5.2.2 使用Map-Reduce的PageRank迭代

PageRank算法的一个迭代包括获取一个估计的页面秩向量v，然后计算下一个估计v ' by

v′=βMv +(1−β)e / n

召回β是一个常数略小于1,e是一个向量的1,n是图中节点的数目,转移矩阵M代表。

如果n足够小，每个Map任务可以在主内存中存储完整的向量v，并且在主内存中也有空间存储结果向量v '，那么这里就只有矩阵-向量乘法了。额外的步骤每个组件的Mv乘以常数β和添加(1−β)/ n每个组件。

然而，考虑到当今Web的大小，v可能太大，无法装入主内存。即2.3.1节,我们讨论了在分段的方法,我们将M为垂直条纹(见图2.4)和v分解成相应的横纹,将允许我们使用映射-规约模式过程有效地执行,没有更多的v在任何地图任务比可以方便适合在主存。

5.2.3 使用组合子合并结果向量有两个原因

第5.2.2节的方法可能不够充分。

1.我们可能希望在Map任务中添加v '的项，即结果向量v的第i个分量。这种改进与使用组合器是一样的，因为Reduce函数只是添加带有公共键的项。回想一下，对于矩阵-向量乘法的map-reduce实现，关键是i的值，其中有一个术语mijvj。

i

2.我们可能根本不使用map-reduce，而是在一台机器或一组机器上执行迭代步骤。

我们将假设我们正在尝试与Map任务一起实现一个组合器;第二种情况使用了基本相同的思想。

假设我们使用条带方法对不适合主存的矩阵和向量进行分区。然后矩阵M的一条垂直条纹和向量v的一条水平条纹将构成结果向量v '的所有分量。因为这个向量和v一样长，所以它也不适合存储在主存中。此外，由于M是逐列存储的，因此一列可以影响v '的任何组件。因此，当我们需要向某个组件v '添加一个项时，该组件不太可能已经在主内存中了。因此，大多数术语都要求将页面放到主内存中，以便将其添加到适当的组件。这种被称为“抖动”的情况需要大量的时间来实现。

i

162 第五章。链接分析

另一种策略是将矩阵划分为k2块，而向量仍然划分为k条。图5.12显示了k = 4的除法。请注意,我们没有显示矩阵的multiplica-tionβ或添加(1−β)e / n,因为这些步骤简单,不管我们使用的策略。

=

4

1

2

3

|  |
| --- |
| **v** ’ |
| **v** ’ |
| **v** ’ |
| **v** ’ |

|  |  |  |  |
| --- | --- | --- | --- |
| *M11* | *M12* | *M13* | *M14* |
| *M21* | *M22* | *M23* | *M24* |
| *M31* | *M32* | *M33* | *M34* |
| *M41* | *M42* | *M43* | *M44* |

|  |
| --- |
| **v1** |
| **v2** |
| **v3** |
| **v4** |

图5.12:将矩阵划分为方块

在这个方法中，我们使用k2 Map任务。每个任务都得到一个矩阵M的平方，比如Mij，以及向量v的一条条纹，它必须是vj。注意，向量的每条条纹都被发送给k个不同的映射任务;对于i的k个可能值，将vj发送给处理Mij的任务，因此v通过网络传输k次。然而，矩阵的每一部分只发送一次。由于矩阵的大小，如第5.2.1节中所描述的正确编码，可以预期是向量大小的几倍，因此传输成本不会比可能的最小值大太多。由于我们正在对Map任务进行大量的组合，所以在将数据从Map任务传递到Reduce任务时保存数据。

这种方法的优点是我们可以同时保留第j条

当我们处理Mij时，v '和主存储器中v '的第i条条纹。注意，所有由Mij和vj生成的术语都与v '有关，而与v '无关。

i

5.2.4 表示转换矩阵的块

由于我们用5.2.1节中描述的特殊方式表示转换矩阵，因此需要考虑图5.12中的块是如何表示的。不幸的是，一列块所需的空间(前面我们称之为“条带”)大于整个条带所需的空间，但不会太大。

对于每个块，我们需要关于所有那些在块中至少有一个非零项的列的数据。如果每个维度中的条纹数k很大，那么大多数列在其条纹的大多数块中都没有任何内容。对于给定的块，我们不仅必须列出该列具有非零项的行，而且必须对该列所表示的节点重复out-degree。因此，out-degree可能会像out-degree本身一样重复多次。这个观察限在

5.2。PAGERANK的高效计算 163

存储条带块所需的空间是存储整个条带所需空间的两倍。

*A B* *C* *D*

*A*

*B*

*C*

*D*

图5.13:一个4节点的图被分成4个2×2的块

例5.8:假设例5.7中的矩阵被划分为块，k = 2。也就是说，左上象限表示从A或B到A或B的链接，右上象限表示从C或D到A或B的链接，以此类推。在这个小例子中，我们唯一可以避免的条目是M22中的C条目，因为C没有到C或d的弧。

如果我们检查图5.14(a)，我们看到左上角象限的表示。请注意，A和B的学位与图5.11相同，因为我们需要知道的是整个继任者的数量，而不是相关区块内的继任者数量。但是，图5.14(A)或图5.14(c)中表示了A或B的每个后继，但没有同时表示两者。还要注意，在图5.14(d)中没有C的项，因为在矩阵的下半部分(行C和d)中没有C的后继项。 2

5.2.5 PageRank迭代的其他有效方法

第5.2.3节中讨论的算法并不是唯一的选择。我们将讨论其他几种使用较少处理器的方法。这些算法与第5.2.3节的算法有一个共同的优点，即矩阵M只读取一次，而向量v读取k次，其中选择参数k，使向量v和v '的1/k可以保存在主存中。回想一下，第5.2.3节的算法使用k2处理器，假设所有Map任务都在不同的处理器上并行执行。

我们可以将一行块中的所有块分配给一个Map任务，从而将Map任务的数量减少到k。例如，在图5.12中，将M11、M12、M13和M14分配给一个Map任务。如果我们将这些块表示为图5.14所示，我们可以一次读取一行块中的块，因此矩阵不会消耗大量的主内存。在读取Mij的同时，我们必须读取矢量条纹vj。因此，每个k映射任务读取整个向量v，以及矩阵的1/k。

164 第五章。链接分析



源 度

A 3.

B 2

目的地

B

A

(a)将A和B连接到A和B的M11的表示



源 度

C 1

D 2

目的地

A

B

(b)连接C和D到A和B的M12的表示



源 度

A 3.

B 2

目的地

C, D

D

(c)连接A和B到C和D的M21的表示



源 度

D 2

目的地

C

(d)连接C和D与C和D的M22的表示

图5.14:矩阵块的稀疏表示

因此，读取M和v的工作与section 5.2.3的算法相同，但是这种方法的优点是，每个achmap任务都可以将其唯一负责的部分v '的所有术语组合起来。换句话说，Reduce任务除了将从k Map任务接收到的v '片段连接起来之外，没有其他事情可做。

i

我们可以将此思想扩展到不使用map-reduce的环境中。假设我们有一个单独的处理器，它的磁盘上存储着M和v，对M使用我们已经讨论过的相同的稀疏表示。我们可以首先模拟第一个Map任务，即使用M11到M1k块和所有v to

′

计算v1。然后我们模拟第二个Map任务，读取M21到M2k，以及所有v来计算v2，等等。对于前面的算法，我们这样读M一次，v k次。我们可以使k尽可能小，受

′

限制是有足够的主内存来存储1/kth v和1/kth v '，以及尽可能小的一部分M(通常是一个磁盘块)。

5.3。主题敏感网页排名 165

. 5.2.6 第5.2节的练习

练习5.2.1:假设我们希望存储一个n×n布尔矩阵(只有0和1个元素)。我们可以用位本身来表示它，或者我们可以把1的位置作为整数对来表示矩阵

整数要求⌈log2 n⌉碎片。前者适用于稠密矩阵;后者适用于稀疏矩阵。矩阵必须有多稀疏(即,什么

为了节省空间，稀疏表示的元素的比例应该是1)?

