中南大学地球科学与信息物理学院2019秋

《空间数据挖掘》大作业



班级： 地信1602

学号： 0106160223

姓名： 李源 张婷

任课老师：刘启亮

报告成绩：

目录

[中南大学地球科学与信息物理学院2019秋 1](#_Toc29505314)

[《空间数据挖掘》大作业 1](#_Toc29505315)

[课程报告 一种轨迹数据的数据挖掘可视化系统 3](#_Toc29505316)

[2【摘要】 3](#_Toc29505317)

[2【Instruction and implements】 4](#_Toc29505318)

[2.1【Instructions】 4](#_Toc29505319)

[2.2【轨迹数据的处理】 5](#_Toc29505320)

[2.3【单个轨迹数据可视化】 9](#_Toc29505321)

[2.4【轨迹数据全局信息可视化】 14](#_Toc29505322)

[3【总结】 19](#_Toc29505323)

[4【参考文献】 20](#_Toc29505324)

[5【小组成员贡献】 20](#_Toc29505325)

# 课程报告 一种轨迹数据的数据挖掘可视化系统

## 2【摘要】

轨迹数据挖掘一直以来都是研究工作的热点，而且跨度十分广泛，从数据产生，到数据处理，到数据模式挖掘。本文根据可视化的最新成果，提出了一种轨迹矩阵的可视化的方式，为城市可视化，和城市间的对比，以及可视化的解译轨迹数据提供了新的思路。本文主要使用了两种可视化手段，来构建整个轨迹的可视化的系统。其中第一个局部轨迹可视化的手段，是使用经典的嵌入方法，比如node2vec，struc2vec等，这种可视化手段，将某一个轨迹可视化成为低维度空间的节点，从而对outline轨迹节点的寻找，以及轨迹模式的挖掘提供了有效助力，但是这些基于采样的方法并不适用于我们轨迹所面临的情况，大规模计算太难，成本太高，这里我们选择了ProNE方法（2019最新图嵌入成果）。而另一个全局轨迹可视化的手段，是使用gcn搭配变分自编码，来寻找一个可以体现城市中所有轨迹的一个更加宏观可视化结果，而且变分自编码的结构也为这种可视化的可解释提供了更好的支撑。但是由于算力不够，我拿到的路网数据的矩阵轻则30多万节点，多则几百万节点，电脑跑的内存溢出也没有跑出来，所以第二种手段仅提供可执行代码，老师看到可执行代码之后，跑了之后，觉得我这个模型还ok，有做的价值，愿意提供算力，可以联系我，我小年才回去。

## 2【Instruction and implements】

### 2.1【Instructions】

轨迹数据挖掘一直是一个非常重要的领域，这个领域的跨度非常的大，从数据的产生（停留点监测，Map-Maping算法，轨迹数据的切分，轨迹数据的压缩—），再到数据的处理（在线版本快速的轨迹采样，离线版本的轨迹采样，主要是滤波，比如kalman滤波，粒子滤波等等），到轨迹数据的快速查询，轨迹数据的索引（如何从多个维度去建立索引，让轨迹数据快速的查找，比如github上的名叫torch的轨迹查询引擎），压缩（和地图综合联系紧密）等等吧。乃至轨迹数据的模式寻找和挖掘(太多方法)，以及轨迹数据的转化，比如转network，转tensor，转matrix等等。具体的分类如下图所示：

图片包含 文字

描述已自动生成

本人在工作的过程中发现，单纯的轨迹数据的挖掘并不能有好的结果，因为轨迹数据太过驳杂，反而将轨迹数据转化为network和tensor后更有利于数据挖掘过程的工作，在此基础上本文提出了一个轨迹数据挖掘可视化的系统。以下是每一部分的工作量的概述。

