图表示学习

Zhou Shen

School of Computer Science, Wuhan University

November 1, 2021

Outline

Word2Vec

统计语言模型

对于一个给定的句子：

$$S=\{w_1,w_2,w_3,w_4,\cdots,w_n\}$$

其概率可以计算为：

$$\begin{aligned} P(S) &= P\left(w_1,w_2,\ldots,w_n\right) \\ &= P\left(w_1\right)P\left(w_2 \mid w_1\right)\ldots P\left(w_n \mid w_1,w_2,\ldots,w_{n-1}\right) \\ &= \prod_{i=1}^{n} p\left(w_i \mid w_1 \ldots w_{i-1}\right) \end{aligned}$$

缺点： 1. 数据稀疏严重 2. 参数空间过大

Word2Vec

N-Gram

马尔科夫假设：语言中的每一个词只与前面长度为 $n-1$ 的上下文相关
例如：2-Gram 模型假设下一个词的出现只依赖于前一个词

$$P(S) = P(w_1)P(w_2 \mid w_1) \cdots P(w_n \mid w_{n-1})$$

1) 当 n 较大时，可以学习到更多语境信息，但是参数多，训练量大，更稀疏
2) 当 n 较小时，提供的语境信息少，参数少，但是在训练语料库中出现的次数多，统计结果更加可靠

Word2Vec

One-Hot Encoding

1-of-N Encoding

apple = [1 0 0 0 0]

bag = [0 1 0 0 0]

cat = [0 0 1 0 0]

dog = [0 0 0 1 0]

elephant = [0 0 0 0 1]

缺点：

- 单词的维度受到语料库的大小影响
- 相近的两个词的向量无法体现其关系

Word2Vec

One-Word Context

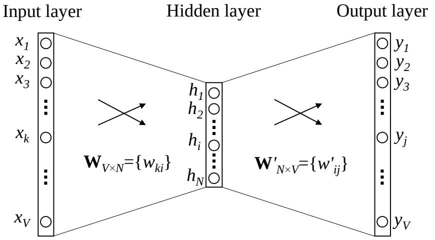


Figure 1: A simple CBOW model with only one word in the context

$$\mathbf{h} = \mathbf{W}^T\mathbf{x} = \mathbf{W}^T_{(k,\cdot)} := \mathbf{v}^T_{w_I}, u_j = \mathbf{v}'_{w_j}h$$

$$p\left(w_j \mid w_I\right) = y_j = \frac{\exp\left(u_j\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)}$$

Word2Vec

Objective Function

$$
\begin{aligned}
\max p\left(w_O \mid w_I\right) &= \max y_{j^*} \\
&= \max \log y_{j^*} \\
&= u_{j^*} - \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) := -E
\end{aligned} \tag{1}
$$

常用变形

$\min E = -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right)$

Word2Vec

CBOW

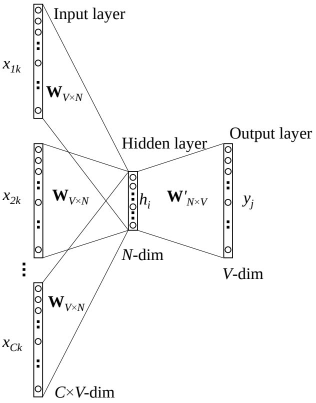


Figure 2: Continuous bag-of-word model

$$\mathbf{h} = \frac{1}{C}(\mathbf{v}_{w_1} + \mathbf{v}_{w_2} + \cdots + \mathbf{v}_{w_C})^T$$

$$\begin{aligned} E &= -\log p\left(w_O \mid w_{I,1}, \cdots, w_{I,C}\right) \\ &= -u_{j^*} + \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right) \end{aligned}$$

