**针对实体关系查询的知识图谱划分算法设计**

1183300118 符建 必修 题号：8

1. **摘要**

本次大作业涉及的是针对实体关系查询的知识图谱划分问题。实体关系查询即为针对两个实体之间是否有关系（可达性）的查询。针对这个问题，我设计了一个划分算法，我称之为两层衍生划分。在LUBM生成的数据集上验证了算法的性能和可靠性，并且与RDF查询引擎（rdf3x）进行了一定程度的比较。

1. **关键词**

知识图谱 RDF数据格式 知识图谱划分 可达性查询

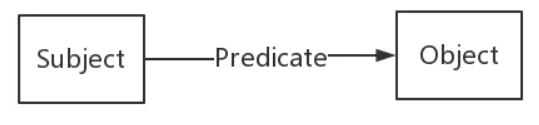
1. **技术背景**
2. **知识图谱和RDF数据格式**

知识图谱，是结构化的语义知识库，用于迅速描述物理世界中的概念及其相互关系，通过将数据粒度从document级别降到data级别，聚合大量知识，从而实现知识的快速响应和推理。下图展示了知识图谱的构建流程：



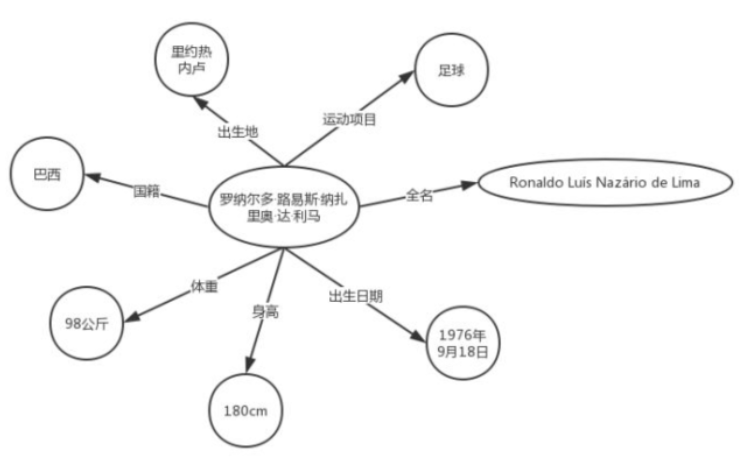
知识图谱的基本结构和单元有：节点，代表实体；边，代表关系。同时每个节点代表的实体还存在着一些属性。知识图谱就是由这些节点和边组成的网络状的知识库。

RDF(Resource Description Framework)，即资源描述框架，其本质是一个数据模型（Data Model）。它提供了一个统一的标准，用于描述实体/资源。简单来说，就是表示事物的一种方法和手段。RDF形式上表示为SPO三元组，有时候也称为一条语句（statement），知识图谱中我们也称其为一条知识。



有了RDF的表示形式和类型，如何将其序列化（Serialization）呢？换句话说，就是如何存储和传输RDF数据？目前，RDF序列化的方式主要有：RDF/XML，N-Triples，Turtle，RDFa，JSON-LD等几种。

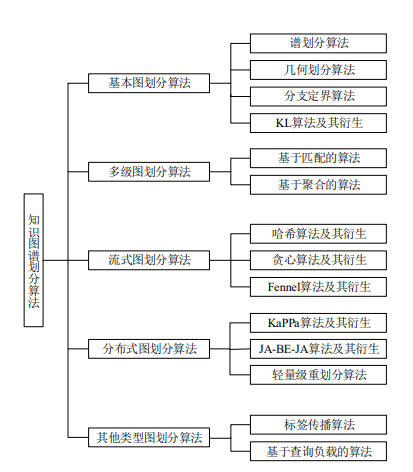
RDF是对具体事物的描述，缺乏抽象能力，无法对同一个类别的事物进行定义和描述。就以罗纳尔多这个知识图为例，RDF能够表达罗纳尔多和里约热内卢这两个实体具有哪些属性，以及它们之间的关系。但如果我们想定义罗纳尔多是人，里约热内卢是地点，并且人具有哪些属性，地点具有哪些属性，人和地点之间存在哪些关系，这个时候RDF就表示无能为力了。不论是在智能的概念上，还是在现实的应用当中，这种泛化抽象能力都是相当重要的；同时，这也是知识图谱本身十分强调的。