### 2.2【轨迹数据的处理】

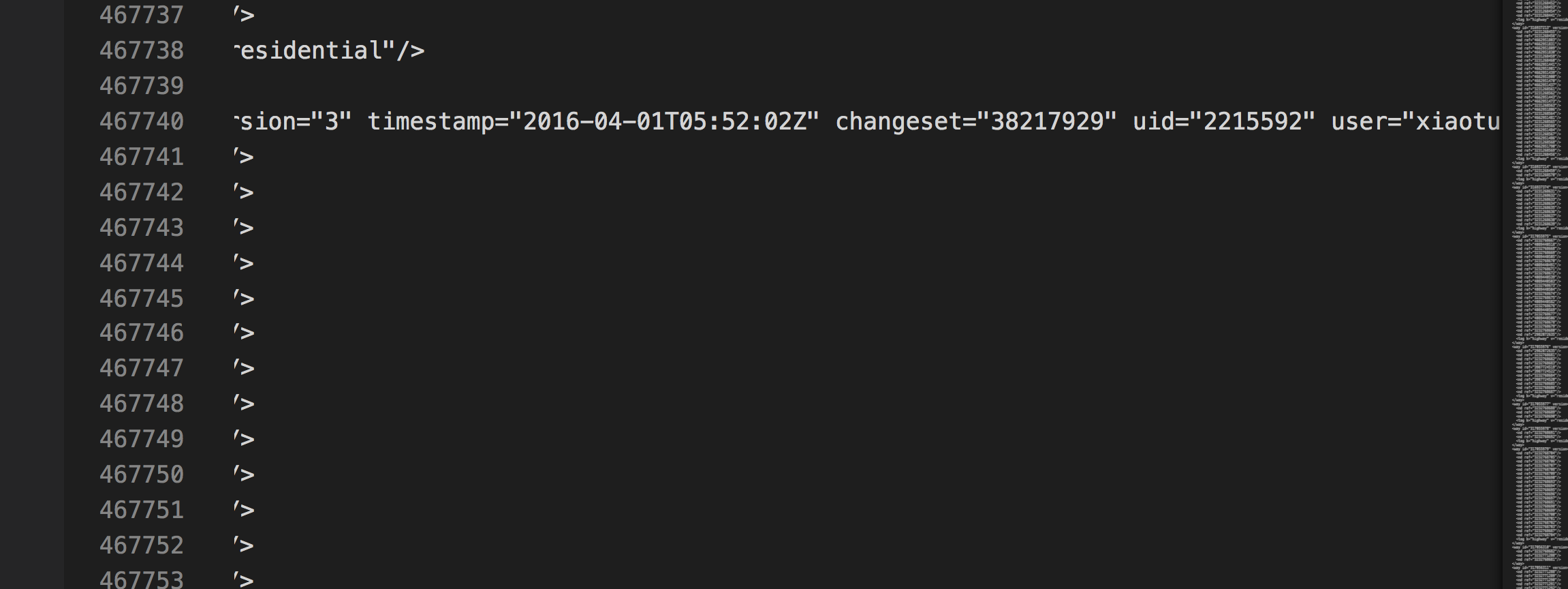
要实现我所需要的数据，这个数据必须要有对应的已经分好段的轨迹数据，必须要有对应的路网（包括节点信息和语义信息），需要这两者数据之后，才能够在这个基础上，进行上面所写两个方案（node2vec等方法处理信息进行局部样例可视化，寻找outline，而使用vaes和gcn进行全局轨迹的可视化，寻找全局表征，并进一步的利用vaes的优势进行可解释的工作）。所以我使用google dataset search和老师给的微软亚洲研究院的Yu Zheng的关于轨迹数据挖掘的数据集，在处理了超过10个数据集之后，排除了kaggle等等数据挖掘竞赛数据集，整个过程历时2天,老师提供的数据集全部失败，大都没有路网匹配数据，更别提标签数据。在阅读老师提供的*Trajectory Data Mining*这篇review论文的时候，我发现了osm提供类似数据，以.osm.pdf和.osm的格式提供对应的信息，同时数据放出来的大小大概是10个g多，必须自己从中根据关键词提取对应的城市。我拿到了超过10个市的数据，包括城市的节点数据，节点的link数据，以及人类轨迹在node和link之间经过的数据。（但是数据质量不敢保证，都是某用户很长时间段内的轨迹信息数据，由于我个人无法对用户的采样负责，所以我无法对这个数据和分析的数据进行负责，但是本文的重点在于从局部角度和全局角度提出一种轨迹信息可视化的系统）