练习5.2.2:利用第5.2.1节的方法，表示如下图的转换矩阵:

图5.4 (a)。

图5.7 (b)。

练习5.2.3:使用第5.2.4节的方法，表示图5.3中图形的转换矩阵，假设块的边长为2。

练习5.2.4:考虑一个Web图，它是一个链，如图5.9所示，有n个节点。作为k的函数，你可以假设除以n，用5.2.4节的方法描述这个图的转换矩阵的表示

5.3 主题敏感网页排名

我们可以对PageRank做几个改进。在本节中要研究的一个问题是，由于主题的不同，我们可以对某些页面施加更大的权重。执行这种权重的机制是为了改变随机冲浪者的行为方式，让他们更愿意停留在已知覆盖所选主题的页面上。在下一节中，我们将看到如何应用主题敏感的思想来抵消一种名为“链接垃圾邮件”的新型垃圾邮件的影响，这种垃圾邮件是为了愚弄PageRank算法而开发的。

5.3.1 对主题敏感的页面排名的动机

不同的人有不同的兴趣，有时在查询中使用相同的术语表示不同的兴趣。典型的例子是搜索查询jaguar，它可能指动物、汽车、MAC操作系统的一个版本，甚至是一个古老的游戏机。例如，如果搜索引擎可以推断出用户对汽车感兴趣，那么它就可以更好地将相关页面返回给用户。

理想情况下，每个用户都应该有一个私有的PageRank向量，该向量表示每个页面对该用户的重要性。为十亿用户中的每一个用户存储数十亿长度的向量是不可行的，因此我们需要做些什么

166 第五章。链接分析

简单。对主题敏感的PageRank方法为少数几个主题创建一个向量，从而使PageRank偏向于该主题的页面。然后，我们努力根据用户对所选主题的兴趣程度对用户进行分类。虽然我们肯定会丢失一些准确性，但好处是我们只为每个用户存储一个短向量，而不是为每个用户存储一个巨大的向量。

例5.9:一个有用的主题集是Open Directory (DMOZ)的16个顶级类别(sports、med-icine等)。我们可以创建16个PageRank向量，每个主题一个。如果我们能够确定用户对其中一个主题感兴趣，可能是通过他们最近浏览过的页面的内容，那么在决定页面的排名时，我们可以使用该主题的PageRank向量。2

5.3.2 偏置随机漫步

假设我们已经确定了一些代表主题的页面，比如“体育”。“为了创建一个对运动话题敏感的网页，我们可以安排随机的浏览者只被介绍到一个随机的运动页面，而不是任何类型的随机页面。这种选择的结果是，随机的冲浪者很可能在一个确定的体育页面，或者从这些已知的体育页面之一的短路径可以访问的页面。我们的直觉是，由体育页面链接到的页面本身很可能是关于体育的。他们链接到的页面也可能是关于体育的，尽管随着与指定体育页面的距离增加，关于体育的可能性肯定会降低。产生主题敏感PageRank的迭代的数学公式类似于我们用于一般PageRank的方程。唯一的区别是我们如何添加新的冲浪者。假设S是由行/列数字组成的整数集，其中包含我们已经标识为属于某个主题(称为teleport set)的页面的行/列号。设eS是一个向量，它的分量是1，其他分量是0。然后是主题敏感的页面-

S的秩是迭代的极限

v′=βMv +(1−β)eS / | |

这里，和往常一样，M是Web的转换矩阵，|S|是集合S的大小。

例5.10:让我们重新考虑图5.1中使用的原始Web图，我们将其重新生成为图5.15。假设我们使用β= 0.8。然后这个图形的转移矩阵,乘以β,

 0 2/5 4/5 0   4/15 0 0 2/5 





βM =

 4/15 0 0 2/5  4/15 2/5 0 0

这个目录，可以在www.dmoz.org上找到，是一个人类分类网页的集合。

5.3。主题敏感网页排名 167

*A* *B*

*C* *D*

图5.15:重复示例Web图

假设我们的主题由传送集S = {B,D}表示。那么向量(1−β)eS / | |有1/10的第二和第四为另外两个组件组件和0。原因在于1−β= 1/5,年代的大小是2,和eS 1 B和D和0的组件的组件和c。因此,必须迭代方程

 0 2/5  4/15 0

 4/15 0 4/15 2/5



′

v =

4/5 0    0 2/5    1/10

0 2/5   0  0 0 1/10

v +



 

下面是这个方程的前几次迭代。我们也只从传送集中的页面开始搜索。虽然初始分布对极限没有影响，但它可以帮助计算更快地收敛。

 0/2  2/10  42/150  62/250   54/210   1/2  3/10  41/150  71/250    59/210





 



 







 0/2  2/10  26/150  46/250    1/2 3/10 41/150 71/250 59/210 38/210

注意，由于冲浪者集中在B和D，这些节点获得的PageRank比示例5.2中更高。在这个例子中，A是PageRank最高的节点。2

5.3.3 使用主题敏感网页排名

为了将对主题敏感的PageRank集成到搜索引擎中，我们必须:

1.决定我们将为哪些主题创建专门的PageRank向量器。

2.为每个主题选择一个传送集，并使用该集合计算该主题的主题敏感PageRank向量。

168 第五章。链接分析

3.找到一种方法来确定与特定搜索查询最相关的主题或主题集。

4.在对搜索查询的响应排序中，为该主题或多个主题使用PageRank向量。

我们有一种选择主题集的方法:使用打开目录中的顶级主题。其他方法是可能的，但是可能至少需要对某些页面进行人工分类。

第三步可能是最棘手的，已经提出了几种方法。一些可能性:

(a)允许用户从菜单中选择主题。

(b)根据用户最近搜索的网页中出现的单词或用户最近发出的查询来推断主题。我们需要讨论一个人如何从一组单词变成一个主题，我们将在第5.3.4节中讨论

(c)根据用户的信息，例如他们的书签或他们在Facebook上声明的兴趣来推断主题。

5.3.4 从单词中推断话题

按主题对文档进行分类的问题已经研究了几十年，我们在这里不做过多的讨论。可以这样说，主题的特征是在有关该主题的文档中出现频率高得惊人的单词。例如，后卫和麻疹都不会经常出现在Web文档中。但是后卫将会比平均水平更多的出现在关于体育的页面上，麻疹将会比平均水平更多的出现在关于药物的页面上。

如果我们检查整个网络，或者一个很大的随机网络样本，我们可以得到每个单词的背景频率。假设我们接着访问一个已知的关于某个主题的页面的大样本，比如按打开目录分类在sports下的页面。检查单词在运动样本中的出现频率，并识别出运动样本中出现频率明显高于背景的单词。在做出这个判断时，我们必须小心避免在体育样本中出现频率相对较高的一些极其罕见的词。这个词可能是一个拼写错误，碰巧只出现在一个或几个体育版上。因此，我们可能想要为一个单词出现的次数设定一个下限，然后才能认为它是一个主题的特征。

一旦我们确定了大量单词，这些单词在sports sample中出现的频率要比在background中出现的频率高得多，并且我们对列表中的所有主题都进行了相同的操作，我们就可以检查其他页面并按主题对它们进行分类。这里有一个简单的方法。假设S1, S2,…，Sk是一组单词，它们被确定为每个主题的特征

5.4。链接的垃圾邮件 169

我们的列表。假设P是出现在给定页面P中的一组单词。计算P和每个Si之间的Jaccard相似性(回忆一下第3.1.1节)。将页面分类为与Jaccard相似度最高的主题。注意，所有Jaccard相似性可能都非常低，特别是当Si集的大小很小时。因此，选择适当大的集合Si是很重要的，以确保我们涵盖了集合所代表的主题的所有方面。

我们可以使用这个方法，或者许多变体，对用户最近检索到的页面进行分类。我们可以说，用户对这些页面中最多的主题感兴趣。或者，我们可以将对主题敏感的PageRank向量按照属于每个主题的页面的比例进行混合，从而构建一个反映用户当前兴趣的PageRank向量。我们还可以对用户当前已书签的页面使用相同的过程，或者将书签页面与最近查看的页面组合起来。

5.3.5 第5.3节的练习

练习5.3.1:计算图5.15中图的主题敏感PageRank，假设传送集为:

(一)。

(b) A和C。

5.4 链接的垃圾邮件

当很明显，PageRank和谷歌使用的其他技术使术语垃圾邮件无效时，垃圾邮件发送者转而使用旨在欺骗PageRank算法使其高估某些页面价值的方法。人为提高页面的PageRank的技术统称为链接垃圾邮件。在本节中，我们将首先研究垃圾邮件制造者如何创建链接垃圾邮件，然后介绍几种降低这些垃圾邮件技术效率的方法，包括TrustRank和垃圾邮件质量的度量。

5.4.1之前 垃圾邮件场的体系结构

旨在增加某个或多个页面的PageRank的页面集合称为垃圾邮件场。图5.16显示了垃圾邮件场的最简单形式。从滥发讯息者的角度看，互联网可分为三个部分:

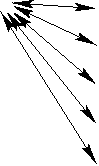
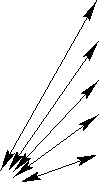
1.无法访问的页面:垃圾邮件发送者无法影响的页面。大部分Web都在这一部分。

2.可访问页面:虽然不受垃圾邮件发送者控制，但可能受到垃圾邮件发送者影响的页面。

170 第五章。链接分析

3.拥有页面:垃圾邮件发送者拥有和控制的页面。

目标页面



访问页面

访问的页面

自己的

页面

图5.16:从垃圾链接者的角度来看Web

垃圾邮件场由垃圾邮件发送者自己的页面(如右图所示)和从可访问页面到垃圾邮件发送者页面的一些链接组成。如果没有来自外部的一些链接，垃圾邮件场将是无用的，因为它甚至不会被一个典型的搜索引擎爬行。

关于可访问页面，一个人可以在不拥有它的情况下修改一个页面，这似乎令人惊讶。然而，今天有许多网站，如博客或报纸，邀请其他人在网站上发表评论。为了让尽可能多的PageRank从外部流向他自己的页面，垃圾邮件发送者发布了许多评论，比如“我同意”。请查看我的文章www.mySpamFarm.com。”

在垃圾邮件场中，有一个页面t，即目标页面，垃圾邮件发送者试图在其中放置尽可能多的PageRank。有大量的支持页面,积累部分的平均分配到所有页面的网页排名(分数1−β的PageRank repre-sents冲浪者会随机页面)。支持页面也在一定程度上防止了t的PageRank丢失，因为每轮都会有一些被征税。注意，t有到每个支持页面的链接，并且每个支持页面只链接到t。

