中國人民大學
RENMIN UNIVERSITY OF CHINA

Part III: Network Embedding with Deep Learning

Xin Zhao

batmanfly@qq.com

Renmin University of China

Outline

- Preliminaries
  - word2vec
- Network Embedding Models
  - DeepWalk
  - Node2vec
  - GENE
  - LINE
  - SDNE
- Applications of Network Embedding
  - Basic applications
  - Visualization
  - Text classification
  - Recommendation
- Conclusion

Preliminaries

- Softmax functions
- Distributional semantics
- Word2vec
  - CBOW
  - Skip-gram

Preliminaries

- Representation learning
  - Using machine learning techniques to derive data representation
- Distributed representation
  - Different from one-hot representation, it uses dense vectors to represent data points
- Embedding
  - Mapping information entities into a low-dimensional space

Softmax function

- It transforms a *K*-dimensional real vector into a *probability distribution*
  - A common transformation function to derive objective functions for classification or discrete variable modeling

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad \text{for } j = 1, \cdots, K.$$

Distributional semantics

- Target word = “stars”

```
he curtains open and the stars shining in on the barely
ars and the cold , close stars " . And neither of the w
rough the night with the stars shining so brightly , it
made in the light of the stars . It all boils down , wr
 surely under the bright stars , thrilled by ice-white
sun , the seasons of the stars ? Home , alone , Jay pla
m is dazzling snow , the stars have risen full and cold
un and the temple of the stars , driving out of the hug
 in the dark and now the stars rise , full and amber a
bird on the shape of the stars over the trees in front
 But I could n't see the stars or the moon , only the
 they love the sun , the stars and the stars . None of
r the light of the shiny stars . The plash of flowing w
man 's first look at the stars ; various exhibits , aer
rief information on both stars and constellations, inc
```

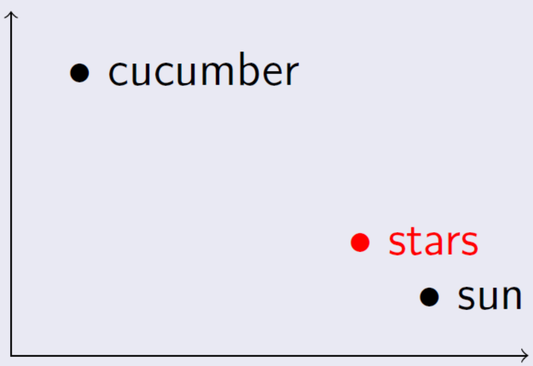
Distributional semantics

- Collect the contextual words for "stars"

Construct vector representations

	shining	bright	trees	dark	look
stars	38	45	2	27	12

Similarity in meaning as vector similarity



Word2Vec

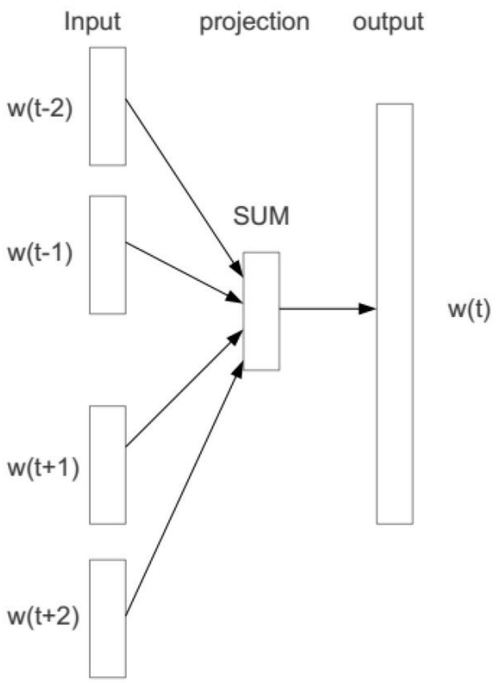
- Input: a sequence of words from a vocabulary ***V***

- Output: a fixed-length vector for each term in the vocabulary
  - $\mathbf{v}_w$

It implements the idea of distributional semantics using a shallow neural network model.

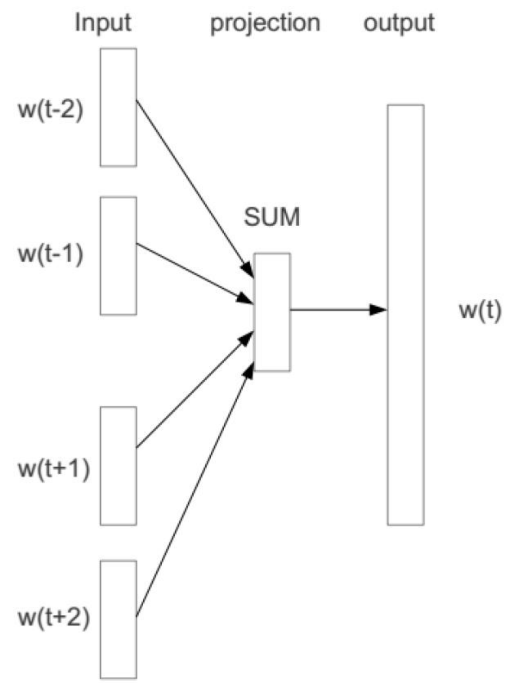
Architecture 1: CBOW

- **CBOW** predicts the current word using surrounding contexts
  - $Pr(w_t \mid \mathrm{context}(w_t))$
    - Window size 2c
    - $\mathrm{context}(w_t)$ = $[w_{t-c}, \ldots, w_{t+c}]$



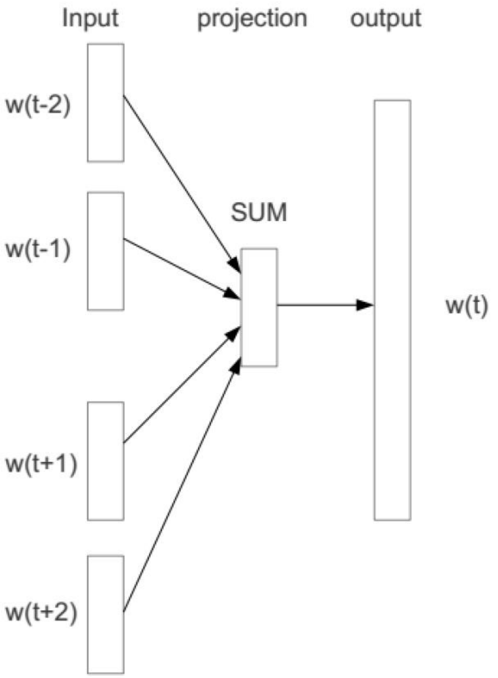
Architecture 1: CBOW

- **<u>CBOW</u>** predicts the current word using surrounding contexts
  - $Pr(w_t | \text{context}(w_t))$
  - Using a K-dimensional vector to represent words
    - $w_t \rightarrow \boldsymbol{v}_{w_t}$
    - $\tilde{\boldsymbol{v}}_{w_t} = \frac{\sum_{i=t-c}^{t+c} \boldsymbol{v}_{w_i}}{2c} \qquad (i \neq t)$



Architecture 1: CBOW

- **<u>CBOW</u>** predicts the current word using surrounding contexts
  - $Pr(w_t \mid \text{context}(w_t))$
  - Basic Idea
    - Given the context of the current word $\tilde{\boldsymbol{v}}_{w_t}$
    - $\text{Sim}(\tilde{\boldsymbol{v}}_{w_t}, \boldsymbol{v}_{w_t}) > \text{Sim}(\tilde{\boldsymbol{v}}_{w_t}, \boldsymbol{v}_{w_j})$



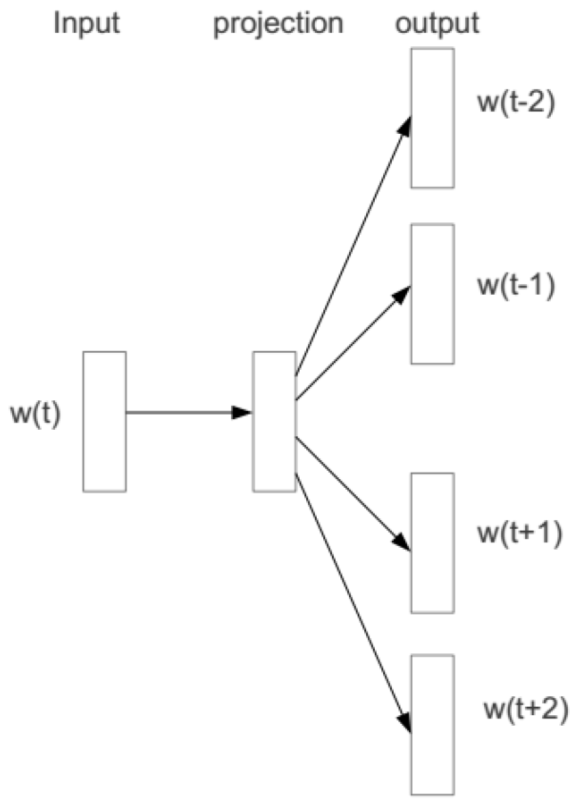
Architecture 1: CBOW

- How to formulate the idea
  - Using a softmax function
  - Considered as a classification problem
    - Each word is a classification label

$$P(w|w_{\text{context}}) = \frac{\exp(sim(\tilde{\boldsymbol{v}}_w , \boldsymbol{v}_w))}{\sum_{w'} \exp(sim(\tilde{\boldsymbol{v}}_w , \boldsymbol{v}_{w'}))}$$