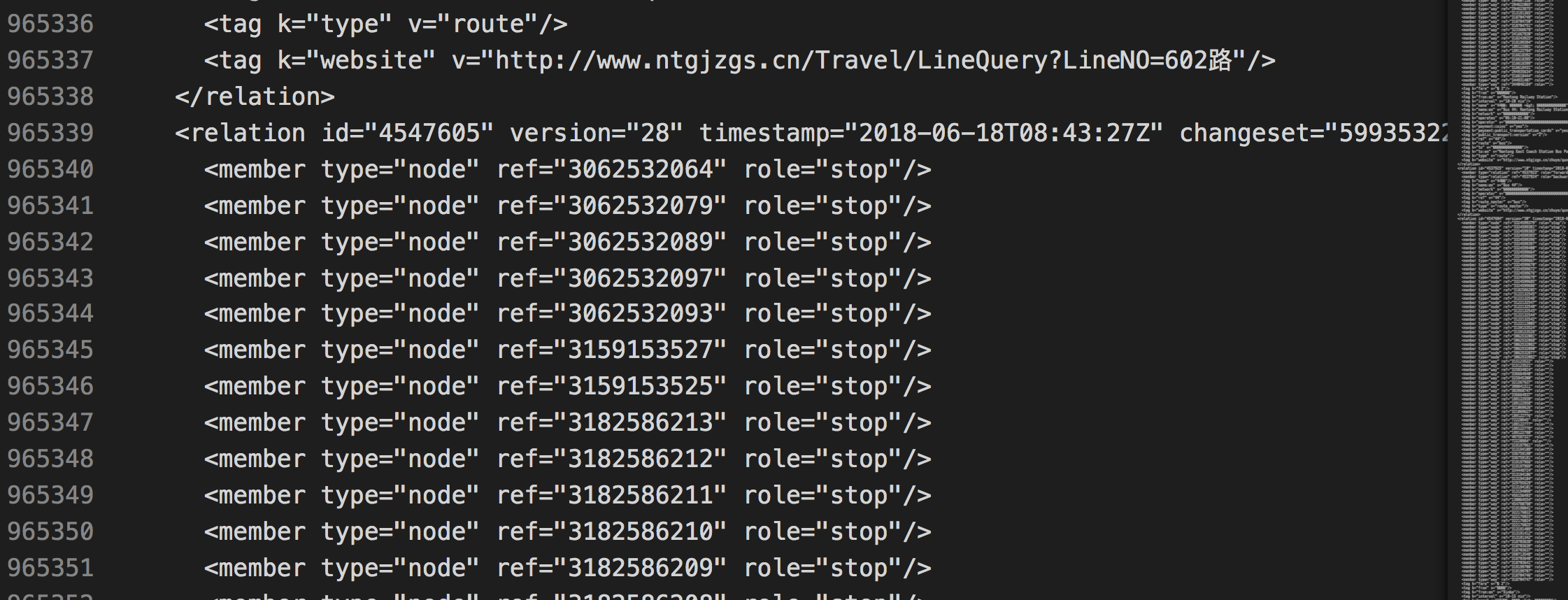
于是我参考老师给出的*Trajectory Data Mining*这篇文章的trajectory的to graph部分，对轨迹数据和路网连接数据进行了转化。

代码首先使用第三方库，将.osm.pbf完全转化为osm数据，数据的格式如下所示：

图片包含 文字, 户外

描述已自动生成



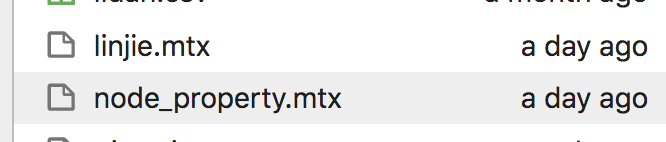


由于这里仅仅是一个地级市的数据，就有超过30万个节点，同时采集的link数据就有超过60万，采集的轨迹数据就有超过2k用户，而北京的数据就更加惊人，因为我之后会使用gcn，但即使我使用稀疏矩阵相乘（复杂度与所有不为0的数据相关）也不行，这个参数量太过惊人。通过熟悉数据格式，最终将数据转化为三个稀疏矩阵，一个是node数据的邻接矩阵（参数量30万\*30万，化为稀疏矩阵之后计算提升并不明显），至于另一个就是轨迹 to graph的稀疏矩阵（参数量30万\*30万），还有一个是我考虑到我们的用途，认为这么高的维度没有必要，所以使用pca对数据进行了降维，让数据转变成了128维度。