Word2Vec

Skip-Gram

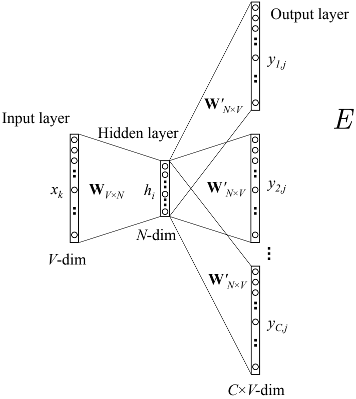


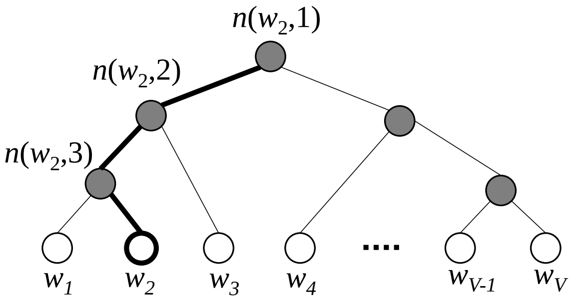
Figure 3: The skip-gram model.

$$
\begin{aligned}
E &= -\log p\left(w_{O,1}, w_{O,2}, \cdots, w_{O,C} \mid w_I\right) \\
&= -\log \prod_{c=1}^{C} \frac{\exp\left(u_{c,j_c^*}\right)}{\sum_{j'=1}^{V} \exp\left(u_{j'}\right)} \\
&= -\sum_{c=1}^{C} u_{j_c^*} + C \cdot \log \sum_{j'=1}^{V} \exp\left(u_{j'}\right)
\end{aligned}
$$

Word2Vec

Hierarchical Softmax

哈夫曼树的构建是基于语料库中单词出现的频率



$$p\left(w=w_O\right)=\prod_{j=1}^{L(w)-1} \sigma\left(\left[n(w, j+1)=\operatorname{ch}(n(w, j))\right] \cdot \mathbf{v}_{n(w, j)}^{\prime}{}^{T} \mathbf{h}\right)$$

1. 词频计算

$$weight(w_i) = \frac{count(w_i)^{0.75}}{\sum_j^V count(w_j)^{0.75}}$$

2. 目标函数

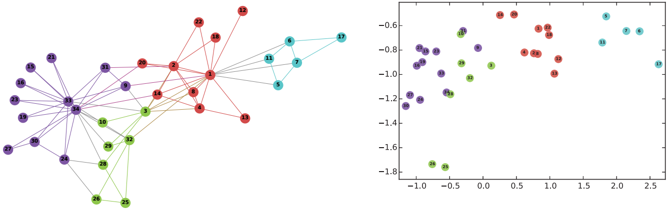
$$E = -\log \sigma \left({\mathbf{v}'_{w_O}}^T \mathbf{h} \right) - \sum_{w_j \in \mathcal{W}_{\text{neg}}} \log \sigma \left(-{\mathbf{v}'_{w_j}}^T \mathbf{h} \right)$$

Word2Vec

总结

1. Word2Vec 主要包含两个模型 CBOW(根据上下文预测中心词的概率) 与 Skip-Gram(根据中心词来预测上下文)
2. Word2Vec 可以被看作是一个多分类模型，但是我们更加关注这个模型的附加产物 (每一个单词的映射向量)
3. Hierarchical Softmax 与 Negative Sample 其实本质上都是在优化 softmax，我们知道 softmax 的时间复杂度是 $O(N)$, 因此，我们可以将其转化为 $log(N)$ 次的二分类计算，这便是 Hierarchical Softmax。负采样则是对（1）式的后半部分进行优化，不需要全部的非目标词都作为负样本，等于把 $O(N)$ 复杂度降为 $O(m)$，m 为负样本的个数。

DeepWalk

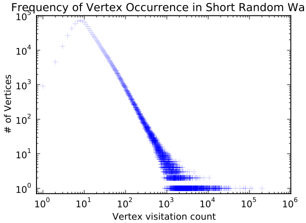


(a) Input: Karate Graph (b) Output: Representation

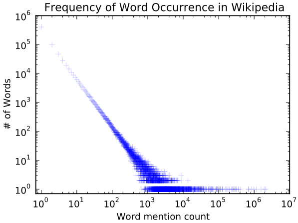
Definition

Problem Definition Let $G = (V, E)$, where V are the members of the network, and E be its edgs. Our goal is to learn $X_E \in \mathcal{R}^{|V| \times d}$, where d is number of latent dimensions.

DeepWalk



(a) YouTube Social Graph



(b) Wikipedia Article Text

Connection

The power-law distribution of vertices appearing in the short random walks follows a power-law, much like the distribution of words in natural language.

DeepWalk

1. give a sequence of random walk from vertice i

$$\mathcal{W}_i^n = \{v_1, v_2, v_3, \cdots, v_n\} \tag{2}$$

2. maximize the likehood

$$\Pr\left(v_i \mid (\Phi(v_1), \Phi(v_2), \cdots, \Phi(v_{i-1}))\right) \tag{3}$$

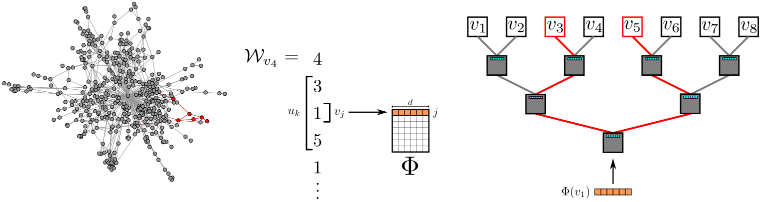
Mapping Function Φ

$\Phi : v \in V \longmapsto \mathcal{R}^{|V| \times d}$ where d is the number of dimensions

DeepWalk

Disadvantages and Optimizations

- As the walk length grows, computing this objective function becomes unfeasible.
- Restricts the order in which vertices appear



(a) Random walk generation. (b) Representation mapping. (c) Hierarchical Softmax.

DeepWalk

Algorithm

Algorithm 1 DeepWalk(G, w, d, γ, t)

Input: graph $G(V, E)$
window size w
embedding size d
walks per vertex γ
walk length t
Output: matrix of vertex representations $\Phi \in \mathbb{R}^{|V| \times d}$

1: Initialization: Sample Φ from $\mathcal{U}^{|V| \times d}$
2: Build a binary Tree T from V
3: **for** $i = 0$ to γ **do**
4: $\quad \mathcal{O} = \text{Shuffle}(V)$
5: $\quad$ **for each** $v_i \in \mathcal{O}$ **do**
6: $\quad\quad \mathcal{W}_{v_i} = RandomWalk(G, v_i, \text{t})$
7: $\quad\quad \text{SkipGram}(\Phi, \mathcal{W}_{v_i}, w)$
8: $\quad$ **end for**
9: **end for**

Corollary

$$\underset{\Phi}{\text{minimize}} - \log \Pr\left(\{v_{i-w}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+w}\} \mid \Phi\left(v_i\right)\right)$$

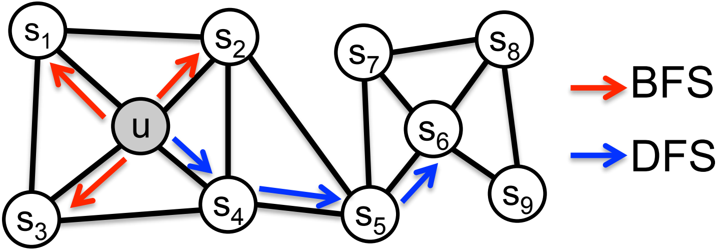
DeepWalk

总结

1. DeepWalk 将图中的每一个点对应成一个单词，将图中随机游走生成的序列看作是句子，基于此使用 Work2Vec 的思想来学习每一个节点的向量，并用作下游任务
2. 正是由于其游走的随机性，DeepWalk 没有确定的相似度优化目标函数，没有办法针对性的捕捉特定的相似性 (homophily and structural equivalence)

Node2Vec

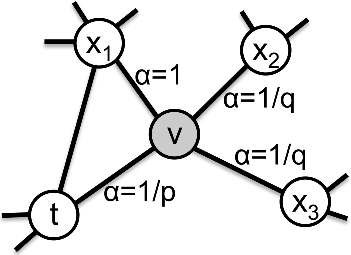
Classic Search Strategies



1. The neighborhoods sampled by BFS lead to embeddings that correspond closely to structural equivalence.
2. DFS can freely explore network neighborhoods which is important in discovering homophilous communities.