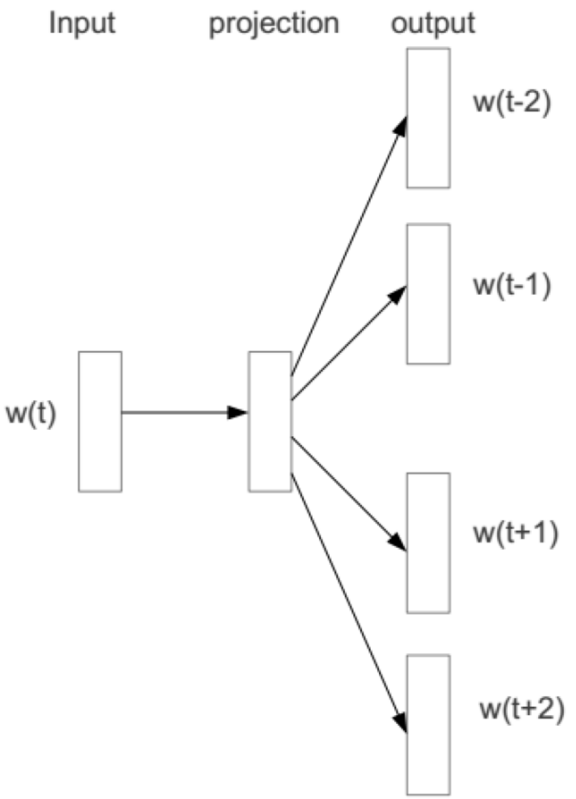
Architecture 2

- **<u>Skip-gram</u>** predicts surrounding words using the current word
  - $Pr(\text{context}(w_t) \mid w_t)$
    - Window size 2c
    - $\text{context}(w_t)$ = $[w_{t-c}, \ldots, w_{t+c}]$



Architecture 2

- **<u>Skip-gram</u>** predicts surrounding words using the current word
  - $Pr(\text{context}(w_t) \mid w_t)$
    - Window size 2c
    - $\text{context}(w_t)$ = $[w_{t-c}, \dots, w_{t+c}]$



$$P(w'|w) = \frac{\exp(sim(\boldsymbol{v}_w , \boldsymbol{v}_{w'}))}{\sum_{w''} \exp(sim(\boldsymbol{v}_w , \boldsymbol{v}_{w''}))}$$

Network Embedding Models

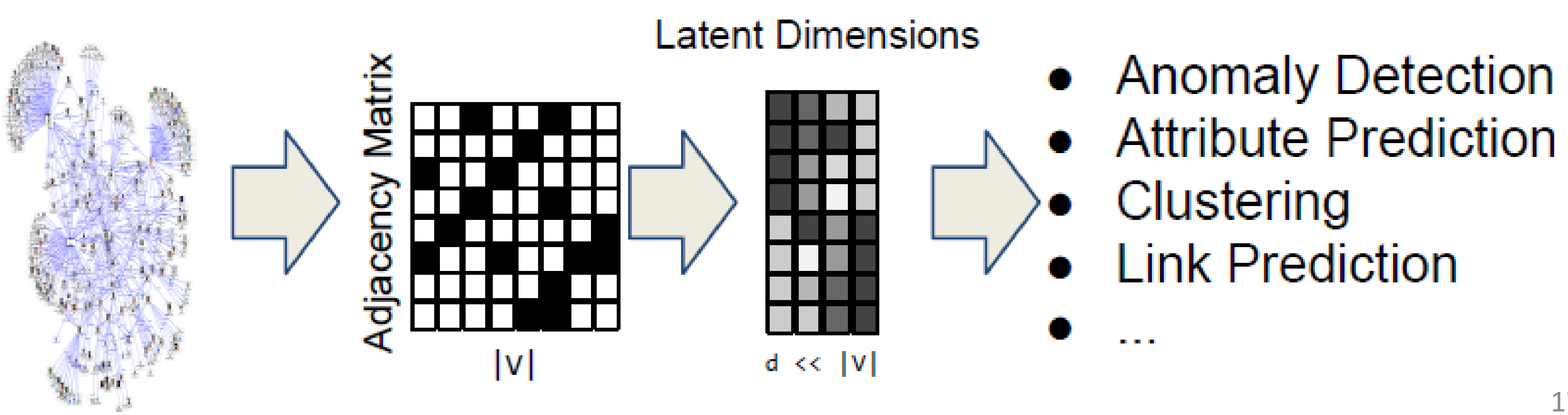
- DeepWalk
- Node2vec
- GENE
- LINE
- SDNE

Network Embedding Models

- **DeepWalk** (Perozzi et al., KDD 2014)
- Node2vec
- GENE
- LINE
- SDNE

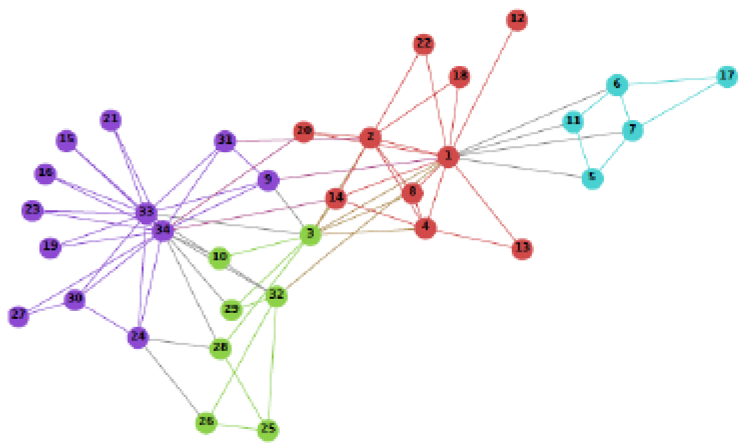
What is network embedding?

- We map each node in a network into a low-dimensional space
  - Distributed representation for nodes
  - Similarity between nodes indicate the link strength
  - Encode network information and generate node representation

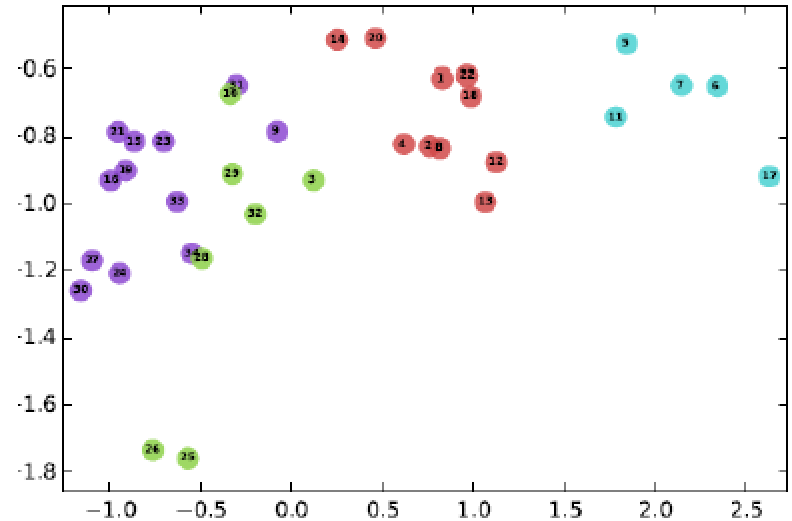


Example

- Zachary’s Karate Network:



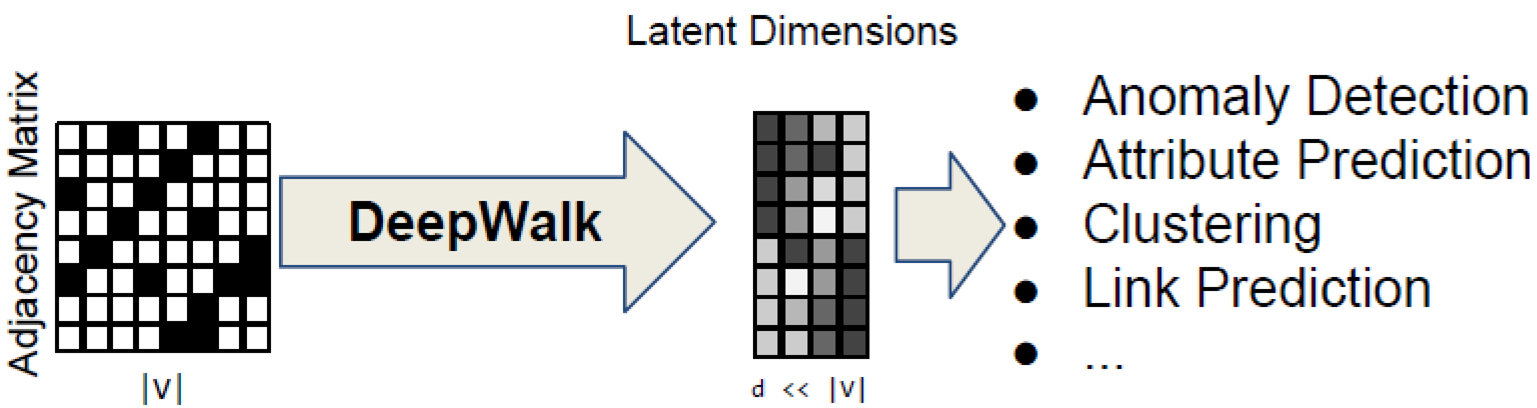
Input



Output

DeepWalk

- DeepWalk learns a latent representation of adjacency matrices using deep learning techniques developed for language modeling