所以这一步的结束，我们拿到了三个数据：一个是路网的邻接数据（数据量为30万\*30万），一个是原始的轨迹to graph的矩阵（数据量为30万\*30万），一个是降低维度后的轨迹to graph属性矩阵数据（30万\*128）这三个数据，如下图所示：

图片包含 屏幕截图

描述已自动生成

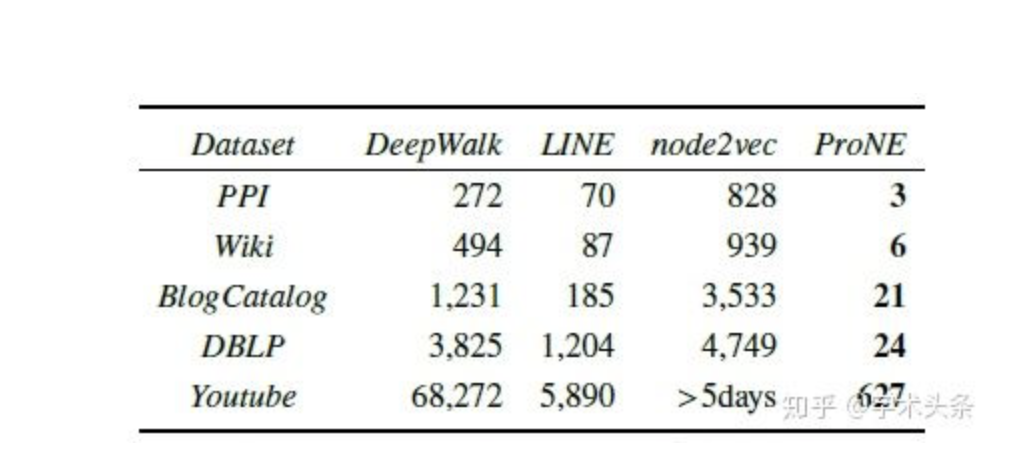


到这里我们即完成了所有数据的清洗和处理的工作。

### 2.3【单个轨迹数据可视化】

**2.3.1 【单个轨迹局部可视化综述】**

在这个过程中我最开始使用了node2vec,struc2vec，等方法进行可视化，最开始的时候我用3行 node2vec进行了实验，因为参数太大，电脑跑的太热，内存占用很容易跑爆炸。所以我认为deepwalk，node2vec，struc2vec这种采样的方式肯定不满足要求，于是我选择去寻求一种快速，同时又能够对高维轨迹向量就行快速嵌入的方式。最终，我在最新的2019年的IJCAI论文中找到了ProNE，以下是论文中演示的效果：

****

结果表明它能够在超过100万的节点上工作的很快，而且效果不错，正适合我这种非常稀疏的矩阵。下面是对这种方法伪代码描述。

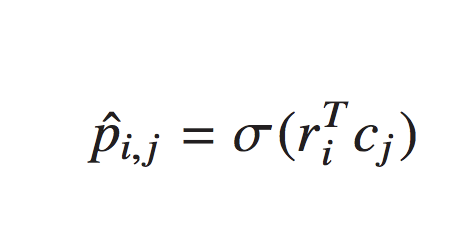
**图片包含 屏幕截图

描述已自动生成**

**2.3.2【分析】**

算法的第一步是构建M相似矩阵，也就是这个矩阵上的每一个值（i,j）都等于i节点和j节点的相似程度。这个相似程度的度量，作者是下了大功夫的。我这里默认您是图嵌入方向的老手，不再阐述嵌入节点和背景节点的概念，其实也简单，嵌入节点是你现在focus的那个节点，背景节点就是该节点的拓扑意义上的相近的邻居。

