中科院计算培训中心

Part 7大数据关联规则和相似项发现

频繁项集及相似项

- 杨文川

1）购物篮模型
2）Apriori算法
3）抄袭文档发现
4）近邻搜索的应用

频繁项集发现

- 频繁项集发现vs关联规则
  - 该问题常常被看成关联规则发现
  - 但关联规则主要是基于频繁项集发现，而实现的一种更复杂的数据刻画方式
- 频繁项集发现问题和相似性搜索不同
  - 频繁项集关注包含某个特定项集的，购物篮的绝对数目
  - 相似性搜索是寻找购物篮之间，具有较高重合度的项集，不管购物篮数的绝对数量是否很低

频繁项集发现- Apriori算法

- Apriori算法的基本思路是
  - 如果一个集合的子集不是频繁项集，那么该集合也不可能是频繁项集。
  - 基于这种思路，该算法可以通过检查小集合，而去掉大部分不合格的大集合。
- Apriori算法还有各种改进
  - 这些改进策略，集中关注那些给可用内存带来很大压力的，极大规模数据集。

购物篮模型

- 购物篮模型用于描述两类对象之间，一种常见的多对多关系。
  - 其中的一类对象是项(item,或商品)，另一类对象是购物篮或交易。
  - 每个购物篮由多个项组成的集合(项集)构成，
  - 通常都假设一个购物篮中项集总数目较小，相对于所有项的总数目而言要小得多。
  - 购物篮的数目通常假设很大，导致在内存中无法存放。
  - 整个数据由一个购物篮序列构成的文件来表示。

频繁项集的定义

- 在多个购物篮中出现的项集，称为<u>频繁项集</u>。
  - 如果I是一个项集
  - I的支持度Support是指包含I(子项集)的购物篮数目
  - 假定有个支持度阈值*s*。
  - 如果I的支持度不小于*s*，则I是频繁项集。

一个购物篮的例子

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}

单元素集合在购物篮中出现情况

- 单元素集合中，{cat}和{dog}非常频繁。
  - dog 支持度为7， cat支持度为6。
  - and的出现也很频繁，其支持度为5。
  - a和training各出现3次
  - 而for和is各出现2次。其他词的出现次数≤1次。
- 假定给出的支持度阈值为s =3
  - 有5个频繁的单元素集合
  - {dog}， {cat}， {and}， {a}， 和 {training}。

双元素集合在购物篮中出现情况

- 双元素集合中的两个元素都必须是频繁的，
  - 这样该集合才有可能是频繁的。
  - 所有可能的双元素频繁集合只有10个

	training	a	and	cat
dog	4, 6	2, 3, 7	1, 2, 8	1, 2, 3, 6, 7
cat	5, 6	2, 7	1, 2, 5	
and	5	2, 3		
a	none			

  - 在s =3的情况下，{dog， training} 其支持度为2，并非频繁项集。
  - 只有如下4个双元素集合是频繁的：{dog, a} {dog, and} {dog, cat} {cat, and}

中科院计算培训中心

三元素频繁项集是否存在？

- 三个元素组成的项集要成为频繁项集，必须其中任意两个元素组成的集合都是频繁的。
  - 例如，集合{dog, a, and}不可能是频繁项集，如果它是，那么必定有{a, and}是频繁项集，但是这个集合并不频繁。
  - {dog, cat, and}有可能是频繁项集，因为其任意两个元素组成的集合都是频繁项集。
  - 不过集合中的三个词只在购物篮(1)和(2)中一起出现，因此该集合实际上并不是频繁项集。
  - **如果不存在三元素频繁项集，那么肯定不会存在四元素或者更多元素组成的频繁项集。**

频繁项集的应用

- 购物篮模型最早应用源于真实购物篮的分析
  - 超市会记录每个客户购物篮(购物车)的内容。这里的项指的是商店出售的不同商品。
  - 购物篮指的是单个购物篮中所装的项集。一个大型的超市或许有10万个不同的项(商品)，每天产生的购物篮数据可能有几百万个
- 通过发现频繁项集，商家可以知道哪些商品通常会被顾客一起购买。
  - 商家尤其关注，那些共同购买频度，远高于各自独立购买频度的，项对(用做捆绑销售的商品对)

热狗和芥末的例子

- 很多喜欢热狗的人，会同时购买芥末。
  - 这个分析结果，能够为超市提供营销的机会。
- **可为热狗做促销广告，同时提高芥末价格**
  - 当人们到商店来购买便宜的热狗时，通常会想起来还需要购买芥末。
  - 他可能并没有注意到芥末的价格较高，
  - 也可能认为不值得为寻找便宜的芥末，而去另外一家商店

