

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 7/00 (2006.01)



[12] 发明专利申请公布说明书

[21] 申请号 200780031842.1

[43] 公开日 2009 年 9 月 2 日

[11] 公开号 CN 101523340A

[22] 申请日 2007.6.27

[21] 申请号 200780031842.1

[30] 优先权

[32] 2006. 6. 27 [33] US [31] 60/805,926

[32] 2007. 6. 24 [33] US [31] 11/767,521

[86] 国际申请 PCT/US2007/072187 2007.6.27

[87] 国际公布 WO2008/002957 英 2008.1.3

[85] 进入国家阶段日期 2009.2.26

[71] 申请人 那哈瓦有限公司

地址 美国加利福尼亚州

[72] 发明人 中野利夫 斯坦利·郑

[74] 专利代理机构 北京安信方达知识产权代理有限公司

代理人 颜 涛 郑 霞

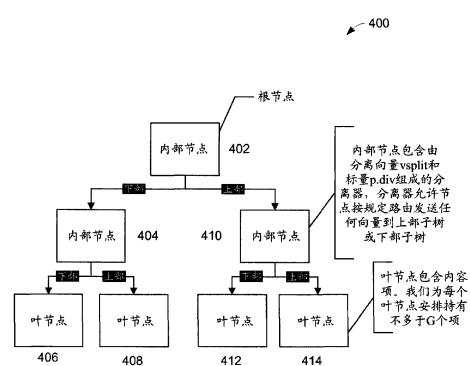
权利要求书 10 页 说明书 23 页 附图 8 页

[54] 发明名称

对大规模高维数据集进行快速的基于相似性的查询、自连接以及连接的方法和设备

[57] 摘要

一种利用相似性索引(400)对大规模高维数据集进行快速的、基于相似性的查询、自连接和连接的方法和设备。



1. 一种计算机实现的方法，其包括：

(a) 指定叶节点的上限为 G 项；

(b) 输入 n 个输入内容项并指示所述 n 个输入项为当前集合；

(c) 确定是否 $n > G$ ；以及

(d) 如果不是，那么

(d1) 建立叶节点；

(d2) 用所述 n 个输入内容项填充所述叶节点；以及

(d3) 接通从父辈到所述叶节点的链接，所述叶节点能够被储存在所述计算机的硬件里并能够显示给用户；

(e) 如果是，那么

(e1) 对所述当前集合的所有项计算向量和，其中所述向量和是 $vsplit = \sum(i; x.i)/n$ ；

(e2) 为在所述当前集合里的每一项计算向量差，其中所述向量差是 $d.i = x.i - vsplit$ ；

(e3) 为在所述当前集合里的每一项计算标量值，其中所述标量值是 $p.i = \langle d.i, vsplit \rangle$ ，并构成为每一项所计算出的所述标量值的集合；

(e4) 确定是否 $p.i < 3$ ；以及

(f) 如果不是，那么

(f1) 从所述集合移除最大的 p.i

(f2) 从所述集合移除最小的 p.i；以及

(f3) 在 (e3) 重新开始；

(g) 如果是，那么

(g1) 确定 1 个还是 2 个计算的值剩余在所述集合里; 以及

(h) 如果是 1 个, 那么

(h1) 让 p. split 成为所述 1 个计算的剩余值; 以及

(h2) 在 (j) 重新开始;

(i) 如果是 2 个, 那么

(i1) 让 p. split 成为所述 2 个计算的剩余值的平均值; 以及

(i2) 在 (j) 重新开始;

(j) 定义由所述 vsplit 和所述 p.split 组成的分离器;

(k) 对于所述当前集合里的所述内容项的每一个, 如果 p.i > p. split, 则将其指示为“上部”名称, 否则为“下部”名称;

(l) 建立由所述分离器组成的内部节点, 并定义到所述“下部”节点和所述“上部”节点的链接;

(m) 将所述“下部”节点作为项输入到新的“下部”当前集合, 让新的“下部” n 指示在所述新的“下部”当前集合中的项的数量, 用所述新的“下部”当前集合代替所述当前集合, 用所述新的“下部” n 代替所述 n, 并在 (c) 重新开始;

(n) 将所述“上部”节点作为项输入到新的“上部”当前集合, 让新的“上部” n 指示在所述新的“上部”当前集合中的项的数量, 用所述新的“上部”当前集合代替所述当前集合, 用所述新的“上部” n 代替所述 n, 并在 (c) 重新开始。

2. 一种计算机实现的方法, 其包括:

(a) 输入数据集, 所述数据集已经建立相似性索引;

(b) 输入查询 q;

(c) 输入期望的相似性阈值 s;

(d) 将当前节点设置为所述相似性索引的根;

(e) 确定所述当前节点是叶节点还是内部节点; 以及

(f) 如果是叶节点, 那么

(f1) 计算在所述叶节点中的每一项和 q 之间的相似性; 以
及

(f2) 返回满足所述期望的相似性阈值 s 的一个或更多的所
述每一项, 所述一个或更多的所述每一项能够储存在所述计算机的硬件里
并能够显示给用户;

(g) 如果是内部节点, 那么

(g1) 从所述内部节点获得分离器 (vsplit,p.split); 以及

(g2) 计算 $r = \langle q - vsplit, vsplit \rangle$;

(h) 确定是否 $r - p.split > 0$; 以及

(i) 如果为是, 那么

(i1) 将所述当前节点设置为“上部”子节点; 以及

(i2) 在 (e) 重新开始;

(j) 如果不是, 那么

(j1) 将所述当前节点设置为“下部”子节点; 以及

(j2) 在 (e) 重新开始。

3. 一种计算机实现的方法, 其包括:

(a) 指定最大集群大小为 G;

(b) 指定最小集群相似性为 s;

(c) 输入 n 个内容项;

(d) 为所述 n 个内容项建立大量相似性索引, 产生具有叶节点的树,
且每个节点有 G 个或更少的内容项;

(e) 为每个叶节点计算项之间的两两相似性;

(f) 确定是否存在相似性超过 s 的至少一对项; 以及

(g) 如果不是, 那么

(g1) 定义“一次性”集群;

(g2) 将所述项放在所述“一次性”集群里; 以及

(g3) 在 (i) 重新开始;

(h) 如果是, 那么

(h1) 计算最高的两两相似性, 并指明一项为集群的“锚”;

(h2) 定义“好”集群;

(h3) 对于所述当前节点里的每个项, 如果它与所述“锚”的相似性超过 s , 则将其放置在所述“好”集群里, 否则指明任何剩余项为“残留”并分别收集所述剩余项;

(i) 确定是否已经处理所有的叶节点; 以及

(j) 如果不是, 那么

(j1) 在 (e) 重新开始;

(k) 如果是, 那么

(l) 确定在这个点是否已经收集任何“残留”; 以及

(m) 如果不是, 那么

(m1) 在所述计算机的硬件里存储任何“好”集群和任何“一次性”集群, 所存储的任何“好”集群和所存储的任何“一次性”集群能够显示给用户,

(n) 如果是, 那么

(n1) 聚集较早收集的所述“残留”; 以及

(n2) 在 (c) 重新开始。

4. 一种计算机实现的方法, 其包括:

- (a) 输入数据集 1 和数据集 2;
- (b) 输入相似性阈值 s^* ;
- (c) 为所述数据集 1 和所述数据集 2 的每一个输入“分组”大小 G_i ;
- (d) 基于数据集 1 和数据集 2 的大小计算树的预期高度, $h.i = \text{ceiling}(\log_2(N.i/G.i))$;
- (e) 使用相似性阈值 $s.i = 1 - [(1 - s^*)/(2 * h.i - 1)]$, 为数据集 1 和数据集 2 建立自连接树, 产生好群和一次性群的树;
- (f) 指明“顶端”集为根好群的锚连同所述一次性群的成员;
- (g) 从所述“顶端”集构成数据集;
- (h) 使用所述相似性阈值 s^* 在所述数据集上执行自连接;
- (i) 从所述数据集上的所述自连接输出具有来自数据集 1 的至少一个成员以及来自数据集 2 的至少一个成员的“好”群, 指明这些“好”群为 s^* -连接集群, 所述 s^* -连接集群能够储存在所述计算机的硬件里并能够显示给用户。

5. 如权利要求 4 所述的计算机实现的方法, 其中提前确定所述“分组”大小 $G.i$ 。

6. 如权利要求 4 所述的计算机实现的方法, 其中所述“分组”大小 $G.i$ 最小化了建立相似性索引的总体计算时间。

7. 一种基于硬件的设备, 其包括:

- (a) 用于指定叶节点的上限为 G 项的装置;
- (b) 用于输入 n 个输入内容项并指示所述 n 个输入项为当前集合的装置;
- (c) 用于确定是否 $n > G$ 的装置; 以及
 - (d) 如果不是, 那么
 - (d1) 用于建立叶节点的装置;
 - (d2) 用于用所述 n 个输入内容项填充所述叶节点的装置;

以及

(d3) 用于接通从父辈到所述叶节点的链接的装置，所述叶节点能够被储存在存储器里，且所述存储器能够显示给用户；

(e) 如果是，那么

(e1) 用于对所述当前集合里的所有项计算向量和的装置，其中所述向量和是 $vsplit = \sum(i; x.i)/n$ ；

(e2) 用于为在所述当前集合里的每一项计算向量差的装置，其中所述向量差是 $d.i = x.i - vsplit$ ；

(e3) 用于为在所述当前集合里的每一项计算标量值且构成成为每一项所计算出的标量值的集合的装置，其中所述标量值是 $p.i = <d.i, vsplit>$ ；

(e4) 确定是否 $p.i < 3$ ；以及

(f) 如果不是，那么

(f1) 用于从所述集合移除最大的 $p.i$ 的装置

(f2) 用于从所述集合移除最小的 $p.i$ 的装置；以及

(f3) 用于在 (e3) 重新开始的装置；

(g) 如果是，那么

(g1) 用于确定 1 个还是 2 个计算的值剩余在所述集合里的装置；以及

(h) 如果是 1 个，那么

(h1) 用于让 $p.split$ 成为所述 1 个计算的剩余值的装置；以及

(h2) 用于在 (j) 重新开始的装置；

(i) 如果是 2 个，那么

(i1) 用于让 $p.split$ 成为所述 2 个计算的剩余值的平均值的装置；以及

(i2) 用于在 (j) 重新开始的装置；

(j) 用于定义由所述 vsplit 和所述 p.split 组成的分离器的装置；

(k) 用于对于所述当前集合里的所述内容项的每一个，如果 $p.i > p.\text{split}$ 则将其指示为“上部”名称，否则为“下部”名称的装置；

(l) 用于建立由所述分离器组成的内部节点、并定义到所述“下部”节点和所述“上部”节点的链接的装置；

(m) 用于将所述“下部”节点作为项输入到新的“下部”当前集合、让新的“下部”n 指示在所述新的“下部”当前集合中的项的数量、用所述新的“下部”当前集合代替所述当前集合、用所述新的“下部”n 代替所述 n、并在 (c) 重新开始的装置；

(n) 用于将所述“上部”节点作为项输入到新的“上部”当前集合、让新的“上部”n 指示在所述新的“上部”当前集合中的项的数量、用所述新的“上部”当前集合代替所述当前集合、用所述新的“上部”n 代替所述 n、并在 (c) 重新开始的装置。

