Web data mining部分：

第三章讨论部分：

读书报告内容：

1. （必填）自己提出的问题的理解（罗列全部）：
2. 提出的问题1：

p16页为什么说在整个计算过程中，我们并不需要将整个数据集加载到内存，这里没有理解。

讨论后的理解：

其实就是整个计算过程，这个算法是属于贝叶斯类的算法，是支持增量计算的。比如说，找到一个频繁集{牛奶，鸡蛋}，那么就可以对对应的count++，然后存入当前数据的内存可以继续存入下一个读取的数据，这样可以不占用内存。

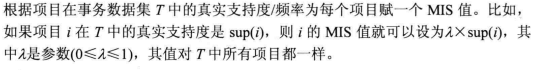
1. 提出的问题2：

p26页最后面说分两步进行，我感觉这是不是分两类就行，感觉第一步和第二步并不相关联，不知道是不是自己理解错了？

讨论后的理解：

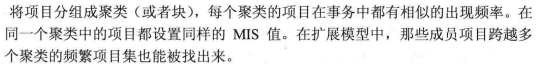
讨论后翻了本来的英文版文献，里面说的是class，应该是两类方法，应该是属于翻译错误。这里具体解释一下：

第一种方法：



其实就是对于每一个项目，根据它的自然频率，选取相同的权重lamda，然后确定相对应的MIS，这样的MIS更加符合真实支持度的比例，能够有效的考虑到频繁集和稀有集的问题

第二种方法：



对于每个聚类设置一个MIS，这样在扩展模型中，那些成员项目跨多个聚类的频繁项目集也能背找出来。

1. （必填）别人提出的问题的理解（选择几个问题罗列，并给出理解）：

问题3：P22第二段，提到为了解决稀有项目问题，一个常见的解决办法是将数据分为几个小块，每个子集只包含那些出现频率比较相近的项目。关于这个分块依据（方法）不是很理解

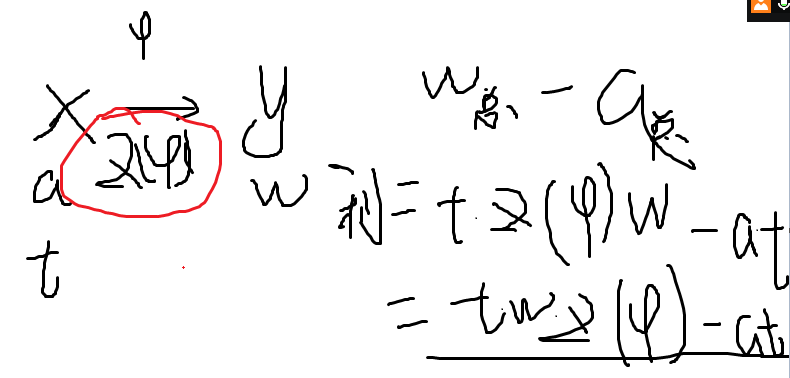
自己的理解：

这里说的数据分为几个小块是指，讲数据的项目分为几个小块，就是比如说{牛奶，面包，冰箱，洗衣机}，那么我们科研根据出现的频率，将牛奶，面包分为一个小块，将冰箱分为一个小块，将洗衣机分为一个小块。然后设置他们各自的MIS，可以解决频繁集和稀有集的问题，后面继续讨论继续优化，直接对每一个项目设置一个MIS，这样可以更加科学。

问题4：假定各个项目的MIS都由用户指定，那么参数φ应该怎么考虑呢？为什么要设置该参数呢（即，在满足各自MIS的情况下，支持度相差很大的事物为什么要过滤掉？P25

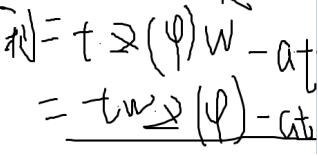
自己的理解：

这个在讨论的时候我给大伙画了这么一个图，建立了一个简单的数学模型，我觉得应该是这样的，当然这个数学模型非常简单和理想，实际情况应该复杂得多，但是应该可以说明问题了。



对于推断规则：X---》y

对于一个给定的fai，置信度必定是fai的函数，那么可以假定这样一个场景：推断规则X---》y用作广告投放，对于每个投放的x，有置信度fai的函数推导可得y，那么可以认为每个x有Z（fai）函数的概率，成功投放广告吸引顾客购买，假定每一个y可以获得利润w，每一个x广告投放成本为a，那么最后获得利润的数学期望为w总-a总，那么利润可以表述为上述公式：



其中t为投放广告的个数，w为单个y的利润，a为单个广告x的成本，那么毛利润必定是z（fai）的某个函数，那么求导必定可以获得利润的最值极值，可以转化为简单的数学模型，由于fai存在自然定义域，那么确定对应最值的的fai值，居然数学模型可能有更复杂的变化，但是大题是这样的思路，最后这个回答也得到了大家的认可。

问题6：P18 怎么得到Apriori算法是指数级的？

自己的理解：

这里可以提供两种解释：

第一种：

如果不加以优化的ap算法，那么对于一个n项目集合，可能的频繁集数目必定等于

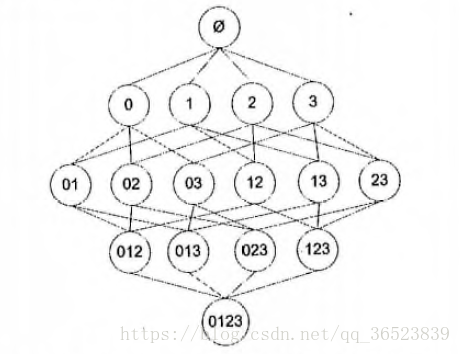
C（1，n）+C（2，n）+C（3，n）+。。。。+C（n，n），由高中排列组合基础知识可得

其结果等于o（2^n-1）。

第二种：

举例说明：

假设我们只有 4 种商品：商品0，商品1，商品 2，商品3. 那么如何得可能被一起购买的商品的组合



上图显示了物品之间所有可能的组合，从上往下一个集合是 Ø，表示不包含任何物品的空集，物品集合之间的连线表明两个或者更多集合可以组合形成一个更大的集合。我们的目标是找到经常在一起购买的物品集合。这里使用集合的支持度来度量其出现的频率。一个集合出现的支持度是指有多少比例的交易记录包含该集合。例如，对于上图，要计算 0,3 的支持度，直接的想法是遍历每条记录，统计包含有 0 和 3 的记录的数量，使用该数量除以总记录数，就可以得到支持度。而这只是针对单个集合 0,3. 要获得每种可能集合的支持度就需要多次重复上述过程。对于上图，虽然仅有4中物品，也需要遍历数据15次。随着物品数目的增加，遍历次数会急剧增加，对于包含 N 种物品的数据集共有 2^N−1 种项集组合

问题6：P23为什么“不把同时含有频繁项目和稀有项目的相集作为频繁项目生成”？

自己的理解：

1. （必填）读书计划
2. 本周完成的内容章节：

（1）2.1到2.5完成并且梳理知识点，寻找问题，自己思考并且于小组会之前完成了自己的思考。

1. 西瓜书和李航的统计学习并未发现任何相关联章节。黑书的算法写的有点晦涩难懂，但是多读几遍，结合后面的例子仍然可以理解，同时参考了csdn上的文章，链接如下：

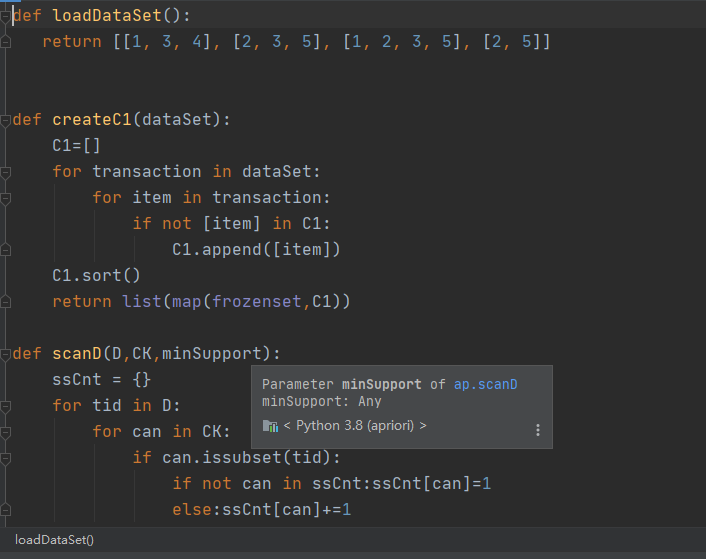
https://blog.csdn.net/qq\_36523839/article/details/82191677

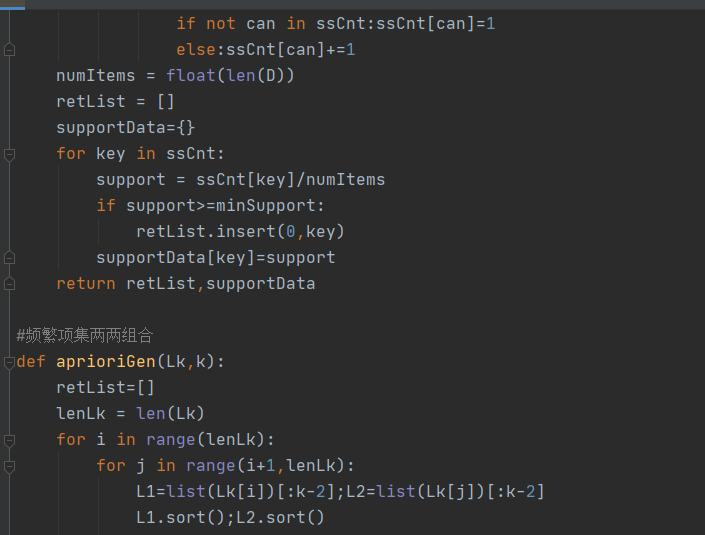
2、下周计划：2.6-2.10的阅读.

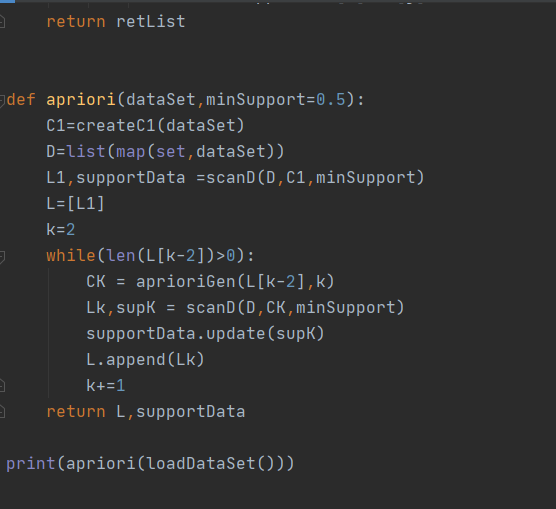
四、（选做）读书摘要及理解或伪代码的具体实现（读书摘要、伪代码的具体实现代码等可以写到这个部分）