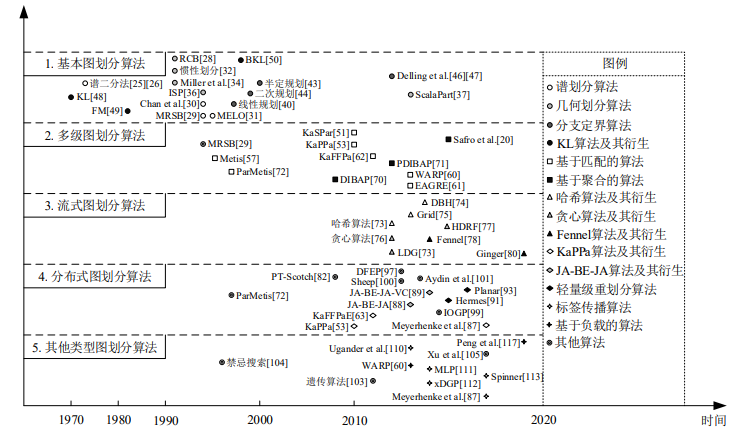


1. **知识图谱划分方法**

知识图谱的数据规模正在不断增大，必须通过分布式集群来存储和处理大规模知识图谱数据。知识图谱数据的分布式存储面临的第一个问题即是知识图谱划分。这就使得如何划分大规模知识图谱，同时满足跨分区边数目最小化且有效提高知识图谱查询处理性能成为迫切需要深入研究的课题。图划分是一个经典的NP难问题，没有恒定因子的近似算法。

KL算法是最早出现的图划分算法，随后出现的是谱划分算法，之后很长一段时间内都是这两类算法，直到上世纪90年代图划分算法的种类才逐渐增多，有了几何划分算法和分支定界算法。早期算法主要在单机下运行，随着图数据规模的增大，多级图划分算法逐渐兴起，其将原始图粗糙化为较小图进行划分，再将结果投射回原始图。之后随着大规模知识图谱的出现，多级算法也不能处理现有的大图，人们开始思考如何将现有算法置于分布式环境下运行，于是有了分布式图划分算法。并且将动态图作为划分的输入图，以适应现实中不断更新的数据。近几年提出的流式图划分算法将顶点或边转化成流输入，是另一种划分大规模 图的思路。当前知识图谱得到广泛关注，一些专门针对RDF图的划分算法开始出现，例如基于查询负载的算法。





1. **针对RDF图的划分研究**

先介绍有向标签图。有向标签图是在有向图G =（V，E）的基础上为每个顶点添加了标签。其形式化定义如下：

有向标签图 G 由三元组(V，E，L)构成。其中

(1) V 是顶点的集合；

(2) E ⊆ V × V是有向边的集合；

(3) L 是顶点和边上标签的集合。

RDF图可以看作是特殊的有向标签图，其特殊之处在于一个三元组中的谓语也可作为另一个三元组的主语或宾语，反映在有向标签图中，即边亦可作为顶点，顶点与边交集非空。

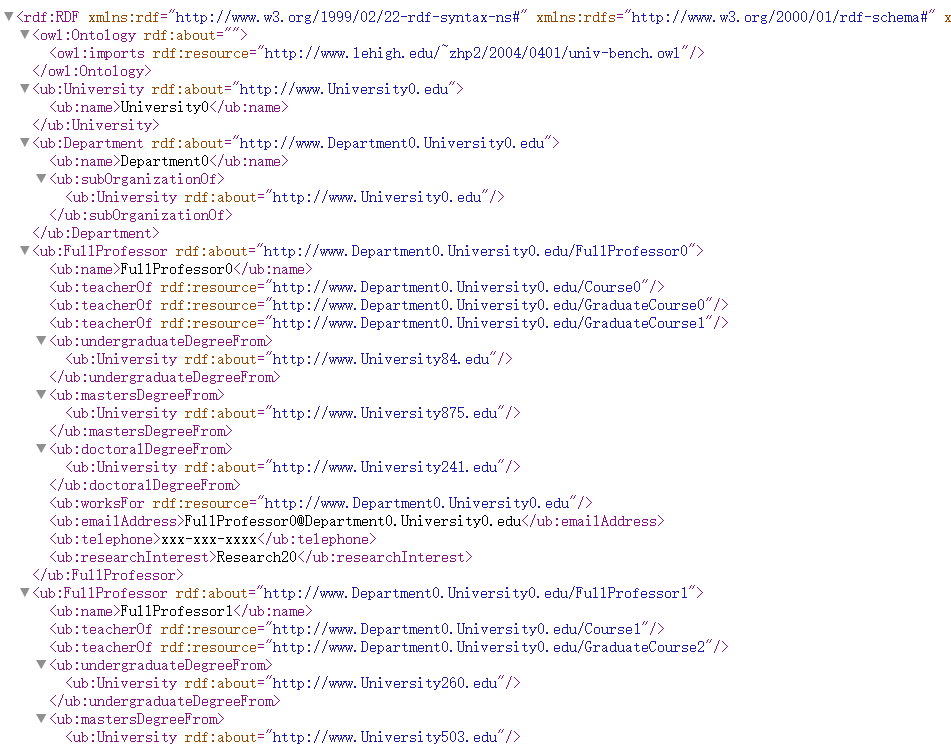
在查找相关资料的过程中阅读了几篇有关RDF图划分的算法。其中一篇提出了一个双目标优化的RDF图分割算法，将边割和分割平衡两项图分割指标融合到一个目标函数，并依据此目标函数，实现了RDF图的静态和动态分割。上述方法是基于负载均衡的，对于可达性查询的帮助有限。

如果将RDF图转化为经典图表示，那么所有的图划分算法都可以被用在RDF图中。三元组的转换是最简单，对于三元组t=<s,p,o>被转换成有向边,其中s′、o′∈ V，p′∈ L。原三元组中p和s、o是相交的，但边中p′和s′、o′不相交。RDF比传统图更通用，有向有标记的图可以很容易转换成RDF图，但是反向变换很麻烦。这意味着每个RDF图的问题的复杂性并不比相应的经典图形问题的复杂性要好。

1. **LUBM数据集**

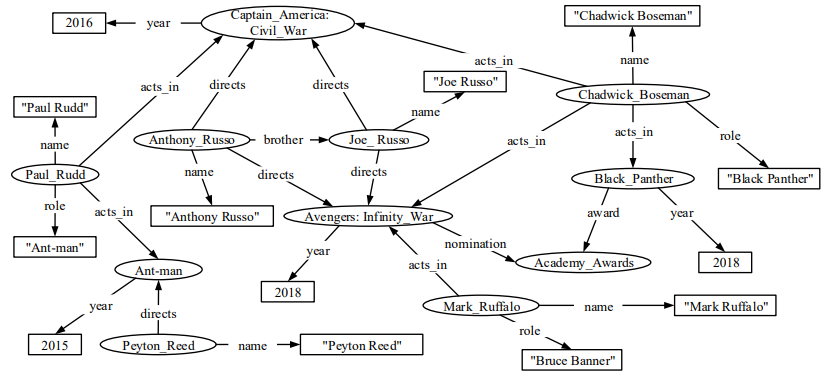
LUBM所做的是提供大学部门，班级和教授的小本体。数据生成器生成任意大型 ABox 语句。该工具在大学单位的Univ-Bench本体上生成合成OWL或DAML + OIL数据。这些数据是可重复和可定制的，允许用户指定随机数生成的种子，大学的数量和大学的起始指数。默认生成OWL数据。

生成的数据样式如下：



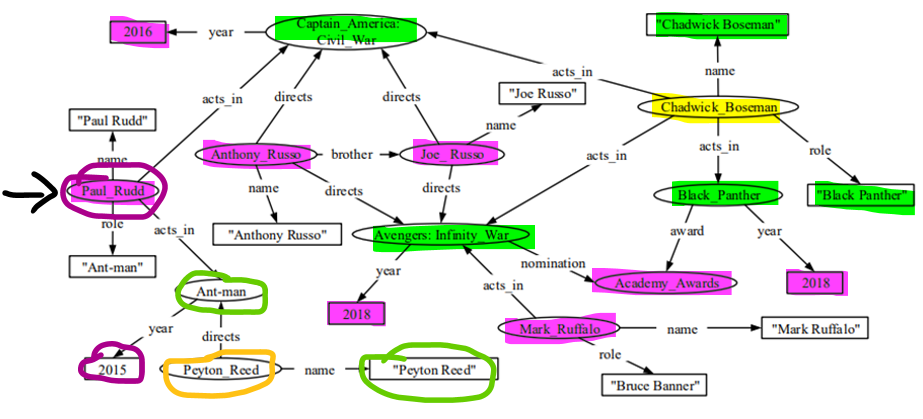
1. **算法设计分析**

想要判断图中个点是否可达，只要可以找到一条连接两点的路径。但是由于数据集巨大，想要获得精确的联通量来判断两点之间是否可达是需要巨大的代价的。对于巨大的RDF图，可以牺牲一些精度，换取一个近似的结果。对于下边的RDF图：