8. 一种基于硬件的设备，其包括：

(a) 用于输入数据集的装置，所述数据集已经建立相似性索引；

(b) 用于输入查询 q 的装置；

(c) 用于输入期望的相似性阈值 s 的装置；

(d) 用于将当前节点设置为所述相似性索引的根的装置；

(e) 用于确定所述当前节点是叶节点还是内部节点的装置；以及

(f) 如果是叶节点，那么

(f1) 用于计算在所述叶节点里的每一项和 q 之间的相似性的装置；以及

(f2) 用于返回满足所述期望的相似性阈值 s 的一个或更多的所述每一项的装置，所述一个或更多的所述每一项能够储存在存储器里且所述存储器能够显示给用户；

(g) 如果是内部节点，那么

(g1) 用于从所述内部节点获得分离器 (vsplit,p.split) 的装置；以及

(g2) 用于计算 $r = \langle q - vsplit, vsplit \rangle$ 的装置；

(h) 确定是否 $r - p. split > 0$ ；以及

(i) 如果是，那么

(i1) 用于将所述当前节点设置为“上部”子节点的装置；以及

(i2) 用于在 (e) 重新开始的装置；

(j) 如果不是，那么

(j1) 用于将所述当前节点设置为“下部”子节点的装置；以及

(j2) 用于在 (e) 重新开始的装置。

9. 一种基于硬件的设备，其包括：

(a) 用于指定最大集群大小为 G 的装置；

(b) 用于指定最小集群相似性为 s 的装置；

(c) 用于输入 n 个内容项的装置；

(d) 用于为所述 n 个内容项建立大量相似性索引、产生具有叶节点的树且每个节点有 G 个或更少内容项的装置；

(e) 用于为每个叶节点计算项之间两两相似性的装置；

(f) 用于确定是否存在相似性超过 s 的至少一对项的装置；以及

(g) 如果不是，那么

(g1) 用于定义“一次性”集群的装置；

(g2) 用于将所述项放在所述“一次性”集群里的装置；以及

(g3) 用于在 (i) 重新开始的装置；

(h) 如果是，那么

(h1) 用于计算最高两两相似性并指明一个项为集群的“锚”的装置；

(h2) 用于定义“好”集群装置；

(h3) 用于对于所述当前节点里的每一项，如果它与所述“锚”的相似性超过 s ，则将其放置在所述“好”集群里，否则指明任何剩余项为“残留”并分别收集所述剩余项的装置；

(i) 用于确定是否已经处理所有的叶节点的装置；以及

(j) 如果不是，那么

(j1) 用于在 (e) 重新开始的装置；

(k) 如果是，那么

(l) 用于确定在这个点是否已经收集任何“残留”的装置；

以及

(m) 如果不是，那么

(m1) 用于在存储器里存储任何“好”集群和任何“一次性”集群的装置，以及所述存储器能够显示给用户；

(n) 如果是，那么

(n1) 用于聚集较早收集的所述“残留”的装置；

以及

(n2) 用于在 (c) 重新开始的装置。

10. 一种基于硬件的设备，其包括：

(a) 用于输入数据集 1 和数据集 2 的装置；

(b) 用于输入相似性阈值 s^* 的装置；

(c) 用于为所述数据集 1 和所述数据集 2 的每一个输入“分组”大小 G_i 的装置；

-
- (d) 用于基于数据集 1 和数据集 2 的大小计算树的预期高度 $h.i = \text{ceiling}(\log_2(N.i/G.i))$ 的装置；
 - (e) 用于使用相似性阈值 $s.i = 1 - [(1 - s^*)/(2 * h.i - 1)]$ 为数据集 1 和数据集 2 建立自连接树、产生好群和一次性群的树的装置；
 - (f) 用于指明“顶端”集为根好群的锚连同所述一次性群的成员的装置；
 - (g) 用于从所述“顶端”集构成数据集的装置；
 - (h) 用于使用所述相似性阈值 s^* 在所述数据集上执行自连接的装置；
 - (i) 用于从所述数据集上的所述自连接输出具有来自数据集 1 的至少一个成员以及来自数据集 2 的至少一个成员的“好”群并指明这些“好”群为 s^* -连接集群的装置，所述 s^* -连接集群能够储存在存储器里，且所述存储器能够显示给用户。

11. 如权利要求 10 所述的基于硬件的设备，其中所述“分组”大小 $G.i$ 提前确定。

12. 如权利要求 10 所述的基于硬件的设备，其中所述“分组”大小最小化了建立相似性索引的总体计算时间。

对大规模高维数据集进行快速的基于相似性的查询、自连接以及连接的方法和设备

相关申请

本专利申请要求 2006 年 6 月 27 日提交的题为 “Method and Apparatus for fast Similarity-based query, self-join, and join for massive, high-dimension datasets” 的美国临时申请序号 60/805,926 的优先权，其与本申请是同一发明人，并因此在这里通过引用并入。本专利申请要求 2007 年 6 月 24 日提交的题为 “Method and Apparatus for fast Similarity-based query, self-join, and join for massive, high-dimension datasets” 的美国申请序号 11/767,521 的优先权，其与本申请是同一发明人，并因此在这里通过引用并入。

发明领域

本发明涉及数据集。更具体地，本发明涉及对大规模高维数据集进行快速的基于相似性的查询、自连接(self-join)以及连接的方法和设备。

发明背景

基于相似性 (Similarity) 的大规模数据集的连接是许多重要问题的核心。例如，在信息检索领域内的一个重要的问题是数据挖掘，数据挖掘寻找在例如文档、图像或其它非结构化(unstructured)的内容的项(item)的集合之间识别模式。一般地，有某种准则来衡量在数据成员之间的相似性，其可作为数学公式表达。一般地，我们有两个大规模数据集，并且我们想“连接”数据集以识别对(pair)或集群(cluster)，其中每个数据集至少有一个成员与来自其他数据集的另一个成员相似。一个重要的特殊情形是“自连接”，识别在单个数据集内的副本、近副本或非常相似的项。一个重要应

用是内容可寻址存储和智能文件存储的出现区域，其中，或者相对于参考集合，或者相对于其自身，连接目标数据集，以识别副本和近副本。虽然计算机变得更迅速，存储变得更广泛，且内容变得更多样，但是对大规模数据集进行有效判断（effective sense）的能力并没有跟上。这就出现了问题。

附图简述

通过实施例的方式说明本发明，并不限于附图的图形，其中：

图 1 说明了其中可实现本发明的方法和设备的网络环境；

图 2 是其中可实现本发明的一些实施方式以及可使用本发明的一些实施方式的计算机系统的结构图；

图 3 说明了显示建立大量相似性索引（bulk similarity index）的本发明的一种实施方式；

图 4 说明了显示相似性索引树的实施例的本发明的一种实施方式；

图 5 说明了显示查询的本发明的一种实施方式；

图 6 说明了显示自连接的本发明的一种实施方式；

图 7 说明了本发明的一种实施方式，显示自连接树的实施例；以及

图 8 说明了显示常规连接（general join）的本发明的实施方式。

详细描述

介绍

为了描述本发明，下述内容可帮助读者。非结构化的内容项是信息的单元(unit)，例如文本、图像、音频、视频、基因组(genomic)序列或能由计算机存储器中的比特表示的任何实体。数据集是非结构化的内容项的集合。一般地，数据集里的项不是具有就是被认为具有某种关联。例如，它们可以是来自照片集合的静止图像，或在数据集里的项可以是来自法律文

档或来自小说的段落。作为附带说明，我们指出，想探究项是否有关联时，可在数据集里插入项，例如当我们组合来自各种各样不同物种的 DNA 序列的片段，或组合来自监视视频的帧时。

将非结构化的内容表示为向量

形式上，我们将内容项 x 指示为来自内积空间的元素，我们也将内积空间称为向量空间。此外，我们将有时将内容项称为来自这个向量空间的向量。作为快速回顾，内积空间 S 所具有的属性列在附录 1。我们使用在之前的申请【参考 R. Nakano 的日期为 2005 年 2 月的美国专利申请第 11/361166 号的“Method and apparatus for efficient indexed storage for unstructured content”】里的内积空间的属性。

内积空间具有距离函数 $D(x,y)$ ，其表示在空间里两个元素之间距离的实数。这对我们非常重要，因为我们感兴趣的是两个内容项是如何相似或相异的。例如，我们可能对两个网页的内部结构的相似性感兴趣，以最有效率地表示逻辑，从而从大集合的网页提取内容。或者我们想在企业政策报告的资料库里或法律条款的系统信息中心库(repository)里识别副本和近副本文本项。

在我们挑选的内积空间里，我们把内容项和它的具体表示区分开。例如，在文本应用中，我们选定把在特定序列的每个词分配给记号(token)的文本的表示。一种提取符号策略 (tokenization strategy) 是“断词(stem)”，以便将复数的词尾和其它不同词尾分配给相同的记号。或者，我们可决定数量映射到相同的记号，以便“3 bears” 和 “100 bears” 是等价的，也就是一个数量跟着是记号 “bear”。在向量表示方面，两个向量可能相等，但是所隐含的内容项可不同。另外的实施例衡量文档相似性，其包括由出现在文档内的关键词和每个关键词出现的数量来对文档进行表示。这是术语 (term) 文档向量。用这种表示形式，例如 “like an arrow” 和 “an arrow like”的文本序列包含每个关键词 “arrow” 和 “like”的一次出现。它们的向量是相等的，且因此在向量空间意义里它们是相等的。但是所隐含的内容项是不同的。我们引入这个特性，是因为我们使用数据集成员的向量空间表

示，但是我们允许内容项维持它们隐含的特征。事实上，我们可对单个项选择引入多种向量表示。

内积、距离、相似性

为了我们的目的，假定我们已经选定了表示形式，其将内容项映射到向量。明确的，给定内容项的集合，我们选定一种表示形式，其将每个项分配到在内积空间 S 里的向量 x 。给定 S 内的两个元素 x, y ，我们将内积记为 $\langle x, y \rangle$ 。附录 1 总结了内积空间的属性。

距离 (Distance) $d(x, y) = \sqrt{\langle x - y, x - y \rangle}$ 。

从这个定义，我们看出如果两个向量 x 和 y 相等，那么在它们之间的距离也是零。记住，我们允许两个向量相等，但是相应的内容项可以不同。

具有内积是方便的，因为它给我们提供了对数据集里两个项的相似性进行表示的方式。当我们具有其对应向量表示非零的两个内容项时，我们引入相似性的概念：

相似性 (Similarity) $(x, y) = \frac{\langle x, y \rangle}{\sqrt{\langle x, x \rangle * \langle y, y \rangle}}$ 。

相似性是有用的概念，因为相似性是在 0 和 1 之间的数字。当在它们之间的距离为零时，两个向量具有一的相似性或 100% 相似性。

相似性是吸引人的，因为它理解简单。给定数据目标的族，可以有表达相似性的数学方式，其容易掌握、计算直接，且最重要的是，与人们关于两个目标相似程度的直觉一致。

然而，在实践中，将相似性应用到大型数据集变得有问题。假设数据集包含 N 个项，且我们想在数据集里寻找相似的项。

当向量空间 S 的维数小时，例如，在一维的情形，向量空间减小到严格有序的集合，且我们能够在 $O(n * \log(n))$ 时间里对数据集分类，并简单计算在有序集里的邻近项的受限集的相似性，。