Language modeling

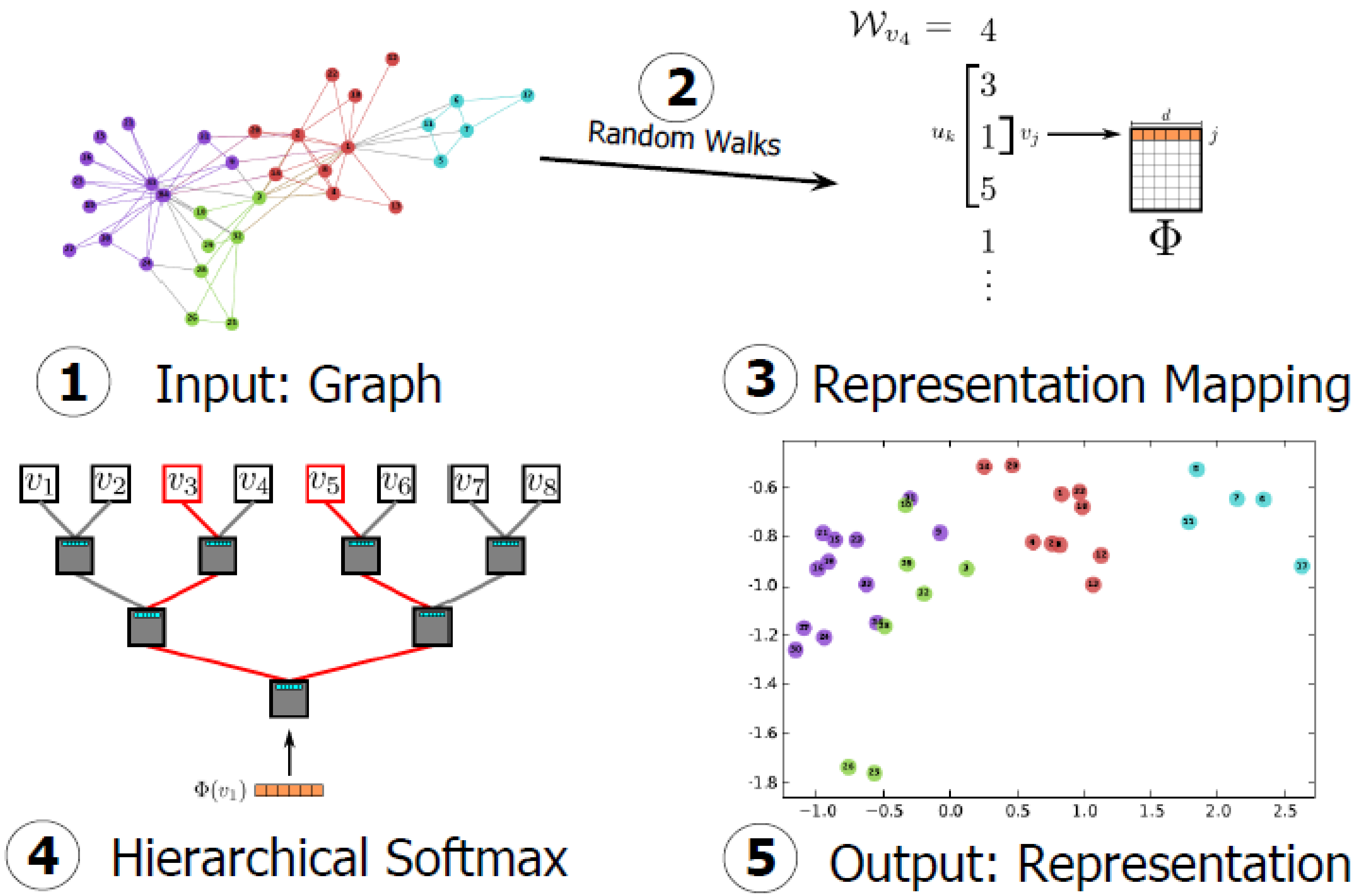
- Learning a representation of a word from documents (word co-occurrence):
  - word2vec: $\Phi: v \in V \mapsto \mathbb{R}^{|V| \times d}$
- The learned representations capture inherent structure
- Example:

$$||\Phi(rose) - \Phi(daisy)|| < ||\Phi(rose) - \Phi(tiger)||$$

From language modeling to graphs

- Idea:
  - Nodes <--> Words
  - Node sequences <--> Sentences
- Generating node sequences:
  - Using random walks
    - short random walks = sentences
- Connection:
  - **Words frequency** in a natural language corpus follows a power law.
  - **Vertex frequency** in random walks on scale free graphs also follows a power law.

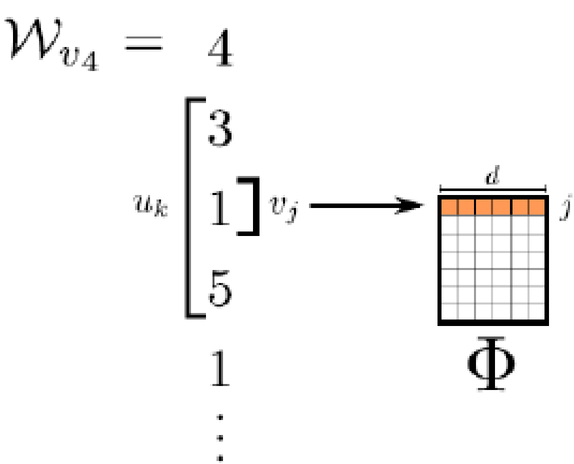
Framework



2
Random Walks
$\mathcal{W}_{v_4} = 4$
3
u_k 1 v_j
5
1
d
j
Φ
1 Input: Graph
3 Representation Mapping
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$\Phi(v_1)$
-0.6 -0.8 -1.0 -1.2 -1.4 -1.6 -1.8
−1.0 −0.5 0.0 0.5 1.0 1.5 2.0 2.5
4 Hierarchical Softmax
5 Output: Representation

Representation Mapping

$$\mathcal{W}_{v_4} = v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_1 \rightarrow v_{46} \rightarrow v_{51} \rightarrow v_{89}$$



- Map the vertex under focus (v_1) to its representation.
- Define a window of size w
- If w = 1 and v = v_1

Maximize: $\Pr(v_3|\Phi(v_1))$

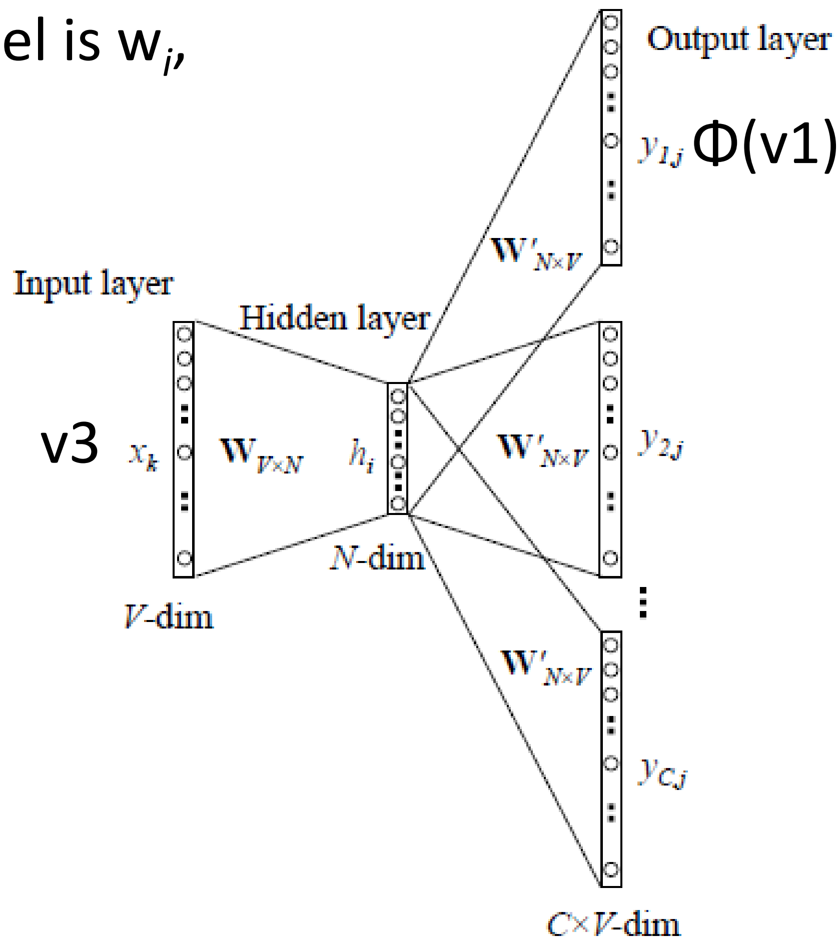
$\Pr(v_5|\Phi(v_1))$

Deep Learning Structure: Skip-gram model

Skip-gram: The input to the model is w_i, and the output could be $w_{i-1}, w_{i-2}, w_{i+1}, w_{i+2}$

Maximize: $\Pr(v_3 | \Phi(v_1))$

$\Pr(v_5 | \Phi(v_1))$



Experiments

- Node Classification
  – Some nodes have labels, some don't
- DataSet
  – BlogCatalog
  – Flickr
  – YouTube

