各位老师，下午好，我是李奇平，我的毕业设计的题目是社交网路中部分信息缺失下的社区发现。这个毕业设计是在我浙大的导师王灿老师和陈英老师的指导下完成的。

下面我将对本论文的主要内容做一个大致的介绍，在课题背景部分，我会介绍本论文的研究目的和研究手段，通过这个部分各位能知道我在这个论文中解决了一个什么问题，以及是如何解决的。在研究内容部分，我会更加详细地介绍本文提出的算法，包括一个距离度量学习算法和一个聚类算法，使用这两个算法我们能完成社交网络中的社区发现问题。在实验部分，我会讲述我们如何验证我们算法的有效性。

首先我们介绍论文的研究目的，从论文的题目可以看出，我们在这个论文中要解决的是如何在社交网络中进行社区发现或者是社区挖掘的问题。我们首先来看什么是社区，社区是社交网络中联系比较密切的一组人，他们之间有较多的共同点。比如说在人人或者微薄上我们有很多的好友，为了便于管理，我们会把我们的好友分为初中好友，高中好友，或者按照不同的领域，比如有些好友是计算机领域的，有些好友是一起打球的。我们可以把这样的一组好友成为一个社区，比如，初中同学是一个社区，他们都来自于同一所中学。本文的目的就是在社交网络中自动地找到这样的社区。同时，我们还需要关注“信息缺失”，因为现在的社交网络无比庞大，里面有很多的用户，因此我们无法把这样一个整个的社交网络的信息全部获取到，我们的目的就是要在这样信息缺失的社交网络中进行社区挖掘。

我们的做法是这样的，虽然我们无法获取整个社交网络的信息，但是我们可以得到少数几个局部信息网络，局部信息网络的定义我待会会介绍，利用局部信息网络的信息我们可以学习到一个距离度量，通过这个距离度量，我们可以得到所有节点之间的距离。然后我们使用一种聚类算法基于这个距离进行聚类，得到的一个个聚类就是我们需要的社区。

下面我具体地介绍本文的算法。首先，我们先对信息缺失做一个定义，正如刚才提到的，我们无法得到社交网络的全部信息，但是我们可以得到少数借个局部信息网络，在局部信息网络中每个节点的信息是已知的。这是可以做到的，比如说获取整个人人网所有用户的信息是不可能的，基本上全国所有的大学生都上人人网，但是我们可以从其中调查几个班的同学，这些同学的信息我们都可以得到，这样的一个班，我们可以把它当成一个局部信息网络。有了这些局部信息网络的信息，我们就可以知道哪些节点是相似的，哪些节点是不相似的。利用这个去进行距离度量学习。

然后我们来介绍一下距离度量学习。距离度量学习的目的就是为了计算所

有节点之间的距离，本来欧氏距离是一种比较简单的计算节点间距离的方法，它仅仅使用节点的特征属性计算节点之间的距离, 但是现在我们有了局部信息网络的信息，我们希望能够利用局部信息网络的信息对欧氏距离距离进行一个调整，让相似的节点之间的距离更近，不相似的节点之间的距离更远。因此，我们使用马氏距离来作为节点之间的距离，可以马氏距离比欧氏距离多了一个矩阵 M, 我们就是通过这个矩阵来对欧氏距离进行调整，我们把矩阵 M 称为距离度量，所以我们现在的目的是为了得到一个这样的矩阵 M, 通过它计算出的距离让相似的节点之间的距离更近。通过建立数学模型，求解 M 可以转化为这样的一个最优化问题，求解这个最优化问题能够得到一个矩阵 A，然后就能求得矩阵 M。这样，我们就能计算节点之间的距离。