对于最感兴趣的非结构化的内容数据集，维度为高。例如，在术语文档表示里，向量空间的维度与可能的关键词的数量相等，其能容易排列到

数百或更多。除了简单关键词的方法，更复杂的表示可使用在文档内的词序列。在这些情况，向量空间的维数与邻近的词的不同的 n 元文法(n-gram)的数量成比例，并且维度扩展到好几万。

对于基因组序列数据集，一般考虑 16 连续核苷酸的 n 元文法，其中，在 n 元文法内的每个位置有 4 个可能的选择。一般地，如果数据集由长度为 k 的核苷酸序列组成，则向量空间的理论维数大概是 $k^4 \cdot 16$ ，或者 k 升高到四十亿幂。考虑到几百核苷酸的序列被认为是短的，我们正在处理巨大维数的问题。

定义

为了描述本发明的目的，术语的下述定义（在括号里）可帮助读者。

（一致的）我们说当它们的向量表示是向量空间 S 的成员，且存在空间 S 的内积时，项 $x.i$ 的集合和集 $x.i$ 是一致的。

例如，假设我们有两个数据集，都由静止图像组成。此外，我们已经选定在两个数据集上应用相同的变换，特定地为亮度、对比度、比例和方位来对像素值进行规范化，为每个图像产生高维度向量。我们想知道，比如说在一对图像里是否有相似的图像，或者是否有彼此相似图像的集群。通过这种构造(construction)，我们说数据集是一致的。

（ s -集群）我们说，当对于在集合里的任何项 x ，对集合里的项具有大于或等于 s 的相似性时，一致的项集合形成 s -集群。

（自连接）给定数据集 D，在 D 里识别 s -集群， s 是在 0 和 1 之间的相似性阈值。

（查询）给定项 q 、数据集 D 和在 0 和 1 之间的相似性阈值 s ，其中 q 与 D 一致。考虑 q 和 D 的联合(union)，在包括 q 的那个联合里识别任何 s -集群。

（ s -连接集群）给定在数据集 D.1 里的向量 $x.i$ 和在数据集 D.2 里的向量 $y.i$ ，其中 D.1 和 D.2 是一致的。 s -连接集群是至少一个成员来自 D.1 且

一个成员来自 D.2 的 s-集群。

(常规连接)给定在数据集 D.1 里的向量 $x.i$ 和在数据集 D.2 里的向量 $y.i$, 其中 D.1 和 D.1 是一致的。识别所有 s-连接集群。

实施方式

我们将以下面的顺序继续讨论本发明的不同实施方式。

首先，我们说明如何建立相似性索引。

第二、我们说明如何相对于相似性索引来执行基于相似性的查询。

第三，我们说明执行自连接的快速技术。因为我们方法的计算复杂性按 $O(n * \log(n))$ 增长, 所以认为该算法是快速的。作为对比, 强力(brute force)方法在考虑数据集里的所有可能时, 要求 $n * (n-1)/2$ 次相似性计算或 $O(n^2)$ 次运算。

第四, 我们说明在两个数据集上如何执行常规连接。这个程序使用自连接技术作为子程序, 并且它还具有 $O(n1 * \log(n1 + n2 * \log(n2)))$ 次运算的计算复杂性, 其中 $n1$ 和 $n2$ 是两个数据集的大小。我们注意到, 我们将要描述的程序比构造这两个数据集的联合要好, 其为在结合的数据集上执行自连接。我们证明用于常规连接的程序的正确性。

建立相似性索引

之前的描述集中在递增树建立的一般情形。【参考 R. Nakano 的日期为 2005 年 2 月的美国专利申请第 11/361166 号 “Method and apparatus for efficient indexed storage for unstructured content”】。这里为了我们的目的, 我们引入大量 (bulk) 建立二进制相似索引树的程序, 其比更早的递增程序更简单且更快速。数据集的初始加载是十分普通的操作, 它使对那种情况的优化有意义。

图 3 一般用 300 举例说明了显示建立大量相似性索引的本发明的一种实施方式。在 302 我们指定叶节点的上限为 G 项。在 302 我们输入内容项,

让 n 代表项的数量，并称所述项为当前集合。在 306 我们核对以查看是否 $n > G$ 。如果不是 $n > G$ ，那么我们进行 308，在 308 建立叶节点，用 n 个内容项将其填充，并从父辈(parent)接通链接，以及然后进行 310 完成。如果 $n > G$ 为真，那么我们进行 312，在 312，对当前集合里的所有项计算下述向量和： $vsplit = \sum(i; x.i)/n$ 。我们然后进行 314，在 314，我们为当前集合里的每一项计算向量差： $d.i = x.i - vsplit$ 。我们接下来进行 316，在 316，我们为当前集合里的每一项计算下述标量值： $p.i = \langle d.i, vsplit \rangle$ ，以及获得这些值的集合。接下来在 318，我们核对以查看 $p.i$ 的数量是否小于 3。如果 $p.i$ 的数量不小于 3，那么在 320 我们从集合移除最大和最小的 $p.i$ ，并然后返回 316。如果 $p.i$ 的数量小于 3，那么在 322，我们核对以查看我们是否具有剩余的 1 或 2 个值。如果我们具有 1 个剩余值，那么在 324，我们让 $p.split$ 成为最后的剩余值，并然后进行 328。如果我们具有 2 个剩余值，那么在 326，我们让 $p.split$ 成为最后两个剩余值的平均值，并然后进行 328。在 328，我们定义“分离器(splitter)”，其由较早(在 312 的 $vsplit$ ，和在 324 或 326 的 $p.split$)计算的($vsplit, p.split$)组成。在 330，对于在当前集合里的每个内容项，如果 $p.i > p.split$ ，那么指明其为“上部”，否则指明其为“下部”。接下来在 332，我们建立由来自之前步骤的分离器组成的内部节点，并定义到待被建立的“下部”和“上部”节点的链接。接下来取决于作为“下部”或“上部”分类，我们分别行进到 334 或 336。在 334，对于“下部”集合，我们称其为当前集合，我们让 n 作为项的数量，并且通过行进到 306 我们再次调用这个整个程序。在 336，对于“上部”集合，我们称其为当前集合，我们让 n 作为项的数量，并且通过行进到 306 我们再次调用这个整个程序。因此图 3 一般举例说明了显示大量建立相似性索引的方法的本发明的一种实施方式，其中输入是内容项的集合且输出是有项在叶节点的二进制树。

图 4 一般在 400 举例说明了显示相似性索引树的实施例的本发明的一种实施方式。每个内部节点(例如 402、404、410)持有一个分离器，其由“平均”向量和标量分离值(split value)组成。分离器的工作是捕获任何输入向量，并决定那个向量是否被引导到“下部的”或“上部的”子树。每个叶节点(例如 406、408、412、414)持有内容项向量。

查询

相似性索引树能被用来执行如下的查询：

- 1、我们给定查询向量 q 、相似性索引树和相似性阈值 s 。
- 2、设置当前节点作为索引树的根。
- 3、如果根节点是叶节点，那么计算在 q 和叶节点里每个项之间的相似性。返回具有大于 s 的相似性的项。完成。
- 4、否则，如果根节点是内部节点，则分离器由向量“ $vsplit$ ”和标量值“ $p.split$ ”组成。计算表达式，
 $r = \langle q - vsplit, vsplit \rangle$
- 5、计算 $\text{delta} = (r - p. split)$ 。
- 6、如果 delta 大于零，那么设置当前节点为上部子节点。否则，设置当前节点为下部子节点。
- 7、行进到步骤 3。

图 5 一般在 500 举例说明了显示查询的本发明的一种实施方式。在 502，我们给定已经建立相似性索引的数据集，给定查询 q 并给定期望的相似性阈值 s 。在 504，我们将当前节点设置到相似性索引的根。在 506，我们确定当前节点是叶节点还是内部节点。如果当前节点是叶节点，那么在 508 我们计算在 q 和在叶节点里的每个项之间的相似性，然后我们进行到 510，在 510 我们返回满足相似阈值准则 s 的项，并然后进行到 512，在 512 完成。如果当前节点是内部节点，那么在 514 我们从内部节点获得分离器($vsplit, p.split$)并计算 $r = \langle q - vsplit, vsplit \rangle$ 。接下来在 516，我们确定是否 $r - p.split > 0$ 。如果 $r - p.split$ 大于零，那么在 518 我们将当前节点设置为“上部”子节点，并进行到 506。如果 $r - p.split$ 不大于零，那么在 520，我们将当前节点设置为“下部”子节点并进行到 506。因此图 5 一般举例说明了显示查询的方法的本发明的一种实施方式，输入是查询 q 、相似索引、阈值 s ，以及输出是使查询至少与阈值 s 匹配的项。

自连接

执行自连接利用了建立相似性索引的重复调用。例如，见图 6 中的流程图。

程序

假设数据集 D 由项 $x.i$ 组成，其中 $i=0,\dots,n-1$ 。我们定义自连接程序，其输出 s -集群的分层级分组。

给定 s 在 $(0,1)$ 中的值，在 D 中的每一项 x 属于具有下面属性的集群 $C(x)$ ：

a、对于在 $C(x)$ 中的所有 y ，相似性(x,y) $> s$ 。

b、至少有一个 $C(x)$ 的成员，称其为锚 (anchor) a ，使得最接近 a 的成员数量大于或等于最靠近 $C(x)$ 的任何其它成员的成员数量。(如何对此说明以包含 $C(x)$ 是有限的、无穷的但是可数的、和无穷但是不可数的情形？)。

c、D 中的每个 x 属于某个集群 $C(x)$ 。

d、在 D 中 x 的集群 $C(x)$ 的集是有限的互为唯一的。

我们注意到对于 s 的给定值，可有不止一种方式定义满足上述属性的集群。换而言之，用自连接技术输出的集群的集可能不是唯一的，并可有对同样满足期望的条件来对项进行分组的其它方式。

图 6 一般在 600 举例说明了显示自连接的本发明的一种实施方式。如上指出的，执行自连接利用了建立相似性索引的重复调用。在 602 我们指定最大集群大小为 G ，以及最小集群相似性为 s 。在 604 我们输入 n 个内容项。在 606，我们大量建立相似性索引，这产生具有叶节点的树，且每个叶节点有 G 个或更少的项。在 608，对于每个叶节点，我们计算各项间的两两 (pairwise) (也称为 pair-wise) 相似性。在 610，我们核对以查看是否至少有一对项的相似性超过 s 。如果没有至少一对项的相似性超过 s ，那么在 612 我们定义“一次性 (oneoff)”(也称为 one-off) 集群，并将项

放在其中，并然后进行到 622。如果有相似性超过 s 的至少一对项，那么在 614 我们计算最高的两两相似性，并指明一个项作为集群的“锚”。接下来在 616，我们定义“好”集群。在 618，对于在当前节点的每一项，如果其与锚的相似性超过 s ，那么将其放在好集群里，否则，如果其与锚的相似性没超过 s ，那么在 620，如果有剩余项，指明剩余项作为“残留(residual)”并分别收集它们。在 622，我们确定我们是否已处理所有的叶节点。如果我们没有处理所有的叶节点，那么我们在 608 重新开始。如果我们已经处理所有的叶节点，那么在 624 我们核对以查看在这个点我们是否有任何收集的残留。如果在这个点我们没有任何收集的残留，那么我们进行到 626，完成。如果在这个点我们具有任何收集的残留，那么我们进行到 628，在 628 我们聚集较早收集的残留，且我们然后进行到 606。于是图 6 一般举例说明了显示自连接方法的本发明的一种实施方式，其中输入是内容项的集合，且输出是项的“好”和“一次性”集群。