Results: BlogCatalog

	% Labeled Nodes	10%	20%	30%	40%	50%	60%	70%	80%	90%
Micro-F1(%)	DeepWalk	**36.00**	**38.20**	**39.60**	**40.30**	**41.00**	**41.30**	41.50	41.50	42.00
	SpectralClustering	31.06	34.95	37.27	38.93	39.97	40.99	**41.66**	**42.42**	**42.62**
	EdgeCluster	27.94	30.76	31.85	32.99	34.12	35.00	34.63	35.99	36.29
	Modularity	27.35	30.74	31.77	32.97	34.09	36.13	36.08	37.23	38.18
	wvRN	19.51	24.34	25.62	28.82	30.37	31.81	32.19	33.33	34.28
	Majority	16.51	16.66	16.61	16.70	16.91	16.99	16.92	16.49	17.26
Macro-F1(%)	DeepWalk	**21.30**	**23.80**	25.30	26.30	27.30	27.60	27.90	28.20	28.90
	SpectralClustering	19.14	23.57	**25.97**	**27.46**	**28.31**	**29.46**	**30.13**	**31.38**	**31.78**
	EdgeCluster	16.16	19.16	20.48	22.00	23.00	23.64	23.82	24.61	24.92
	Modularity	17.36	20.00	20.80	21.85	22.65	23.41	23.89	24.20	24.97
	wvRN	6.25	10.13	11.64	14.24	15.86	17.18	17.98	18.86	19.57
	Majority	2.52	2.55	2.52	2.58	2.58	2.63	2.61	2.48	2.62

Network Embedding Models

- DeepWalk
- **Node2vec** (Grover et al., KDD 2016)
- GENE
- LINE
- SDNE

Node2Vec

- A generalized version of DeepWalk
  - Objective function

$$\max_f \quad \sum_{u \in V} \log Pr(N_S(u)|f(u)).$$

  - Conditional independence

$$Pr(N_S(u)|f(u)) = \prod_{n_i \in N_S(u)} Pr(n_i|f(u)).$$

  - Symmetry in feature space

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))}.$$

Node2Vec

$N_S(u) \subset V$

– a network neighborhood of node *u* generated through a neighborhood sampling strategy ***S***.

– The key lies in how to find a neighbor on the graph

– How *DeepWalk* solve this?

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

where π_{vx} is the unnormalized transition probability between nodes v and x, and Z is the normalizing constant.

How Node2vec Do this?

- Motivation

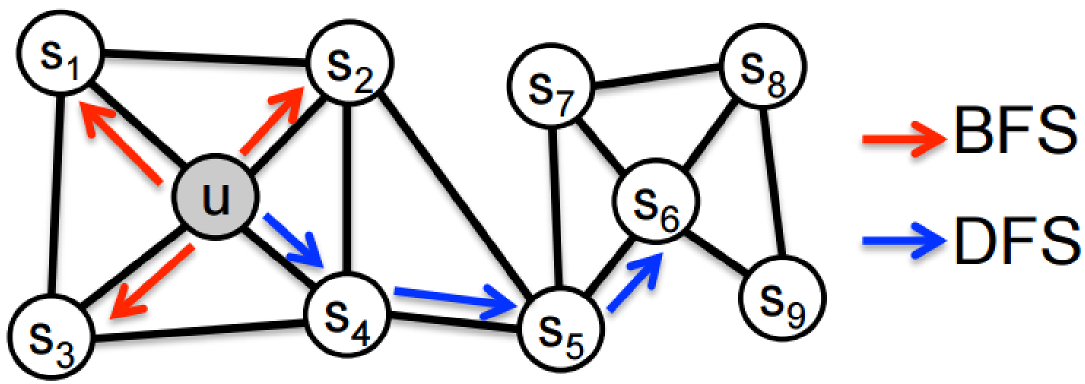


Figure 1: BFS and DFS search strategies from node u ($k = 3$).

  - BFS: broader → ***homophily***
  - DFS: deeper → ***structural equivalence***

How Node2vec Do this?

- Can we combine the merits of DFS and BFS
  - BFS: broader → ***homophily***
  - DFS: deeper → ***structural equivalence***

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

How Node2vec Do this?

- Explaining the sampling strategy

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v, x) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$\pi_{vx} = \alpha_{pq}(t, x) \cdot w_{vx}$$

$$\alpha_{pq}(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Return parameter, p. Parameter p controls the likelihood of immediately revisiting a node in the walk.

In-out parameter, q. Parameter q allows the search to differentiate between "inward" and "outward" nodes.

Node2vec Algorithm

Algorithm 1 The *node2vec* algorithm.

LearnFeatures (Graph $G = (V, E, W)$, Dimensions d, Walks per node r, Walk length l, Context size k, Return p, In-out q)
    π = PreprocessModifiedWeights(G, p, q)
    $G' = (V, E, \pi)$
    Initialize $walks$ to Empty
    for $iter = 1$ **to** r **do**
        for all nodes $u \in V$ **do**
            $walk$ = node2vecWalk(G', u, l)
            Append $walk$ to $walks$
    f = StochasticGradientDescent($k, d, walks$)
    return f

node2vecWalk (Graph $G' = (V, E, \pi)$, Start node u, Length l)
    Inititalize $walk$ to $[u]$
    for $walk\_iter = 1$ **to** l **do**
        $curr = walk[-1]$
        V_{curr} = GetNeighbors($curr, G'$)
        s = AliasSample(V_{curr}, π)
        Append s to $walk$
    return $walk$

Comparison between DeepWalk and Node2vec

- They actually have the same objective function and formulations

- The difference lies in how to generate random walks

- BEAUTY: node → word, path → sentence

Network Embedding Models

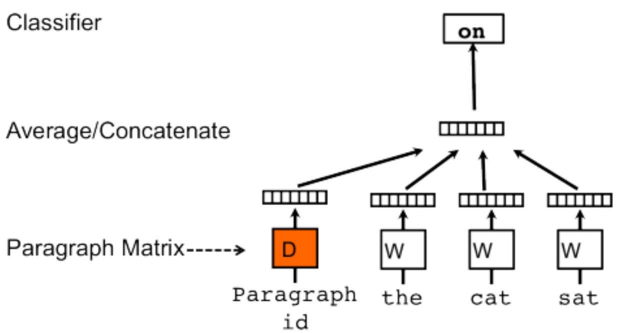
- DeepWalk
- Node2vec
- **GENE** (Chen et al., CIKM 2016)
- LINE
- SDNE

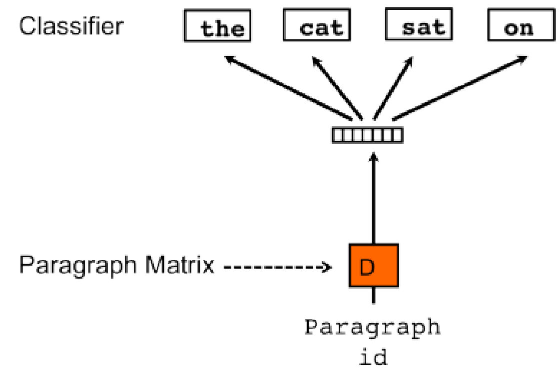
GENE

- Incorporate Group Information to Enhance Network Embedding
  - When group information is available, how to model it?
    - Group $\rightarrow_{\text{control}}$ member

GENE

- Recall doc2vec

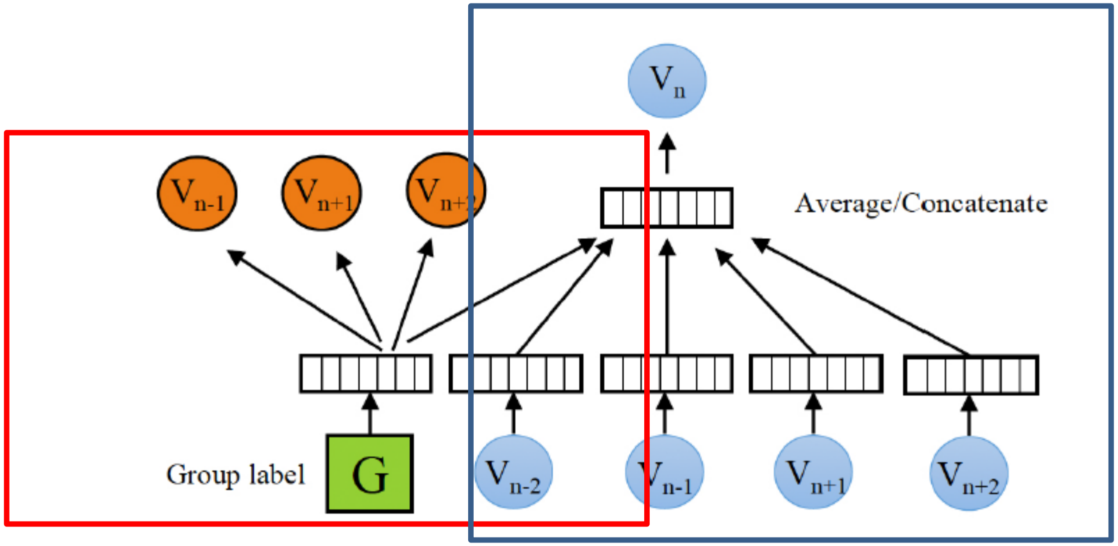




- How to use doc2vec to model group and member vectors

GENE

- Incorporate Group Information to Enhance Network Embedding
  - When group information is available, how to model it?