然后就通过下面的公式计算：

****

**图片包含 物体

描述已自动生成**

在这里是singmoid函数，符号ri,ciRd分别表示节点vi的嵌入表示向量和背景表示向量。至于D我就不解释了，非常容易看出D就是edge集合。

在我们这种应用场景，也就是使用原始的轨迹to graph的矩阵的轨迹向量去计算相似性。我们可以很明显的发现图片包含 物体, 天空, 室内

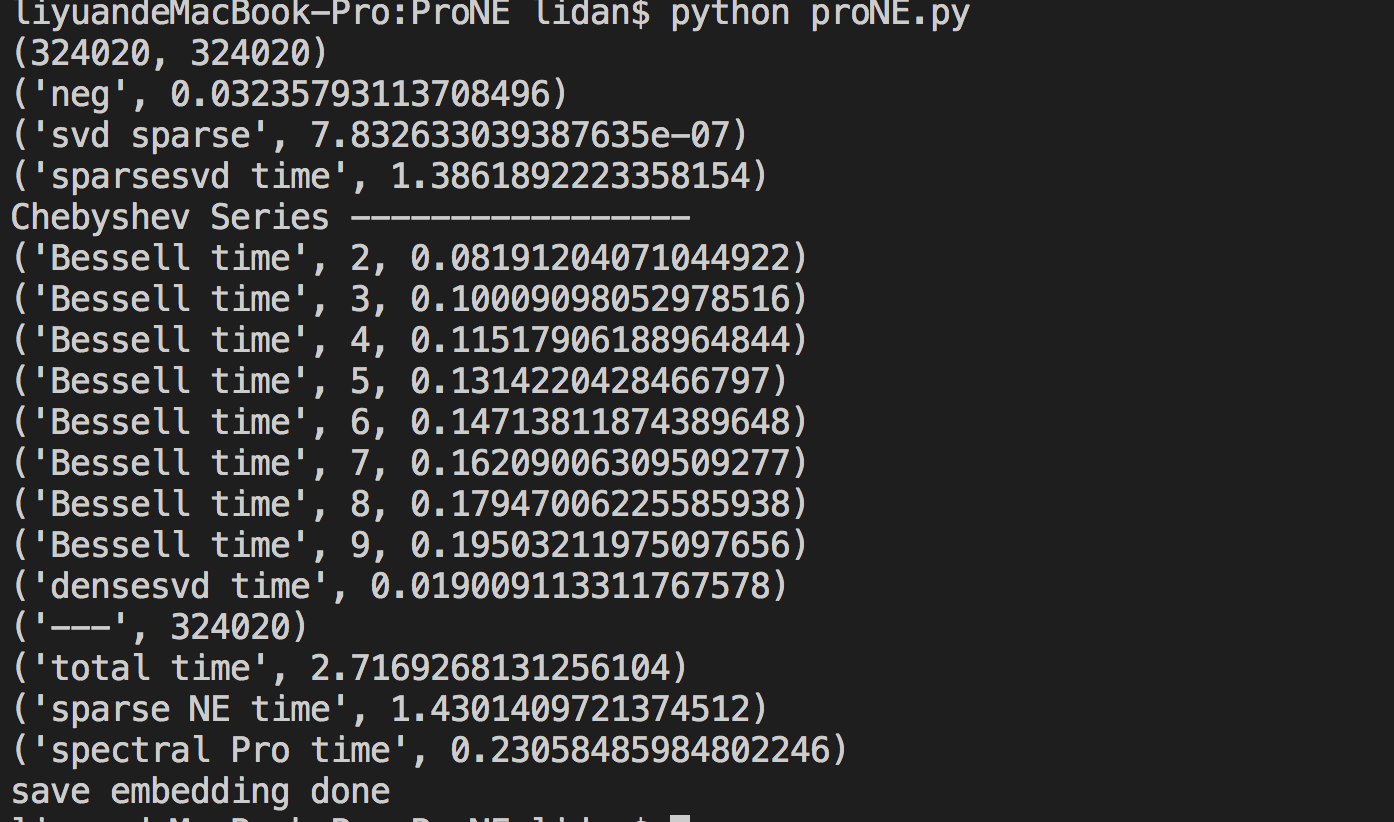
描述已自动生成这一项就为了刻画着两个节点之间的直接相关性，图片包含 物体

描述已自动生成这一项是为了刻画和j和其他节点的相关性，这样就能刻画两个节点的真正相对于邻域的相关性了。

所以很自然，根据nlp现在很火的分布假设的相似性进行的图嵌入的问题就转化为了这个相似矩阵的分解问题。将M分解为你想要的维度的就可以了，相当于SVD进行降维操作。这里作者使用了TSVD，所以才有了后面的计算的步骤。这里我也就不再多说。