图 7 一般在 700 举例说明了显示自连接树的实施例的本发明的一种实施方式。显示各种各样的等级，等级 0(702)、等级 1(704)和等级 2(706)。显示了顶端的集 (708、720 和 728)。在顶端 708 和“好”群 712 之间显示指向“锚” 710 的指针。同样在 714 和 716 显示“好”群。在 718 是没有锚的“一次性”群。同样在 722 和 724 显示“好”群。726 显示“一次性”群，其中没有一项与该群中的另一项的相似性满足相似性阈值 s 。

常规连接

给定两个一致的数据集 D.1 和 D.2 以及相似性阈值 s^* ，这个程序识别集合 s^* -连接集群。每个 s^* -连接集群显示关于两个数据集的重要信息，因为它识别在一个数据集中的哪一项具有其他数据集中的相似的对应部分。而且，由于我们可以设置 s^* 参数，因此我们能够确定，匹配条件(match)是否滤除了几乎全部项，而只留下了那些非常接近匹配条件的项 (s^* 非常接近 1)，或者我们是否乐于找出非常宽泛的匹配条件但是仍然筛选出完全不匹配的项 (例如，使用 s^* 的 70%)。

方法

如我们早前所提到的，大规模数据集的一个主要挑战是尽可能避免 n^2 两两相似性计算阶段。例如，在自连接程序期间，仅当我们到每个包含 G 个项的叶节点时，我们采取两两相似性计算。因为我们能控制叶节点 G 的大小，所以我们在 $G^2/2$ 的两两计算（pairwise computation）上设置边界（bound）。

在常规连接背后的想法是尽可能地平衡自连接程序产生的局部层级。当在数据集内有高度的内部相似性时，会有许多“好”群。参考图 7。每个好群从它的中间选定“锚”项，以表示该群在群层级之上。由于每个锚代表 G 个其他群成员，因为外部项相对于锚的相异性（Dissimilarity）等于知道相同的项将也与群的每个成员相异，这导致在计算成本中巨大的节约。

然而，使用此代理表示方案（proxy representation scheme）用于相似性是有代价的，并且这种表示方案不得不接收这一事实：锚仅是每个群成员的不完美的代替物。为了对此进行补偿，我们必须加强（tighten）好群层级的等级之间的相似性阈值，以维持我们对总体 $s*$ 相似性准则的遵守。换言之，因为在自连接分组层级里的每个等级失去一些相似性精确度，所以我们调整自连接中所使用的阈值来补偿。我们通过得到用于相似性的“链规则（chain rule）”来计算在每个等级我们能够允许多少相似性预算，列举如下。

假设我们在两个项 x 和 z 之间有一距离，且我们想引入中间项 y :

$$\begin{aligned} & \|x - z\|^2 \\ &= \| (x - y) + (y - z) \|^2 \\ &\leq \|x - y\|^2 + \|y - z\|^2 \quad (\text{三角不等式}) \end{aligned}$$

如果我们假设 x 、 y 和 z 被规范化以具有单位标准，或者 $\langle x, x \rangle = \langle y, y \rangle = \langle z, z \rangle = 1$ ，我们能够使用论据：相似性(x, y) = $\langle x, y \rangle / \sqrt{\langle x, x \rangle * \langle y, y \rangle} = \langle x, y \rangle$ 。

使用定义 $\|x - y\|^2 = \langle x - y, x - y \rangle$, 并使用内积的属性展开第二项 (second term), 我们获得相似性的链规则:

$$1\text{-相似性}(x, z) \leq (1\text{-相似性}(x, y)) + (1\text{-相似性}(y, z)) \quad (*)$$

顺便说一句, 这能够根据相异性用更值得储存的术语表示。

$$\text{相异性}(x, z) \leq \text{相异性}(x, y) + \text{相异性}(y, z),$$

$$\text{其中相异性 } (x, y) = 1 - \text{相似性}(x, y).$$

改写上述的等式 (*), 我们得到等价链规则:

$$\text{相似性}(x, z) \leq \text{相似性}(x, y) + \text{相似性}(y, z) - 1.$$

这能扩展到两个中间项:

$$\text{相似性}(x, z) \leq \text{相似性}(x, y_1)$$

$$+ \text{相似性}(y_1, y_2)$$

$$+ \text{相似性}(y_2, z)$$

- 2。

这能扩展到多个中间项, y_i , 其中 $i=l, k$.

$$\text{相似性}(x, z) \leq \text{相似性}(x, y_l)$$

$$+ \text{sum}(i=l, k-1; \text{相似性}(y_i, y_{i+1}))$$

$$+ \text{相似性}(y_k, z)$$

- k。

这个表达式能解释为意味着每次我们在 x 和 z 之间插入附加的中间项, 在这两个端点间的相似性的下边界被在每个附加的中间项之间的相似性降低。

我们将使用这个链规则确定我们需要在每个自连接上使用什么样的相似性阈值, 使得当我们试图推断从在树里的叶节点向上传播到锚和一次性节点的相似性时, 我们能够控制因树中的多个等级而给解决方案引入的

精确度损失。在此情况下，我们将总体相似性预算均匀地分配在各等级间。

为了给出在操作中的链规则的实施例，下面的表格显示出，如果需要要求当两个叶项（leaf items）通过某个锚项（anchor item）而彼此相关时，在最坏的情形下，我们能够断言两个项满足 85% 的相似性阈值，则必要的“内部”相似性阈值是必需的。如表格指示的，随着树变得更高，为了维持对总体相似性的保障，我们必须在等级间强制更大程度的相似性。

在自连接树中的等级	获得总体 85% 相似性所需的内部相似性
1	85.00%
2	95.00%
3	97.00%
4	97.86%
5	98.33%
6	98.64%

解决方案准则

常规连接程序可达到的明确的条件是，每个 s^* -连接集群具有集群的每个成员与另一数据集的某个成员至少在 s^* 相似性阈值内的这个属性。而且，如果连接集群的任何成员是锚，那么可通过所述锚直接或间接可达到的所述成员的数据集中的所有成员与另一数据集的某个成员至少在 s^* 相似性阈值内。另外，如果能够从连接集群建立一个连接对，对的每个成员是锚，那么每个锚的整体分组子树是满足 s^* 阈值的备选。相反的，如果有来自 D.1 的 x.1 和来自 D.2 的 x.2，且 $\text{相似性}(x.1, x.2) > s^*$ ，那么来自常规连接的解决方案包含 s^* -连接集群，在该 s^* -连接集群中，或者 a) x.1 和 x.2 都是成员，或者 b) 有是 s^* -连接集群的成员的锚，其中通过那个锚，x.1 和/或 x.2 可直接或间接获得。

程序

- 1、给定相似性阈值 s^* 。
- 2、假设数据集 1 和数据集 2 的大小分别为 N_1 和 N_2 。
- 3、挑选分组大小 $G_1 > 0$ 。（我们希望 G 足够大，使得分层级的自连接树的预期的大小可管理，但是同时， G 足够小，使得 $G_1^{1/2}/2$ 两两相似性计算保持有界。我们计算最优分组大小，其最小化计算两两相似性的全部计算开销，加上计算建立自连接二进制树所需的上部/下部分离的开销。）根据高度 $h_1 = \text{ceiling}(\log_2(N_1/G_1))$ ，分组大小 G_1 的选择确定二进制树的高度。
- 4、按以下方式为数据集 1 挑选相似性阈值 s_1 。使用自连接树 h_1 的预期高度，计算“内部”相似性 s_1 为

$$s_1 = 1 - (1 - s^*) / (2 * h_1 - 1)$$

这是当在数据集 1 上建立自连接树时我们所需要使用的相似性，以确保考虑了树的完整高度，以使每个锚对它的群的相似性足够接近，以保存的所有备选连接对。
- 5、对数据集 2 实施相同操作。即，挑选分组大小 G_2 、高度 h_2 和内部相似性阈值 s_2 。
- 6、使用较早确定的分组大小和内部相似性阈值，为每个数据集计算自连接。
- 7、识别每个数据集的自连接树的终端元素，其由根集群的锚加上所有一次性群的成员组成。
- 8、将来自数据集 1 和数据集 2 的自连接树的终端元素结合到单个集合里，并使用 s^* 的相似性阈值和对应于在这个集合里的项的数量的最优分组大小来计算这个集合上的自连接。
- 9、检查由自连接产生的集群，并对 s^* 的相似性阈值挑选所有的连接集群。将每个挑选出的连接集群加到结果集。
- 10、按以下方式对连接的结果进行解释。根据对 s^* -连接集群的定义，每个这样的集群包含来自数据集 1 和数据集 2 的至少一个成员。集群的成

员满足 s^* 相似性阈值条件。此外，集群的一些成员是在上述步骤 6 中在其自身的数据集上执行的自连接中的锚。在那种情形，我们说，通过该锚直接或间接可达到的数据集的成员是连接集的一部分。特别地，如果连接集群包含分别来自数据集 1 和数据集 2 的锚 a_1 和锚 a_2 ，那么从锚 a_1 可达到的数据集 1 的成员显示为被连接到从锚 a_2 可达到的数据集 2 的成员。

参考图 8 中的流程图。

图 8 一般在 800 举例说明了显示常规连接的本发明的一种实施方式。在 802，给定有阈值 s^* 的数据集 1 和数据集 2。在 804，我们能够挑选预先确定的固定的“分组”大小 G_i ，或任选地，为每个数据集确定将建立相似性索引的总体计算时间减到虽小的“分组”大小 G_i 。接下来在 806，给定数据集 1 和数据集 2 的大小，我们计算树的预期高度 $h_i = \text{ceiling}(\log_2(N_i/G_i))$ 。在 808，对于数据集 1 和数据集 2，我们使用相似性阈值 $s.i = 1 - [(1-s^*)/(2*h.i - 1)]$ 建立自连接树。接下来在 810，每个自连接产生好群和一次性群的树，且我们指明“顶端”集为根好群 (root good group) 的锚连同一次性群的成员。在 812，我们从之前步骤 (810) 中识别的“顶端”集形成数据集。接下来在 814，使用 s^* 作为相似性阈值，我们在之前步骤 (812) 中形成的数据集上执行自连接。现在，在 816，根据在之前步骤 (814) 中的自连接，我们输出具有来自数据集 1 的至少一个成员和来自数据集 2 的至少一个成员的“好”群，并称这些为 s^* -连接集群。我们接下来进行到 818，完成。因此图 8 一般举例说明了显示常规连接的方法的本发明的一种实施方式，其中输入是内容项的集合，以及输出是项的“ s -连接”集群。

正确性论据

常规连接算法的正确性要求：对于在数据集 1 里的 $x.1$ 和在数据集 2 里的 $x.2$ ，当且仅当相似性($x.1, x.2$) $> s.0$ 时， $x.1$ 和 $x.2$ 或者 a) 在结果集里识别为成对的，或者 b) $x.1$ 和 $x.2$ 的一个或两者由在它们的自连接树内的