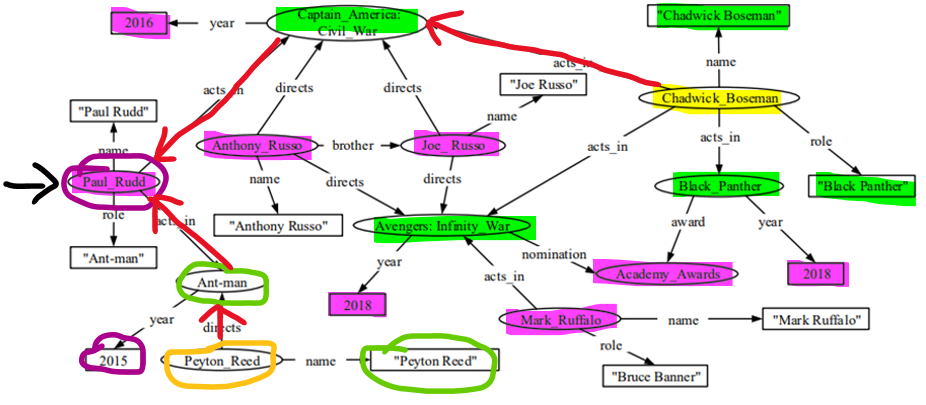


一个节点存在多个属性，这些属性可能是节点也可能是文本。如果节点的属性还是节点的话，那么从子节点也可以延伸出更多的节点。传统的BFS方法就是一层一层这样扩展出去，看看有没有到达另一个节点的路径。但是对于现在要处理的图来说，以这样的扩展规模，那么维护的节点集合会十分的巨大。

为了维护一个较小的节点集合，可以限定扩展的范围。从一个节点出发，首先衍生出其子节点，然后子节点然后再衍生它的子节点，限制这个衍生的层数，假设为两层。从需要查询可达性的两个节点都做两层衍生，对于这两个衍生出来的子图，查看其是否有交集，有交集的话就说明这两个节点之间存在可达性，路径也可以找出来；如果没有交集的话，就判断其没有交点。



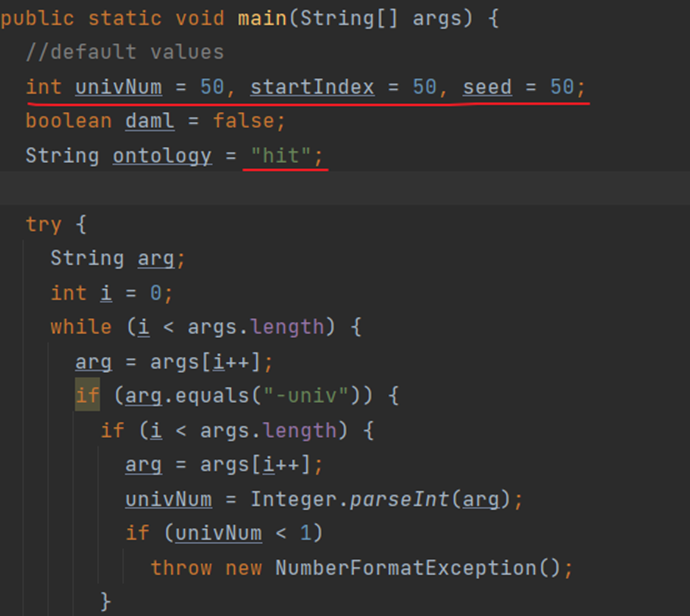
还是刚刚上面的图，衍生的节点被用颜色填充或者圈出来。用颜色填充和用颜色圈的代表从两个不同的源节点衍生出来，黄色代表源节点，绿色代表第二层衍生，紫色代表第三层衍生。可以发现这两个衍生出来的子图存在一个交点（使用黑色箭头指出），那么我们就可以找到一条路径连接两个源节点（红色标出）。



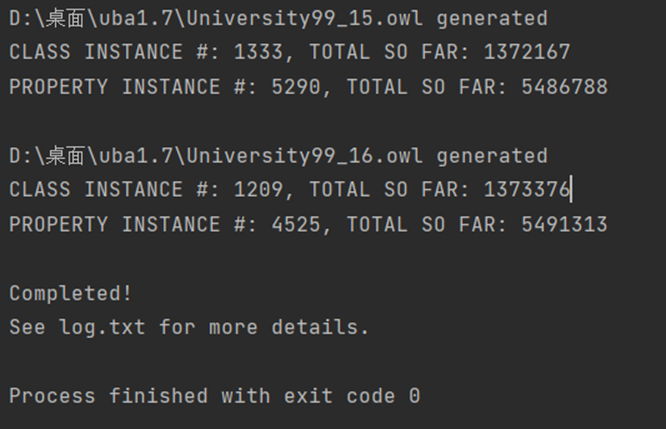
为了处理巨大的数据集合，这样的策略维护的是一个小的集合，牺牲了一些精度，通过对于LUBM数据集的观察，发现其节点分布还是比较扁平的，所以即使是限定了衍生的层数应该还是可以取得一定的效果。

