估计大数据集的重复数据实际删除率

学生姓名：田浩翔

学　　号：2017282110197

专　　业：计算机技术

班　　级：5 班

目 录

一、概述

（一）简介

通过研究在特定数据集上，重复数据删除和压缩实现的数据缩减比率的准确估计问题，进而实现对于海量数据不进行全盘扫描的情况下，估计整个数据集的重复数据删除比例，提供可信结果，并尽可能给出可靠的范围保障。

通过对现实世界中的企业数据集进行测试，结合需要考虑的实际使用的障碍——内存消耗、采样等，改进算法与方案，使其能够顺利投入实践，最大程度达到误差最小化、收益最大化的效果。

（二）作用与意义

随着数据世界的持续爆炸性增长，对数据存储技术提出了越来越高的要求。面临着巨大的现实挑战，重复数据删除与压缩技术愈发普及，这些技术实际收益大小的问题，逐渐被更多地关注，很多情况下成为存储的最重要关键点之一。

用户需要重复数据删除的估计结果，为其存储规模、容量等决策提供可参考的价值信息。在备份技术发展多年之后，重复数据删除已经成为存储的关键技术之一。随着闪存的兴起，结合现代虚拟环境的普及及其高度重复性的特点，重复数据删除能给高端存储系统带来巨大的优势。

二、背景

（一）重复数据删除传统技术

传统的重复数据删除方法主要有两种：第一种是根据先验知识，给出有根据的猜测，并根据手中的工作量的信息，给出估计的结果；第二种是对手头的数据进行全面扫描，得到最终结果。第一种方法无疑是高度不准确的模糊界定，第二种方法对于海量数据集而言，意味着大量的内存和磁盘操作，耗费大，成本高。

一个简单的改进方法是简单地在整个数据集上运行数据缩减技术，但往往只是记录节省而不是存储减少的数据集。由此可见，重复数据删除估计地传统方案在现实世界中可行性均偏低，效果无法达到预期。

（二）基于的以往研究

1. 减少重复数据删除估计传统方案的直接成本

针对传统重复数据删除方案内存耗费大、磁盘 IO 成本高等缺陷，剔除了克服资源限制地最直接方法：对数据的一个子集进行采样，并详尽地计算采样数据的数据压缩率。然而，经过简单地研究，即发现这种方法必然会失败：如果不在数据的重复结构中包含特定的知识，就不可能仅通过查看随机子集准确地预测重复数据删除率的数据。另外，实验结果显示，这种改进后的方案，会产生不可控的估计值的偏差。

2.Unseen 算法——Valiant，2011 年

给出并证明了一个准确的方法，在只有一小部分数据的情况下，估计全局的重复数据。

用 S 表示一个数据集，并将其视为 N 个数据块（即 N 个哈希值）组成。 数据集由 D 个不同的块组成，第 i 个元素在数据集中出现 ni 次，即$\sum_{i=1}^{D} n_i = N$。定义数据集 S 的重复次数（率）直方图 DFH 是由 x = {x1，x2，...}组成的，xi 的值是在 S 中正好出现 i 次的不同块的数量，$\sum_i x_i \cdot i = N$。最终目标是找到 D 的估计值，或与其等效的比值$r = \frac{D}{N}$。

Unseen 算法的主要思想是找到值 x，使得预期的 DFHy'和观测到的 DFH y 之间的距离度量最小，使用线性规划进行最小化。

该算法实际的实践遇到了重大障碍：Unseen 算法的准确性证明在理论上是渐近的，不能转化为现实世界的具体数字；此外，它们的返回值不会对数据集提供任何保证。在现实世界中的数据集可行性差、实际效果不佳。

三、研究工作

（一）研究重点

1.提供估计的可靠范围与最坏情况分析

根据简单的测试，重复数据采样率越低的样本，估计值与实际值的误差越大。经过对样本的分析，发现问题是出在关于数据结构的环节，这无疑是对于算法实际应用效果的极大限制。需要认识到：只有一小部分数据集，特别是对数据集中的部分数据进行抽样，无论是随机的还是根据各种抽样方法，都可能产生任意不准确的重复数据删除率。这个潜在的事实得到了分析证明的支持，需要设计一个测量算法精度的方法。

2.减少算法的内存消耗

在实际系统中，能够以小的内存占用和无需额外的磁盘 IO 来执行估计是非常需要的，在某些情况下也是必须的；当在分布式系统中执行估计时，低内存使用也有利于较低的通信带宽。