尿布和啤酒的关联故事

- 这类例子中最著名的一个。
  - 人们可能很难想到这样两个商品有关联，但是通过分析某个超市的数据发现，购买尿布的人非常可能会购买啤酒。
  - 推测其主要原因在于，如果某个人购买尿布，那么他很可能家里有个婴孩，但是如果有个婴孩，那么他就更可能带啤酒回家喝。
- 频繁项集分析的应用并不仅限于购物篮数据
  - 同样的模型，可用于挖掘很多其他类型的数据

关联规则

- 关联规则
  - 从数据中抽取频繁项集，抽取结果用if then形式的规则集合来表示，这些规则称为关联规则。
- 关联规则的形式为
  - I→ j，其中I是一个项集，j是一个项。
  - 该关联规则的意义是，如果I 中所有项出现在某个购物篮，那j有可能也出现在这一购物篮

规则的可信度定义

- 规则$I \rightarrow j$的可信度Confidence
  - 等于集合$I \cup \{j\}$的支持度与I的支持度的比值。
- 该规则的可信度
  - 等于所有包含I的购物篮中，同时包含j的购物篮的比例

考查购物篮

- 规则{cat, dog} → and 可信度为3/5
  - 词语cat和dog同时出现在5个购物篮中，而and出现在3个中，也就是5个购物篮的3/5 当中。
- 规则{cat} → kitten 的可信度为1/6
  - cat出现在6个购物篮中，其中仅有1个包含词kitten.
- 如果关联规则左部项集的支持度相当大，那么单独的可信度就会有用。
  - 例如，只要知道很多人购买热狗，且不少人同时购买热狗和芥末就行，并不一定需要知道人们在购买热狗时会有极大的可能性购买芥末。

兴趣度

- 关联规则I → j的兴趣度Interest
  - 定义为其可信度，及包含j的购物篮比率之间的差值
- 如果I对j没有任何影响，那么包含I的购物篮中包含j的比率，就应该等于所有购物篮中包含j的比率，即该规则的兴趣度为0。
  - 若一条规则的兴趣度很高，或是某个绝对值很大的负值，都十分令人关注。
  - 前者意味着某个购物篮中I的存在，会促进j的存在；而后者意味着I的存在，会抑制j的存在

啤酒和尿布规则有很高的兴趣度

- 啤酒和尿布的故事实际上是
  - 关联规则{diapers} → beer 具有很高的兴趣度
  - 也是说，购买尿布的人中，购买啤酒的比率，显著高于所有顾客中，购买啤酒的比率。
  - 一条负兴趣度值的规则是{coke} → pepsi。
    - 购买可口可乐的顾客，一般不会同时购买百事可乐，尽管在所有顾客中，购买百事可乐的比率不低
    - 但一般只购买这两者中的一种，而不会同时购买两者
  - 类似，规则{pepsi} → coke 的兴趣度预计也为负值

高可信度关联规则的发现

- 识别有用的关联规则
  - 假定已经可以找到那些，支持度不低于某个支持阈值s的频繁项集。
- 若希望寻找的关联规则$I \rightarrow j$，能够应用于很多购物篮，那么I的支持度一定要相当高
  - 对于传统超市的销售而言，大概相当于所有购物篮的1%左右。
- 也希望规则的可信度相当高，
  - 或许是50%，否则规则的实际用处不大。
  - 这样，集合$I \cup \{j\}$的支持度也相当高

购物篮及A-Priori算法

- 如何寻找频繁项集，或从频繁项集推出有用信息，比如具有高支持可信度的关联规则。
  - Apriori是一个最早对朴素算法，进行改进的频繁集发现算法。
  - 介绍一些对它的进一步改进算法和措施。
- 在介绍Apriori算法前
  - 将对频繁项搜索时，数据存储和处理方式，做一些介绍

购物篮数据的表示

编号	牛奶	果冻	啤酒	面包	花生酱
T_1	1	1	0	0	1
T_2	0	1	0	1	0
T_3	0	1	1	0	0
T_4	1	1	0	1	0
T_5	1	0	1	0	0
T_6	0	1	1	0	0
T_7	1	0	1	0	0
T_8	1	1	1	0	1
T_9	1	1	1	0	0

- 一个样本称为一个"**事务**"
- 每个事务由多个属性来确定，这里的属性就是"**项**"
- 多个项组成的集合称为"**项集**"

一个 Apriori 算法示例

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3rd scan

L_3

Itemset	sup
{B, C, E}	2

多个并行处理器上的任务组合

- 有可能一台机器接收了整个文件
  - 但更多可能，是用MapReduce或类似的工具，将整个任务分配到多个处理器中，其中每个处理器只接收文件的一部分。
  - 证据表明，通过将多个并行处理器上的任务组合，来获得满足全局支持度阈值的精确项集集合，是很难的