1. **具体实现**
2. **生成LUBM数据集**

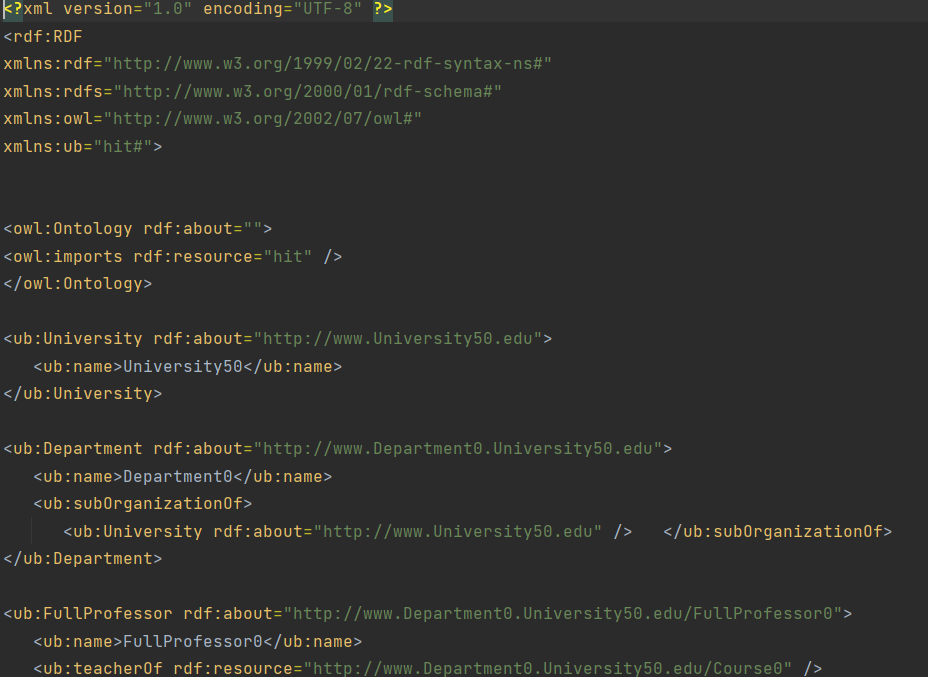
使用UBA1.7生成器以下列参数生成数据：



运行完毕：



这个数据集生成了991个owl文件，大小超过500M，超出了我机器的处理能力，所以我只使用了一部分（78个文件，42M）。数据形式如下：

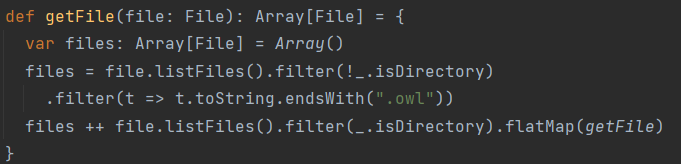


1. **预处理**

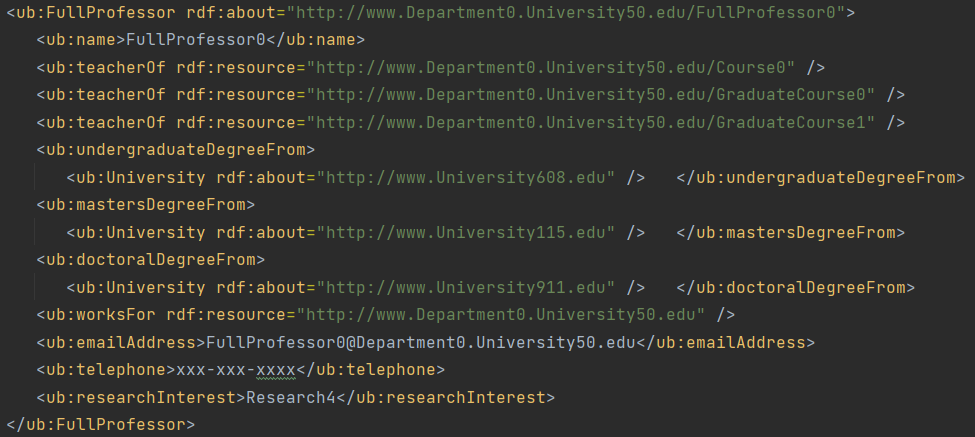
首先要获取RDF图的完整信息。需要将所有节点及其属性以及关系存储下来。使用了如下的数据结构来存储：



