5
u
v

Reconstruction Loss

Reconstruction Loss: Autoencoder aims to minimize the distance between inputs and the reconstructed outputs. The reconstruction loss is shown as:

$$\mathcal{L}_{rec} = ||\mathbf{s} - \hat{\mathbf{s}}||. \quad (5)$$

Reconstruction Loss

Reconstruction Loss: Autoencoder aims to minimize the distance between inputs and the reconstructed outputs. The reconstruction loss is shown as:

$$\mathcal{L}_{rec} = ||\mathbf{s} - \hat{\mathbf{s}}||. \quad (5)$$

However, due to the sparsity of the input vector, the number of zero elements in $\mathbf{s}$ is much larger than that of non-zero elements.

Reconstruction Loss

$$\mathcal{L}_{rec} = ||\mathbf{s} - \hat{\mathbf{s}}||. \tag{5}$$

However, due to the sparsity of the input vector, the number of zero elements in s is much larger than that of non-zero elements.

$$\mathcal{L}_{ae} = ||(\mathbf{s} - \hat{\mathbf{s}}) \odot \mathbf{x}||, \tag{6}$$

where $\mathbf{x}$ is a weight vector and $\odot$ means the Hadamard product. For $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{|T|}$, $\mathbf{x}_i = 1$ when $\mathbf{s}_i = 0$ and $\mathbf{x}_i = \beta > 1$ otherwise.

Overall Cost

$$\mathcal{L} = \mathcal{L}_{trans} + \alpha[\mathcal{L}_{ae}(l) + \mathcal{L}_{ae}(\hat{l})] + \eta\mathcal{L}_{reg}. \tag{7}$$

Here, we introduce two hyper-parameters α and η to balance the weights of different parts. Besides, $\mathcal{L}_{reg}$ is an $L2$-norm regularizer to prevent overfitting, which is defined as

$$\mathcal{L}_{reg} = \sum_{i=1}^{K}(||W^{(i)}||_2^2 + ||b^{(i)}||_2^2). \tag{8}$$

One More Thing

Thanks for Listening

Preferences

- [1]First-order Proximity
  - https://blog.csdn.net/github_36326955/article/details/72528656
- [2]KL-divergence
  - https://baike.baidu.com/item/%E7%9B%B8%E5%AF%B9%E7%86%B5/4233536?fr=aladdin
- [3] 《PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks》