DeepWalk深度游走算法在数字供应链网络图谱构建中的作用

锦素数科 刘卫星 锦素数字供应链 2020-04-14

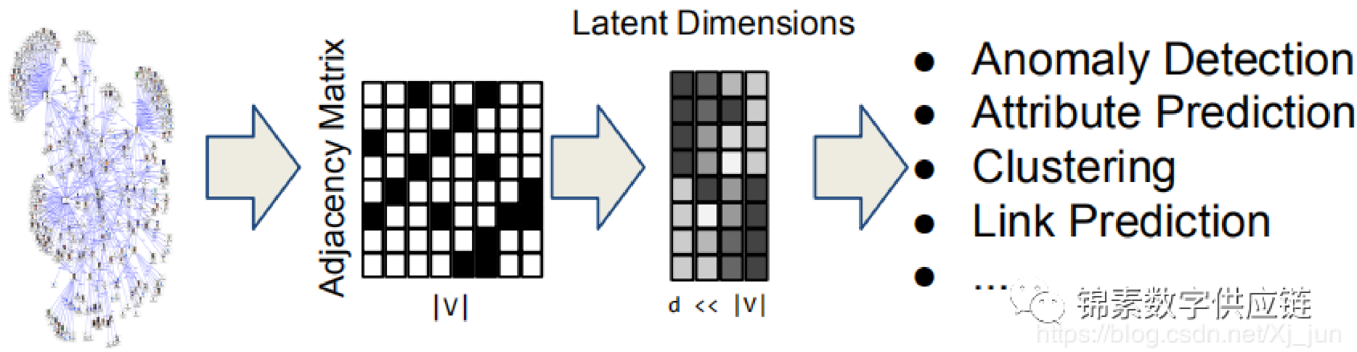
DeepWalk深度游走算法在数字供应链网络图谱

在建立供应链图谱的基础上，在以下领域有关键作用：

1. **反欺诈模型的建立**
2. **供应链节点企业的分类**
3. **供应链节点的自繁殖技术。**

NetworkEmbedding/Graph Embedding目的是希望能够将网络中的节点用比较低维的向量去表达，同时在这个向量空间中，网络结构的一些性质仍能够保持。

正文



如图所示可以将网络中的节点用低维的向量表达，然后来执行实际的任务（异常检测，分类，链接预测等等）

DeepWalk

论文中作者提出自己的三点主要贡献：

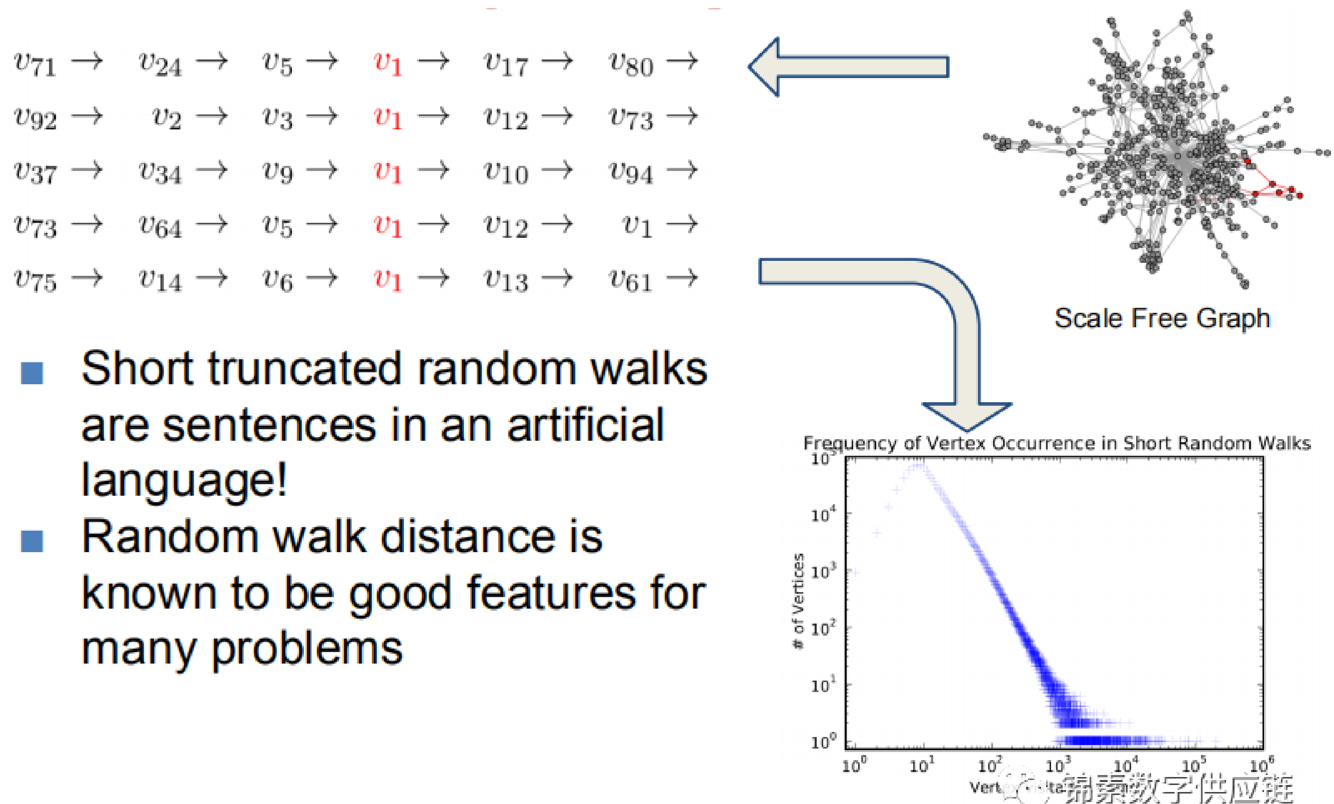
- 作者使用深度学习作为工具去分析图，建立了一个适合复杂模型的RobustRepresentations。DeepWalk根据shortrandom walks来学习结构化表示
- 作者在考虑稀疏问题上，在多标签分类任务上有很大进步，在MicroF1MicroF_1MicroF1上有着5%-10%的提升。在一些例子上，即使提取40%的训练数据依然能获得很好的效果
- 作者通过采用并行的方法构建web-scalegraphs（例如youtube）的representations表明了算法的可扩展性。

文章中的Definition

We consider the problem of classifying members of a social network into one or more categories. Let $G = (V, E)$, where V represent the members of the network, E are their connections, $E \subseteq (V \times V)$, and $G_L = (V, E, X, Y)$ is a partially labeled social network, with attributes $X \in \mathbb{R}^{|V| \times S}$ where S is the size of the feature space for each attribute vector, and $Y \in \mathbb{R}^{|V| \times |\mathcal{Y}|}$, $\mathcal{Y}$ is the set of labels.

DeepWalk的目的是要学习XE∈R∣V∣∗d

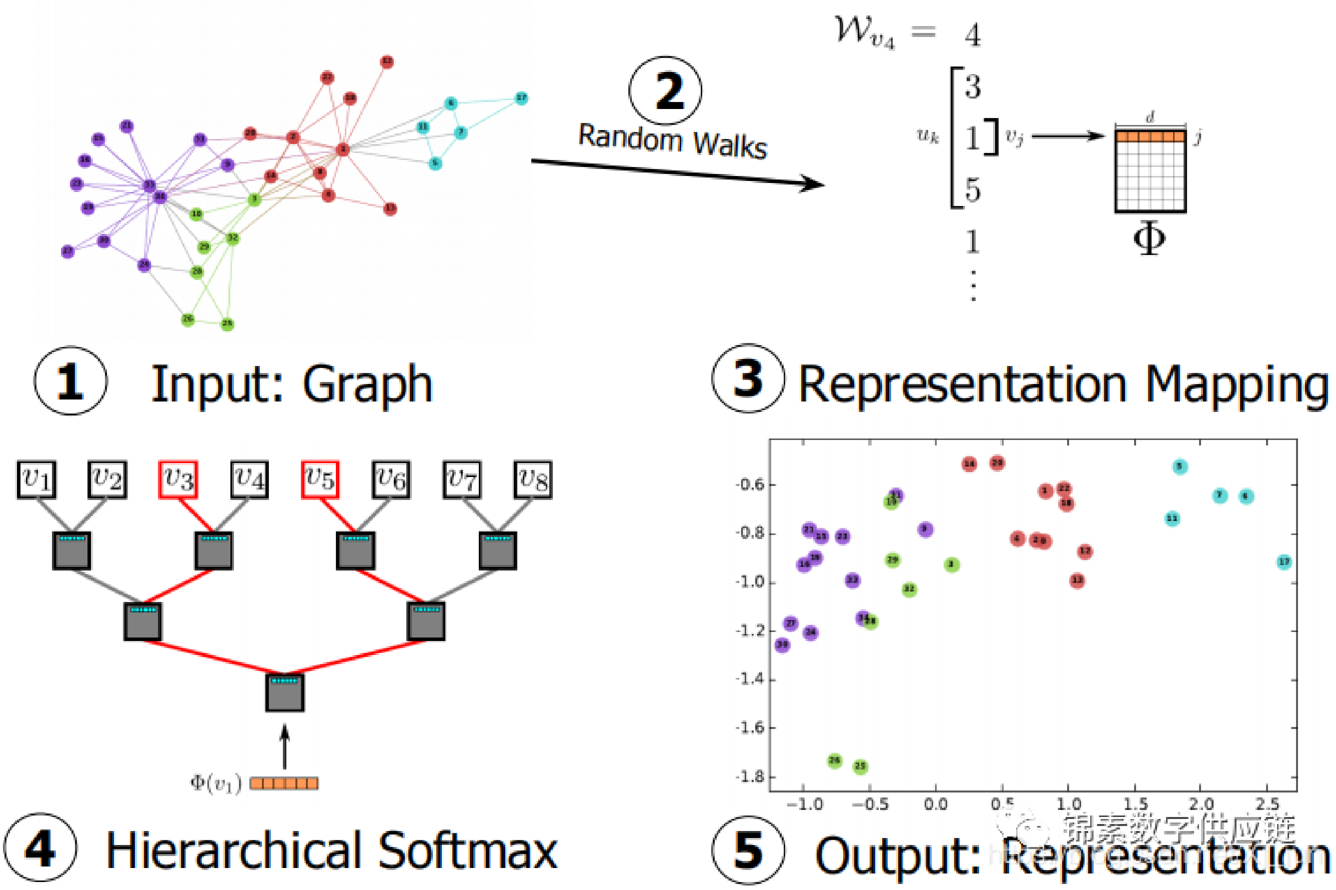
学习SocialRepresentations



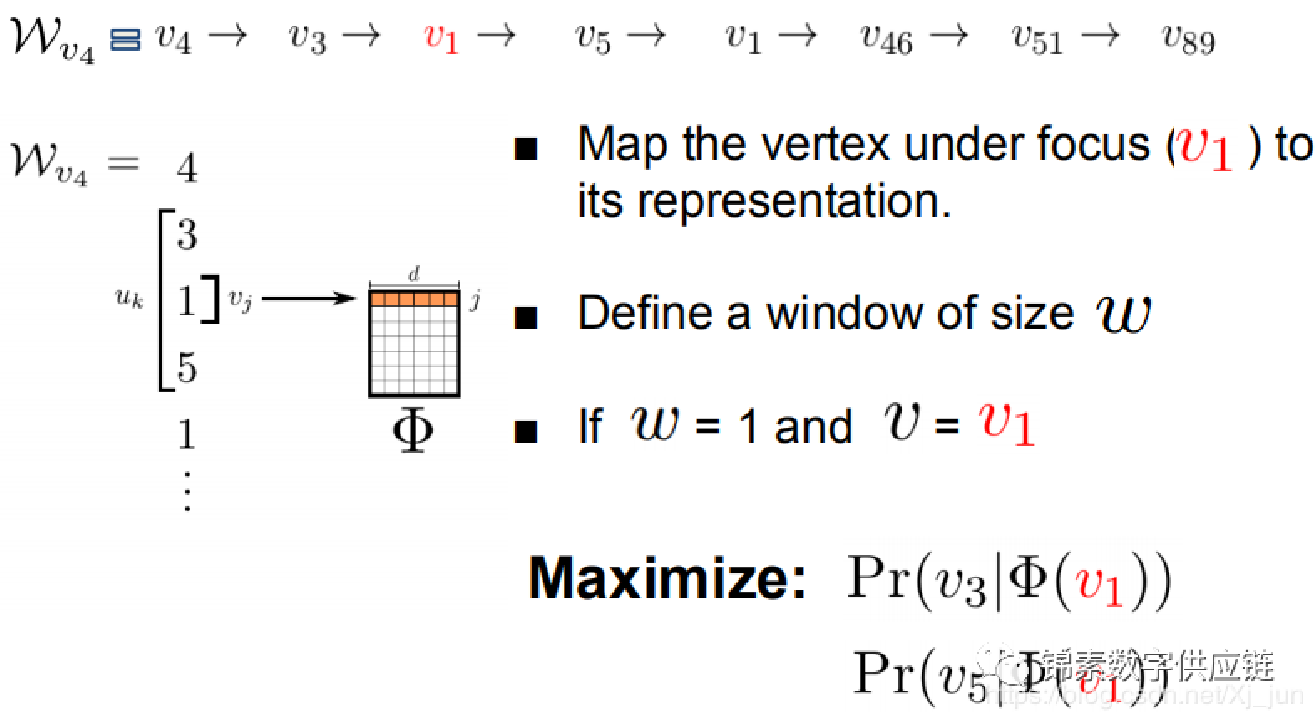
(a) YouTube Social Graph

- DeepWalk采用随机游走的思想在节点中随机游走生成节点序列，然后引用Word2Vec思想，将节点序列看做为语句。

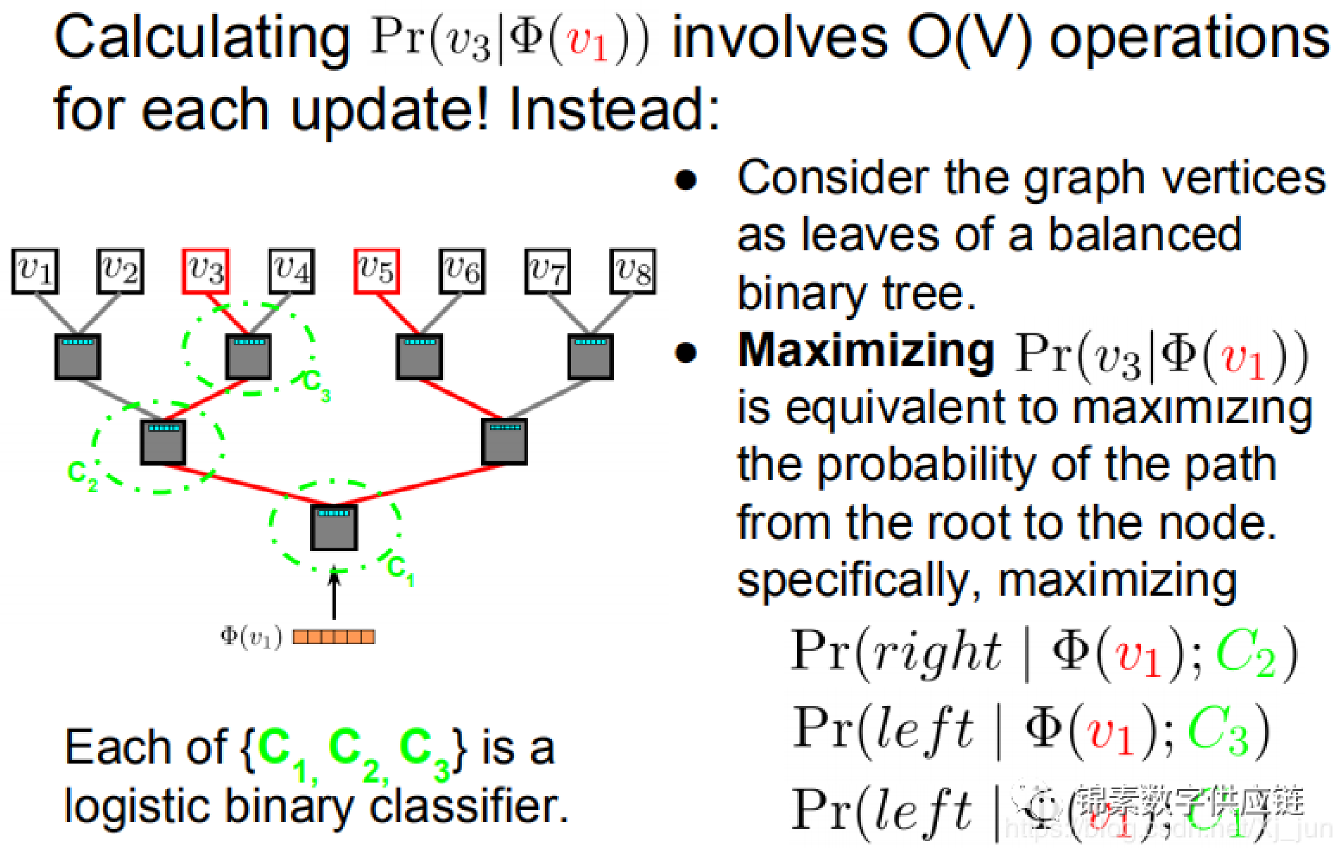
模型示意图如下



2 Random Walks
1 Input: Graph
3 Representation Mapping
4 Hierarchical Softmax
5 Output: Representation
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8
$\Phi(v_1)$
$\mathcal{W}_{v_4} = 4, 3, 1, 5, 1, \ldots$
Φ
锦素数字供应链



$\mathcal{W}_{v_4} = v_4 \rightarrow v_3 \rightarrow v_1 \rightarrow v_5 \rightarrow v_1 \rightarrow v_{46} \rightarrow v_{51} \rightarrow v_{89}$
$\mathcal{W}_{v_4} = 4, 3, 1, 5, 1, \ldots$
Φ
Map the vertex under focus (v_1) to its representation.
Define a window of size w
If w = 1 and v = v_1
Maximize: $\Pr(v_3|\Phi(v_1))$
$\Pr(v_5|\Phi(v_1))$
锦素数字供应链
https://blog.csdn.net/Xj_jun



采用Hierarchical Softmax来计算条件概率

算法

算法包括两个步骤

- 在图中的节点上随机游走生成随机序列
- 根据随机序列，运行skip-gram，来学习每个节点的embedding

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 $\Phi \in \mathbb{R}^{|V| \times d}$

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

Algorithm 2 SkipGram(Φ, $\mathcal{W}_{v_i}$, w)

for each $v_j \in \mathcal{W}_{v_i}$ **do**
  for each $u_k \in \mathcal{W}_{v_i}[j - w : j + w]$ **do**
    $J(\Phi) = -\log \Pr(u_k \mid \Phi(v_j))$
    $\Phi = \Phi - \alpha * \frac{\partial J}{\partial \Phi}$
  end for
end for

多层的softmax利用二叉树来解决softmax计算成本问题。在二叉树中，所有叶子节点（上面所说的图中的v1，v2，… v8）都是图中的顶点。在每个内部节点中（除了叶子节点以外的节点，也就是分枝结点），都通过一个二元分类器来决定路径的选取。为了计算某个顶点v_k的概率，可以简单地计算沿着从根节点到叶子节点v_k的路径中的每个子路径的概率。由于每个节点的孩子节点的概率和为1，因此在多层softmax中，所有顶点的概率之和等于1的特性仍然能够保持。如果n是叶子的数量，二叉树的最长路径由O（log（n））限定，因此，元素的计算时间复杂度将减少到O（log | V |）。