5.4。链接的垃圾邮件 171

5.4.2 垃圾邮件场的分析

假设PageRank是用税收计算参数β,通常约为0.85。,β是一个页面的一部分的PageRank运输都需要燃烧到下一轮的继任者。假设Web上共有n个页面，其中一些为图5.16所示的形式的垃圾邮件场，目标页面t和m支持页面。设x为可访问页面贡献的PageRank。x是总和,所有p t链接访问页面,PageRank的p乘以β,除以的继任者p。最后,让y t的未知的PageRank。我们将求解y。

首先，每个支持页面的PageRank为

βy / m +(1−β)/ n

第一项代表的贡献从t, t的PageRank y是缴税,所以只有βy t分布的继任者。该PageRank平均分配给m个支持页面。第二项是支持页面的份额比例1−β的PageRank划分同样在所有网络上的网页。

现在，让我们计算目标页面t的PageRank y，它的PageRank有三个来源:

1.假设x来自外部。

2.PageRank的β倍每个支持页面;也就是说,ββy / m +(1−β)/ n

3.(1−β)/ n的比例分数1−β的PageRank属于t。这个数量是微不足道的,并将简化分析。

因此，从上面的(1)和(2)，我们可以写

y = x +βm m + 1−β= x +β2y +β(1−β)n我们可能解决上述方程y,屈服

βy



n

m



y = 1−β2 + c n

x



m



其中c =β(1−β)/(1−β2)=β/(1 +β)。

例5.11:如果我们选择β= 0.85,然后1 /(1−β2)= 3.6,和c =β/(1 +β)= 0.46。也就是说，该结构将外部PageRank的贡献放大了360%，并且获得了一个PageRank的数量，占代表垃圾邮件场中Web m/n部分的百分比m/n的46%。2

172 第五章。链接分析

5.4.3 打击垃圾邮件链接

对于搜索引擎来说，检测和消除链接垃圾邮件已经变得非常重要，就像在过去十年中消除术语垃圾邮件一样。有两种方法可以链接垃圾邮件。一种是寻找结构，如图5.16中的垃圾邮件场，其中一个页面链接到非常多的页面，每个页面都链接回该页面。搜索引擎肯定会搜索这样的结构并从索引中删除这些页面。这导致垃圾邮件发送者开发不同的结构，这些结构在本质上具有与为一个或多个目标页面捕获PageRank相同的效果。图5.16的变化本质上没有结束，所以垃圾邮件制造者和搜索引擎之间的战争很可能会持续很长一段时间。

然而，还有另一种方法可以消除链接垃圾邮件，它不依赖于定位垃圾邮件场。相反，搜索引擎可以修改它对PageRank的定义，自动降低链接垃圾页面的排名。我们将考虑两种不同的公式:

1.TrustRank是对主题敏感的网页排名的一种变体，旨在降低垃圾网页的得分。

2.垃圾邮件量，一种识别可能是垃圾邮件的页面的计算方法，它允许搜索引擎删除这些页面或大幅降低它们的PageRank。

5.4.4 TrustRank

TrustRank是对主题敏感的PageRank，其中“主题”是一组被认为是可信的页面(而不是垃圾邮件)。该理论认为，虽然垃圾邮件页面可以很容易地链接到一个值得信任的页面，但一个值得信任的页面不太可能链接到一个垃圾邮件页面。如5.4.1节所述，borderline区域是一个有博客或其他机会让垃圾邮件发送者创建链接的站点。这些页面不能被认为是值得信赖的，即使它们自己的内容是高度可靠的，就像一家声誉良好的报纸允许读者发表评论一样。

为了实现TrustRank，我们需要开发一组合适的可信任页面。已经尝试了两种方法:

1.让人类检查一组页面，并确定其中哪些值得信任。例如，我们可能会选择PageRank最高的页面来检查，因为理论上，虽然链接垃圾邮件可以将页面的排名从底部提升到包的中间，但基本上不可能让垃圾邮件页面的PageRank接近列表的顶部。

2.选择一个成员受控制的域，假设垃圾邮件发送者很难将其页面放入这些域。例如，我们可以选择.edu域，因为大学页面不太可能是垃圾邮件场。我们也可以选择。mil或。gov。然而,问题

5.4。链接的垃圾邮件 173

有了这些特定的选择，它们几乎完全是美国网站。为了得到一个良好的值得信赖的网页分布，我们应该包括类似的网站从国外，如ac.il，或教育。sg。

今天的搜索引擎很可能例行公事地执行第二种策略，所以我们所说的PageRank实际上是TrustRank的一种形式。

5.4.5 垃圾邮件的质量

垃圾邮件数量背后的想法是，我们衡量它的PageRank中来自垃圾邮件的部分。我们通过计算普通的PageRank和基于一些可信任页面的传送集的TrustRank来做到这一点。假设页面p有PageRank r和TrustRank t，则p的垃圾邮件量为(r - t)/r。负的或小的正的垃圾邮件质量意味着p可能不是垃圾邮件页面，而接近1的垃圾邮件质量表明该页面可能是垃圾邮件。有可能从搜索引擎使用的Web页面索引中删除具有高垃圾邮件量的页面，从而消除大量链接垃圾邮件，而不必确定垃圾邮件农民使用的特定结构。

例5.12:让我们分别考虑例5.2和例5.10中对图5.1所计算的PageRank和主题敏感页级。在后一种情况下，传送集是节点B和D，所以让我们假设这些是可信页面。图5.17列出了四个节点的PageRank、TrustRank和spam质量。

节点 PageRank

A

3/9

B 2/9 C 2/9 D

2/9

TrustRank

54/210

59/210 38/210 59/210

垃圾邮件质量

0.229

-0.264 0.186 -0.264

图5.17:计算垃圾邮件的数量

在这个简单的示例中，唯一的结论是节点B和D具有负的垃圾邮件质量，因此不是垃圾邮件，而节点B和D是预先确定为垃圾邮件的。另外两个节点A和C的垃圾邮件数量都为正值，因为它们的pagerank高于trustrank。例如，计算A的垃圾邮件量，取差3/9−54/210= 8/105，再除以PageRank 3/9，得到8/35，即约0.229。然而，它们的垃圾邮件质量仍然接近于0而不是1，所以它们很可能不是垃圾邮件。2

5.4.6 第5.4节的练习

练习5.4.1:在5.4.2节中，我们分析了图5.16中的垃圾邮件场，其中每个支持页面都链接回目标页面。对a重复分析

174 第五章。链接分析

其中:

(a)每个支持页面都链接到其本身，而不是目标页面。

(b)每个支持页面都没有链接。

(c)每个支持页面都链接到其本身和目标页面。

练习5.4.2:对于图5.1的原始Web图，假设只有B是可信页面:

(a)计算每一页的TrustRank。

(b)计算每页的滥发讯息量。

!练习5.4.3:假设两个垃圾邮件处理员同意链接他们的垃圾邮件处理场。如何链接这些页面，以尽可能地增加每个垃圾邮件场的目标页面的PageRank ?链接垃圾邮件场有什么好处吗?

5.5 中心和有关部门

在PageRank首次实施后不久，一个名为“中心和当局”的想法被提出。计算中心和权限的算法与PageRank的计算有一定的相似之处，因为它还涉及到一个固定点的迭代计算，涉及重复的矩阵-向量多次迭代。然而，这两种思想之间也存在着显著的差异，两者都不能替代。

这hubs-and-authorities算法,有时被称为(hyperlink-induced主题搜索),本来是不作为预处理步骤处理搜索查询之前,网页排名,但作为一个步骤要做搜索查询的处理,只排名查询的反应。但是，我们将把它描述为一种分析整个Web或搜索引擎爬行的部分的技术。我们有理由相信，Ask搜索引擎实际上使用了类似的方法。

5.5.1 热门歌曲背后的直觉

PageRank假定页面的重要性是一维的，而HITS认为重要的页面有两种重要性。

1.某些页面很有价值，因为它们提供了关于某个主题的信息。这些页面被称为权威。

2.其他页面之所以有价值，并不是因为它们提供了关于某个主题的信息，而是因为它们告诉您在哪里可以找到关于该主题的信息。这些页面称为中心。

5.5。中心和有关部门 175

例5.13:大学里一个典型的系会维护一个网页，列出系所提供的所有课程，每个课程都有一个网页的链接，说明课程内容——讲师、课文、课程大纲等等。如果你想了解某门课程，你需要那门课程的页面;系里的课程列表不行。课程页面是该课程的权威。然而，如果你想知道部门提供什么课程，搜索每个课程的页面是没有帮助的;您首先需要包含课程列表的页面。这一页是课程信息的集散地。2

正如PageRank使用递归定义的重要性”很重要,如果重要的页面链接页面,“点击使用相互递归两个概念的定义:“一个页面是一个很好的中心如果它链接好的部门,和一个页面是一个不错的权威,如果与良好的中心。”

5.5.2 将傲慢和权威正式化

为了使上述直观概念形式化，我们将为每个Web页面分配两个分数。一个分数代表一个页面的大小——也就是说，它是一个好的中心的程度，第二个分数代表这个页面是一个好的权威的程度。假设页面被枚举，我们用向量h和a表示这些分数。h的第i个分量表示第i个页面的hubbiness, a的第i个分量表示同一页面的authority。