在计算出了节点之间的距离之后，我们使用本文提出的聚类算法 DSHRINK 对节点进行聚类。DSHRINK 是一种层次化的聚类算法，经典的层次化聚类算法每次把两个最近的节点聚在一起，形成一个超级节点，不断地重复这个过程，最后所有的节点都会聚成一个节点，因此，如果要形成 k 个聚类，在还剩 k 个节点的时候停止聚类即可。DSHRINK 判断聚类终止的条件和这个不一样，它使用模块性准则作为聚类终止的判断条件。我们待会会给出模块性准则的定义。

DSHRINK 把节点聚到一起的过程和经典的层次化聚类算法也有所不同，它的过程是这样的：首先，找到每一个节点的节点的最近邻居，也就是距离它最近的节点，然后找到所有的互近邻节点对，节点 u 和节点 v 为互近邻当且仅当 u 是 v 的最近邻，v 也是 u 的最近邻。有了互近邻之后，我们可以得到一系列的当地社区，当地社区其实就是一个节点的集合，它满足这样的三个性质，通俗的说，当地社区中所有节点之间的距离都是很近的，也正因如此，整个 DSHRINK 算法就是通过不断地把当地社区中的节点合并为一个超级节点来实现。合并之后，超级节点又被放入原来的信息网络中，开始下一轮的合并。不断的重复这个过程，直到遇到终止条件。

那么 DSHRINK 算法什么时候终止呢，我们使用模块性准则作为聚类终止的判断条件，所谓的模块性准则，就是聚类质量的好坏一个评判标准，如果一个把一些节点聚成一个聚类导致聚类质量不好，那么就不应该继续这个聚类的过程。我们定义的基于距离的模块性准则如下：每次把当地社区中的节点聚成一个超级节点时，我们都会计算两次的 Qd 的变化，如果 Qd 变小，说明聚类质量变好了，聚类继续进行，否则停止聚类。最终的一个个聚类也就是一个个的社区。

最后，为了验证本文算法的有效性，我们进行了大量的实验，我们从

Blogcatalog 这个网站上的数据生成了我们所需要的两组数据集，分别包含 5000 多个节点，同时每一个节点有类的标签，用于作为 ground truth 验证最终的结果。在这样的两组数据集上，我们使用 kmeans 算法和我们的算法做对比，观察哪个算法具有更好的效果。我们使用纯度作为效果好坏的评判标准，纯度是这么定义的：因为不管是我们的算法还是 kmeans 算法，最终的节点都是一个个的聚类, 对于每一个聚类，我们可以知道哪个标签所拥有的节点数最多，拥有最多节点的标签叫做这聚类的标签 label，这个 label 在这个聚类中有的节点个数叫做这个聚类的标签计数，把所有聚类的标签计数加起来，除以总的节点个数就是整个结果的纯度。

下面我们来看一下实验结果，我们先看本文算法和 kmeans 算法在不同的局部信息网络大小下，纯度的变化。可以看到，本文算法始终比 kmeans 算法具有更高的纯度，而且整体趋势是局部信息网络越大，纯度越高。接下来我们看在不同的局部信息网络个数下纯度的变化。同样的，本文算法始终比 kmeans 算法具有更高的纯度，而且，局部信息网络个数越多，纯度越高。

本文算法相对于 keans 算法具有跟高的纯度是因为它利用局部信息网络的信息，在欧式距离的基础上进行调整，使得计算出的距离与真实情况更加符合。而且，局部信息网络个数越多，大小越大，信息越多，计算出的距离越准确，纯度也就越高。

同时，本文算法相对与 kmeans 算法还有一个优势是它不需要知道聚类的个数，它能够基于模块性准则自动地选择一个合适的聚类个数。而对于 kmeans 而言，如果不知道聚类的个数 k，就需要尝试各种不同的 k，来选择一个最好的结果，这是一个非常耗时的过程。

好的，关于本文就先介绍到这里，感谢各位老师抽出时间来听我讲解。在这里想王灿老师和陈英老师表示感谢，谢谢他们的指导。同时，感谢大学四年陪我一起走过的同学和朋友，从他们身上我学到了很多东西。最后，向我的家人表示感谢。