GENE

- Formulate the idea

$$
\begin{aligned}
\mathscr{L} = \sum_{g_i \in C} (\alpha \sum_{W \in W_{g_i}} \sum_{v_j \in W} \underline{\log p(v_j | v_{j-k}, \ldots, v_{j+k}, g_i)} + \\
\beta \sum_{\hat{v}_j \in \hat{W}_{g_i}} \underline{\log p(\hat{v}_j | g_i)}),
\end{aligned}
\tag{1}
$$

$$
\log p(v_j | v_{j-k}, \ldots, v_{j+k}, g_i) = \frac{\exp(\bar{u}^T u'_j)}{\sum_{n=1}^{M} \exp(\bar{u}^T u'_n)}, \tag{2}
$$

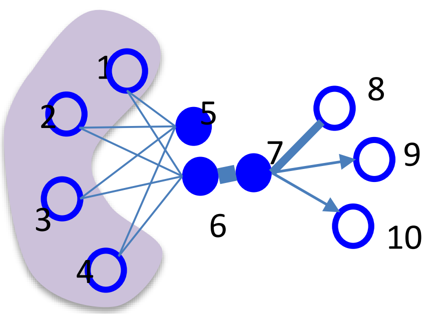
$$
\log p(\hat{v}_j | g_i) = \frac{\exp(u_{g_i}^T \hat{u}_j)}{\sum_{n=1}^{M} \exp(u_{g_i}{}^T \hat{u_n})}, \tag{3}
$$

Network Embedding Models

- DeepWalk
- Node2vec
- GENE
- **LINE** (Tang et al., WWW 2015)
- SDNE

LINE

First-order Proximity



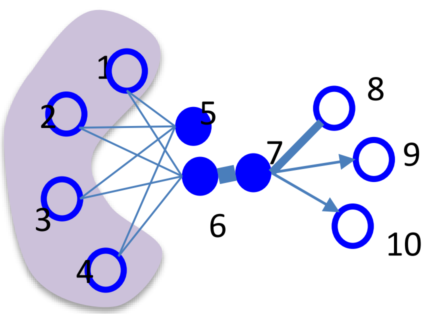
Vertex **6** and **7** have a large first-order proximity

- The local pairwise proximity between the vertices
  - Determined by the observed links
- However, many links between the vertices are missing
  - Not sufficient for preserving the entire network structure

From Jian Tang's slides

LINE

Second-order Proximity



Vertex **5** and **6** have a large second-order proximity

$\hat{p}_5 = (1,1,1,1,0,0,0,0,0,0)$

$\hat{p}_6 = (1,1,1,1,0,0,5,0,0,0)$

- The proximity between the *neighborhood structures* of the vertices
- Mathematically, the second-order proximity between each pair of vertices (u,v) is determined by:

$$\hat{p}_u = (w_{u1}, w_{u2}, \dots, w_{u|V|})$$

$$\hat{p}_v = (w_{v1}, w_{v2}, \dots, w_{v|V|})$$

From Jian Tang's slides

LINE

Preserving the First-order Proximity

- Given an ***undirected*** edge (v_i, v_j), the joint probability of v_i, v_j

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \cdot \vec{u}_j)}$$

$\vec{u}_i$: Embedding of vertex v_i

$$\hat{p}_1(v_i, v_j) = \frac{w_{ij}}{\sum_{(i',j')} w_{i'j'}}$$

- Objective:

$$O_1 = d(\hat{p}_1(\cdot,\cdot), p_1(\cdot,\cdot))$$

$$\propto - \sum_{(i,j)\in E} w_{ij} \log p_1(v_i, v_j)$$

KL-divergence

From Jian Tang's slides

LINE

Preserving the Second-order Proximity

- Given a ***directed*** edge (v_i, v_j), the conditional probability of v_j given v_i is:

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}_j'^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}_k'^T \cdot \vec{u}_i)}$$

$\vec{u}_i$: Embedding of vertex *i* when *i* is a source node;
$\vec{u}_i'$: Embedding of vertex *i* when *i* is a target node.

$$\hat{p}_2(v_j|v_i) = \frac{w_{ij}}{\sum_{k \in V} w_{ik}}$$

- Objective:

$$O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot\,|v_i), p_2(\cdot\,|v_i))$$

$$\propto -\sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i)$$

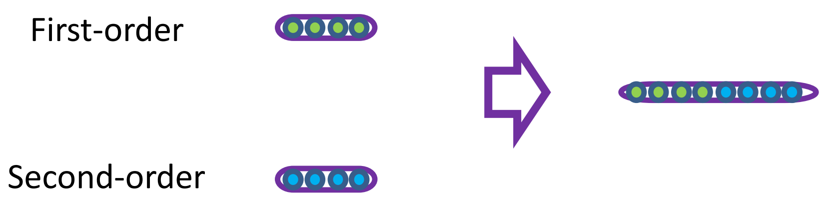
λ_i: Prestige of vertex in the network
$\lambda_i = \sum_j w_{ij}$

From Jian Tang's slides

LINE

Preserving both Proximity

- Concatenate the embeddings individually learned by the two proximity



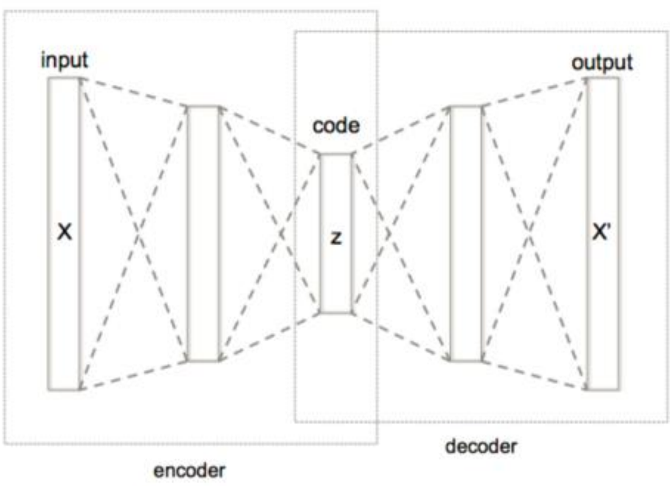
From Jian Tang’s slides

Network Embedding Models

- DeepWalk
- Node2vec
- GENE
- LINE
- **SDNE** (Wang et al., KDD 2016)

SDNE

- Preliminary
  - Autoencoder



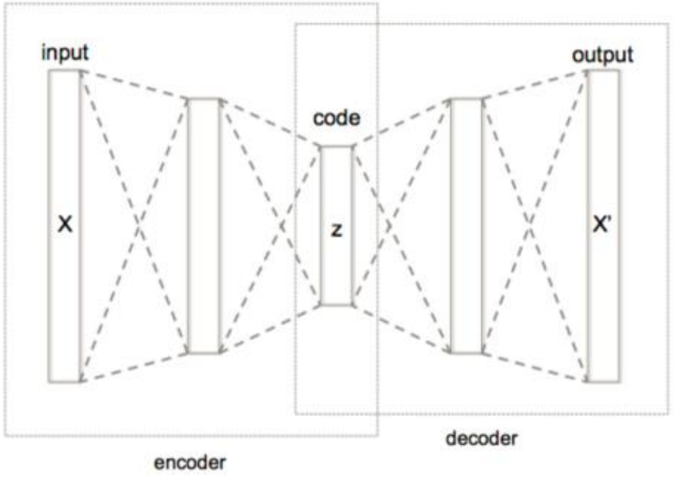
$$\phi : \mathcal{X} \rightarrow \mathcal{F}$$

$$\psi : \mathcal{F} \rightarrow \mathcal{X}$$

$$\arg\min_{\phi,\psi} \|X - (\psi \circ \phi)X\|^2$$

SDNE

- Preliminary
  - Autoencoder
    - The simplest case: a single hidden layer



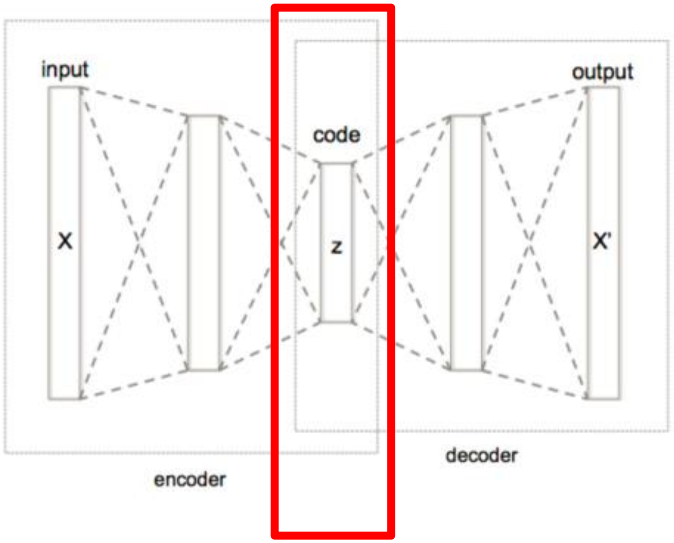
$$\mathbf{z} = \sigma_1(\mathbf{W}\mathbf{x} + \mathbf{b})$$

$$\mathbf{x}' = \sigma_2(\mathbf{W}'\mathbf{z} + \mathbf{b}')$$

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$$

SDNE

- Preliminary
  - Autoencoder
    - The simplest case: a single hidden layer



$$\mathbf{z} = \sigma_1(\mathbf{W}\mathbf{x} + \mathbf{b})$$

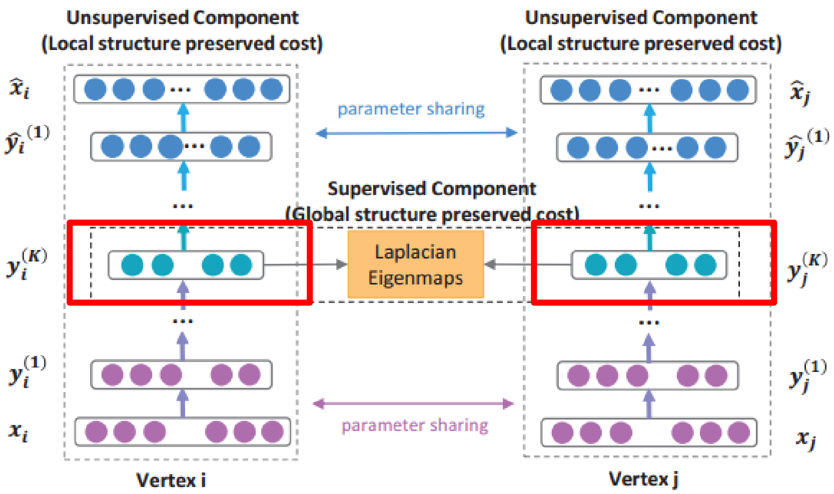
$$\mathbf{x}' = \sigma_2(\mathbf{W}'\mathbf{z} + \mathbf{b}')$$

$$\mathcal{L}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|^2$$

SDNE

- First-order proximity
  - Linked nodes should be coded similarly

$$\mathcal{L}_{1st} = \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i^{(K)} - \mathbf{y}_j^{(K)}\|_2^2$$
$$= \sum_{i,j=1}^{n} s_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2$$