Node2Vec

Search Bias α



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

这里表示的是当上一个经过的点是 t，且当前位于 v，则下一个点转移的影响因子 α 的计算公式，转移权重公式为 $\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}$。在实现中，我们需要使用 Alias Sample 来预先处理，将确定转移点的时间复杂度从 $O(n)$ 降为 $O(1)$，也正因此需要存储大量的转移概率，用空间换时间。

Node2Vec

Algorithm

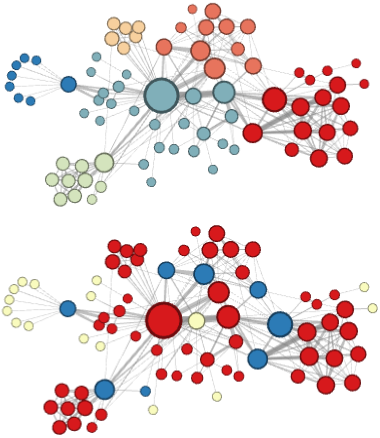
Algorithm 1 The *node2vec* algorithm.

```
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, π)
  Initialize walks to Empty
  for iter = 1 to r do
    for all nodes u ∈ V do
      walk = node2vecWalk(G', u, l)
      Append walk to walks
  f = StochasticGradientDescent(k, d, walks)
  return f

node2vecWalk (Graph G' = (V, E, π), 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
```

Corollary

$$\max_f \sum_{u \in V} \left[\sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) - \log Z_u \right]$$



1. 左图上半部分使用的参数 $p > q$，相同社区的点都被划分成了相同的颜色，更多体现的是同质性；而下半部分则是 $p < q$，更多的是结构相似的点被划分成了相同的颜色，更多体现的是结构相似性。
2.Node2Vec 本质就是通过 BFS 与 DFS 两种随机游走的策略来控制随机游走的方向以此来偏向捕获特定的相似性。
3.Node2Vec 也是有一些缺陷存在，例如如果两个点如果相距太远，很难通过游走策略来捕捉其结构相似性。

Struc2Vec

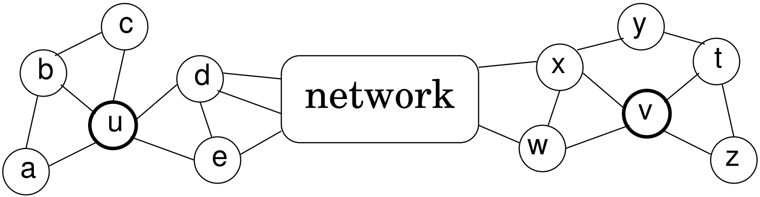


Figure 1: An example of two nodes (u and v) that are structurally similar (degrees 5 and 4, connected to 3 and 2 triangles, connected to the rest of the network by two nodes), but very far apart in the network.

Limitation

Structurally similar nodes will never share the same context if their distance (hop count) is larger than the Skip-Gram window.

Struc2Vec

Structural Similarity

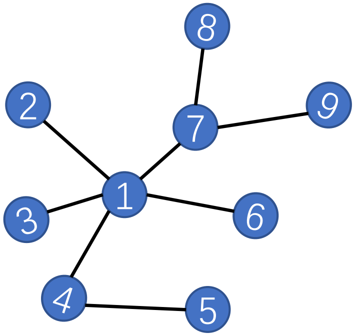
$$f_k(u,v) = f_{k-1}(u,v) + g\left(s\left(R_k(u)\right), s\left(R_k(v)\right)\right)$$
$$k \geq 0 \text{ and } \left|R_k(u)\right|, \left|R_k(v)\right| > 0 \tag{4}$$

符号解释

$f_k(u,v)$	structural distance when consider k-hop neighborhoods
$R_k(u)$	the ring of nodes at distance k
$s(S)$	the ordered degree sequence
$g(a,b)$	DTW Function