首先外层是一个Map，key是String类型，代表节点的类型，value是一个List，存放这个类型所有的节点。表示一个节点使用的是Map，key表示属性，value对应的是文本或者URL。Object RDFreader用来预处理数据集。



getFile函数获取数据集所有的文件名，然后read函数顺序读取这些owl文件对每个节点进行处理。观察一个节点的结构：



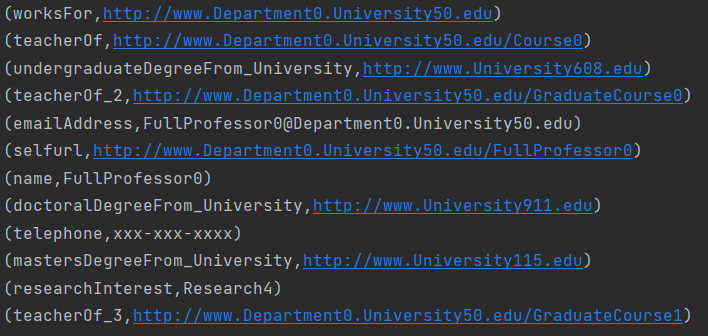
处理一个节点首先要获取节点的类型，这个节点就是FullProfessor。如果某个节点的类型是第一次出现，要在存储的RDF中生成这个类型的键。之后如果还有相同类型的节点出现，那么就添加到这个类型的键对应的List列表里。

scala中有直接处理xml格式的库：

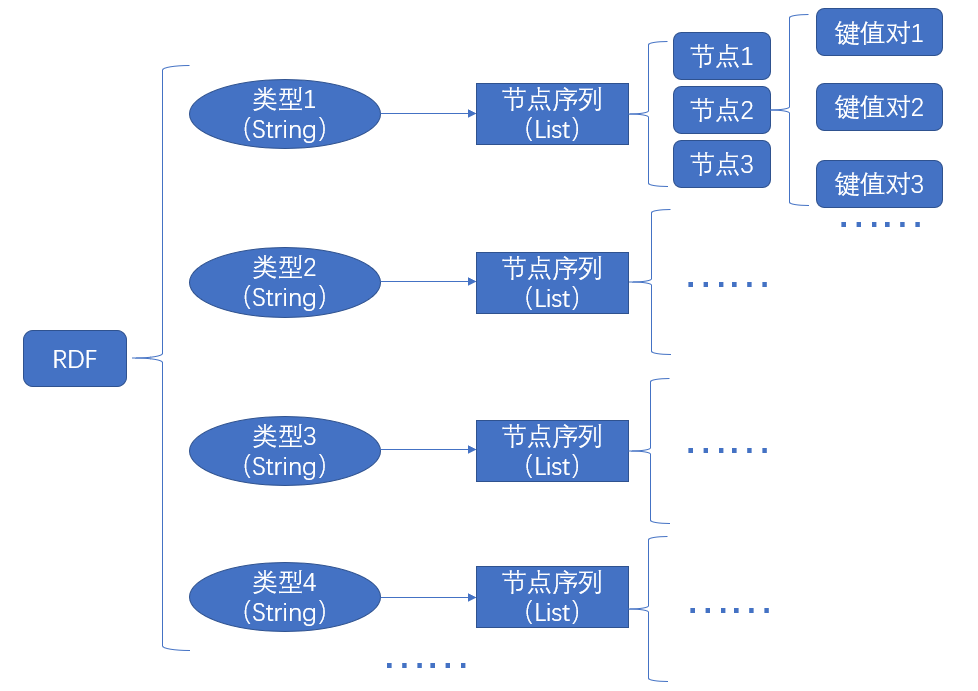


可以对owl或者xml文本进行内容提取，将一个xml节点变为保存节点序列的NodeSeq类，然后我可以对于节点信息进行处理。

对于一个具体的节点，可以将其存储为Map，包含属性和属性值的键值对。delnode函数负责从节点中提取信息，首先节点本身的URL要进行存储；如果某个属性对应的是文本，那么将（属性，文本）作为键值对加入；如果某个属性对应的是URL，那么将（属性，URL）作为键值对加入；如果某个属性对应的是另外的子节点，那么将（属性值，子节点URL）作为键值对加入。如果一个节点有相同类型，但是内容不同的子节点，那么还要加以一定程度的区分。上面FullProfessor0节点的处理结果如下：



数据集处理完毕后就拿到了RDF图的所有信息，视图如下：

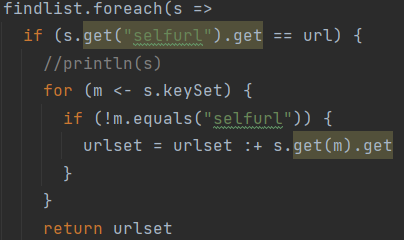


1. **查询功能**

因为任务时间有限，所以只实现了条件性的SELECT（依据SparQL查询语言）。首先获取要查询的域，要查询的条件。确定了查询的域之后，也就是获得了要查询节点的URL，那么可以根据这个URL拿到这个节点的所有属性信息，然后筛选是否有满足查询条件的结果，如果没有就返回不存在，查询功能的实现效果比较单一。