3.抽样技术的实施

抽样是算法在现实世界中实施的最大障碍之一，完全的随机抽样，读取性能会大幅下降，特别是存储介质是基于硬盘的。同时，改进后的方法，需要对于闪存等支持快速随机读取的存储介质也能适用，至少性能不会出现下降；更重要的是，最终操作后的速度要比运行全面扫描快得多，否则是没有增益的，对于算法的实际执行而言，一般情况下，扫描时间主导运行时间，解决线性程序的时间普遍可以忽略不计。抽样运行速度需要明显快于全盘扫描速度，这是时间收益的主要来源。

4.与压缩结合

压缩的过程中，实际上也起到了一定“重复数据删除”的效果，但程度不明显，且想要很好的控制，可行性不高。但在算法的执行方案中，匹配各种存储介质，相应地与适合的压缩技术相结合，对算法执行的实际效果，和最终结果的准确度，均能起到一定的正面作用。

（二）核心算法

1．Range—Unseen 算法

为了解决 Unseen 算法存在的问题，设计了一种新的方法，返回一系列可信的重复数据消除比率，而不是一个单一的估计数。不只针对被观察的样本数据集有“最佳解释”DFH，而是测试所有对于 y 的“合理解释”DFH，并确定这些 DFH 中可能的范围：Range-Unseen。

Range—Unseen 算法是 Unseen 算法的高阶呈现，算法的输入是样本 Sp 的 DFH，用 y 表示；从 y 中得到整个数据集 S 的 DFH 估计值，作为样品观察到 DFH y 的“最佳解释”。在技术上，定义观察到的样本 DFH y 的所有合理解，找到最小的重复数据删除率和最大重复数据删除率。实际上在第一个初始优化中增加了两个额外的线性程序：第一个线性程序确定合理解决方案的邻域，第二个线性程序找到合理范围的两个限制；在线性程序中，用距离测量的优化替代不同块数量的优化。

算法的具体步骤如下：

（1）设定一个矩阵 Ap，可以通过多个二项概率的组合而成；

（2）假设一个 x’的值，之后通过y' = Ap • x'，得到其样本的 DFH y’，称之为期望的 DFH，通过 yi 和 y’i 计算 y 和 y’的 L1 范式距离 L(y,y’) = $\sum_i \frac{1}{\sqrt{y_i+1}}|y_i-(A_p\cdot x')_i|$ L 值越小，则 y 与 y’距离越近，也就是 x 与 x’越近似；

（3）求 x’的线性规划问题：约束条件是满足$\sum_i x'_i \cdot i = N$和$\forall i\, x'_i \geq 0$，即解释 x’是合法的 DFH。

最终得到一个 x’，它的样本 DFH 与观察值 DFH 距离最近，可以认为 x’就是所求的 x。

求解一个具有太多变量的线性规划问题是不切实际的，因此，不去求解一个完整的 DFHx，使用稀疏网格值（意味着 x 中不允许全部是重复值）。这种不准确性对高频计数影响很小，使线性规划的运行时间降低，与扫描时间相比基本上可忽略不计。

2.Chunk-Scan 预测算法

样本集为 S，共有 m 个数据项 e，对样本集中的每个数据项依次遍历：

（1）如果文件 Le 的长度匹配 Li，继续；否则忽略文件 e。

（2）在 e 的第一个块上计算散列以得到其哈希值 h1e，如果存在 i∈B，Le=Li 且 h1e=h1i，继续；否则忽略文件 e。

（3）计算文件 e 上的完整哈希得到 he，如果存在 i∈B，he= hi，则 $count_i$=$count_i$+1；否则忽略文件 e。

Chunk 阶段，需要考虑文件长度，因为文件大小差异很大（对于可变大小的块也是如此）；此外，数据集总大小不再按自然块来计算，而是根据存储文件长度的公分母，可以是文件系统的页面大小（如果文件存储为整个页面的集合），也可以是单个字节的最常见情况。用 N 表示数据集中的总字节数，每个这样的字节都有 m/ N 的独立概率被选择。对于基本样本，可能会多次选择相同的文件，所以需要记录此重复。在{1，...，N}中选择 m 个偏移量，如果文件包含选定的偏移量，则选择文件作为基本样本的一部分；基本计数器是相应的二项式随机变量的结果，其中 0 表示它不在样本中）。