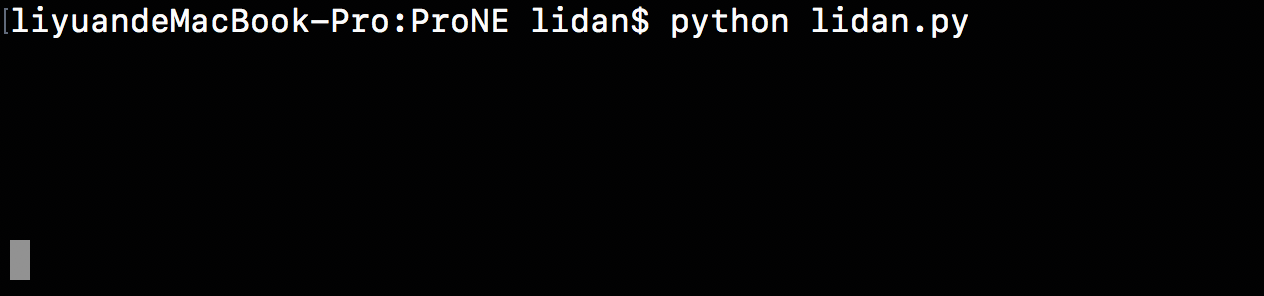
**2.3.3【成果】**

（实现见代码，同时附上THU提供的C++代码，它的代码直接进行了mkl的cpu计算的高性能优化）：

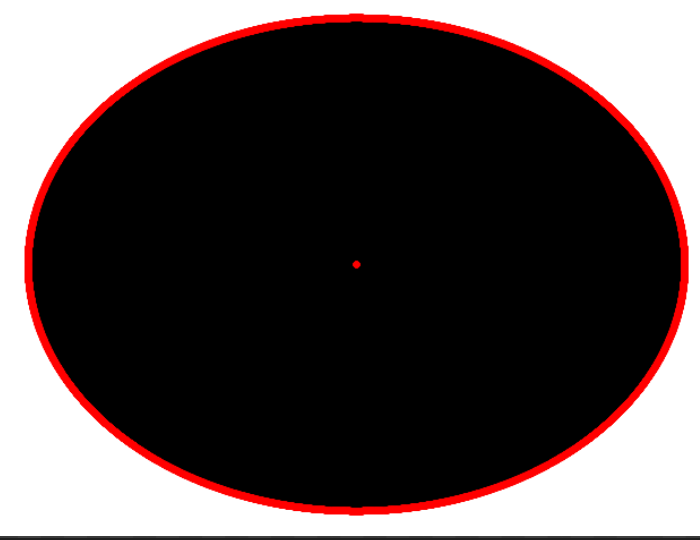
****

**图片包含 屏幕截图

描述已自动生成**

****

嵌入效果直接跑到崩溃,我的电脑支撑不了这样的可视化。请老师分配算力，这样我本人就可以跑出来了，事后老师如果有算力可以直接联系我。网络可视化（30万node为红色，100万边为黑色）为下图：

****

由于缺少提取的标签数据，所以最后效果只能如上图所示。OSM拿不到地物node的标签数据。

由于缺少label数据，所以接下来的工作无法进行，但是降维工作做的非常出色。

### 2.4【轨迹数据全局信息可视化】

**2.4.1【综述】**

我进行轨迹全局信息可视化的idea来自于A Deep Generative Model for Graph Layout这篇论文（由于该论文没有提供代码，需要一行一行自己写），这篇论文对于网络的所有节点的layout属性提供了一个共同可视化，也就是寻找一个向量作为所有节点的全局表征，这篇论文是数据挖掘可视化方向的最新论文。我想我们既然已经拿到了邻接矩阵，同时也拿到了轨迹to graph的矩阵，那么我们就可以有一番作为了。

但是，想要得到轨迹数据的全局可视化是一件非常难得事情，需要经过以下两个步骤：

1. 首先你要拿出来有说服力的证据，你得到的东西，是所有轨迹数据的嵌入，也就是说一个城市所有路网上的轨迹数据的信息都凝练到你的嵌入中，也就是你的嵌入要包含路网信息，也要包含所有的轨迹信息。
2. 其次，要想得到这样一个全局轨迹信息可视化的手段，你还必须对可视化进行一些综合性的可解释性的分析，因为如果你的可视化无法进行可解释性的分析，虽然我们可以进行一些城市间的对比分析，但是你的分析就没有太大意义了。所以我们的分析必须能够可以解释，至少是可视化效果的可以解释。

