**论文复现：**Random Sample over Join Revisit

姓名：杨开来

学号：1170300709

**一、复现概述**

1. 目的

进一步理解抽样过程

学习和掌握一种大数据抽样方法

1. 内容

复现论文：Random Sample over Join Revisit， SIGMOD 2018。

**二、对实验步骤的详细阐述**

1. 论文选择：Random Sampling over Joins Revisited
2. 任务定义：自然连接是数据库操作中很重要的一种操作。自然连接是关系R和S在所有公共属性上的等接。但在得到的结果中公共属性只保留一次，其余删除。这篇论文关注数据库中多个表的自然连接问题。在数据库应用中，很多时候需要对多个表进行自然连接，得到结果后再对其某些性质进行研究。然而在大数据环境下，这种操作时间复杂性可能是无法接受的。因此需要研究合理的对该整体结果进行均匀抽样的方法。在原论文中，作者关注了chain join，star join和acyclic join三种join方式，在本实验中由于时间关系，我们只关注chain join的均匀抽样方法。
3. 方法描述：

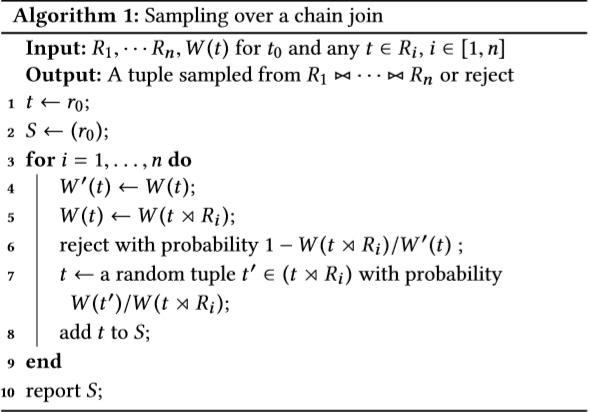
对于自然连接的抽样，基本思路是对于连接中每一个表的随机抽样，得出一个对于连接规模上界的估计，然后利用这个上界替代实际的连接规模对该表进行一个抽样。对于连接规模，我们给出如下精确定义：



其中t∈Ri

W即为该值的上界估计。显然，为了使抽样过程更加高效，这个上界应该尽量小。在对Olken et al.[1]和Chaudhuri et al.[2]进行研究后，作者提出了新的连接抽样框架。

该框架主要包含两部分，第一部分是一个抽样的框架。该框架接收一个表的列表和这些列表的每一个元素对应的连接规模上界的估计值，在自然连接上进行均匀抽样。重要的算法流程如下：



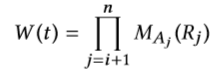
可以看到，在该算法中某一次的sampling有一定的几率reject。在每一个表上进行迭代时都可能reject，具体的概率也已经在算法中给出。根据论文中的证明，这个算法以1 / W(r0)的概率返回整体join中的任意一个。其中W(r0)是整体join size的一个上界。

另一部分是对于W上界的计算。作者对[1]和[2]中的方法进行了泛化，此后又提出了另一种基于random walk的方法对W的值进行估计和计算。值得注意的是，提出的三种方法中仅仅是计算W的方法不一样，进行抽样均使用上面提到的方法。下面对这三种方法进行阐述：

第一种方法对[1]中的方法进行了泛化和改进。论文中将其命名为Extended Olken (EO)方法。作者先把Olken el al.在量个列表自然连接中的上界估计方法推广到多个列表的连接上。作者定义了某一个表上的主键最大频率如下：

M Ai (Ri) = max v d Ai (v,Ri)

即获得某一个列表主键上最大频率主键的频率。此后，利用如下公式估计W：



其中t∈Ri。

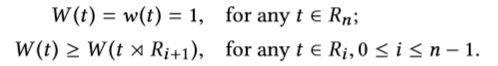
作者还引入了AGM Bound[3]。在chain join的情况下，作者给出了AGM Bound的简化形式如下：



这两种方法在不同的情况下的优劣程度也不一样。如果某个表的最大频率很小，那么Olken的方法得到的上界可能更小一些。而当表的最大频率很大时（比如接近表本身的大小），Olken方法得到的上界可能很大。因此此时选用AGM Bound可以使得整体的上界更小。基于这样的观察，作者提出整合两种方法的混合方法。

该混合方法通过一个阈值h统计每个表R上频率大于h的元素和小于h的元素，并将该表分为RH（heavy部分）和RL（light部分）两部分，并在计算中对这两部分使用不同的方法进行计算，得到整体较低的上界。对于RH倾向于使用AGM方法，对于RL部分倾向于使用Olken方法。