Scan 阶段，在完整文件的情况下，利用典型文件系统中容易获得的元数据，减少在扫描阶段从磁盘读取所有数据的需要。重点是需要处理与基本样本相关的

文件，它们的散列在基本样本中。通过查看文件长度，可以排除许多文件相关的可能性，因为如果在基本示例中具有相同长度的文件，文件只能存在，这与 Bloom 过滤器的作用类似，即只有在数据有可能相关时才查看数据。在许多文件系统中，第一个块在文件的 i 节点中，可以在元数据扫描期间快速读取，不需要额外的磁盘搜索。

3.Full-File 预测算法

在上一算法中，对于 Chunk 和 Scan 两个阶段而言，可变大小的组块是一个挑战，因为既不能确定文件中有多少块、也不知道第 j 块的偏移量是多少，无法读取和分块整个文件。

Full-File 算法对其进行了改进。采样时设定精确的偏移量，然后选择包含该偏移量的块。假设在一个文件中选择了偏移量 k，那么可以通过读取偏移量为 k-maxchunk 的文件，并从那里分块直到找到包含相关偏移量的块。这既减轻了读取整个文件所需的消耗，也达到了给予每个块的可能性与块的实际长度成线性关系的所需效果。

（三）创新点

1.改进算法返回的不确定输出

改进的出发点来自于对同一数据集的多次实验，返回的结果竟然均不一致。针对这个问题，算法需要对样本中所有不同的块进行映射和计数，对于海量数据集而言，这个数字可能非常大。

由此引入松弛参数，不再仅返回一个估计值，而是输出预期实际结果的范围。一个小的松弛参数将形成更严格的估计范围，松弛参数的选择是启发式的，并且适合于所需的置信水平。使用该值估计期望值和标准差，并给出合理 DFH 范围的界限。进而进行了形式化的证明和实证检验，最终形成了一个近似线性的下界，用于估计一组中不同元素的数量（近似线性）。这在许多方面是实用的，并且具体允许逐步执行：首先抽取小样本并评估其结果；如果范围太大，则继续增加样本大小。

2.结合重复数据删除和压缩

在压缩的过程中，实际上也起到了一定的重复数据删除效果，只是它不易控制，并且不可逆。部署重复数据删除的许多系统，都使用压缩来补充数据简化。

估计二者结合的效益的，种自然方法是分别估计每一种，并将两者的比例相乘；但这在某些情况下是不正确的，因为重复数据删除的有效性和压缩有效性是独立的。作者提出了一种将压缩集成到算法中的方法，并显示产生的准确结果。

基本原理是用压缩加权 DFH 代替 DFH，多次实验证明，平均压缩率可以用一个小的随机样本来进行很好的估计。

与压缩不同，为了看到来自重复数据删除的数据缩减的效果，必须在样本中包含重复项目的多个副本。如果没有复制位置的先验知识，就有两次击中相同元素的可能性很小，并且击中相同元素的 k 个副本的概率极低（除非样本非常大）。

重复数据删除和压缩技术空间有效估计的一般框架。包括两个阶段：

（1） 样本阶段：从整个数据集中获取元素的基本样本。随机抽取样本，其中每个元素以相对于元素大小的概率独立出现（即固定块大小相同的概率，但对于文件大不相同）计算样本中每个元素的哈希值和压缩率。

（2） 扫描阶段：浏览整个数据集，仅存储基本样本中元素的统计信息。计算每个元素的散列值，但只有在匹配基本样本元素的散列值时才被记录。

在扫描阶段记录的统计数据被用来导出数据缩减率估计。

3.支持顺序读取的“随机”抽样策略

在磁盘中读取数据时，顺序读取相比随机读取，具有很大的时间与性能优势。从这个技术关键点出发，设计一个可以在磁盘中顺序读取，又能起到随机效果的抽样策略。

于是，在研究工作中，提出了一个适应算法实际执行的框架：在每个文件的第一个数据块上使用文件长度和散列值（通常这是文件 inode 的一部分），可以避免读取实际上几乎所有与基本样本不相关的数据。

遍历系统中的所有区块 id，该 id 必须是唯一的标识符，如卷名和区块偏移量，在区块 id 上计算快速哈希函数；然后使用此哈希值确定该块是否在当前示例中。这种简单的技术很容易并行，实际上对枚举顺序没有限制。在每个磁盘中，它几乎是无状态的，只需要记住当前索引 j 和输入参数，主循环可以非常快速。测试表明，迭代和散列的开销是可以忽略的（小于采样数据集 1%时间的 0.5%），这足以用于所有实际的手段。这样，实际操作可以顺序执行，且能起到随机效果，采样结果的随机性并未有明显的削弱。