一个页面的重要性是由页面的后继者来划分的，用Web的转换矩阵来表示，描述hubbiness和authority计算的一般方法是添加后继者的authority来估计hubbiness，添加前人的hubbiness来估计authority。如果这就是我们所做的一切，那么hubbiness和authority值通常会增长到超出界限。因此，我们通常缩放向量h和a的值，使最大的分量为1。另一种方法是缩放，使组件的和为1。

为了形式化地描述h和a的迭代计算，我们使用Web的链接矩阵L。如果我们有n个页面，那么L就是一个n×n矩阵，Lij = 1

如果有一个链接从页面i到页面j，如果没有，Lij = 0。我们还需要LT, l的转置，也就是说，如果有从j页到i页的链接，Lij = 1，否则Lij = 0。注意，它类似于我们在PageRank中使用的矩阵M，但是当它有1时，M有一个分数- 1除以由该列表示的页面外链接的数量。

T

T

例5.14:对于一个正在运行的示例，我们将使用图5.4的Web，这里我们将其复制为图5.18。一个重要的观察是，死角或蜘蛛陷阱并不会阻止命中迭代收敛到一个有意义的向量对。因此，我们可以直接处理图5.18，而不需要对图进行“征税”或修改。链接矩阵L及其转置如图5.19所示。2

176 第五章。链接分析

*A* *B*

*C* *D*

E

图5.18:HITS示例使用的示例数据

 0 1 1 1  1 0 0 1

L =  0 0 0 0





 0 1 1 0 0 0 0 0

0   0

0  LT =  1 0   1







1







1

0 0

1 0 0 0 0 1 0 0 1 1 0 0 0 1 0

0  0 

0  0







0

图5.19:图5.18网络的链接矩阵及其转置

页面的hubbiness是成正比的总和所表达的权威的继任者方程h =λLa,λ是一个未知常数代表所需的比例因子。同样的,这一事实的权威页面hubbinesses之和成正比的前任=μLTh表示,μ是另一个比例常数。这些方程使得我们可以独立计算出hubbiness和authority，将一个方程代入另一个方程，得到:

•h =λμLLTh。

•一个=λμLTLa。

然而，由于LLT和LTL不像L和LT那么稀疏，我们通常最好在真正的相互递归中计算h和a。也就是说，从所有1的向量h开始。

1.计算a = LTh，然后缩放，所以最大的分量是1。

2.接下来，计算h = La并再次缩放。

5.5。中心和有关部门 177

现在，我们有了一个新的h，可以重复步骤(1)和(2)，直到在某些迭代中，这两个向量的变化足够小，我们可以停止并接受当前值作为极限。

 1   1   1/2   3   1   1   2   1   3/2   1/2   1   2   1   1/2    1/6









 

 

 

 





 

 





 1   2   1   2   2/3  1 1 1/2 0 0

h

 1/2   5/3   5/3 

 

 

 3/2  1/6

LTh LTh

  3/10

1

 

 

 

 1    9/10

1/10

a

a

 29/10   6/5    1/10

 

 

 2  0

La

La h

 1   12/29    1/29

 

 

 20/29  0

h

图5.20:HITS算法的前两次迭代

例5.15:让我们在图5.18的Web上执行HITS算法的前两次迭代。在图5.20中，我们看到计算出的向量序列。第一列是初始的h，都是1。在第二列中，我们通过计算LTh估计了页面的相对权威，从而给出了每个页面的前一列的hubbinesses的总和。第三列给出了a的第一个估计值，它是通过缩放第二列得到的;在本例中，我们将每个分量除以2，因为这是第二列中最大的值。

第四列是La。也就是说，我们通过对每一页的继承者的权威估计的总和来估计每一页的大小。然后，第五列缩放第四列。在这种情况下，我们除以3，因为这是第四列中最大的值。第6列到第9列重复了我们在解释第2列到第5列时所概述的过程，但是第5列给出了更好的hubbiness估计。

这个过程的极限可能不明显，但可以用一个简单的程序来计算。限制:

 1    0.3583

h =  0 









0.7165   0

0.2087  

1 =     0.7913













1

0

178 第五章。链接分析

这个结果是有道理的。首先，我们注意到E的hubbiness肯定是0，因为它没有指向任何地方。C的hubbiness只取决于E的authority，反之亦然，所以两者都为0也就不足为奇了。A是最大的中心，因为它连接到三个最大的中心B、C和D。同样，B和C也是最大的中心，因为它们由两个最大的中心A和D连接。

对于web大小的图，计算hub -and- authority方程解的惟一方法是迭代。然而，对于这个小例子，我们可以通过解方程来计算解。我们将使用方程h =λμLLTh。首先,图象





3 1 0 2 0

 1 2 0 0 0  LLT =  0 0 1 0 0 









  2 0 0 2 0 0 0 0 0 0

让ν= 1 /(λμ)和h为节点的组件通过E是一个E,分别。然后就可以写出h的方程

νa = 3 a + b + 2 dνc = c

νe = 0

νb = a + 2 bνd = 2 + 2 d

方程告诉我们b = /(ν−2)和方程告诉我们d = 2 /(ν−2)。如果我们用这些表达式代替b和d的方程,我们得到νa = 3 + 5 /(ν−2)。从这个方程,因为双方是一个因素,我们只剩下一个二次方程ν,简化了ν2−5ν+ 1 = 0。积极的根是ν= (5 + 21)/ 2 = 4.791。现在我们知道ν既不是0或1,c和e的方程告诉我们立即c = e = 0。

√



最后，如果我们知道a是h的最大的分量，令a = 1，我们得到b = 0.3583 d = 0.7165。随着c = e = 0，这些值给出了h的极限值，a的值可以通过h乘以LT并缩放得到。2

5.5.3 第5.5节的练习

练习5.5.1:计算原始Web图5.1中每个节点的hubbiness和authority。

!练习5.5.2:假设我们的图是由n个节点组成的链，如图5.9所示。计算中心和权威向量，作为n的函数。

5.6 第五章小结

✦术语垃圾邮件:早期搜索引擎无法提供相关的结果,因为他们容易受到任期垃圾邮件——词的引入网页歪曲的页面。

5.6。第五章小结 179

✦谷歌解决方案项垃圾邮件:谷歌被两种技术能够抵消项垃圾邮件。首先是PageRank算法，用于确定Web上页面的相对重要性。第二种策略是相信其他页面对给定页面的描述，而不是只相信页面对自己的描述。

✦PageRank: PageRank算法,分配一个实数,叫做PageRank,每个页面。页面的PageRank衡量的是页面有多重要，或者它对搜索查询的响应有多好。在最简单的形式中，PageRank是递归方程的一个解“如果重要的页面链接到某个页面，那么这个页面就是重要的”。

✦转移矩阵网络的:我们代表链接在Web的一个矩阵i行和i列代表第i个页面的Web。如果从第j页到第i页有一个或多个链接，那么第i行和第j列中的条目是1/k，其中k是链接到第j页的页面数。转换矩阵的其他元素是0。

✦计算PageRank强连通网络图:强连通网络图(那些任何节点可以达到任何其他节点),PageRank是过渡的主要特征向量矩阵。我们可以通过从任何非零向量开始计算PageRank，并反复将当前向量乘以转换矩阵，得到更好的估计。在大约50次迭代之后，估计会非常接近极限，这就是真正的PageRank。

✦随机冲浪者模型:网页排名的计算模拟的行为可以被认为是许多随机冲浪者,每个从一个随机的页面,在任何一步行动,随机,当前页面的页面的链接。冲浪者在给定页面的极限概率是该页面的PageRank。人们的直觉是，人们倾向于创建到他们认为有用的页面的链接，所以随机的冲浪者往往在一个有用的页面上。

✦死胡同:死胡同是没有链接的网页。死胡同的存在将导致迭代计算中部分或全部页面的PageRank趋于0，包括非死胡同的页面。在进行PageRank计算之前，我们可以通过递归删除没有弧的节点来消除所有死角。请注意，删除一个节点会导致只链接到它的另一个节点成为死胡同，因此该过程必须是递归的。

从技术上讲，这种方法工作的条件不仅仅是“强连接”。“然而，其他必要条件肯定会被任何非人工构建的大型强连接网络组件所满足。

180 第五章。链接分析

✦蜘蛛陷阱:蜘蛛陷阱是一组节点,当他们互相链接,没有链接到其他节点。在PageRank的迭代计算中，spider trap的存在会导致在该节点集中捕获所有PageRank。

✦税收计划:计数器的影响蜘蛛陷阱(死角,如果我们不排除),网页排名通常是计算的方式修改简单的迭代矩阵乘法的过渡。选择一个参数β,通常约为0.85。给出一个估计的PageRank,下一个估计是乘以计算的β倍转移矩阵的估计,然后添加(1−β)/ n的估计为每一个页面,其中n是页面的总数。

✦税收和随机冲浪者:PageRank使用taxa-tion参数β的计算可以被认为是给每个随机冲浪者prob-ability 1−β离开网络,并引入随机冲浪者的整个网络。

✦过渡矩阵的有效表示:由于过渡矩阵非常稀疏(几乎所有的条目都是0),它可以节省时间和空间来表示,其非零项清单。然而，非零项除了具有稀疏性外，还有一个特殊的性质:它们在任何给定列中都是相同的;每个非零项的值是该列中非零项数的倒数。因此，首选的表示方法是逐列表示，其中列的表示是非零项的数量，然后是这些项所在行的列表。