综合前面两个手段，我们发现很难做好这样的轨迹信息的可视化，即使我们已经拿到了路网matrix和轨迹matrix。所以这里我选择了如下所示的结构，下面针对该结构做一些说明。

**2.4.1【分析】**

以下是整个网络的结构：

**图片包含 屏幕截图

描述已自动生成**

下面分为encoder和outcoder进行分别叙述：

**图片包含 音乐

描述已自动生成**

在结构中，A是路网的邻接矩阵，P是每个node的轨迹属性矩阵(已经pca降维的)，第一步，我们对属性信息进一步的提取特征，也就是128维度继续进行降低维度，也许10-20维度是一个比较好的选择，也就是a标记是一个dense layer，之后我们拿到了每个节点的10-20维度属性（轨迹属性），然后通过一个3层的gcn，在通过一个contact Dense layer（见实现代码），然后在通过一个MLP（多层感知机）我们就拿到了一个轨迹信息的全局的表征图片包含 物体

描述已自动生成。这就是是一个encoder的过程。

如果图片包含 物体

描述已自动生成是一个全局的表征，那么我们通过给zl的每一维度都加上一个one-hot分量，也就是下图中的fusion操作，在经过另一个三层gcn之后，再经过一个三层感知机，然后一层denselayer，经过训练能够恢复原来的图片包含 物体, 时钟, 室内, 墙壁

描述已自动生成。那就说明图片包含 物体

描述已自动生成确实是一个全局表征，他就能够用于可视化的操作。这就是一个decoder的过程。

**图片包含 音乐

描述已自动生成**

然后我花了4天时间来写这个模型，总共19层。很可惜，在本篇文章的这部分，我实现了非常多的代码，为了更好的提升速度，我没有使用keras，而是从头到尾写了自己的代码，一共1000行左右，但是最后这个算法跑的太慢了，因为有gcn层，所以必须一次性把数据feed进去，的计算机不支持这样强度的计算，所以仅仅提供给老师可执行的代码，老师如果有兴趣可以联系我，我们一起把这个工作做完。

**2.4.3【成果】**

下图即是证明：图片包含 屏幕截图, 文字

描述已自动生成

图片包含 屏幕截图

描述已自动生成

这部分的代码工作除去注释一共953行，写了4天。

在实现过程中遇到了非常多的问题，所以比如tensorflow的各种API版本问题。

图片包含 屏幕截图

描述已自动生成

比如tensorflow的api更改的问题，比如tf.sparse\_placeholder，tf.add\_n,tf.concat等api在不同tensorflow版本中均有巨大不同。

将可执行代码提供给老师，老师如果感兴趣，愿意提供算力，可以做下一步的工作。

## 3【总结】

我们参考了最新成果的5篇论文和技术文档，构建了一个空间数据挖掘的可视化系统，比如2019年的IJCAI论文中找到了ProNE以及tsvd的实现技术文档，提出了单个轨迹可视化的方法，同时参考2019年IEEE vis的A Deep Generative Model for Graph Layout方法，结合vaes和gcn提出了的到全局所有轨迹的统一表征的方法。整个系统的构建非常成功。

纯算法代码代码量超过1700行，除ProNe算法参考别人代码修改应用到轨迹数据图之外，其他均为自己一点一点手写。参考了keras的部分源码。

## 4【参考文献】

[1]. ProNE: Fast and Scalable Network Representation Learning – Jie. Zhang, Yuxiao Dong, Yan Wang, Jie Tang, Ming Ding - IJCAI 2019

[2]. Deep Generative Model for Graph Layout IEEE Transactions on Visualization and Computer Graphics. In Proc. IEEE VIS 2019 (InfoVis); doi:10.1109/TVCG.2019.2934396

[3]. Kingma D P, Welling M. Auto-Encoding Variational Bayes[J]. stat, 2014, 1050: 10.

[4]. DOERSCH C. Tutorial on Variational Autoencoders[J]. stat, 2016, 1050: 13.

[5]. Tsvd技术文档

[6] Zheng Y . Trajectory Data Mining: An Overview[J]. ACM Transactions on Intelligent Systems and Technology, 2015, 6(3):1-41.