1.代码实现

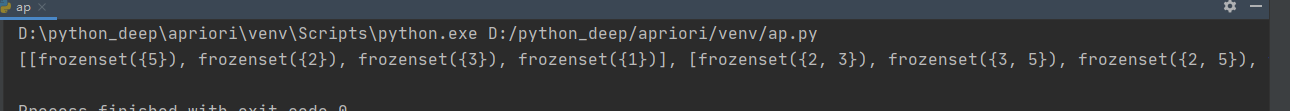
Ap算法的代码实现找到频繁集：







输出结果：

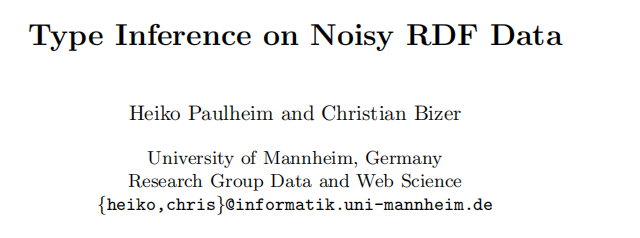


知识图谱部分：

#### 知乎专栏

已提交，并且对应已经发的专栏进行了排版和修改，后续如果有问题请漆老师提出，马上修改

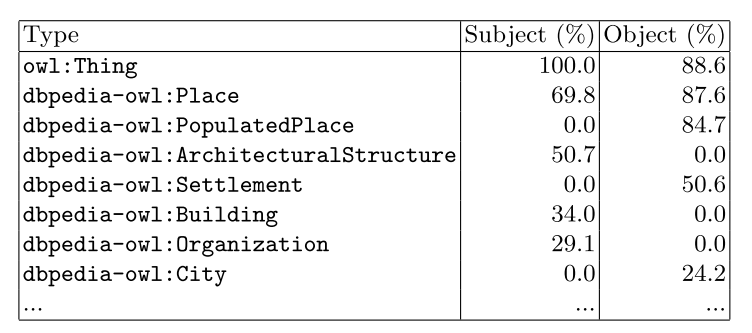
#### 百科领域图谱构建



阅读了上述论文

总结：RDF知识库由A-box(实例的定义以及它们之间的关系)与T-box(本体论)构成。SDType算法利用了实体之间的链接，通过加权投票的方式进行类型推断。其基本思路是如果某种关系仅仅和特定的类型一起出现，那么如果某个实例通过该关系连接到另一个实例，则可以启发式地判断该实例属于与该关系相关联的特定类型。比如，给出的例子是对于一个形如:x dbpedia-owl:location :y的声明，与关系dbpedia-owl:location相关联的宾语可以直观上推断:y 是一个location。

SDType的思路就是使用资源之间的链接作为类型推断的依据，一般这种链接有资源指出的(资源作为主语)或者指向资源的(资源作为宾语)。具体来说，对于每一种链接，算法会计算其两边的主语与宾语所对应类型的数据分布，来作为接下来推断实例类型的依据。例如，给出的dbpedia-owl: location属性在DBPedia中的247,601个三元组中出现，其主语和宾语位置的类型分布如下：



也就是说，如果观察到诸如:x dbpedia-owl:location :y形式的三元组时，以Place属性为例，则可以判断P(?x a dbpedia-owl:Place) = 0.698，P(?y a dbpedia-owl:Place) = 0.876，说明三元组中和location相关的所有三元组中，类型为Place的主语占比69.8%，87.6的比例是宾语。

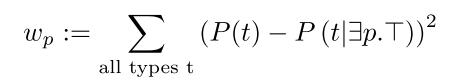
进一步来说，SDType的基础是一个条件概率：

T代表type，p代表property，即上文提到的关系，r代表的是实例(resource)。这个公式的意义是，如果一个resource存在属性(incoming property或outgoing property)p的话，那么其类型是 T的概率有多大。而对于每一个属性，其会被赋予一个权值wp，用来表征其预测类型的能力。公式可以进一步转化如下：

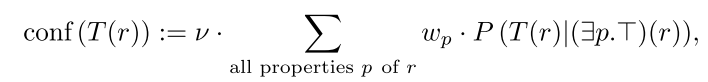


其中N代表的是一个resource链接到其他resource的所有链接数，这里的作用是平均。这样处理后，因为噪声数据本身数量很少，所以平均后的结果置信度很低，噪声数据的影响会大大减小。

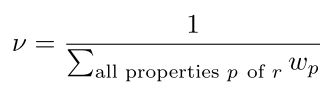
SDType使用了数据的分布进行推断。而使用数据的分布而非预先设定的域和范围来进行类型推断时，如果数据库严重倾斜，也就是某些类的数量规模是其他类规模的好几个数量级时，会出现明显的问题，如遇到rdfs:label 和owl:sameAs这样的常用关系时，那些更加频繁的类型的概率将会被大大夸大。举例来说，一个国家可能一般包含10000个城市，而它们都具有这些常见关系，则推断出城市的概率就会大得多(数量大)，尽管这样的关系存在于所有实例中。为了处理这个问题，给每种property设定一个权值wp，测量这种性质与所有类型的先验分布之间的偏差，来降低数量级的影响(如果该属性的预测能力不强，那么差值会很小)。