第二种方法对Chaudhuri el al.的方法进行了推广，把该方法从两个列表推广到多个列表。该方法被命名为Exact Weight(EW)方法。该方法的基本思想是对每一个列表中的每一个元素精确计算其值。为了降低时间复杂度，该方法使用动态规划来进行计算。该方法利用如下的限制：

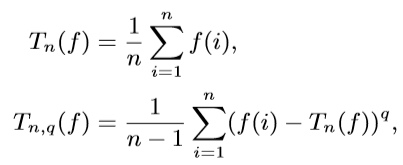


其中，将第二项的大于等于号改为等于号。对于join列表中的最后一个表，将其初始化为1。再从后向前计算每一个表中的结果。

第三种方法借鉴了Li et al.[4]的思想。该方法被命名为Online Exploration(OE)方法。该方法在该join列表上进行大量基于自然连接的随机游走（在每一个参与连接的表上，在下一个表中可以与当前元素进行连接的元素中均匀随机的选取一个），得到大量随机游走的结果序列，并在这个过程中保存得到每一个序列的概率。

此后对于每一个表上的每一个元素，设置一个阈值。其中在random walk中到达该元素达到该阈值次数的，使用wander join estimator对上界进行估计。对于未到达该阈值的，使用Exact Weight中的动态规划方法进行计算。

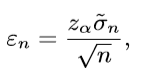
下面介绍wander join estimator。wander join estimator利用random walk的结果，对不同的数据库函数利用中心极限定理进行估计。具体的估计方式使用wander join estimator得到估计值的期望和方差后，给予一定的置信度，利用上(α+1)/2-分位数计算置信区间。此后，使用期望+置信区间/2作为最终的W估计值。对于具体的计算公式，由于W函数应为数据库函数中的COUNT函数，因此我们选用[4]中给出的COUNT函数的估计公式。给定如下的函数定义：



对于COUNT函数，我们有：



由以上公式得到期望和方差的值后，我们给定置信区间α，使用如下公式计算置信区间的大小：



其中zα是均值为0，方差为1的正态分布函数的(α+1)/2-分位数。

1. 实验

由于计算资源和时间的限制，我们在相对原论文中的数据集较小的数据集上进行试验。我们在相同的query上采样不同规模的的sample，并记录其运行时间，用来比较三种方法的时间效率。除此之外，我们比较三种方法在输出第一个采样结果时间的比较，以分析三种方法计算W值的效率。

根据原论文中的设置，我们设定如下三种查询的query：

1. Query T: popular user, twitter user,和twitter user的三角join query。
2. Query S: popular user, twitter user, twitter user, 和twitter user的四角join query。
3. Query F: 两个popular user和两个twitter user之间的snowflake join query。

**三、实验**

**1. 实验设置**

**实验环境**： Python 3.7

**数据库：**Sqlite3数据库引擎

**数据**：

之前尝试使用原论文中提供的数据集[[1]](#footnote-1)进行试验。经过统计，发现整个数据集共有36G，超过19亿条数据。且原论文中使用的popular user列表并未直接提供，需要进行预处理，从整体的列表中抽取popular user组成新的表。而这个过程以现有计算资源无法在有效时间内完成。除此之外，原论文中提到数据集是被整个加载到内存中进行实验的，然而对于现有内存资源，个人电脑上不可能将整个数据集加载到内存中。因此，我们选择相对适合当前实验的数据集予以替代。

通过寻找，我们找到斯坦福SNAP提供的推特用户关系数据集。该数据集[[2]](#footnote-2)由[5]提出，数据规模在百万级别，大小约为100MB.我们使用SNAP for python工具建立该数据集的关系图，并从中提取入度超过一定阈值的节点作为popular user，提取与这些节点有关的记录组成popular user表，用做实验。

**2. 实验结果**

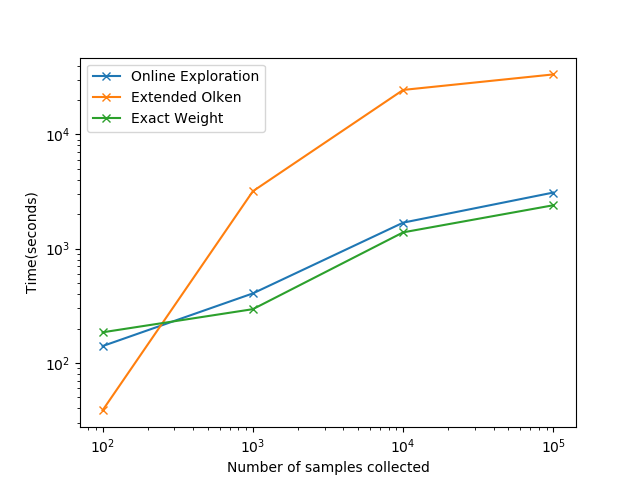
1. 针对三个不同的query我们分别从中抽取不同个数的sample，并比较其运行时间。在代码实现的过程中，解决了一些困难。