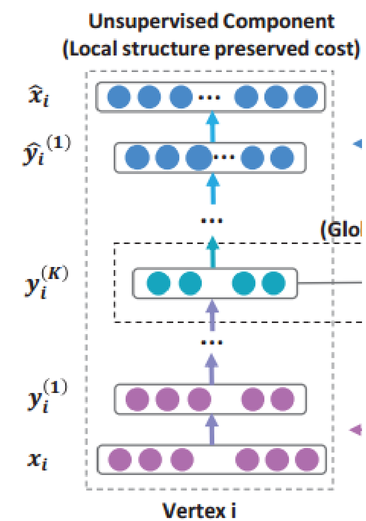


$$\mathbf{y}_i^{(1)} = \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)})$$
$$\mathbf{y}_i^{(k)} = \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, ..., K$$

SDNE

- Second-order proximity
  - The model should reconstruct the neighborhood vectors
  - Similar nodes even without links can have similar codes
    - Or we can not reconstruct the neighborhood

$$\mathcal{L}_{2nd} = \sum_{i=1}^{n} \|(\hat{\mathbf{x}}_i - \mathbf{x}_i) \odot \mathbf{b_i}\|_2^2 = \|(\hat{X} - X) \odot B\|_F^2$$



$$\mathbf{y}_i^{(1)} = \sigma(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)})$$
$$\mathbf{y}_i^{(k)} = \sigma(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}), k = 2, ..., K$$

SDNE

- Network reconstruction

Table 4: MAP on ARXIV-GRQC and BLOGCATALOG on reconstruction task

Method	ARXIV-GRQC					BLOGCATALOG				
	SDNE	*GraRep*	*LINE*	*DeepWalk*	*LE*	*SDNE*	*GraRep*	*LINE*	*DeepWalk*	*LE*
MAP	**0.836****	0.05	0.69	0.58	0.23	**0.63****	0.42	0.58	0.28	0.12

Significantly outperforms GraRep at the: ** 0.01 level.

- Link prediction

Table 5: *precision@k* on ARXIV GR-QC for link prediction

Algorithm	P@2	P@10	P@100	P@200	P@300	P@500	P@800	P@1000	P@10000
SDNE	**1**	**1**	**1**	**1**	**1***	**0.99****	**0.97****	**0.91****	**0.257****
LINE	1	1	1	1	0.99	0.936	0.74	0.79	0.2196
DeepWalk	1	0.8	0.6	0.555	0.443	0.346	0.2988	0.293	0.1591
GraRep	1	0.2	0.04	0.035	0.033	0.038	0.035	0.035	0.019
Common Neighbor	1	1	1	0.96	0.9667	0.98	0.8775	0.798	0.192
LE	1	1	0.93	0.855	0.827	0.66	0.468	0.391	0.05

Significantly outperforms Line at the: ** 0.01 and * 0.05 level, paired t-test.

Network Embedding Models

- DeepWalk
  - Node sentences + word2vec
- Node2vec
  - DeepWalk + more sampling strategies
- GENE
  - Group~document + doc2vec(DM, DBOW)
- LINE
  - Shallow + first-order + second-order proximity
- SDNE
  - Deep + First-order + second-order proximity

Applications of Network Embedding

- Basic applications
- Data Visualization
- Text classification
- Recommendation

Basic Applications

- Network reconstruction
- Link prediction
- Clustering
- Feature coding
  - Node classification
    - Demographic prediction

Applications of Network Embedding

- Basic applications
- **Data Visualization** (Tang et al., WWW 2016)
- Text classification
- Recommendation

Data Visualization

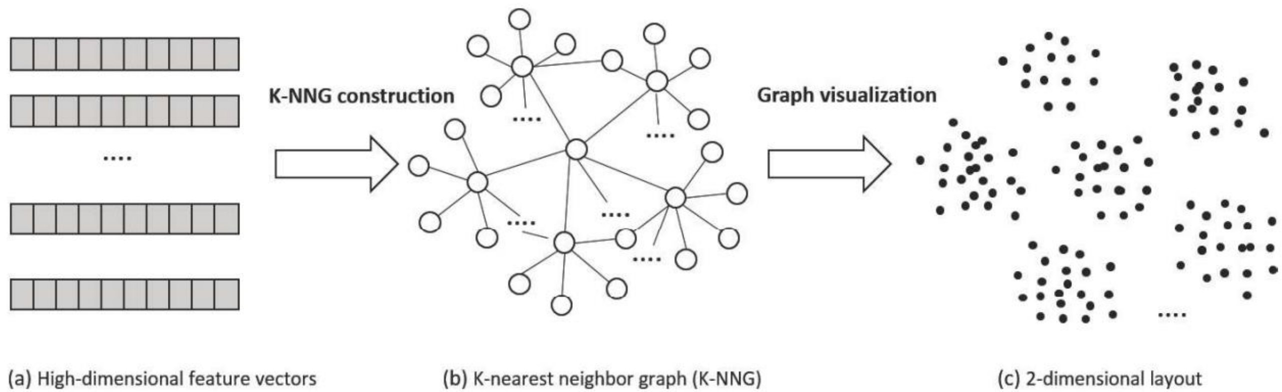


Figure 1: A typical pipeline of data visualization by first constructing a K-nearest neighbor graph and then projecting the graph into a low-dimensional space.

Data Visualization

- Construction of the KNN graph

For the weights of the edges in the K-nearest neighbor graph, we use the same approach as t-SNE. The conditional probability from data $\vec{x}_i$ to $\vec{x}_j$ is first calculated as:

$$p_{j|i} = \frac{\exp(-||\vec{x}_i - \vec{x}_j||^2/2\sigma_i^2)}{\sum_{(i,k)\in E}\exp(-||\vec{x}_i - \vec{x}_k||^2/2\sigma_i^2)}, \text{ and} \quad (1)$$
$$p_{i|i} = 0,$$

Then the graph is symmetrized through setting the weight between $\vec{x}_i$ and $\vec{x}_j$ as:

$$w_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}. \quad (2)$$

Data Visualization

- Visualization-based embedding

$$P(e_{ij} = 1) = f(||\vec{y}_i - \vec{y}_j||),$$

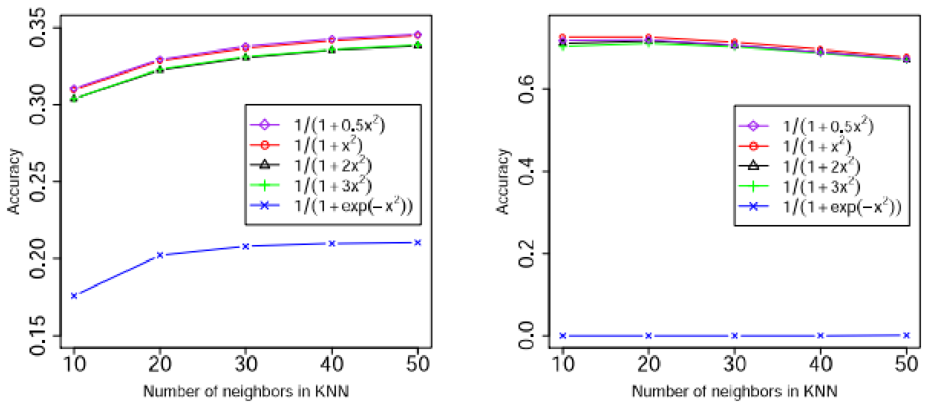
$$P(e_{ij} = w_{ij}) = P(e_{ij} = 1)^{w_{ij}}.$$

$$\begin{aligned} O &= \prod_{(i,j)\in E} p(e_{ij} = 1)^{w_{ij}} \prod_{(i,j)\in \bar{E}} (1 - p(e_{ij} = 1))^{\gamma} \\ &\propto \sum_{(i,j)\in E} w_{ij} \log p(e_{ij} = 1) + \sum_{(i,j)\in \bar{E}} \gamma \log(1 - p(e_{ij} = 1)), \end{aligned}$$

Data Visualization

- Non-linear function

$$P(e_{ij} = 1) = f(||\vec{y}_i - \vec{y}_j||),$$



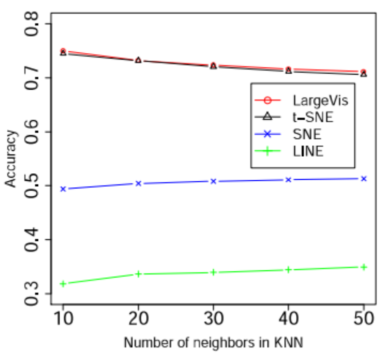
(a) WikiDoc (b) LiveJournal

Figure 4: Comparing different probabilistic functions.