✦非常大规模的矩阵向量乘法:Web-sized图表,它可能不是可行的整个网页排名估计向量存储在一台机器的内存。因此，我们可以将向量分解成k段，并将转换矩阵分解成k2平方，称为块，将每个平方分配给一台机器。每个向量段都被发送到k台机器上，因此复制这个向量的额外成本很小。

✦代表过渡矩阵块:当我们过渡矩阵划分为方形块,列分为k段。要表示列的段，如果该段中没有非零项，则不需要任何东西。但是，如果有一个或多个非零项，那么我们需要用列中非零项的总数表示列的段(这样我们就可以知道非零项的值是多少)，然后是具有非零项的行列表。

✦主题敏感网页级别:如果我们知道cer-tain查询感兴趣的话题,然后是有意义的偏见PageRank的页面主题。要计算这种形式的PageRank，我们确定一组已知与该主题相关的页面，并将其用作“传送集”

5.6。第五章小结 181

PageRank计算被修改，因此只有传送集中的页面被分配了一部分税，而不是在Web上的所有页面之间分配税。

✦创建传送集:主题敏感网页排名工作,我们需要确定的页面很可能是关于一个给定的话题。一种方法是从open directory (DMOZ)与该主题标识的页面开始。另一种方法是识别已知与主题相关的单词，并为teleport集选择出现此类单词数量异常高的页面。

✦垃圾链接:傻瓜PageRank算法,肆无忌惮的演员创造了垃圾邮件的农场。这些页面集合的目的是将高PageRank集中在特定的目标页面上。

✦垃圾农场的结构:通常,一个垃圾邮件农场由一个目标页面和很多支持页面。目标页面链接到所有支持页面，而支持页面仅链接到目标页面。此外，创建来自垃圾邮件场之外的一些链接也是非常重要的。例如，垃圾邮件发送者可能通过在其他人的博客或讨论组中写评论来引入指向其目标页面的链接。

✦TrustRank:改善垃圾链接的效果的一种方法是计算一个主题敏感PageRank称为TrustRank,可信的传送是一个集合页面。例如，大学的主页可以作为可信集，这种技术避免了在PageRank计算中与垃圾邮件场中的大量支持页面共享税收，从而优先降低了它们的PageRank。

✦垃圾质量:识别垃圾邮件的农场,我们可以计算conven-tional PageRank和TrustRank所有页面。那些TrustRank比PageRank低得多的页面很可能是垃圾邮件的一部分。

✦中心和部门:尽管PageRank给一维视图页面的重要性,一个算法称为点击试图测量两种不同方面的重要性。权威就是那些包含有价值信息的页面。集线器是一些页面，虽然它们本身不包含信息，但是它们链接到可以找到信息的地方。

✦命中算法的递归公式:计算中心和政府成绩的页面取决于解决递归equa-tions:“许多政府中心的链接,和一个权威与许多中心。“这些方程的解本质上是一个由iter化的矩阵-向量乘法，就像PageRank一样。但是，死角的存在或蜘蛛陷阱的存在并不影响对

182 第五章。链接分析

以PageRank的方式计算公式，所以不需要征税方案。

5.7 第5章的参考资料

PageRank算法首先用[1]表示。在[2]中描述了我们用来证明死角和蜘蛛陷阱存在的蛛网结构实验。执行PageRank迭代的块条带方法取自[5]。

主题敏感的PageRank取自[6]。TrustRank在[4]中进行了描述，垃圾邮件量的概念来源于[3]。

hit (hub和authority)的想法在[7]中进行了描述。

1.“大型超文本网络搜索引擎之剖析”，国立台湾科技大学资讯工程学研究所硕士论文。万维网会议，第107-117页，1998。

2.“网路上的图表结构”，中华民国资讯科技大学资讯工程学研究所硕士论文，国立台湾科技大学资讯工程学研究所硕士论文，1999年。

3.“基于质量估计的链接垃圾邮件检测”，第32卷第1期。《大型数据库研究》，439-450页，2006。

4.“用trustrank来对抗垃圾链接”，国立台湾科技大学资讯工程学研究所硕士论文。《大型数据库研究》，第576 - 587页，2004年。

5.胡伟华，“有效计算PageRank”，国立台湾科技大学资讯工程学研究所硕士论文，1999年9月。可作为

http://infolab.stanford.edu/~taherh/papers/efficient-pr.pdf

6.主题敏感的PageRank，第11国际版。万维网会议，517-526页，2002

7.“超连结环境中的权威资料来源”，中华民国资讯工程学研究所硕士论文，1999年。

第六章

频繁项集

我们将在本章转向描述数据的主要技术家族之一:频繁项集的发现。这个问题通常被看作是“关联规则”的发现，尽管后者是一个更复杂的数据字符化，其发现从根本上依赖于频繁项集的发现。

首先，我们介绍数据的“市场篮子”模型，它本质上是两种元素之间的许多关系，称为“项目”和“篮子”，但对数据的形状有一些假设。频繁项集的问题是寻找出现在(与)许多相同的篮子中的项集。

发现频繁项集的问题与第三章讨论的相似度搜索不同。这里我们感兴趣的是篮子的绝对数量，其中包含一组特定的项目。在第三章中，我们想要的是那些篮子中有很大一部分是相同的，即使篮子的绝对数量很小。

这种差异导致了一种新的算法用于寻找频繁项集。我们从a - priori算法开始，它的工作原理是通过首先查看较小的集合，并认识到一个较大的集合不可能是频繁的，除非它的所有子集都是。然后，我们考虑对基本的先天概念的各种改进，集中于强调可用内存的非常大的数据集。

接下来，我们考虑工作速度更快但不能保证找到所有频繁项集的近似算法。在这类算法中还有利用并行性的算法，包括我们可以通过map-reduce公式获得的并行性。最后，我们简要讨论了如何在数据流中查找频繁项集。

183

184 第六章。频繁项集

6.1 市场购物篮模型

数据的市场篮子模型用于描述两种对象之间的一种常见的多-多关系形式。一方面，我们有商品，另一方面，我们有篮子，有时被称为“交易”。“每个篮子由一组物品组成(一个物品集)，通常我们假设篮子里的物品数量很小——远远小于物品的总数。篮子的数量通常被认为是非常大的，比能装进内存的要大。假设数据在一个由篮子序列组成的文件中表示。在2.1节描述的分布式文件系统中，篮子是文件的对象，每个篮子的类型都是“项目集”。

但是 频繁项集的定义

直观地说，出现在许多篮子中的一组项目是“频繁的”。“正式地说，我们假设有一个数字s，叫做支持阈值。如果I是一组项目，那么对I的支持就是I是其中一个子集的篮数。如果它的支持是s或更多，我们说I是频繁的。

例6.1:图6.1中是单词集。每一组是一个篮子，单词是项目。我们用谷歌搜索cat dog，并从排名最高的网页上截取片段，就得到了这些图片。如果一个单词在一个篮子里出现两次，不要担心，因为篮子是一组，并且原则上项目只能出现一次。同时,忽略大小写。

1. {Cat, and, dog, bites}

2. {Yahoo, news, claims, a, cat, mated, with, a, dog, and, produced, viable, oﬀspring}

3. {Cat, killer, likely, is, a, big, dog}

4. {Professional, free, advice, on, dog, training, puppy, training}

5. {Cat, and, kitten, training, and, behavior}

6. {Dog, &, Cat, provides, dog, training, in, Eugene, Oregon}

7. {“Dog, and, cat”, is, a, slang, term, used, by, police, oﬃcers, for, a, male– female, relationship}

8. {Shop, for, your, show, dog, grooming, and, pet, supplies}

图6.1:这里有八个篮子，每个篮子由单词组成

由于空集是任意集的子集，所以对∅8的支持为8。但是，我们一般不关心空集，因为它告诉我们

6.1。市场购物篮模型 185

什么都没有。

在单例集合中，显然{cat}和{dog}非常频繁。除了篮子(5)，所有篮子里都有“Dog”，所以它的支撑力是7，而除了篮子(4)和(8)，所有篮子里都有“cat”，所以它的支撑力是6。“和”这个词也很常见;它出现在(1)(2)(5)(7)(8)中，所以它的支持度是5。单词“a”和“training”出现在三组单词中，而“for”和“is”分别出现在两组单词中。没有其他单词出现超过一次。

假设我们将阈值设为s = 3。然后有5个频繁的单例项集:{dog}、{cat}、{and}、{a}和{training}。

现在，让我们看看双峰。除非集合中的两个项本身都是频繁的，否则一个双例不能是频繁的。因此，只有十种可能的频繁双峰。图6.2是一个表，指出哪些篮子包含哪些对偶子。

training a

dog 4, 6 2, 3, 7 cat 5, 6 2, 7 and 5 2, 3



a none

and cat

1, 2, 8 1, 2, 3, 6, 7 1, 2, 5

图6.2:双峰的出现

例如，从图6.2的表格中我们可以看出，doubleton {dog, training}只出现在basket(4)和basket(6)中，所以它的support是2，并不常见。如果s = 3，则有四个频繁的对偶子;他们是

{dog, a} {dog, cat}

{dog, and} {cat, and}

除了出现了5次的{dog, cat}之外，每一次都恰好出现3次。

接下来，让我们看看是否存在频繁的三元组。为了成为一个频繁的三元组，集合中的每对元素必须是一个频繁的二元组。例如，{dog, a, and}不能是一个频繁项集，因为如果它是，那么{a, and}肯定是频繁项集，但它不是。三元组(dog、cat和)可能是频繁的，因为它的每个双子集都是频繁的。不幸的是，这三个单词只同时出现在购物篮(1)和(2)中，因此实际上没有频繁出现的三元组。如果没有频繁的三元组，那么肯定就没有频繁的四元组或更大的集合。2