Struc2Vec

Example



Example

$$R_1(1) = \{2, 3, 4, 6, 7\}$$
$$R_2(1) = \{5, 8, 9\}$$
$$s(R_1(1)) = \{1, 1, 1, 2, 3\}$$
$$s(R_2(1)) = \{1, 1, 1\}$$

Dynamic Time Warping

利用动态规划的思想来计算两条不等长序列的相似度

动态转移方程如下：

$$g(i,j) = min\{g(i-1,j), g(i,j-1), g(i-1,j-1)\} + d(i,j)$$

$$d(a,b) = \frac{max(a,b)}{min(a,b)} - 1$$

Struc2Vec

Optimization

OPT1: 对于每一个节点的有序度序列，其空间复杂度为 $O(n)$，为了减少花费，将其转化为 (度数，出现次数) 这样的形式，距离方程变成如下：

$$\operatorname{dist}(a, b) = \left(\frac{\max(a_0, b_0)}{\min(a_0, b_0)} - 1\right) \max(a_1, b_1)$$

OPT2: 两个度数相差很大的点没有必要计算他们的距离，使用双指针寻找与度相近的节点进行计算，将时间复杂度从 $O(n^2)$ 降为 $O(nlog(n))$
OPT3: 限制层次带权图层数

Struc2Vec
总结

1. Struc2Vec 基于节点之间的结构相似性来构建多层加权图，并在这张多层加权图上进行游走生成序列，使用 Word2Vec 的思想进行优化，不像 Node2Vec 受限于 Window Size 的限制导致无法捕捉距离很远的两个点之间的结构相似度
2. 这篇文章的核心就是公式 (4)，通过计算两个节点之间的度序列的相似性来计算其结构相似性

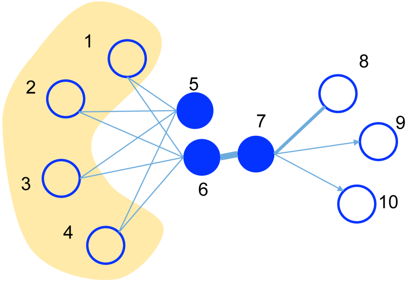


Figure 1: A toy example of information network. Edges can be undirected, directed, and/or weighted. Vertex 6 and 7 should be placed closely in the low-dimensional space as they are connected through a strong tie. Vertex 5 and 6 should also be placed closely as they share similar neighbors.

LINE

First-Order Proximity

The Joint Probability between v_i and v_j:

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

Empirical Probability $\hat{p}_1(i,j) = \frac{w_{ij}}{W}$

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

where $d(\ \cdot\ ,\ \cdot\ )$ is the distance between two distributions.

Objective Function

$\min\ O_1 = -\sum_{(i,j)\in E} w_{ij} \log p_1\left(v_i, v_j\right)$

LINE

Second-Order Proximity

The Probability of Context v_j generated by vertex v_i as:

$$p_2\left(v_j \mid v_i\right)=\frac{\exp\left(\vec{u}_j^{T} \cdot \vec{u}_i\right)}{\sum_{k=1}^{|V|} \exp\left(\vec{u}_k^{\prime T} \cdot \vec{u}_i\right)}$$

Empirical Probability $\hat{p}_1(v_j \mid v_i)=\frac{w_{ij}}{d_i}$

Objective Function

$O_2=-\sum_{(i,j)\in E} w_{ij} \log p_2\left(v_j \mid v_i\right)$

1. Negative Sample

$$\log\sigma\left(\vec{u}_j'^{T}\cdot\vec{u}_i\right)+\sum_{i=1}^{K}E_{v_n\sim P_n(v)}\left[\log\sigma\left(-\vec{u}_n^{T}\cdot\vec{u}_i\right)\right]$$

where $\sigma(x)=\frac{1}{1+exp(-x)}$ is the sigmoid function.