锚代表，以及它们的表示直接出现或间接连接在结果集里。

根据结果集的结构，我们知道为结果集挑选的任何对满足相似性($x_1, x_2) > s_0$ 。这样建立“仅当”说明的部分 a)。

为了建立“仅当”说明的部分 b)，我们注意到，我们将 D.1 和 D.2 的自连接树构造成，从顶端锚或一次性项到树中的在其之下的任何项的总体相似性的下界是 s^* 。换言之，我们考虑到了任何项到它的锚的距离，并合计基于树的高度的总体相异性，以确保在树内的项的任何连接对至少是相似性 s^* 。由于所说明的在每个自连接树内的总体相似性，我们只需要考虑在 s^* -连接集群里任何其它成员的附加的相似性，其由锚到另外的锚或直接到项而产生。这个附加的距离可导致在最后的解决方案里的一些可能成对的消除。

对于论据的“当”说明，假设相反的存在 x_1 和 x_2 ，其中 x_1 和 x_2 分别来自数据集 1 和 2，其中，相似性($x_1, x_2) > s_0$ ，而且在结果集里未出现 x_1 和 x_2 。换言之，存在来自数据集的一对匹配项，而我们未能通过在结果集里它们的直接出现，也未能通过在结果集里的代表性锚来识别所述匹配。我们希望建立矛盾(contradiction)。

这意味着我们具有如下情况：在连接算法期间的任何时间， x_1 和 x_2 中的任一个或两者在结果集里没有出现，而且，它们的自连接树中向上的锚中的任何一个也没有出现在结果集。我们将通过归纳继续。

对于基本情形，假设 x_1 是顶端的锚，或它出现在一个一次性群中。假设 x_2 同样如此。连接算法的第一步将两个数据集的所有顶端的锚和一次性成员结合到单个集合里，并使用 s^* 阈值在其上执行自连接。为进一步处理，识别包含来自数据集 1 和数据集 2 的每个的至少一个成员的任何集群；我们将其称为 s^* -连接集群。但是这导致矛盾，因为我们具有的情形是 x_1 和 x_2 在顶端集里，且它们满足 s^* 条件，但是在所述情形中，自连接程序将它们放入结果集合里。

对于归纳的情形，假设 x_1 既不是顶端锚，也不在一次性群里。这意味着它在包含锚的群里。并反之，每个锚或者在一次性群里，或者它也在

有锚的群里。通过自连接树的构造，我们知道 $x.1$ 与顶端锚或与一次性群成员的相似性至少是 s^* 。

相同的推理适用于 $x.2$ 。由此，我们知道 $x.1$ 和 $x.2$ 在顶端集里具有满足 s^* 条件的表示。但是在程序的过程中，我们从 D.1 和 D.2 构成顶端项的 s^* -连接集群。因此，对于我们来说，不直接构成具有 $x.1$ 和 $x.2$ 的表示的群的唯一的方法是，假设所述表示被分离到未由公共锚连接的不同的群中。但是这与自连接的属性发生矛盾。

由此，我们已经建立了论据的“当”和“仅当”说明。

讨论

我们注意到，在两个数据集间执行连接的工作发生在两部分。首先，我们在每个数据集上单独地进行自连接。第二，我们在每个数据集的顶端元素上进行自连接。实际上，我们看到，由于锚作为各自的群成员的代理而出现，因此，当数据集在它自身内具有相当程度的相似性时，顶端集可以大大地小于整体数据集。当两个数据集具有高度的内部相似性时，那么顶端成员的数量将相对小，且在结合的顶端集上的自连接将相对迅速。

另一方面，当数据集在其自身内包含非常小的相似性，那么自连接产生了大量一次性群，而锚很少。在这种情形，顶端成员将几乎像全部数据集一样大。如果第二数据集同样如此，那么我们看到，在顶端集上的自连接将构成大量工作。

我们指出，用于连接两个数据集的这个程序能扩展成连接多个数据集。首先，在每个数据集上分别进行自连接。第二，将每个自连接的顶端成员收集到组合的顶端成员集里，并在所述集上执行自连接。

实际上，当增加或移除项时，为数据集递增地更新相似性索引是有意义的，因为自连接是与其它数据集连接的前提，并且反之，相似性索引是自连接的必要组成。参考文献 R. Nakano 的日期为 2005 年 2 月的美国专利申请第 11/361166 号 “Method and apparatus for efficient indexed storage for unstructured content” 中描述的程序可适用于这种情况。

“群”和“集群”

本领域技术人员将认识到，通常用几个术语描述相同的事物。如在这个描述使用的，使用术语“群”和“集群”或相似的短语指相同的事物。例如，在上述定义部分中，我们这样定义 s-集群：(s-集群)我们说当对于在集合中的任何项 x ，对于在集合中的项具有大于或等于 s 的相似性时，一致的项集合构成 s -集群。词“集群”指“项”的集合，在别处指“内容项”。通过说它们在“集群”里，强调了在集群里的所有元素具有 s 或更大的两两相似性。例如，如果相似性阈值 s 是 99%，我们具有 10 个图像 $p.1, \dots, p.10$ ，并在被适当转变为它们各自的内积空间向量后，对于在 $\{1, \dots, 10\}$ 中的所有 i 和 j ，所述相似性 $(p.i, p.j) \geq 0.99$ 。

另一例子在自连接部分的讨论中，我们规定：给定在 $(0, 1)$ 中的 s 的值，在 D 中的每个项 x 属于具有如下属性的集群 $C(x)$ ：....这里词“集群”指相同的事物。所以继续上述的例子，存在值 $s=0.99$ ，其中，图像集合里的每个图像 x 具有的属性为，图像的两两相似性满足或超过 99% 阈值。

另一例子是关于群的讨论，例如在图 8 中公布了“每个自连接产生好群和一次性群的树”。这里，自连接操作被描述为产生好群和一次性群。这与说好群是满足相似性阈值 s 的 s -集群是相同的。一次性群是未能满足 s 阈值的内容项的集。例如，如果我们已经设置 s 为 99%，那么好群是 0.99-集群。但是一次性群是准则未能满足 0.99-集群条件的集合。如果保持相似性阈值条件，则我们能够使用术语“好集群”、“好集群”和“ s -集群”来表示相同的事物“。否则，如果未保持相似性条件，则我们能够使用术语“一次性群”和“一次性集群”或相似短语来表示相同的事物。

从这里我们看到，“群”和“集群”（和相似短语）指具有一些已知两两相似性值的内容项的集。因为我们知道这个，我们能够断言是否获得了给定的相似性阈值 s 。特别地，如果内容项的所有对超过 s 相似性，我们具有 s -集群、好群或好集群。否则，我们不具有 s -集群，但是改为具有一次性群或一次性集群。

附带说明，注意，如果我们具有 n 个内容项且我们不知道它们的所有 $n*(n-1)/2$ 个两两相似性值，那么我们不能判定它们是否构成好群、一次性群、好集群、一次性集群或 s -集群。在简并的情形下，任何 n 个项构成 0-集群。换言之，内容项的任何对具有相似性 ≥ 0 ，这成立，但是无助于我们。

要说明或证明的(Quod erat demonstrandum)

我们已经显示对于大规模高维数据集如何执行基于相似性查询、自连接和常规连接。

因此描述了用于大规模高维数据集的快速的基于相似性的查询、自连接和连接的方法和设备。

返回参考图 1，图 1 举例了网络环境 100，其中可使用描述的技术。网络环境 100 具有连接 S 个服务器 104-1 到 104-S 以及 C 个客户端 108-1 到 108-C 的网络 102。如所示，几个计算机系统以 S 个服务器 104-1 到 104-S 以及 C 个客户端 108-1 到 108-C 的形式通过网络 102 彼此连接，网络 102 可能是例如基于企业的网络。注意，可选择地，网络 102 可以是或包括一个或更多下列项：因特网、局域网（LAN）、广域网（WAN）、卫星链路、光纤网络、电缆网或它们的组合和/或其它。服务器可代表，例如仅仅磁盘存储系统或存储器以及计算资源。同样地，客户端可具有计算、储存和浏览能力。这里描述的方法和设备可被用于实质上任何类型的通讯装置或计算设备，而不论是本地的还是远程的，例如 LAN、WAN、系统总线、微处理器、主机、服务器等等。

返回参考图 2，图 2 以框图形式举例说明了计算机系统 200，其是显示在图 1 中的任何客户端和/或服务器的代表。该框图是高级概念表示形式，并可以用种种方式及通过不同结构实现。总线系统 202 与中央处理器（CPU）204、只读存储器（ROM）206、随机存取存储器（RAM）208、储存器 210、显示器 220、音频 222、键盘 224、指示器(pointer)226、不同种类的输入/输出（I/O）设备 228 和通信系统(communications)230 互相连

接。总线系统 202 可以是例如作为系统总线的一个或更多这样的总线外设部件互连 (PCI)、加速图形端口 (AGP)、小型计算机系统接口 (SCSI)、电气和电子工程师协会 (IEEE) 标准号 1394 (FireWire)、通用串行总线 (USB) 等等。CPU 204 可以是单个、多重甚至是分布式计算资源。储存器 210 可以是压缩磁盘 (CD)、数字化视频光盘 (DVD)、硬盘 (HD)、光盘、磁带、快闪记忆体、记忆棒、录影机等等。显示器 220 可以是例如电子射线管 (CRT)、液晶显示器 (LCD)、投影系统、电视机 (TV) 等等。注意，依赖于计算机系统的实际实现，计算机系统可包括框图中的组件的一些、全部、更多或重新排列。例如，瘦客户端 (thin client) 可由缺少例如传统键盘的无线手持设备组成。因此，在图 2 的系统上的许多变化是可能的。

为了讨论和理解本发明，应理解，本领域技术人员使用各种各样的术语描述技术和方法。而且，在描述中，为了解释的目的，为了提供本发明的通透的理解而提出许多明确的细节。然而对于本领域普通技术人员将是明显的是，可实践本发明而无需这些明确的细节。在一些实例里，为了避免使本发明模糊不清，众所周知的结构和设备以框图形式显示，而不是详细地显示。已足够详细地描述了这些实施方式，以使本领域普通技术人员能够实践本发明，并应理解，可利用其它的实施方式，且可进行逻辑的、机械的、电的和其它改变而不背离本发明的范围。

根据算法和对例如在计算机存储器内的数据位的操作的象征性表示，介绍了描述的一些部分。这些算法描述和表示是数据处理领域的普通技术人员使用的方式，以便最有效地将其工作的实质传达给本领域的其他普通技术人员。算法在这里通常被设想为导致期望结果的自一致的动作序列。这些是要求物理量的物理处理的动作。虽然不一定，但通常这些量采取能够被储存、转移、结合、比较和以其他方式处理的电或磁信号的形式。主要是由于公共使用的原因，有时将这些信号称为位、值、单元、符号、字符、术语、数字或等，已证明是方便的。

然而，应该牢记，所有这些以及相似的术语与适当的物理量相关联，且仅仅是适用于这些量的方便的标签。除非特别声明，否则从讨论中明显

的，应认识到，贯穿描述，利用了例如“处理”或“计算”(computing)或“计算”(calculating)或“确定”或“显示”或之类的术语的讨论，可以指计算机系统或类似电子计算设备的动作和处理，这些动作和处理将在计算机系统的寄存器和存储器内的表示为物理(电)量的数据，操纵并转换为在计算机系统存储器或寄存器或其它这样的信息存储器、传输设备或显示设备内的类似地表示为物理量的其它数据。