6.1.2 频繁项集的应用

市场篮子模型的最初应用是分析真实的市场篮子。也就是说，超市和连锁商店会记录下每一个市场购物篮(实体购物车)的内容

186 第六章。频繁项集

网上零售和实体零售



我们在第3.1.3节中建议，在线零售商应该对商品使用相似性度量，以发现虽然许多顾客可能不会购买，但有很大一部分顾客是相同的商品对。然后，在线零售商可以向少数购买了这双鞋的顾客宣传其中一件。这种方法对于实体零售商来说没有任何意义，因为如果没有更少的人购买一件商品，那么在商品上做广告的成本效益就不高。因此，第三章的技术对于实体零售商来说并不常用。

相反，在线零售商不需要我们在本章讨论的分析，因为它的目的是搜索频繁出现的商品集。如果在线零售商被限制在频繁的商品集中，他们就会错过“长尾”中为每个客户单独选择广告的所有机会。

结帐。这里的“项目”是商店销售的不同产品，而“篮子”是一个市场篮子中的一组项目。一家大型连锁店可能销售10万种不同的商品，并收集有关数百万个市场篮子的数据。

通过查找频繁项集，零售商可以了解通常一起购买的商品。特别重要的是成对或更大的项目集，它们比单独购买的项目出现的频率要高得多。我们将在第6.1.3节中讨论这个问题的这一方面，但目前让我们只考虑搜索频繁项集。通过这个分析，我们会发现很多人会一起买面包和牛奶，但这没有什么意义，因为我们已经知道这些东西都是各自受欢迎的。我们可能会发现，很多人一起买热狗和芥末。对于喜欢热狗的人来说，这一点也不奇怪，但它为超市提供了一个做一些巧妙营销的机会。他们可以在热狗上做广告，提高芥末的价格。当人们来店里买便宜的热狗时，他们通常会记得他们需要芥末，然后也买了。要么他们不会注意到价格太高，要么他们认为不值得麻烦去其他地方买更便宜的芥末。

最著名的例子就是“尿布和啤酒”。人们很难想象这两件事会有关联，但通过数据分析，一家连锁商店发现，买尿布的人买啤酒的可能性非常高。理论是，如果你买尿布，你可能在家有一个孩子，如果你有一个孩子，那么你不太可能在酒吧喝酒;因此，你更有可能把啤酒带回家。同样的营销策略，我们建议热狗和芥末可以用于尿布和啤酒。

6.1。市场购物篮模型 187

然而，频率项集分析的应用并不局限于市场篮子。同样的模型可以用来挖掘许多其他类型的数据。一些例子:

1.相关概念:让项目成为单词，让篮子成为文档(例如，Web页面、博客、tweet)。篮子/文件包含文件中出现的项目/单词。如果我们查找在许多文档中同时出现的单词集，这些单词集将由最常见的单词(停止单词)来标识，就像我们在示例6.1中看到的那样。在那里，尽管目的是寻找谈论猫和狗的片段，但“and”和“a”这两个停止词在frequent项目集中很突出。然而，如果我们忽略所有最常见的单词，那么我们希望在频繁出现的单词对中找到一些表示共同概念的单词对。例如，我们期望一对像布拉德、安吉丽娜这样的情侣出现的频率会高得惊人。

2.剽窃:让项目成为文件，篮子成为句子。如果一个项目/文件在一个篮子/句子中，那么这个句子就在这个篮子/句子中。这种安排似乎是向后的，但这正是我们所需要的，我们应该记住，项目和篮子之间的关系是一种任意的多对多的关系。也就是说，“in”不必有其传统意义:“部分的”。“在这个应用程序中，我们查找出现在多个篮子中的成对项目。如果我们找到这样的一对，那么我们有两个文档，它们共享几个句子。在实践中，即使是一两个句子的共同点也是抄袭的一个很好的指标。

3.生物标记:将项目分为两类——生物标记，如基因或血液蛋白，以及疾病。每个篮子都是一组关于病人的数据:他们的基因组和血液化学分析，以及他们的病史。由一种疾病和一个或多个生物标记物组成的频繁项目集建议对该疾病进行测试。

6.1.3 关联规则

虽然本章的主题是从数据中提取频繁项集，但这些信息通常表示为if-then规则的集合，称为关联规则。关联规则的形式是I→j，其中I是一组项，j是一项。这个关联规则的含义是，如果I中的所有项都出现在某个篮子中，那么j也“可能”出现在该篮子中。

我们“可能”的概念形式化定义规则的信心我→j的比例支持我∪{j}的支持。也就是说,规则的信心的分数是篮子的我,也包含j。

188 第六章。频繁项集

例6.2:考虑图6.1中的篮子。规则(cat, dog)→and的置信度为3/5。“猫”和“狗”出现在五个篮子里:(1)、(2)、(3)、(6)和(7)。

另一个例子是，{cat}→小猫的置信度是1/6。单词“cat”出现在6个篮子里，(1)、(2)、(3)、(5)、(6)和(7)。“2

只要对规则左侧的支持相当大，那么只有信心是有用的。例如，我们不需要知道人们在买热狗的时候会不会特别喜欢买芥末，只要我们知道很多人会买热狗，而且很多人会同时买热狗和芥末。我们仍然可以使用第6.1.2节中讨论的热狗促销技巧。然而，如果关联规则反映了一个或多个项以某种方式影响了右边的项，那么它通常更有价值。

因此,我们定义一个关联规则的利益我→j的信心和分数的区别篮子包含j。也就是说,如果我对j没有影响,那么我们会认为篮子的一部分,包括我包含j是一模一样的分数篮子包含0 j。这样的规则有兴趣。然而，有趣的是，无论从非正式意义还是技术意义上讲，如果一个规则具有很高的兴趣，这意味着I在篮子中的出现会以某种方式导致j的出现，或者具有很高的负兴趣，这意味着I的出现会阻碍j的出现。

例子6.3:关于啤酒和尿布的故事实际上是在宣称asso-ciation rule(尿布)→beer有很高的兴趣。换句话说，买尿布的人中买啤酒的比例明显大于所有买啤酒的人的比例。负兴趣规则的一个例子是{coke}→pepsi。也就是说，买可口可乐的人不太可能也买百事可乐，即使有很大一部分人买百事可乐——人们通常喜欢其中之一，但不是两者都喜欢。同样，规则(百事可乐)→可口可乐可以预期为负权益。

对于一些数值计算，让我们回到图6.1的数据。规则{dog}→猫有信心5/7，因为“狗”出现在7个篮子里，其中5个篮子里有“猫”。然而，“猫”在8个篮子里出现了6个，所以我们预计7个篮子里有“狗”的篮子里75%也会有“猫”。因此，规则的兴趣是5/7 - 3/4 = - 0.036，本质上是0。规则{cat}→小猫有兴趣1/6 - 1/8 = 0.042。理由是六个篮子里有一个篮子里有“猫”，而“小猫”只出现在八个篮子里。这种兴趣虽然是积极的，但接近于0，因此表示关联规则不是很“有趣”。“2

6.1。市场购物篮模型 189

6.1.4 找到有信心的关联规则

识别有用的关联规则并不比查找频繁项集困难多少。我们将在本章的其余部分讨论寻找频繁项集的问题，但目前，假设有可能找到那些支持在或高于支持阈值s的频繁项集。

如果我们要寻找的关联规则I→j适用于篮子的合理比例，那么I的支持度必须相当高。在实际操作中，比如在实体店的营销中，“合理的高”通常只占购物篮的1%左右。我们还希望规则的置信度相当高，可能是50%，否则规则几乎没有实际效果。因此，集合I∪{j}也将得到相当高的支持。

假设我们已经找到了满足支持阈值的所有项集，并且为每个项集计算了精确的支持。在这些规则中，我们可以发现所有的关联规则都有很高的支持度和可信度。也就是说，如果J是一组n个被发现是频繁项的集合，那么涉及这组项的关联规则只有n种可能，即J - {J}→J对应J中的每个J。因此，它也是一个频繁项集，我们已经计算了J和J - {J}的支持度。它们的比值是J - {J}→J规则的置信度。

必须假设没有太多的频繁项集，因此没有太多高支持、高信任关联规则的候选项。原因是每一个发现都必须付诸行动。如果我们给商店经理一百万个关联规则来满足我们的支持和信心阈值，他们甚至不能阅读它们，更不用说执行它们了。同样地，如果我们生产出一百万个生物标志物候选项，我们就无法承担检测这些候选项所需的前期费用。因此，调整支持阈值是正常的，这样我们就不会得到太多的频繁项集。在后面的部分中，这个假设将导致关于查找频繁项集的算法效率的重要结果。

6.1.5 第6.1节的练习

练习6.1.1:假设有100个项目，编号为1到100，还有100个篮子，编号为1到100。项目i在篮子b中，当且仅当我除b没有余数。因此，项目1在所有的篮子里，项目2在所有50个偶数篮子里，以此类推。篮子12由{1、2、3、4、6、12}组成，因为这些都是除12的整数。回答以下问题:

(a)如果支持阈值为5，哪些项目是经常出现的?