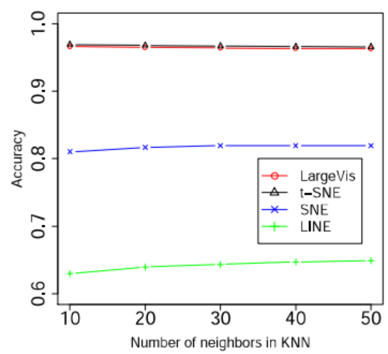
$$f(x) = \frac{1}{1+x^2}$$

Data Visualization

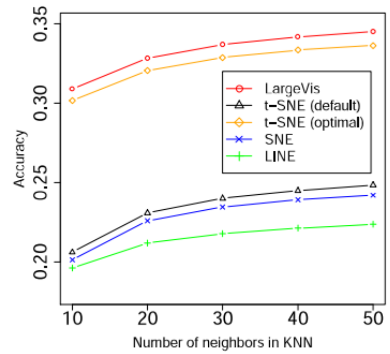
- Accuracy



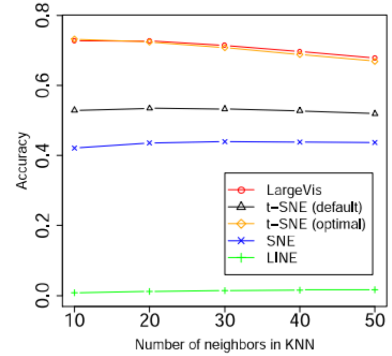
(a) 20NG



(b) MNIST



(c) WikiDoc



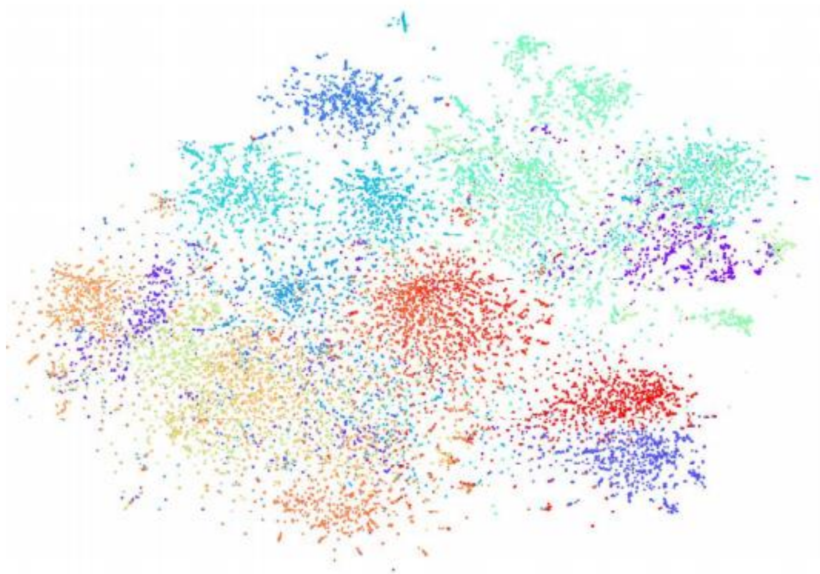
(d) LiveJournal

- Running time

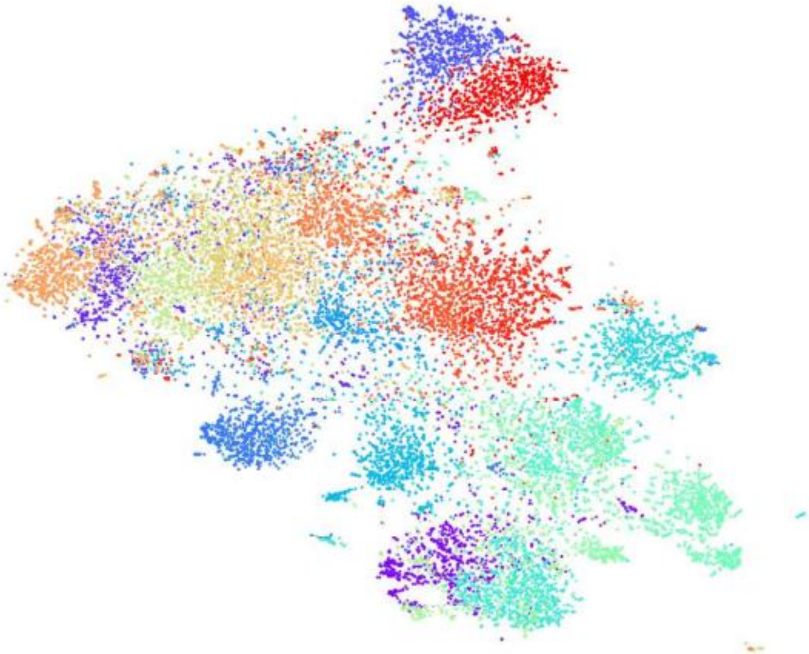
Table 2: Comparison of running time (hours) in graph visualization between the t-SNE and LargeVis.

Algorithm	20NG	MNIST	WikiWord	WikiDoc	LiveJournal	CSAuthor	DBLPPaper
t-SNE	0.12	0.41	9.82	45.01	70.35	28.33	18.73
LargeVis	0.14	0.23	2.01	5.60	9.26	4.24	3.19
Speedup Rate	0	0.7	3.9	7	6.6	5.7	4.9

Data Visualization



(a) 20NG (t-SNE)



(b) 20NG (LargeVis)

Applications of Network Embedding

- Basic applications
- Data Visualization
- **Text classification** (Tang et al., KDD 2015)
- Recommendation

Text Classification

Network embedding helps text modeling

Text representation, e.g., word and document representation, …

Deep learning has been attracting increasing attention …

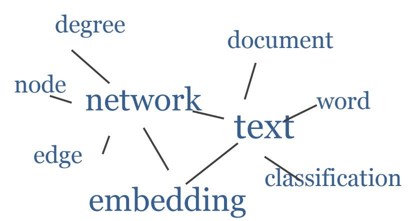
A future direction of deep learning is to integrate unlabeled data …

…

The Skip-gram model is quite effective and efficient …

Information networks encode the relationships between the data objects …

Free text



word co-occurrence network

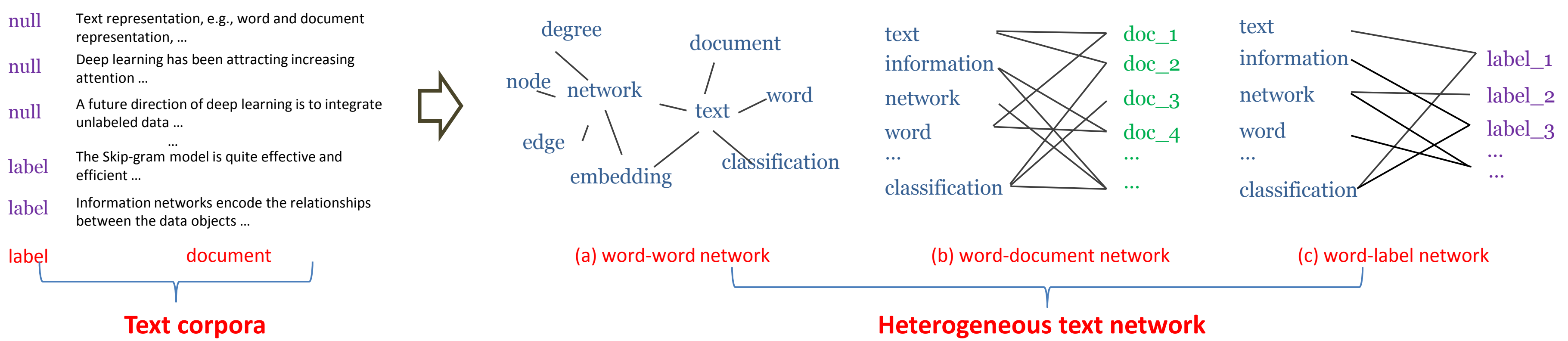
If we have the word network, we can a network embedding model to learn word representations.