用于执行这里的操作的设备能实现本发明。这样的设备可被特别地构建以实现所需目的，或者它可包括通用计算机，其被储存在计算机里的计算机程序有选择地激活或重新配置。这样的一个计算机程序可存储在计算机可读存储介质里，例如但不限于，包括软盘、硬盘、光盘、压缩磁盘-只读存储器(CD-ROM)、以及磁光盘、只读存储器(ROM)、随机存取存储器(RAM)、电可编程只读存储器(EPROM)、电可擦除可编程只读存储器(EEPROM)、闪存、磁或光卡等等的任何类型的盘，或适合于电子指令的本地计算机储存或远程计算机储存的任何类型的介质。

这里提出的算法和显示不是固有地涉及任何特殊的计算机或其它设备。依照这里教授的，可通过程序使用各种各样的通用系统，或者可证明构建更专门的设备来执行要求的方法是方便的。例如，通过对通用处理器编程，或通过硬件和软件的任何组合，可用硬连线电路实现根据本发明的任何方法。本领域的普通技术人员将立即认识到，本发明能用不同于所描述的计算机系统结构来实践，包括手持设备、多处理器系统、基于微处理器或可编程的消费类电子产品、数字信号处理(DSP)设备、机顶盒、网络个人计算机、小型计算机、大型计算机及类似处理。本发明还可在分布式计算环境中实践，其中，任务通过由通信网络链接的远程处理设备执行。

本发明的方法可使用计算机软件实现。如果是用符合公认标准的程序设计语言编写，被设计成实现该方法的指令序列能够被编译，用于在多种硬件平台上执行，并用于通过接口连接到许多操作系统。另外，本发明不参考任何特殊的程序设计语言来描述。将理解，可使用多种程序设计语言实现如这里描述的本发明教授的内容。此外，在本领域中以一种形式或其它形式(例如程序、过程、应用程序、驱动程序、....)提及软件作为采取

行动或造成结果是很普遍的。这样的表达仅仅是一种简要表达方式，用来指示计算机的软件执行使得计算机的处理器实现动作或产生结果。

应理解，本领域技术人员使用各种各样的术语和技术描述通信、协议、应用、执行、机制等等。根据算法或算术表达式，一种这样的技术是对技术实现的描述。即，当技术可以例如实现为在计算机上执行代码时，所述技术的表达可更适宜地且更简便地作为公式、算法或算术表达式传送和通信。因此本领域的普通技术人员将表示 $A+B=C$ 的模块看作是一加法函数，该模块的硬件和/或软件实现接收两个输入 (A 和 B) 并产生总和输出 (C)。因此，如描述的公式、算法或算术表达式的使用应理解为至少具有硬件和/或软件的物理实施方式 (例如一个计算机系统，其中本发明的技术被实践，并被实现为实施方式)。

机器可读介质被理解为包括用于以机器 (例如，计算机) 可读形式储存或传播信息的任何机制。例如，机器可读介质包括只读存储器 (ROM)；随机存取存储器 (RAM)；磁盘存储介质；光存储介质；闪存设备；电、光、声或其它形式的传播信号 (例如载波、红外信号、数字信号等等)；等等。

正如在本说明中使用的，“一种实施方式”或“一个实施方式”或相似的短语意味着描述的特征包括在本发明的至少一种实施方式里。本说明里提及“一种实施方式”不一定指相同的实施方式；然而，这样的实施方式也不互相排斥。“一种实施方式”也不意味着仅是本发明的单个实施方式。例如，在“一种实施方式”中描述的特征、结构、动作等等可也包括在另外的实施方式里。因此，本发明可包括这里描述的实施方式的多种组合和/或结合。

因此，描述了用于大规模高维数据集的快速的基于相似性的查询、自连接和连接的方法和设备。

附录 1

(内积空间)给定在向量空间中的任意两项 x, y , 有满足下列属性的内积:

- a. $\langle x, x \rangle \geq 0$
- b. $\langle x, x \rangle = 0$ iff $x = 0$.
- c. $\langle x, y \rangle = \langle y, x \rangle$
- d. $\langle a^*x, y \rangle = a * \langle x, y \rangle$
- e. $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$

在两个非零元素 x 和 y 之间的相似性可表示为,

$$\text{相似性}(x, y) = \langle x, y \rangle / \sqrt{\langle x, x \rangle * \langle y, y \rangle}.$$