!(b)如果支持阈值为5，哪些项目对是经常出现的?

!(c)所有篮子的大小之和是多少?

!练习6.1.2:对于练习6.1.1的项目篮数据，哪个篮子最大?

190 第六章。频繁项集

练习6.1.3:假设有100件物品，编号从1到100，还有100个篮子，编号也从1到100。当且仅当b除i没有余数时，项i在篮子b中。例如，篮子12由项目组成

{12、24、36,48岁,60岁,72年,84年,96年}

对这些数据重复练习6.1.1。

!练习6.1.4:这个问题涉及到一些数据，从这些数据中我们对频繁项集没有任何有趣的了解，因为没有相关的项集。假设项目编号为1到10，每个篮子由概率为1/i的项目i组成，每个决策独立于所有其他决策。也就是说，所有篮子都包含项目1，一半包含项目2，第三个包含项目3，以此类推。假设篮子的数量足够大，篮子的总体行为符合统计上的预期。假设支持阈值是篮子的1%。查找频繁项集。

练习6.1.5:对于练习6.1.1的数据，下列关联规则的置信度是多少?

(a){5,7}→2。

(b){2,3,4}→5。

练习6.1.6:对于练习6.1.3的数据，下列关联规则的置信度如何?

(a){24,60}→8。

(b){2,3,4}→5。

! !练习6.1.7:描述对下列市场一篮子数据100%有信心的所有关联规则:

(一)但是锻炼。

6.1.3 (b)运动。

!练习6.1.8:证明在练习6.1.4的数据中没有有趣的关联规则;即。，每个关联规则的兴趣值为0。

6.2 市场篮子和a -先验算法

现在，我们将开始讨论如何找到频繁项集或从中派生的信息，例如具有高度支持和信心的关联规则。本文将介绍对已有许多变体的明显算法(即众所周知的“先天算法”)的最初改进。接下来的两节将讨论一些进一步的改进。在讨论a -先验算法本身之前，我们先概述一下在搜索频繁项集时数据是如何存储和操作的假设。

6.2。市场篮子和a -先验算法 191

6.2.1 表示一篮子市场数据

如前所述，我们假设市场篮子数据存储在一个篮子一个篮子的文件中。可能，数据在一个分布式文件系统中，如第2.1节所示，并且篮是文件包含的对象。或者数据可以存储在一个传统的文件中，用字符代码表示篮子和它们的项目。

例6.4:我们可以想象这样一个文件的开头:

{23456、1001}{3,92145}{…

这里，字符{开始一个篮子，字符}结束它。篮子中的项目用整数表示，并用逗号分隔。因此，第一个篮子包含项目23、456和1001;第二个篮子包含项目3、18、92和145。2

可能是一台机器接收了整个文件。或者我们可以使用map-reduce或类似的工具在多个处理器之间分配工作，在这种情况下，每个处理器只接收文件的一部分。事实证明，组合并行处理器的工作来获得满足全局支持阈值的项集的精确集合是困难的，我们将在第6.4.4节中讨论这个问题。

我们还假设篮子文件的大小足够大，不适合放在主内存中。因此，任何算法的主要开销都是从磁盘读取购物篮所需的时间。一次磁盘块的篮子在主内存,我们可以扩展它,生成的所有子集大小k。因为我们模型的假设之一是一篮子的平均规模很小,产生的所有对内存应该花时间远小于篮子从磁盘读取的时间。例如，如果一个篮子中有20个项目，那么篮子中就有20 = 190对项目，这些项目可以很容易地在一对嵌套的For循环中生成。

2

随着我们想要生成的子集的大小变大，所需的时间也变长;实际上大约需要nk/k的时间!为一个有n个项目的篮子生成大小为k的所有子集。最终，这段时间支配了从磁盘传输数据所需的时间。然而:

1.通常，我们只需要小的频繁项集，因此k永远不会超过2或3。

2.当我们确实需要一个大k的项目集时，通常可以消除每个篮子中的许多项目，因为它们不能参与一个频繁的项目集，所以n的值随着k的增加而下降。

我们想得出的结论是，检查每个篮子的工作通常可以假定与文件的大小成正比。因此，我们可以通过读取数据文件的每个磁盘块的次数来测量频率项集算法的运行时间。

192 第六章。频繁项集

此外，我们讨论的所有算法都具有按顺序读取购物篮文件的特性。因此，算法可以通过它们通过购物篮文件的次数来描述，它们的运行时间与它们通过购物篮文件的次数乘以该文件的大小的乘积成正比。由于我们无法控制数据量，所以只有算法通过的次数才重要，在测量频项集算法的运行时间时，我们将重点关注算法的这一方面。

6.2.2 使用主存进行项集计数

然而，我们必须研究第二个与数据有关的问题。所有的频项集算法都要求我们在遍历数据时维护许多不同的计数。例如，我们可能需要计算每对物品在篮子中出现的次数。如果我们没有足够的内存来存储每个计数，那么向随机计数添加1很可能需要从磁盘加载一个页面。在这种情况下，如果我们确定要在主内存中找到每个计数，那么算法的运行速度将比这慢很多个数量级。结论是，我们不能计算任何不适合主存的东西。因此，每种算法都有它能处理多少项的限制。

例6.5:假设某个算法必须计算所有对项，并且有n个项。因此，我们需要空间来存储n个整数，或者大约n2/2个整数。如果整数占用4个字节，则需要2n2个字节。如果我们的机器有2g，或者231字节的主内存，那么我们需要n≤215，或者大约n < 33,000。2

2

以一种容易找到一对{i,j}的计数的方式存储n个计数并不简单。首先，我们没有假设任何关于项如何表示的内容。例如，它们可能是“面包”之类的字符串。“用从1到n的连续整数表示项目更节省空间，其中n是不同项目的数量。除非项目已经以这种方式表示，否则我们需要一个哈希表，将文件中出现的项目转换为整数。也就是说，每次我们在文件中看到一个条目时，我们都会对它进行散列。如果它已经在哈希表中，我们可以从它在表中的条目中获得它的整数代码。如果项目不在那里，我们将为它分配下一个可用的编号(从到目前为止所看到的不同项目数量的计数中)，并将项目及其代码输入到表中。

2

三角矩阵的方法

即使将项目编码为整数，我们仍然有一个问题，即必须在一个位置上计算一对{i,j}。例如，我们可以对这一对进行排序，使i < j，并且只在二维数组a中使用条目a[i,j]。一个更节省空间的方法是

6.2。市场篮子和a -先验算法 193

使用一维三角形数组。我们将一对{i,j}的计数存储在一个[k]中，其中1≤i < j≤n

i



k = (i - 1) n - 2 +j - i

这种布局的结果是对按字典顺序存储，即首先{1,2}，{1,3}，…，{1,n}，然后{2,3}，{2,4}，…n},{2,等等,到{n−2 n−1}, {n−2 n}, {n−1, n}。

三元组法

还有一种存储计数的方法可能更合适，这取决于实际出现在某个篮子中的可能对项的比例。我们可以存储数量为三元组(i, j c),这意味着对{i, j}的计数,和我< j c。是一个数据结构,如一个哈希表i和j作为搜索键,所以我们可以使用告诉如果有三重对于一个给定的i和j,如果是这样,迅速找到它。我们将这种方法称为存储计数的三元组方法。

与三角形矩阵不同，如果一对的计数为0，三元组方法不需要存储任何东西。另一方面，三元组方法要求我们为出现在某个篮子中的每对整数存储三个整数，而不是一个。此外，还有散列所需的空间

用于支持有效检索的表或其他数据结构。结论是，如果n个可能对中至少有1/3出现在某个篮子中，那么三角形矩阵将会更好，而如果明显少于1/3的可能对出现在篮子中，我们应该考虑使用三元组方法。

2

例6.6:假设有100,000个项目，每个项目包含10,000,000个篮子。那么三角矩阵法需要100000 = 5×109(近似)个整数计数。另一方面，所有篮子中对的总数是107对10 = 4.5×108。即使在每对物品只出现一次的极端情况下，也可能只有4.5×108对非零计数。如果我们使用三元组方法来存储计数，我们只需要3倍的整数，即1.35×109个整数。因此，在这种情况下，三元组方法肯定会比三角形矩阵占用更少的空间。

2

2

然而，即使篮子的数量是原来的10倍或100倍，如果篮子里的物品分布非常不均匀，这也是正常的，因此使用三元组方法可能会更好。也就是说，有些对会有很高的计数，而在一个或多个篮子中出现的不同对的数量将远远少于理论中此类对的最大数量。2

在这里，在整个章节中，我们将使用n ' = n2/2的近似值来表示较大的n。

2

194 第六章。频繁项集

6.2.3 单调性的项集

我们将讨论的算法的大部分有效性是由一个单一的观察驱动的，称为项目集的单调性:

•如果一组I项是频繁的，那么I的每个子集也是频繁的。

原因很简单。让J⊆。然后每个篮子里包含的所有项目我肯定包含所有项目的J .因此,J的数必须至少为伟大的作为我的计算,如果计数我至少是s,然后计数J至少年代。因为可能包含在一些篮子,缺少一个或多个元素的我−J,完全有可能,J的计数是我严格大于计数。

除了使a -先验算法有效外，单调性还为我们提供了一种压缩频繁项集信息的方法。如果我们有一个支持阈值s，那么如果没有频繁的超集，我们就说itemset是最大的。如果我们只列出最大项集，那么我们就知道一个最大项集的所有子集都是频繁的，并且没有不属于某个最大项集子集的集合可以是频繁的。