首先是对于EO方法，Olken bound需要获得当前表的最大频率主键的频率。在原论文中，作者提出在数据库大规模查询的过程中，最大主键频率一般是会被统计和保存的。然而在本次实验中，是直接将数据集中的数据加载到数据库中做各种操作，因此不可能获得这样的统计数据，因此，我们显式统计所有元素的频率，然后记录最大频率元素的频率。

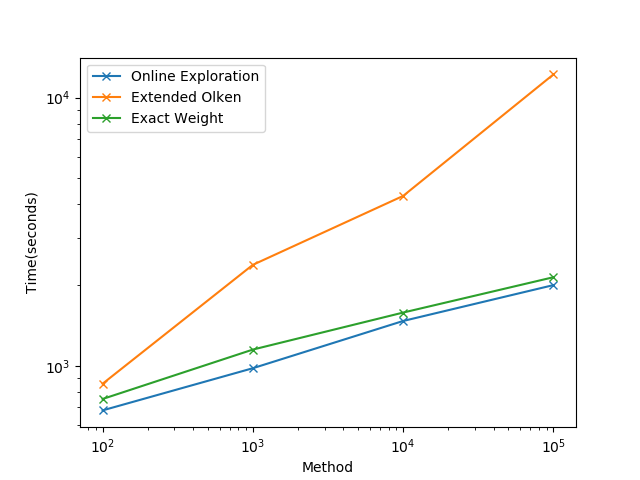
对于EO方法，需要将AGM bound 和 Olken bound结合使用。该思路的思想是当表的最大频率较大时（例如接近表的频率），Olken bound可能会过大，此时适合使用AGM bound。否则适合使用Olken bound。因此将每一个表利用一个阈值频率，将其分为Heavy和Light部分。然而论文中只给出了三个表的连接区分计划，并没有给出其它情况下的区分计划，因此，我们将除第一个表外所有的表区分为Heavy和Light部分，并分别给出AGM bound和Olken bound下的结果，取其中较小的结果，最后相加。

对于Online Exploration方法，我们设置random walk次数为2000000次，其中经过次数大于200次的利用estimator进行估计，小于200次的使用动态规划进行估计。对于EO方法，我们设置阈值为400进行heavy和light的区分。基于现有计算能力，我们分别抽取100，1000，10000，100000个sample进行实验。利用以上设置，分别使用三种不同的query进行测试，得到如下结果：