（四）研究对比

1.横向对比

Chunk-Scan 算法和 Full-File 算法，均采用两阶段式框架；是从现实操作技术的角度出发，对已有技术进行组合和改进，设计能够将技术组合实际运用有效的算法，进而生成最终的预测算法。算法的实际可行性与置信度，已经在现实世界中得到过大量的实例证明，且具备一定的经验积累与错误预警和诊断。

相比之下，Range—Unseen 算法是对于一个在现实世界中难以实际执行的算法进行了修改，通过实验对实际操作的每一个阶段进行设计与改进，最终使得算法能够运用到实践中。

2.纵向对比

以往的研究工作中，研究重点在于重复数据删除，压缩及其组合对现实世界场景的影响，尽管获得了相当多的结果，并且可以从各种工作负载的不同技术得到一些经验法则的估计，但实际上，这远远没有对特定工作负载和特定用户给出准确的估计，很多时候同一种方案、不同次实验给出的同一数据集的重复数据删除估计值，甚至是变化的；这些变化来源于很多因素，包括后备制度、不同的设置，以及用户的行为和倾向，不能得到一个精确的估计。

新的研究工作设计了一个总体框架，提供一个有效的、可证明准确的估计方法。这个框架是一般性的，并不受限于某种特定的重复数据删除技术或其他。对这个框架进行了深入的实证研究，基于四种不同的工作负载：个人工作站，企业文件系统存储库和两种类型的备份数据；最大的数据集包含大约 7TB 的数据。可见，在实际操作与实施方面，相比之前的研究，算法投入现实世界的实践应用层面，做出了巨大的突破。实证研究表明，只有一部分数据被检查在实践中获得的准确性非常接近于分析的范围。

四、准确度分析与实验测试

通过测试一些真实世界的工作负载来评估技术，数据集来自不同的环境，包括：个人工作站、企业的主要文件存储库、小型企业的备份存储、交换数据库的定期备份等；每个工作负载运行 1000 次独立测试，以捕捉不同的数据行为，测试算法在所有这些情况下的工作情况和反馈，并评估涉及真实世界分布时的精度。算法随着样本大小的增长而运行，以便随着样本大小的增长来查看行为。图表可

以清楚地显示在所有工作负载中呈现的明显趋势。

1. Range-Unseen 算法

15%的样本已经足以收敛到一个稳定值，并给出小幅度波动范围的上下界，可以对所有工作负载进行良好的估计，现实生活中，可以在达到足够严格的估计时提早停止。测试结果如图 1 所示。

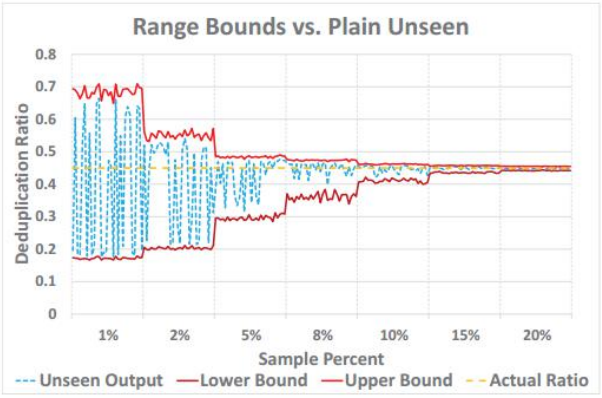

图 1　Range—Unseen 测试结果

2. Chunk-Scan 算法

针对文件存储库的工作负载，图 2 可以看到算法一个不错的收敛行为；且随着样本容量的增加，误差远低于 1%，可见大部分测试产生了很好的估计。

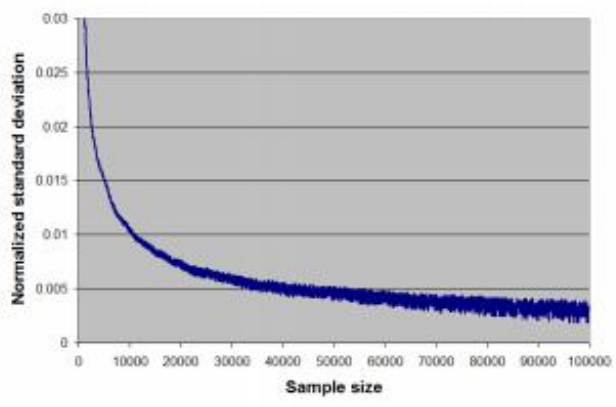

图 2 Chunk-Scan 算法测试结果

但与此同时，同样可以清晰发现，存在的一个隐患是，采样读取的数据量较大，才能得到合理的收敛估计；随着采样率的减少，置信度降低，不足以放心采

用，错误隐患较大。

3.Full-File 算法

可以减少从数据集中读取大部分数据，只读取与基本样本相关的文件（如果在基本样本中具有文件的长度，则读取第一个块，并且如果第一个块与具有相同长度的基本样本的第一个块相同，则读取整个文件）。但这意味着工作量的特征对文件读取量有很大的影响，如果基本样本中的文件具有高重复数据删除率，则会导致读取更多文件。