例6.7:让我们重新考虑例6.1的数据，支持阈值-old s = 3。我们发现有5个频繁的单身人士，他们的名字分别是“猫”、“狗”、“a”、“和”和“训练”。除了“训练”之外，这些都包含在一个频繁的双例中，因此一个最大的频繁项集是{training}。也有4个s = 3的频繁双峰，即

{dog, a} {dog, cat}

{dog, and} {cat, and}

由于不存在支持阈值3的频繁三元组，所以这些都是最大频繁项集，没有其他频繁项集。注意，我们可以从频繁的双子推断出像{dog}这样的单子是频繁的。2

6.2.4 计数对的专制

正如您可能已经注意到的，到目前为止，我们的讨论主要集中在数对的问题上。这样做有一个很好的理由:在实践中，确定频繁对需要最多的主内存。项目的数量虽然可能非常大，但很少会大到不能同时计算主内存中的所有单例集。

那么更大的集合呢——三元组、四元组等等?回想一下，为了使频率项集分析有意义，结果必须是少量的集，否则我们甚至不能全部读取它们，更不用说考虑它们的重要性了。因此，在实践中，支持阈值设置得足够高，以至于只有很少的情况下才会频繁设置支持阈值。单调性告诉我们，如果有一个频繁三重态，那么其中包含三个频繁对。当然，也有可能在非频繁三重中包含频繁对。因此，我们期望找到

6.2。市场篮子和a -先验算法 195

更频繁的配对比频繁的三元组多，更频繁的三元组比频繁的三元组多，以此类推。

如果不可能避免计算所有的三元组，那么这个参数将是不够的，因为三元组比成对多得多。避免计算许多三元组或更大的集合是先验算法和相关算法的工作，我们将看到，它们在这方面是有效的。因此，在接下来的内容中，我们将重点讨论计算频繁对的算法。

6.2.5 先验算法

现在，让我们只专注于寻找频繁出现的配对。如果我们有足够的内存来计算所有对，使用6.2.2节(三角形矩阵或三元组)中讨论的任何一种方法，那么一次性读取篮子的文件就很简单了。对于每个篮子，我们使用一个双循环来生成所有对。每次生成一对，我们就在它的计数上加1。最后，我们检查所有对，看看哪些计数超过支持阈值s;这是最常见的一对。

然而，如果在主存中有太多对项，无法全部计数，则此简单方法将失败。先验算法的目的是减少必须计算的对的数量，代价是执行两次数据传递，而不是一次。

先验的第一次传递

在第一次传递中，我们创建了两个表。如有必要，第一个表将项目名称转换为从1到n的整数，如第6.2.2节所述。另一个表是计数数组;第i个数组元素计算编号为i的项的出现次数。最初，所有项的计数为0。

当我们阅读篮子时，我们查看篮子中的每一项，并将其名称转换为整数。接下来，我们使用该整数索引到count数组中，并在其中找到的整数上添加1。

在先验传递之间

第一次通过后，我们检查物品的数量，以确定哪些是常用的assingletons。许多单身人士并不频繁，这可能会让人感到惊讶。但是请记住，我们设置的阈值足够高，因此不会得到太多的频繁集;一个典型的s是1%的篮子。如果我们想到自己去超市，我们肯定会买超过1%的东西:也许是牛奶、面包、可口可乐或百事可乐，等等。我们甚至可以相信1%的顾客会购买尿布，尽管我们可能不会这么做。然而，货架上的许多商品肯定不是1%的顾客买的:比如奶油凯撒沙拉酱。对于a -先验的第二遍，我们为频繁项创建一个从1到m的新编号。该表是一个索引为1到n的数组，以及条目

196 第六章。频繁项集

因为i要么是0，如果i不是频繁项，要么是1到m范围内唯一的整数，如果i是频繁项。我们将把这张表称为“经常项目表”。

先验的第二遍

在第二遍中，我们计算由两个频繁项组成的所有对。请回忆第6.2.3节，一对不能是频繁的，除非它的两个成员都是频繁的。因此，我们不会错过任何频繁的配对。如果我们用三角矩阵法计算，第二遍需要的空间是2m2，而不是2n2。注意，如果我们要使用合适大小的三角形矩阵，则需要对频繁项重新编号。在第一和第二遍中使用的完整的主内存结构如图6.3所示。

|  |  |  |  |
| --- | --- | --- | --- |
| 项目名称  整数 |  | 1 2  *n* | 项数 |
| 未使用的 | | | |

通过1

|  |  |  |  |
| --- | --- | --- | --- |
| 项目名称  整数 |  | 1 2  *n* | 最近  项目 |
|  | | | |
| 计数数据结构  的双 | | | |

通过2

图6.3:A-Priori算法两次遍历过程中主存使用示意图

还要注意，消除不经常使用的项目的好处被放大了;如果只有一半的项目是频繁的，我们需要四分之一的空间来计数。同样，如果我们使用三元组方法，我们只需要计算至少在一个篮子中出现的两对频繁项。

第二遍的机制如下。

1.对于每个篮子，查看频率项表，看看其中哪些项是频率项。

2.在双循环中，生成所有频繁对。

6.2。市场篮子和a -先验算法 197

3.对于每个频繁对，在用于存储计数的数据结构中向其计数添加一个。

最后，在第二遍结束时，检查计数的结构，以确定哪些对是频繁的。

6.2.6 所有频繁项集的先验

在不计算所有对的情况下查找频繁项对时使用的相同技术允许我们在不计算所有集的详细计数的情况下查找更大的频繁项集。在a -先验算法中，每个集合大小k取一次遍历。如果没有找到一定大小的频繁项集，那么单调性告诉我们不可能有更大的频繁项集，所以我们可以停止。

从一个尺寸k移动到下一个尺寸k + 1的模式可以归纳如下。对于每个尺寸k，有两组itemset:

1.Ck是大小为k的候选项集的集合——我们必须计算这些项集，以确定它们实际上是否频繁。

2.Lk是大小为k的真正频繁项集的集合。

从一个集合到下一个集合，从一个大小到下一个大小的移动模式如图6.4所示。

*C1*

过滤器

|  |  |  |
| --- | --- | --- |
| *L1* | 构造 | *C2* |
|  |  |

过滤器

|  |  |  |
| --- | --- | --- |
| *L2* | 构造 | *C3* |
|  |  |

过滤器

|  |  |
| --- | --- |
| *L3* | 构造 |
|  |

**。**

所有 经常项目项

频繁项对

频繁的对

图6.4:A-Priori算法在构造候选集和筛选之间交替进行，以找到真正频繁的候选集

我们从C1开始，C1是所有单例项集，即。，项目本身。也就是说，在我们检查数据之前，就我们所知，任何项目都可能是频繁的。第一个筛选步骤是计数所有项，那些计数至少为支持阈值s的项构成频繁项集L1。

候选对集合C2是候选对的集合，候选对的项都在L1中;也就是说，它们是经常出现的项目。注意，我们没有显式地构造C2。相反，我们使用C2的定义，通过测试它的两个成员是否都在L1中来测试C2中的成员关系。A-Priori算法的第二遍计算所有候选对，并确定哪些至少出现了s次。这些对形成L2，频繁对。

只要我们愿意，我们可以遵循这种模式。候选三元组的集合C3(隐式地)构造为三元组的集合，其中任意两个三元组在L2中是一对。概述了我们对频繁项集稀疏性的假设

198 第六章。频繁项集

在第6.2.4节中暗示不会有太多的频繁对，因此可以在主内存表中列出它们。同样，也不会有太多的候选三元组，所以这些都可以通过推广三元组方法来计算。也就是说，虽然三元组用于计数对，但是当我们想要计数三元组时，我们将使用三元组，它由三个项目代码和相关的计数组成。类似地，我们可以使用包含k + 1个组件的元组来计数大小为k的集合，最后一个是计数，第一个k是按顺序排列的项目代码。

为了找到L3，我们第三次遍历篮子文件。对于每个篮子，我们只需要查看L1中的那些项。从这些项中，我们可以检查每一对，并确定该对是否在L2中。篮子中任何一项如果没有出现在至少两对频繁项中(这两对频繁项都由篮子中的项目组成)，都不能成为篮子所包含的频繁项三的一部分。因此，我们对篮子中包含的三元组和C3中的候选三元组的搜索相当有限。发现的任何这样的三元组都要在它们的计数中添加1。

例6.8:假设我们的篮子由项目1到10组成。其中1至5项是经常出现的项目，经常出现的项目有:{1,2}，{2,3}，{3,4}和{4,5}。首先，我们删除不常见的项，只留下1到5个。然而，1和5只出现在itemset中的一个频繁对中，因此不能构成篮子中包含的一个频繁对。因此，我们必须考虑增加包含在{2,3,4}中的三元组的数量。当然，只有一个这样的三倍。然而，我们将不会在C3中找到它，因为{2,4}显然不常见。2

更大的频繁项集和候选项集的构造基本上以相同的方式进行，直到在某个阶段我们没有发现新的频繁项集并停止。那就是:

1.定义Ck为大小为k的所有项集，其中k - 1是Lk - 1中的项集。

2.通过遍历篮子找到Lk，并只计算Ck中大小为k的所有项集。那些至少有s计数的项集在Lk中。