周报\_2018\_10\_14

1. Heterogeneous Network Embedding via Deep Architectures （2015）

该文章所给出的异构图的定义与之前的有少许不同，之前网络的节点的类型是指user、college等，而该文章所定义的异构图是指的是网络的节点类型是document、image等。该文章对于异构网络的嵌入可以粗略的分为两个部分：一是使用CNN对image进行特征提取，使用全连接网络对document进行特征提取，生成各自节点的特征向量；而后再利用网络中“相互连接的两个节点的相似度最高，距离越远的节点相似度越低”这一特征，以及之前生成的特征向量构造出关于相似度的损失函数，采用坐标下降法（coordinate descent methods）对目标函数进行优化即可。总结：该文章使用CNN与全连接网络提取了网络中节点的特征，它保存了各个节点的特性，之后构造的损失函数是利用了网络的连接信息；这样节点特性+网络特征就得到了较好的embedding向量。这篇文章的算法只是Proximity Search，不是Semantic Proximity Search。

（这样的话，是否可以将Semantic Proximity Search任务扩展到这样的异构图上？再者，它表示结构的时候，是在之前得到的内容的embedding基础上进行的，那么是否单独训练表示结构的embedding向量，在将它与内容的embedding向量连接起来会得到更好的效果？因为它在表示内容的embedding向量上进行结构的embedding的时候，它对之前embedding向量的修正可能会丢失一部分的内容信息。）

1. 问题：Semantic Proximity Search on Heterogeneous Graph by Proximity Embedding是如何针对不同的语义类进行embedding的？换句话说，它embedding的向量是适用于所有的语义类，还是说对于给出不同的语义类，需要进行不同的embedding？

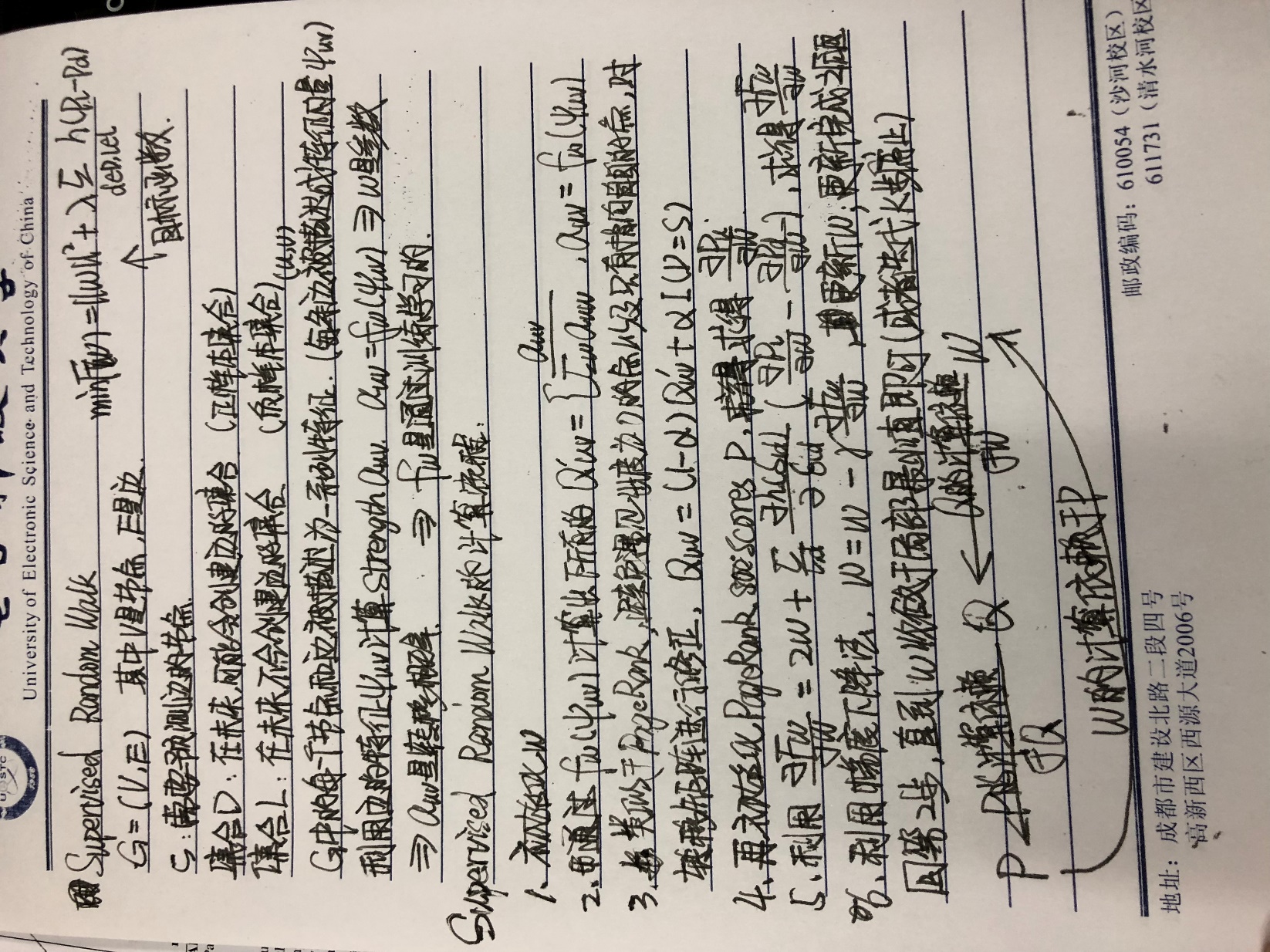
通过看该论文所给出的代码得到了答案（代码的分析放在了github上）：就是它对不同的关系的嵌入是通过加载不同的数据集来实现的。即是说对于不同的关系，需要进行多次嵌入才行。还有就是它的三元组是如何生成的在文中并没有提及到，在它的源代码中也并没有生成三元组的代码。