1. **可达性查询**

对于可达性查询，输入是两个表示节点的URL。对于某个URL，定义了函数DerivativeUrl，用来输出这个URL衍生出的集合。具体方法是寻找RDF图中所有的节点，如果匹配到节点的URL和输入的URL相同，那么取得存储这个节点的Map的所有键值对，可以不需要键，取得值就行。因为之后想要获得这个值和原来的URL是什么关系可以用SELECT进行查询。



对于要查询的两个节点，FindConnect函数对其做了两层衍生，首先由源URL衍生出第一层，可能有文本也可能有URL。如果是文本的话，它不能再衍生了，但是如果是URL的话还可以继续向下衍生。对于集合中的每一个URL，在调用DerivativeUrl函数，生成第二层的衍生集合。记录好第一层和第二层的衍生关系，也就是由谁衍生的。对于源URL衍生的两层集合使用List存储：

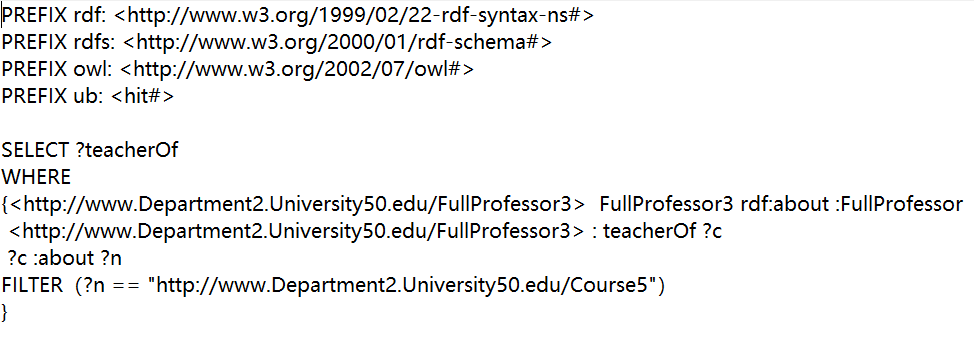


每个项包含第一层URL和其衍生的一个第二层的项。当获取了两个源URL衍生的集合之后，可以比较两个子图有没有相同点。比较的话，具体起来有3 X 3 = 9中情况，不在赘述，不同的情况有不同的输出形式。

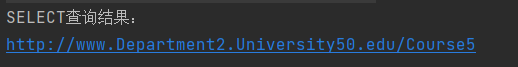
1. **运行效果**
2. **SELECT查询**

query.txt给出如下的查询，表达的意思是：

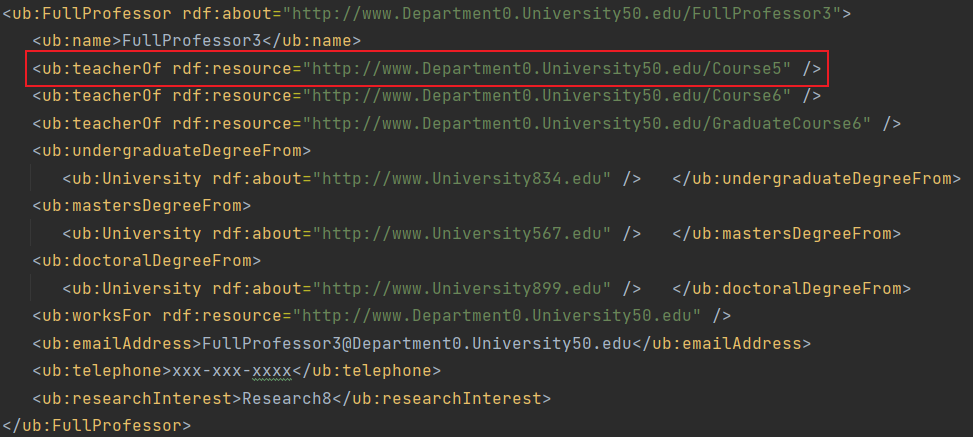
查询<http://www.Department2.University50.edu/FullProfessor3>这个域里边是否有一个关系teacherOf，内容是<http://www.Department2.University50.edu/Course5>。