五、实践改进

1.抽样粒度

以较小的粒度（例如 4KB，与绝大多数存储系统底层页面大小匹配）进行简单的采样，在基于 HDD 的系统中是非常昂贵的（在某些情况下，少至 2%的采样可能已经超过全扫描）。测试了完全扫描时，以不同块大小采样不同百分比所花费的时间，如图 3 所示。

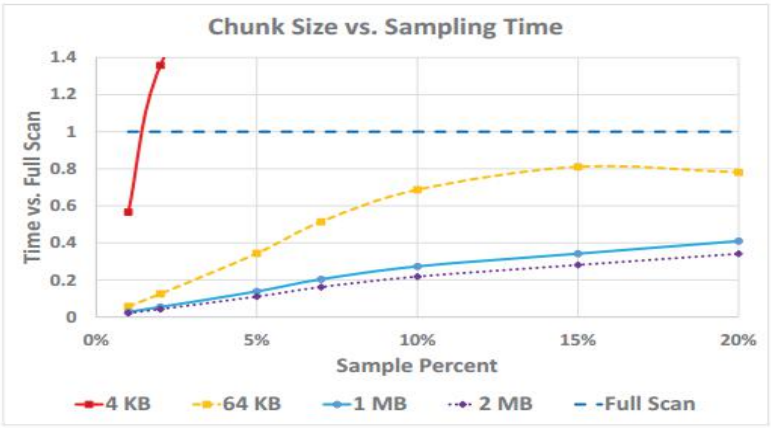

图 3 不同块大小采样不同百分比时耗

从图中可以看出，1MB 的粒度是临界点：1MB 前，随着粒度大小的增加，时间耗费减少显著；大于 1MB，随着粒度大小的增加，时间耗费虽继续减少，但减少幅度并不显著，相比于伴随增加的磁盘操作与 IO 成本，1MB 的粒度最为合适。

2.系统实施问题

（1）维护基本样本

在扫描阶段，有很多方法来保存基本样本。内存空间方面，最经济的方法是，

根据样本阶段结束时的散列值，对基本样本进行排序。由于扫描阶段无需插入，因此更新基本样本中的计数器只需要在已排序的数组上执行查找即可。

另一种方法是使用更复杂的哈希结构，处理好内存利用率，获得更快的平均查找时间。但将需要更多的内存，此外，在处理数据结构方面还有一些开销。

算法只在扫描阶段执行查找，而不是插入到和删除基本样本，这样减少了维护数据结构相关操作的运行时间。

（2）并行执行

扫描阶段可以在分布式系统上并行运行。基本样本需要循环到所有扫描节点，每个节点在本地执行扫描，并计数与该节点相邻的数据累计。此过程结束时，所有计数都被集中累加，并计算出数据压缩率，这很适合并行计算的 Map Reduce 框架。请注意，由于无法在每个节点处保留完整的基本样本，因此并行运行在 k 个节点的过程，需要同时保持 k 个基本样本的副本。

六、结论及存在不足

1.Range-Unseen 算法

基于已有的算法研究，提高了 Unseen 算法的可行性，改进了算法输出结果的置信度；将新技术引入重复数据删除领域，并进行了大量的现实世界中的实验与实践，证明了新技术的稳定性。

虽然采用现实世界的企业数据集、经过了多次的实践检验，但在结果的置信度和可靠性上，最终也未能给出理论上的保障，只是实际结果的归纳总结。虽没有出现反例，但终究存在理论上的隐患。

2.Chunk-Scan 与 Full-File 算法

根据现有的可行技术，给出了一个有效方法，能够随着存储系统规模的不断扩大，在更大的范围，给出大型存储库的重复数据删除的估计结果。

缺陷是算法局限于已有技术，没能彻底突破旧有瓶颈；未将重点集中在算法本身，创新性存在一定不足。此外，操作的实际执行中，所有文件的元数据都将被扫描，但实际数据只需要读取一小部分与基本样本相关的文件。