而公式则可以进一步转化如下：



v被定义为所有权值和的倒数：

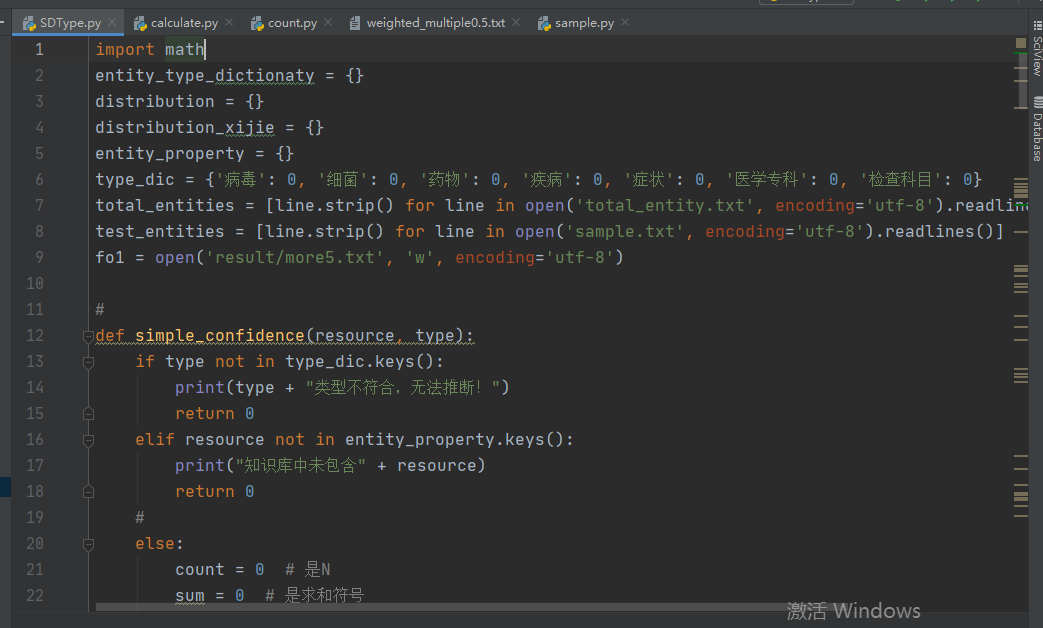


这些权重表现出了一个属性分辨类型的能力。比如说，可以观察到最大的权重一般赋予那些仅仅和某个类型一起出现的property，其可以理解为该类型的一种特征。比如在DBPedia中，dbpedia-owl:maximumBoatLength仅仅和dbpedia-owl:Canal关联。而对于诸如name这样的property来说，因为其非常普遍，权值会非常低，。

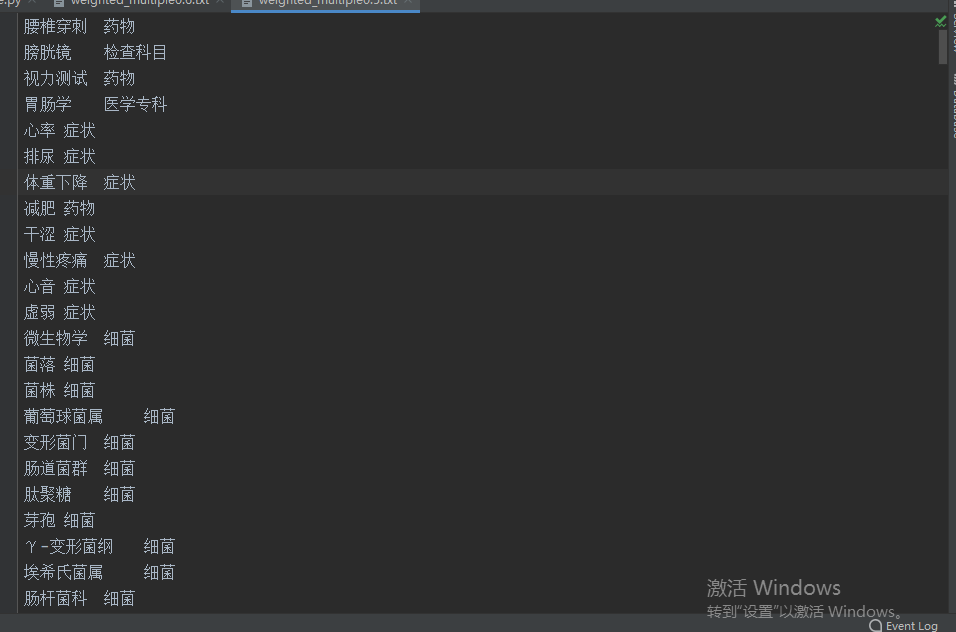
实验表明，权值上的优化相比直接平均有着显著的优势。一般选择0.4-0.6之间的值作为置信度的阈值可以得到0.95-0.99的准确率。而在DBpedia与OpenCyc数据集上，F值可以达到88.9%和63.7%，说明其有着较好的通用性。

代码实现：

基于新冠域的：



输出结果：



基于geo域的：



输出结果：



存在问题：

1. 大文件处理算法较慢，因参考的论文算法本身需要将所有数据读入内存再进行计算分布。
2. 头条百科数据较为准确，百度百科存在大量脏数据，需要对获得的数据进行数据处理，在头条百科中获得准确率较为合理。

解决办法：

1. 后续进行数据处理，解决乱码问题，本质上较少的乱码可以作脏数据直接忽略，在加权投票的时候并不会起任何作用，并不影响算法运行。
2. 这就去连服务器跑。