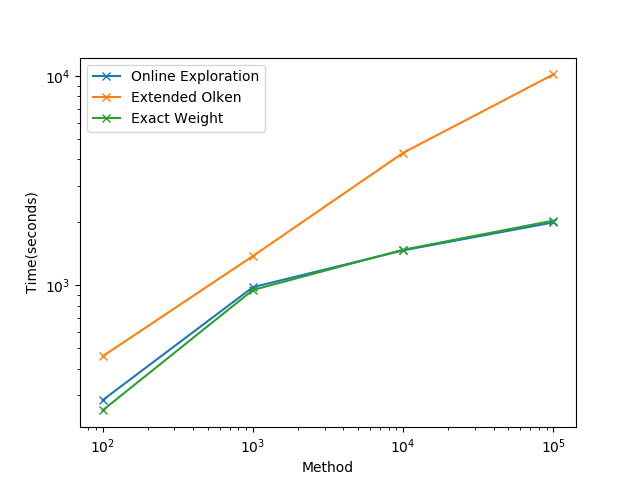
对于Query T，我们有如下运行时间结果：



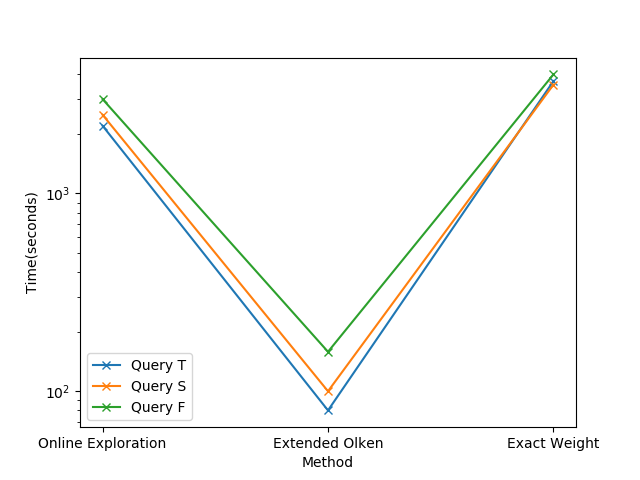
对于Query S，有如下结果：



对于Query F，有如下结果：



1. 我们通过统计三种不同的方法在三种不同的query中获得第一个sample所需时间来比较三种方法进行sample的准备所需的时间。统计结果如下：



**四、对实验结果的理解和分析**

1. 限于计算资源和时间，我们选用的抽取数量与原文中不完全相同。又由于选用不同数据集，因此无法直接比较。但根据论文中的实验结果和复现后的实验结果，运行时间的走势有一定相似。由于OE方法和EW方法都会在某种情况下使用动态规划算法，因此其W值相似度可能较高，因此运行时间也是相似的。可以看到，OE方法和EW方法在时间上随sample大小的波动相对较小。对于OE方法，尽管初始化时W的值不如EW那么精确，但随着时间的增长W的更新较快。而对于EW方法，尽管W的初始化时间较长，但由于其得到的W的值很精确，后期sample时间很快。对于EO方法，可以看到其时间复杂度随sample数增加很快。分析其原因，当sample数较少时，由于EO的W值计算速度相对较快，因此尽管该方法计算得到的bound不如EW精确，仍可以以相对较快的速度进行sample。然而当sample数量迅速增加时，由于W上界计算的不精确性导致sample的拒绝率较高，因此sample所需时间迅速上升。
2. 对于产生第一个sample所需时间的统计结果可以印证上述的推断。根据统计结果，三种方法所需的准备时间EO是最少的，OE方法次之，EW方法最高。对于OE方法，具体的准备时间与random walk次数有关系。由于EO方法在计算W值的过程中不需要额外的准备时间（而OE方法需要random walk过程，EW方法需要初始化最后一行的W值），因此时间最短。OE在random walk过程中需要进行下一个结点的选择和概率的记录，在所有random walk结束后才开始计算W值，因此所花时间较长。EW对每一个节点使用动态规划精确计算W值，因此所需时间最长。
3. 可以看到，尽管在相同的sample数量下实验取得和原论文中相同量级的结果，但实际上复现模型效果仍与原论文中的结果有很大差距，因为复现选用的数据集相比原论文中使用的数据集小了很多，因此复现模型的效果实际上比原模型差。究其原因，论文中有大量未交代清楚的实现细节，因此复现结果可能有所差距。本次复现使用Sqlite3数据库引擎，因此数据处理和查找效率与原论文中不同。模型在不同的处理器上运行，因此算力不同导致实验结果相对较差。

**五、实验过程中最值得说起的几个方面**

1. 较为准确地理解论文的思想，复现了论文提出的对于chain join的sample算法和三种计算w值上界（W值）的方法。

2. 进行试验，复现了论文有关Social graph数据集的相关实验。根据实验环境的实际情况，对原有的数据集进行更换，对原有的部分实验方案进行等价代替（例如频率的统计），使得实验得以正常进行。

3. 分析了实验中出现相关时间统计结果的可能原因。

**六、参考论文**

[1] F.Olken. 1993. Random Sampling from Databases. Ph.D. Dissertation. University of California at Berkeley.

[2] Surajit Chaudhuri, Rajeev Motwani, and Vivek Narasayya. 1999. On Random Sampling over Joins. In Proc. ACM SIGMOD International Conference on Management of Data.

[3] Albert Attsertia, Martin Grohe, and Daniel Marx. 2013. Size bounds and query plans for relational joins. SIAM J. Comput. 42, 4(2013), 1737–1767.

[4] Feifei Li, Bin Wu, Ke Yi, and Zhuoyue Zhao. 2016. Wander Join: Online Aggregation via Random Walks. In SIGMOD. 615–629.

[5] Julian McAuley, Jure Leskovec, Learning to Discover Social Circles in Ego Networks, 2012

1. 数据集地址：http://twitter.mpi-sws.org/data-icwsm2010.html [↑](#footnote-ref-1)
2. 数据集地址：http://snap.stanford.edu/data/ego-Twitter.html [↑](#footnote-ref-2)