2. Edge Sample - Alias Sample

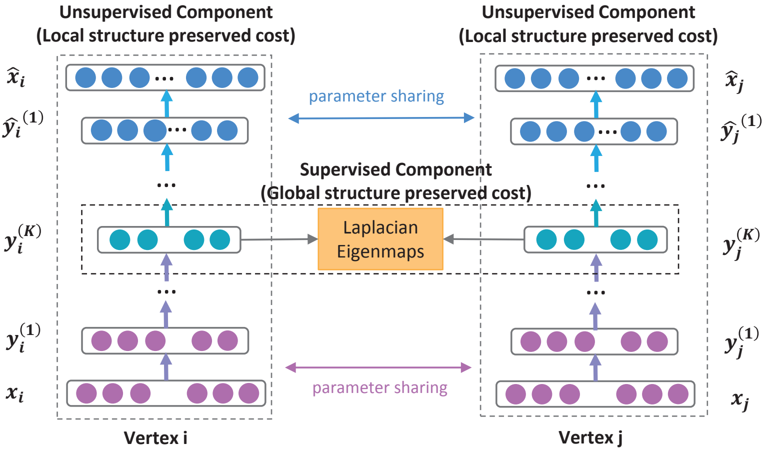
LINE

总结

1.LINE 的核心思想是通过捕获图中的一阶相似度 (两个点之间有连接) 和二阶相似度 (两个点有相同的邻居)
2. 使用 KL 散度来计算相似度，最小化条件概率与经验概率之间的差距
3. 对于一阶相似度，如果只计算两个向量的乘积会导致平凡解，因此也加入负采样的思想进行优化，这样就将一阶相似度与二阶相似度统一使用一个目标函数进行表示，唯一的差别就在于 u_j，在优化一阶与二阶时对其使用不同的 Embedding。

SDNE

Overview



SDNE

First-Order Proximity

Encoder and Decoder

$$
\begin{aligned}
\mathbf{y}_i^{(1)} &= \sigma\left(W^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}\right) \\
\mathbf{y}_i^{(k)} &= \sigma\left(W^{(k)}\mathbf{y}_i^{(k-1)} + \mathbf{b}^{(k)}\right), k = 2, \ldots, K
\end{aligned}
$$

First-Order Proximity Loss Function

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

SDNE

Second-Order Proximity

The autoencoder goal is to minimize:

$$\mathcal{L} = \sum_{i=1}^{n} \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|_2^2$$

Structural Similarity $x_i = s_i$

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

SDNE

Objective Function

正则化部分

$$\mathcal{L}_{reg} = \frac{1}{2}\sum_{k=1}^{K}\left(\left\|W^{(k)}\right\|_F^2 + \left\|\hat{W}^{(k)}\right\|_F^2\right)$$

损失函数

$$\begin{aligned}\mathcal{L}_{mix} &= \mathcal{L}_{2nd} + \alpha\mathcal{L}_{1st} + \nu\mathcal{L}_{reg} \\ &= \|(\hat{X} - X) \odot B\|_F^2 + \alpha\sum_{i,j=1}^{n} s_{i,j}\left\|\mathbf{y}_i - \mathbf{y}_j\right\|_2^2 + \nu\mathcal{L}_{reg}\end{aligned}$$

SDNE

总结

1.DeepWalk 使用随机游走与 Skip-Gram 来学习节点表示，但是缺少一个明确的优化函数来学习图结构特征，LINE 只是结合了一阶相似度与二阶相似度，其节点表示受限于它是一个浅层模型效果不够好。

2.SDNE 的构建思路围绕三点：模型的高度非线性来更好的学习节点的深层次表示；能够很好的保留结构特性；如何处理稀疏性问题。

3.SDNE 使用自编码器来优化二阶相似度 (其中输入 x 是度序列，跟 Struc2Vec 一样都是通过度序列来计算结构上的相似度)，使得结构相似的顶点具有相似的 Embedding，而对于一阶相似度则是使得相邻两个顶点对应的 Embedding 接近。

复现结果

Dataset	Method	LabelNode	macro-F1	参考值
Wiki	Node2Vec	50%	51.10	50.52
	SDNE	50%	49.70	49.80
BlogCatalog	DeepWalk	50%	21.80	21.10
	Node2Vec	50%	23.50	23.00
Wikipedia	LINE	50%	11.40	11.64
Brazil-Flight	Struc2Vec	50%	67.94	68.30