执行结果：



原节点信息：

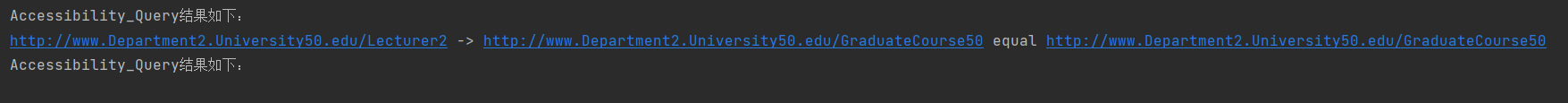


1. **可达性查询**

Accessibility\_Query.txt给出了两个两组URL，看看这两个节点之间是否有联系。

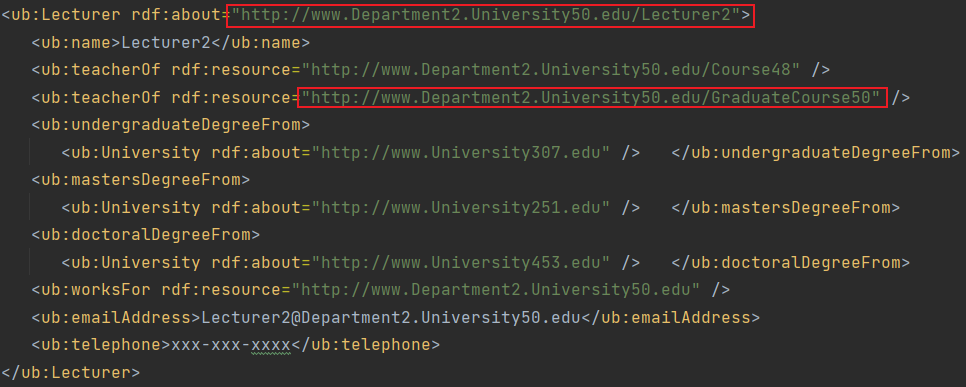


执行结果：

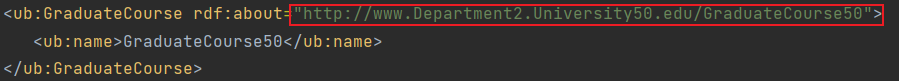


第二个查询没有任何输出，说明判定的是没有联系。对于第一个查询，原来两个节点信息如下：

<http://www.Department2.University50.edu/Lecturer2>：



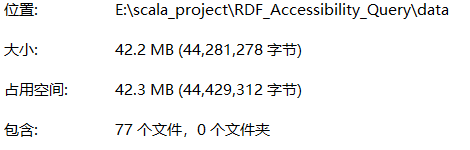
<http://www.Department2.University50.edu/GraduateCourse50>：



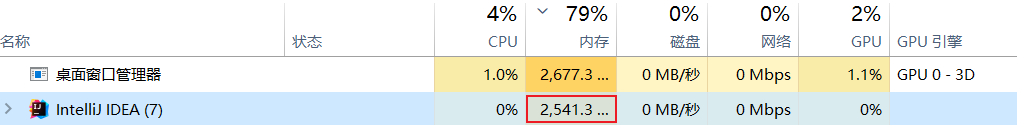
可达路径正确。

1. **性能及与其他平台比较**

虽然说算法的实现比较简单，但是可能采用的数据结构比较臃肿，处理数据过程中生成的中间结果太多，所以消耗的内存比较大。处理的数据集有48个文件，42M大小。



处理这些数据消耗的内存达到了将近2G：



程序运行时间十秒左右。所以说如果处理更大规模（超过300M）的数据我的电脑应该是处理不了了，所以说应该在分布式数据平台处理，例如spark。为了比较其他平台，我选取了rdf3x，一个RDF存储和推理平台，因为没有可达性查询的实现，所以比较了查询的性能，比我的算法快上很多。

1. **可能的改进之处**

首先可以想到的是使用更加简洁的数据结构，因为我现在存储的数据结构比较复杂，而且有很多中间处理结果存储有重复的内容。然后是在看图分割算法的过程中，有一个方法比较有意思，就是可以对图进行聚类和粗化。比如将相同或近似属性的节点合到一起，如果某两个节点被聚合到了一起，他们两个有关系的可能性就会很大。还有可以统计类型之间关联程度，根据这个关联程度直接估计两个节点之间是否有关系。

**参考文献：**

1. 王鑫 陈蔚雪 杨雅君 张小旺 冯志勇. 知识图谱划分算法研究综述. 天津市认知计算与应用重点实验室.
2. 陈志奎 冷泳林. 双目标优化的RDF图分割算法. 计算机工程与应用. 2017-09-25
3. Peng Peng, Lei Zou, Lei Chen, and Dongyan Zhao. Adaptive Distributed RDF Graph Fragmentation and Allocation based on Query Workload. IEEE. 2018
4. Dominik Tomaszuk, ukasz Skonieczny, and David Wood. RDF Graph Partitions: a Brief Survey. Communications in Computer and Information Science · May 2015