我们注意到内部空间是有如下度量的度量空间。

$$\|x\| = \sqrt{\langle x, x \rangle}.$$

三角不等式为,

$$\|x - y\|^2 \leq \|x\|^2 + \|y\|^2.$$

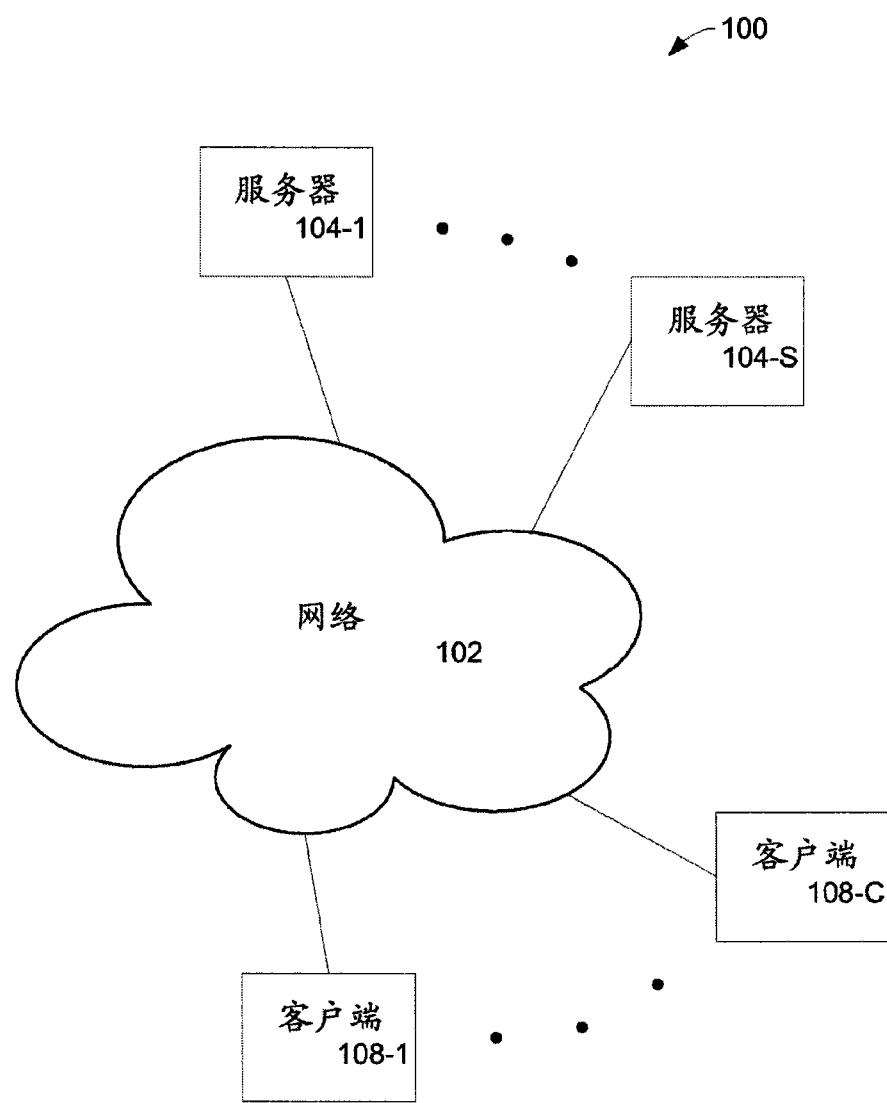


图1

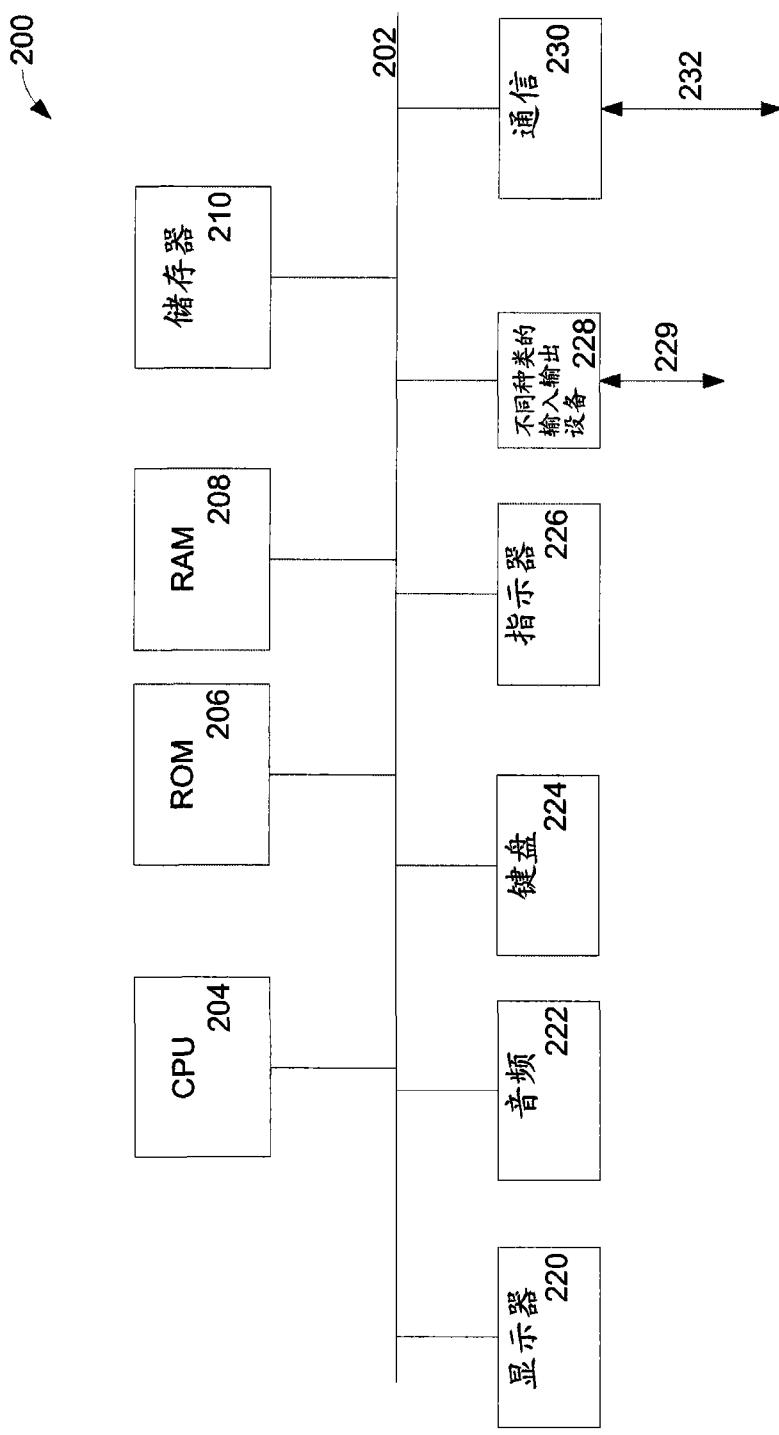


图2

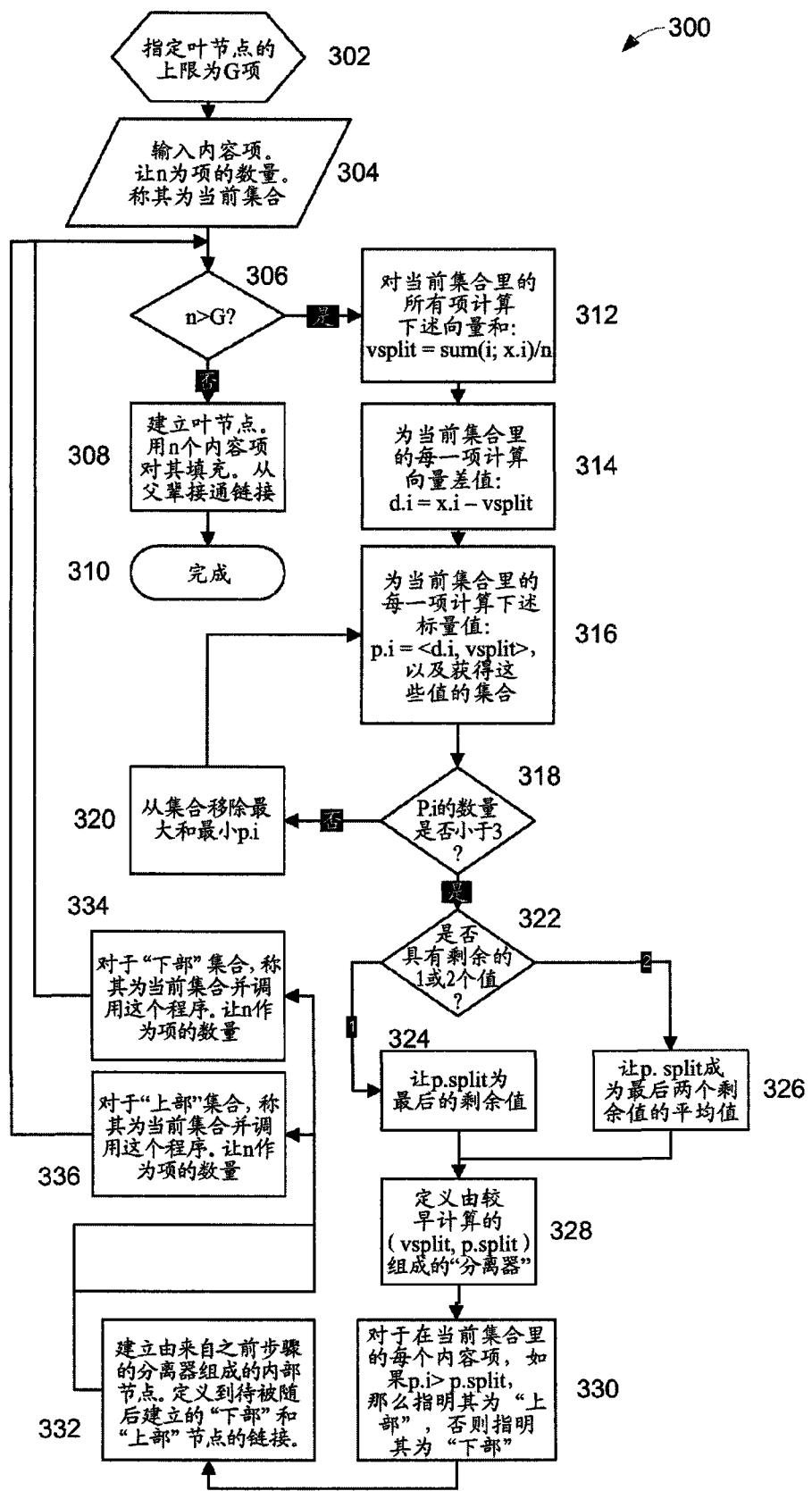


图3: 建立相似性索引

输入: 内容项的集合
输出: 二进制树, 项在叶节点中

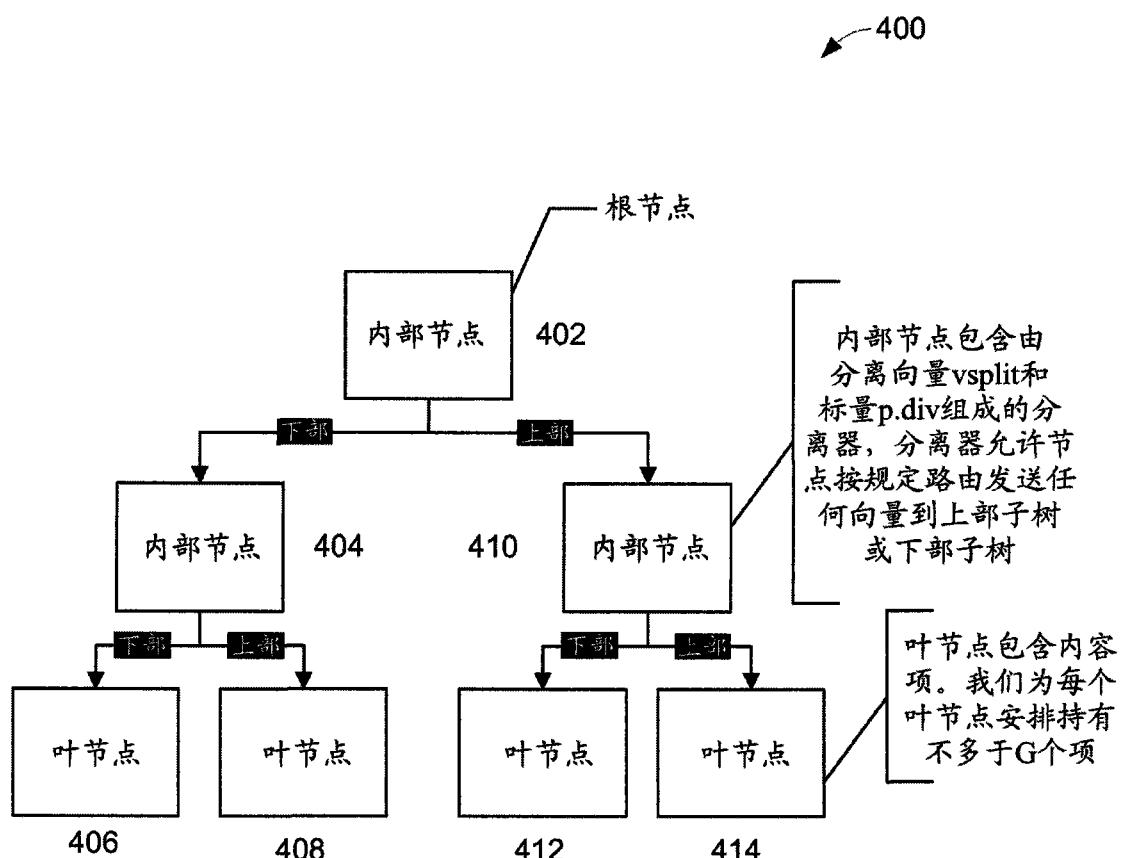


