DeepWalk原理与实践

# 介绍Deep Walk

Deep Walk[3]是KDD 2014年的文章，它属于Graph Embedding里比较经典算法。理清Deep Walk里的工作原理，方便后续更深层次的理解。

一言以蔽之，Graph Embedding是Word2Vec方法在图结构上的大胆尝试，将图上的结点转为低维向量表示。而Deep Walk是在物品组成的图(结点，邻接矩阵，权重矩阵等)中随机游走获得序列数据，对序列数据做word2vec获取embedding。（经典到一句话就能总结出它的做法。）

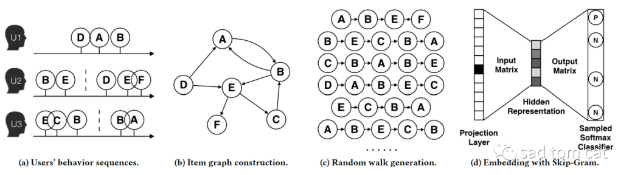
那么技术难点就在于：

* 如何获取物品组成的图？
* 怎么随机游走，获取序列？
* 怎么做embedding？

下面逐一解释。

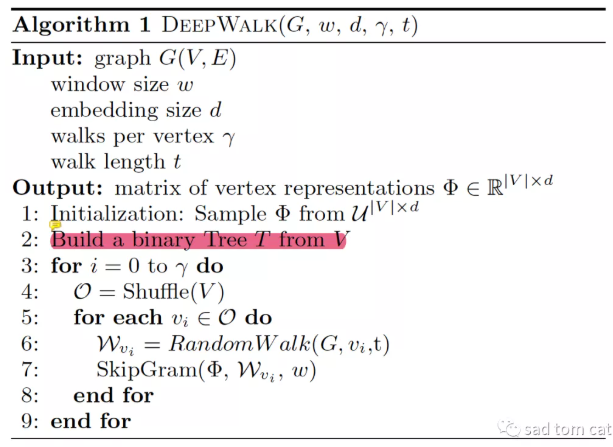
## 获取物品的图

物品之间的边连接是通过用户行为序列得出的，如下图所示。U2的序列被分成两个子序列，子序列内item的顺序就可以转为图中vertex的指向关系。总结全部用户的序列，item前后关系出现的次数也可以相对应地转为vertex的边权重。



## 随机游走的方法

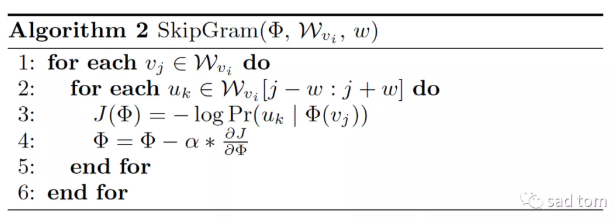
直接上Deep Walk论文里的原图。



事先规定随机游走的序列长度t，外循环的次数。所有的结点都进行一次游走(特别注明，第4行的shuffle操作是方便后续的随机梯度下降)。

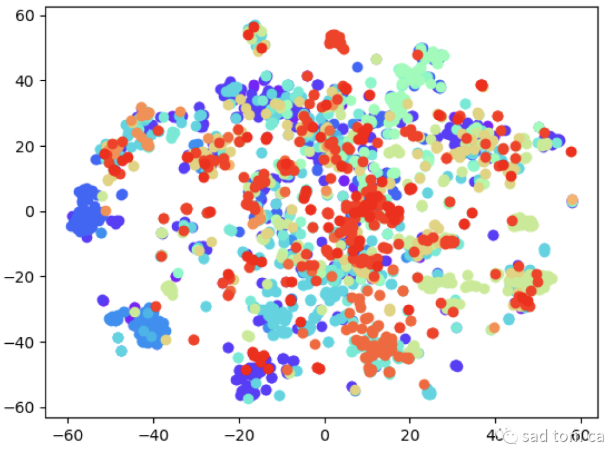
## skip-gram作embedding

这里的skip-gram没有太多的考究，如下所示。



不过原文中特地强调使用hierarchical softmax进行加速训练。(虽然笔者觉得用negative sampling也可以)

最后插一张图，是对DeepWalk的embedding结果做t-sne的结果图（颜色相同代表结点属于相同的类）。



# 总结

Deep Walk通过随机游走获得序列数据，对图中结点做embedding。但成也萧何败也萧何，随即游走也是它的一个弱点——没有记忆特定的结构，结点的空间关系全部要依托序列数据，这一点在LINE[7]的论文也有强调。LINE在损失函数里加入了结点间一级邻接(First-order Proximity)和二级邻接(Second-order Proximity)的考虑。

参考：

<https://mp.weixin.qq.com/s/CQaG7a1E9xGhUCygR9dEhA>