通过回顾文章Semantic proximity search on graphs with metagraph-based learning，发现了对语义邻近搜索任务而言有“the desired class of proximity can be substituted by a set of training examples”，即是说将特定的语义类由一系列的训练数据所代替。

（代码的分析放在了github上，dbgroup-uestc/[dengliwei](https://github.com/dbgroup-uestc/dengliwei)/[Interactive Paths Embedding for Semantic Proximity Search on Heterogeneous Graphs](https://github.com/dbgroup-uestc/dengliwei/tree/master/Interactive%20Paths%20Embedding%20for%20Semantic%20Proximity%20Search%20on%20Heterogeneous%20Graphs)/[相关文章](https://github.com/dbgroup-uestc/dengliwei/tree/master/Interactive%20Paths%20Embedding%20for%20Semantic%20Proximity%20Search%20on%20Heterogeneous%20Graphs/%E7%9B%B8%E5%85%B3%E6%96%87%E7%AB%A0)/Proximity Embedding）

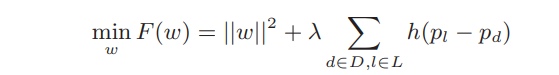
1. Supervised Random Walks: Predicting and Recommending Links in Social Networks（2011）

SRW算法的输入是(s,D,L)三元组，其中s是所要进行Link prediction或者link recommendation的开始节点；D是一个集合，表示未来可能与s连接的节点的集合；L是一个集合，表示未来不会与s形成边的节点的集合。SRW的计算流程如下：



该算法与PageRank算法特别类似，只是在PageRank中是使用节点的出度来定义的状态转移矩阵Q，而在本算法中使用的Q是通过权值w以及边的特征计算得到的。

目标函数：

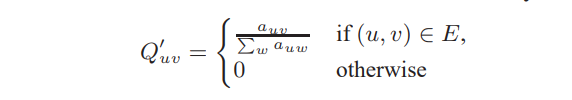


pd表示d节点可能与s节点之间存在边的概率，它的计算公式：（pd是p向量的第d个分量）



Q的计算公式：





每次迭代计算的时候都要首先使得p迭代收敛，然后再迭代使得收敛。

该算法的缺点是明显的，计算量过于巨大，每更新一次参数w，都需要计算整个图来计算收敛的p以及p对w的偏导数。