图4：相似性索引的实施例

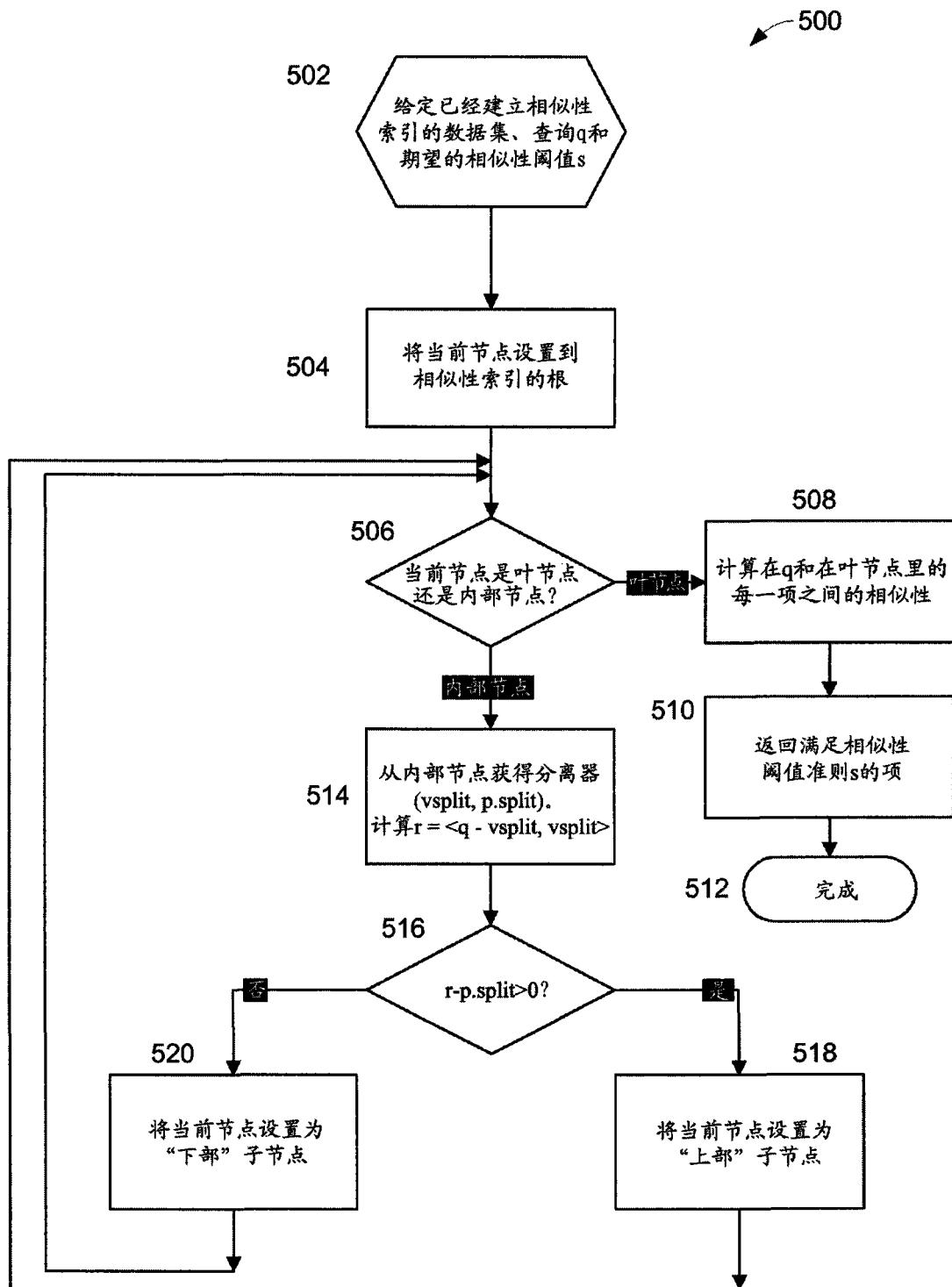


图5：查询

输入：查询q、相似性索引、阈值s

输出：使查询至少与阈值s匹配的项

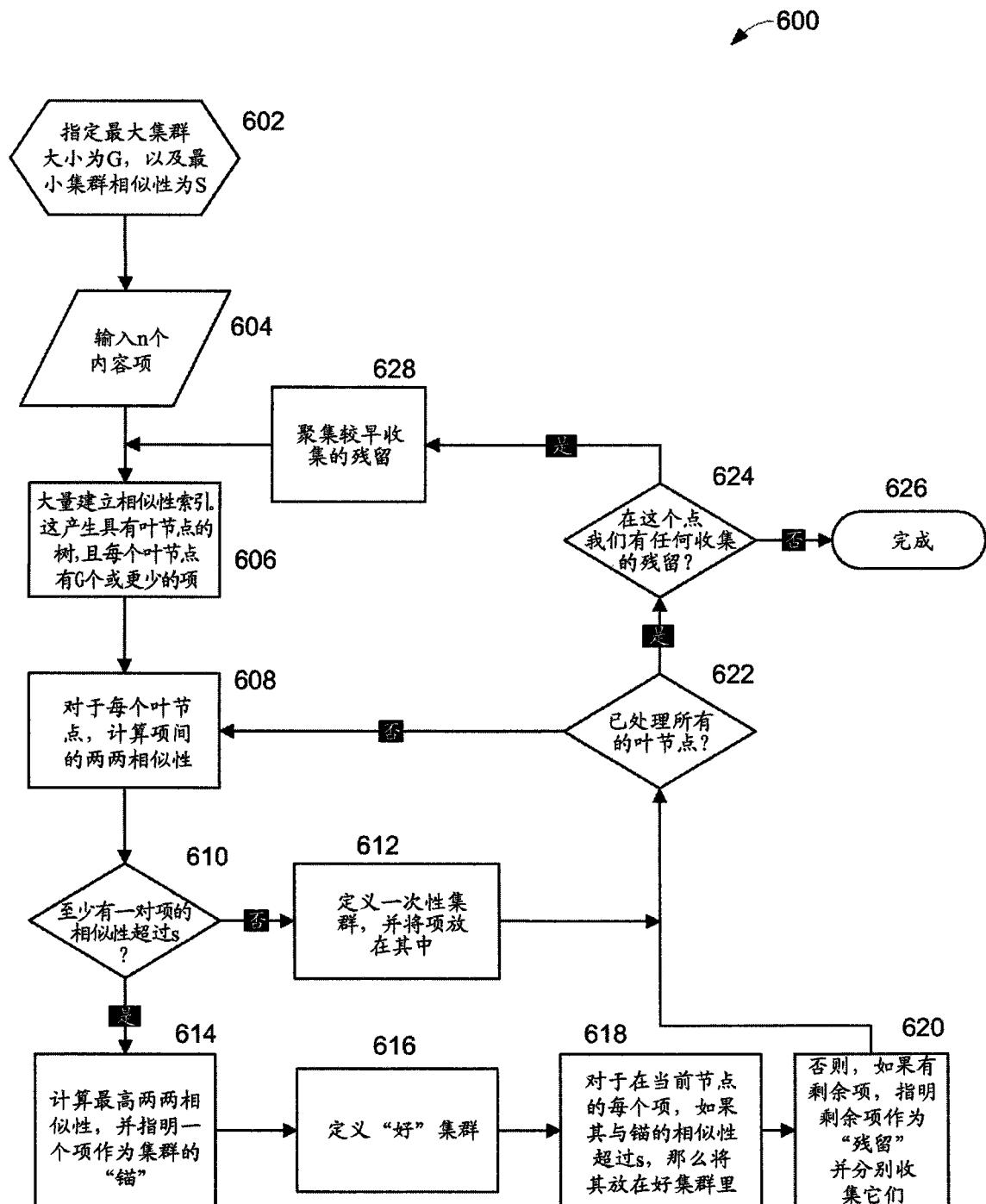


图6: 自连接

输入: 内容项的集合
 输出: 项的“好”和“一次性”集群

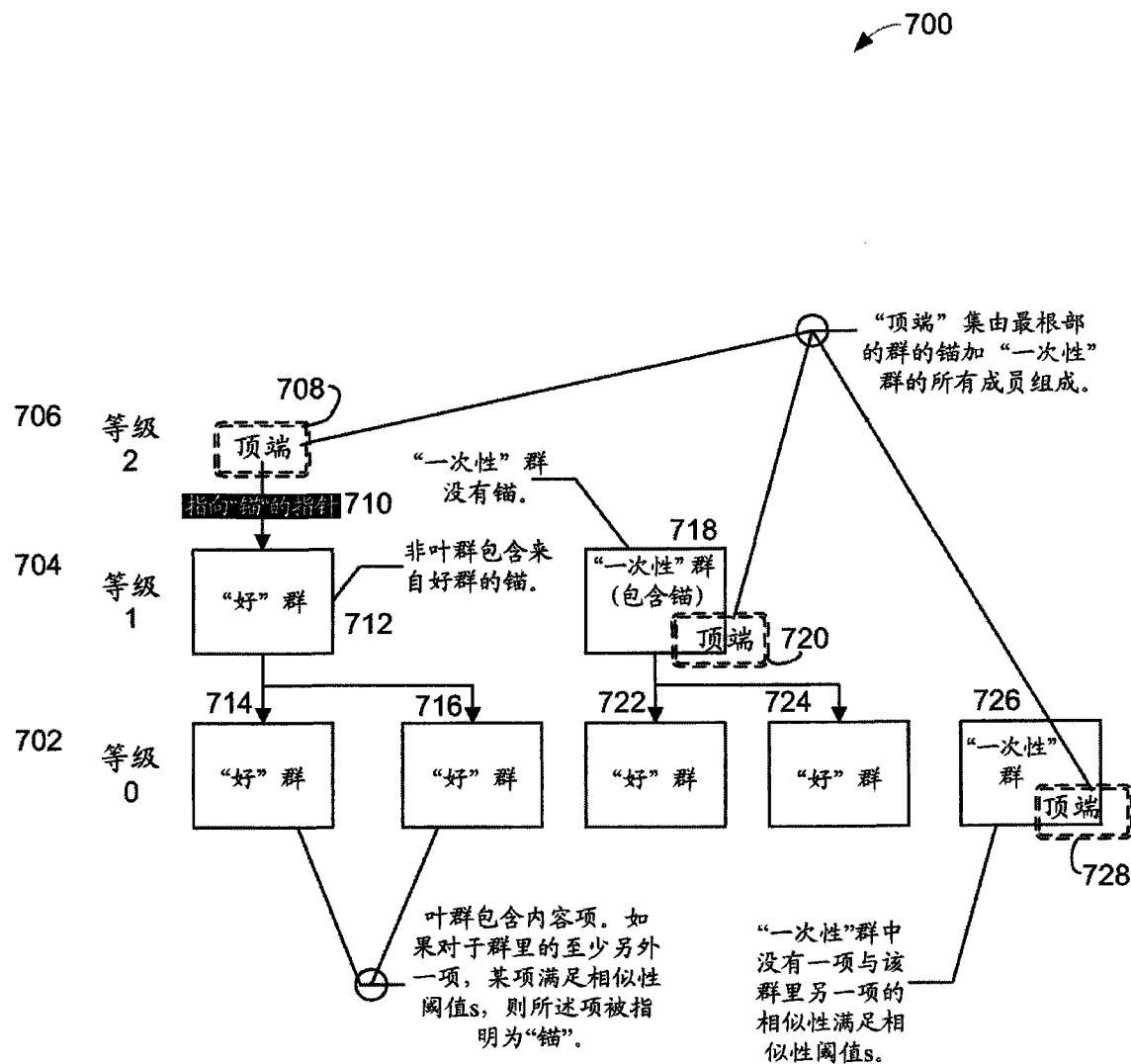


图7：自连接树的实施例

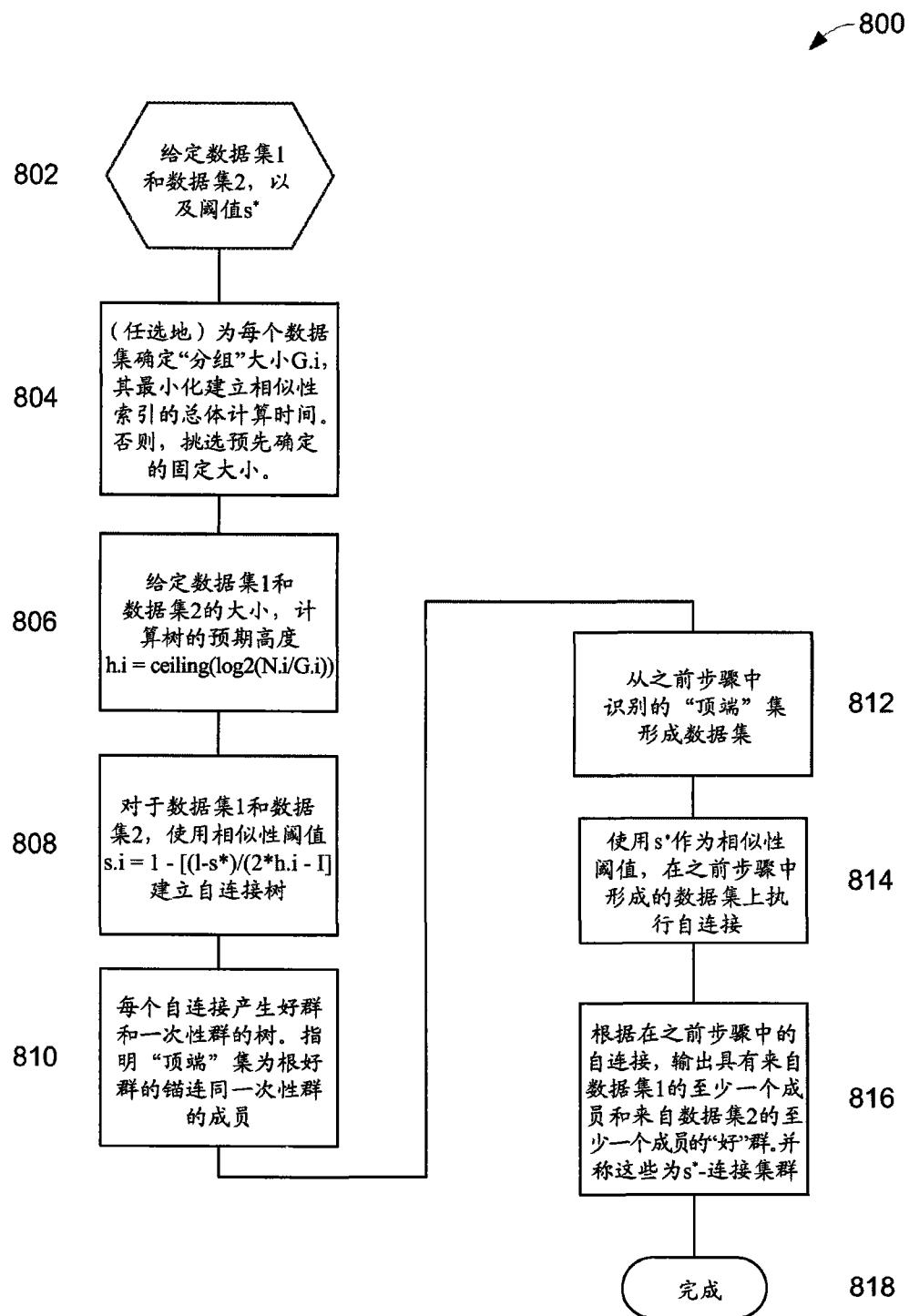


图8：常规连接

输入：内容项的集合
输出：项的“s-连接”集群