From Jian Tang’s slides

Text Classification

- Adapt the advantages of unsupervised text embedding approaches but naturally utilize the ***labeled*** data for specific tasks

- Different levels of word co-occurrences*: local context-level, document-level, label-level*



From Jian Tang's slides

Text Classification

Bipartite Network Embedding

– Extend previous work **LINE** (Tang et al. WWW'2015) on large-scale information network embedding
  – Preserve the *first-order* and *second-order* proximity
  – Only consider the *second-order* proximity here

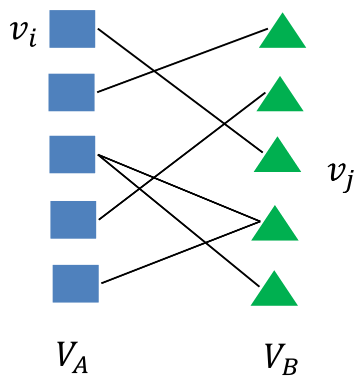
- For each edge (v_i, v_j), define a conditional probability

$$p(v_j|v_i) = \frac{\exp(\vec{u}_j^T \cdot \vec{u}_i)}{\sum_{j' \in B} \exp(\vec{u}_{j'}^T \cdot \vec{u}_i)}$$

- Objective:

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

- Edge sampling and negative sampling for optimization



Tang et al. **LINE: Large-scale Information Network Embedding.** WWW'2015

From Jian Tang's slides

Text Classification

Heterogeneous Text Network Embedding

- Heterogeneous text network: three bipartite networks
  - Word-word (word-context), word-document, word-label network
  - Jointly embed the three bipartite networks
- Objective

$$O_{pte} = O_{ww} + O_{wd} + O_{wl}$$

- where

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

Objective for **word-word** network

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

Objective for **word-document** network

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

Objective for **word-label** network

From Jian Tang's slides

Text Classification

Results on **Long** Documents: Predictive

		20newsgroup		Wikipedia		IMDB	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Unsupervised	LINE(G_{wd})	79.73	78.40	80.14	80.13	89.14	89.14
Predictive embedding	CNN	78.85	78.29	79.72	79.77	86.15	86.15
	CNN(pretrain)	80.15	79.43	79.25	79.32	89.00	89.00
	PTE(G_{wl})	82.70	81.97	79.00	79.02	85.98	85.98
	PTE($G_{ww} + G_{wl}$)	83.90	83.11	81.65	81.62	89.14	89.14
	PTE($G_{wd} + G_{wl}$)	**84.39**	**83.64**	82.29	82.27	89.76	89.76
	PTE(pretrain)	82.86	82.12	79.18	79.21	86.28	86.28
	PTE(joint)	84.20	83.39	**82.51**	**82.49**	**89.80**	**89.80**

PTE(joint) > PTE(pretrain)

PTE(joint) > PTE(G_{wl})

PTE(joint) > CNN/CNN(pretrain)

From Jian Tang’s slides

Text Classification

Results on **Short** Documents: Predictive

		DBLP		MR		Twitter	
Type	Algorithm	Micro-F1	Macro-F1	Micro-F1	Macro-F1	Micro-F1	Macro-F1
Unsupervised embedding	LINE $(G_{ww} + G_{wd})$	74.22	70.12	71.13	71.12	73.84	73.84
Predictive embedding	CNN	76.16	73.08	72.71	72.69	**75.97**	**75.96**
	CNN(pretrain)	75.39	72.28	68.96	68.87	75.92	75.92
	PTE(G_{wl})	76.45	72.74	73.44	73.42	73.92	73.91
	PTE$(G_{ww} + G_{wl})$	76.80	73.28	72.93	72.92	74.93	74.92
	PTE$(G_{wd} + G_{wl})$	**77.46**	**74.03**	73.13	73.11	75.61	75.61
	PTE(pretrain)	76.53	72.94	73.27	73.24	73.79	73.79
	PTE(joint)	77.15	73.61	**73.58**	**73.57**	75.21	75.21

PTE(joint) > PTE(pretrain)

PTE(joint) > PTE(G_{wl})

PTE(joint) ≈ CNN/CNN(pretrain)

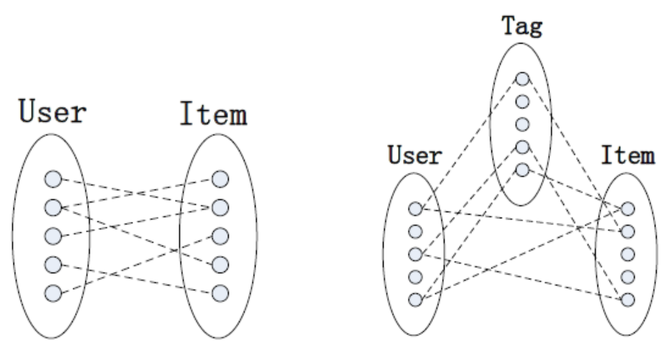
From Jian Tang's slides

Applications of Network Embedding

- Basic applications
- Data Visualization
- Text classification
- **Recommendation** (Zhao et al., AIRS 2016)

Recommendation

- Learning Distributed Representations for Recommender Systems with a Network Embedding Approach
  - Motivation



(a) User-item bipartite network. (b) User-item-tag tripartite network.

Recommendation

- From training records to networks

Definition 3. ***Bipartite User-Item (UI) Network***. *Let $\mathcal{U}$ denote the set of all the users, and $\mathcal{I}$ denote the set of all the items. A bipartite user-item network can be denoted by $\mathcal{G}^{(bi)} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$, the edge set $\mathcal{E} \subset \mathcal{U} \times \mathcal{I}$, the weight matrix $\mathbf{W}$ stores the edge weights, and $W_{u,i}$ denote the link weight between a user u and an item i.*

Definition 4. ***Tripartite User-Item-Tag (UIT) Network***. *Let $\mathcal{U}$ denote the set of all the users, $\mathcal{I}$ denote the set of all the items, and $\mathcal{T}$ denote the set of all the tags. A tripartite user-item-tag network can be denoted by $\mathcal{G}^{(tri)} = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where the vertex set $\mathcal{V} = \mathcal{U} \cup \mathcal{I} \cup \mathcal{T}$, the edge set $\mathcal{E} \subset \big((\mathcal{U} \times \mathcal{I}) \cup (\mathcal{U} \times \mathcal{T}) \cup (\mathcal{I} \times \mathcal{T})\big)$, and the weight matrix $\mathbf{W}$ stores the edge weights.*

Recommendation

- Given any edge in the network

$$P(e_s, e_t) = \sigma(\boldsymbol{v}_{e_s}^{\top} \cdot \boldsymbol{v}_{e_t}) = \frac{1}{1 + \exp(-\boldsymbol{v}_{e_s}^{\top} \cdot \boldsymbol{v}_{e_t})}.$$

$$\hat{P}(e_s, e_t) = \frac{W_{e_s, e_t}}{\sum_{(e_{s'}, e_{t'}) \in \mathcal{E}} W_{e_{s'}, e_{t'}}}.$$

$$L(\mathcal{G}) = D_{KL}(\hat{P}(\cdot, \cdot) || P(\cdot, \cdot)) \propto \sum_{(e_s, e_t) \in \mathcal{E}} W_{e_s, e_t} \log P(e_s, e_t).$$

Recommendation

- User-item recommendation

Table 2. Performance comparisons of the proposed method and baselines on item recommendation.

Methods	JD				MovieLens			
	P@10	R@10	MAP	MRR	P@10	R@10	MAP	MRR
BPR	0.171	0.360	0.337	0.564	0.097	0.169	0.148	0.195
DeepWalk	0.259	0.443	0.502	0.806	0.203	0.243	0.249	0.358
NERM	**0.275**	**0.477**	**0.528**	**0.819**	**0.206**	**0.256**	**0.258**	**0.368**

Recommendation

- User-item-tag recommendation

Table 4. Performance comparisons of the proposed methods and baselines on tag recommendation.

Methods	Last.fm						Bookmarks					
	P@1	R@1	F@1	P@5	R@5	F@5	P@1	R@1	F@1	P@5	R@5	F@5
PITF	0.305	0.125	0.178	0.189	0.351	0.245	0.381	0.132	0.197	0.204	0.304	0.244
DeepWalk	0.088	0.044	0.059	0.040	0.099	0.057	0.064	0.024	0.035	0.038	0.074	0.050
NERM	**0.327**	**0.165**	**0.220**	0.182	**0.370**	0.244	**0.396**	**0.135**	**0.201**	**0.228**	**0.323**	**0.267**

Conclusions

- There are no boundaries between data types and research areas in terms of mythologies
  - Data models are the core
- Even if the ideas are similar, we can move from shallow to deep if the performance actually improves

Disclaimer

- For convenience, I directly copy some original slides or figures from the referred papers. I am sorry but I did not ask for the permission of each referred author. I thank you for these slides. I will not distribute your original slides.

References

- Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. NIPS 2013: 3111-3119
- Bryan Perozzi, Rami Al-Rfou', Steven Skiena. DeepWalk: online learning of social representations. KDD 2014: 701-710
- Aditya Grover, Jure Leskovec. node2vec: Scalable Feature Learning for Networks. KDD 2016: 855-864
- Jifan Chen, Qi Zhang, Xuanjing Huang, Incorporate Group Information to Enhance Network Embedding. CIKM 2016
- Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei. LINE: Large-scale Information Network Embedding. WWW 2015: 1067-1077
- Daixin Wang, Peng Cui, Wenwu Zhu. Structural Deep Network Embedding. KDD 2016: 1225-1234
- Jian Tang, Jingzhou Liu, Ming Zhang, Qiaozhu Mei. Visualizing Large-scale and High-dimensional Data. WWW 2016: 287-297
- Jian Tang, Meng Qu, Qiaozhu Mei. PTE: Predictive Text Embedding through Large-scale Heterogeneous Text Networks. KDD 2015: 1165-1174
- Wayne Xin Zhao, Jin Huang, Ji-Rong Wen. Learning Distributed Representations for Recommender Systems with a Network Embedding Approach. AIRS 2016.

Advanced Readings

- Shiyu Chang, Wei Han, Jiliang Tang, Guo-Jun Qi, Charu C. Aggarwal, Thomas S. Huang: Heterogeneous Network Embedding via Deep Architectures. KDD 2015: 119-128
- Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, Wenwu Zhu: Asymmetric Transitivity Preserving Graph Embedding. KDD 2016: 1105-1114
- Thomas N. Kipf, Max Welling: Semi-Supervised Classification with Graph Convolutional Networks. CoRR abs/1609.02907 (2016)
- Mikael Henaff, Joan Bruna, Yann LeCun: Deep Convolutional Networks on Graph-Structured Data. CoRR abs/1506.05163 (2015)