假定购物篮组成的文件太大

- 若文件在内存无法存放
  - 算法的主要时间开销，都集中在将购物篮从磁盘读入内存这个过程。
  - 一旦一个装满购物篮的磁盘块处于内存时，可以对它进行扩展，产生所有规模为k的子集。
  - 由于模型中的一个基本假设是，购物篮的平均规模很小，所以在内存中产生所有项对所花费的时间，会比购物篮的读入时间少很多。
  - 如果某购物篮中有20个项，则该购物篮中有
  - $\binom{20}{2}$ = 190个项对，可通过两层嵌套for循环来生成

- 通常情况下
  - (1) 往往只需较小的频繁项集，k远不会超过2或3;
  - (2)当确实需要一个更大的k的项集时，可去掉每个购物篮中，不太可能成为频繁项的那些项，保证k增长的同时n却下降。
- 结论是
  - 通常可以假设每个购物篮上的检查工作时间，正比于文件的大小。
  - 这样就可通过数据文件的每个磁盘块读取次数，来度量频繁项集算法的执行时间。

相似项发现

- 另外一个基本的数据挖掘问题是：
  - 从数据中获得相似项
- 将相似度问题表述为：
  - 寻找具有相对较大交集的集合问题

近邻搜索的应用

- Jaccard相似度
  - 一个特定的“相似度”概念，即通过计算交集的相对大小，来获得集合之间的相似度。
  - 具体应用，包括文本内容相似的文档查找，及协同过滤中相似顾客，和相似产品的查找。

文档的相似度

- 采用Jaccard相似度时，取得较好效果的应用
  - 在大语料库(Web或新闻语料)中，寻找文本内容相似的文档。
  - 这里主要侧重于字面上的相似，意义上的相似，需要通过其他技术来解决。
- 还有很多非常重要的应用
  - 如检查两篇文档之间，是否完全重复或近似重复

抄袭文档

- 抄袭文档的发现，可以考验文本相似度发现的能力。
  - 抄袭者可能会从其他文档中，将某些部分的文本据为己用
  - 也可能对某些词语，或者原始文本中的句序进行改变。
  - 尽管如此，最终的文档中仍然有50%，甚至更多的内容来自别人的原始文档。
  - 当然，一个复杂的抄袭文档，很难通过简单的字面比较来发现

镜像页面

- 重要或流行的Web站点，会在多个主机上，建立镜像以共享加载内容。
  - 这些镜像站点的页面十分相似，但也不是完全一样。
  - 例如，这些网页可能包含与其所在的特定主机相关的信息，或者包含对其他镜像网站的链接
- 另一个现象就是课程网站的互相套用。
  - 能够检测出这种类型的相似网页非常重要
  - 能避免在返回的第一页结果中，包含几乎相同的两个网页

同源新闻稿

- 通常一个记者会撰写一篇新闻稿，然后分发到各处
  - 比如通过美联社到多家报纸，然后每家报纸会在其Web网站发布该新闻稿。
  - 每家报纸会对新闻稿进行某种程度的修改。比如去掉某些段落或者加上自己的内容。
  - 在新闻稿周围会有报纸自己的徽标、广告或者指向自己Web站点的其他文章的链接等。
  - 但是每家报纸的核心内容，还是原始的新闻稿。

文栏Shingling

- 为了识别字面上相似的文档，将文档表示成集合的最有效方法，是构建文档中的短字符串集合。
  - 如果文档采用这样的集合表示，那么有相同句子，甚至短语的文档之间，将会拥有很多公共的集合元素
  - 即使两篇文档中的句序并不相同，也是如此。
  - 介绍一个最简单，最常用的Shingling方法，及一个有趣的变形

k-shingle

- 文档是一个字符串，文档k-shingle定义为其中任意长度为k的子串。
  - 每篇文档可表示成，文档中出现的k-shingle的集合
- 例：假设文档D为字符串abcdabd
  - 选择k=2，文档D中的所有2-shingle组成的集合是
  - {ab, bc, cd, da, bd}。
    - 注意：子串ab在文档中出现2次，但在集合中只算1次
  - shingle的一个变形，是将文档表示成包，而不是集合，这样每个shingle的出现次数也被考虑在内

shingle大小的选择

- 理论上，可以选择任意的常数作为k。
  - 但如果选择的k太小，那么可以推测大部分长度为k的字符串，会出现在大部分文档中。
  - 到底要选择多大的k，依赖于文档的典型长度，以及典型的字符表大小。
  - k应该选择得足够大，以保证任意给定的shingle，出现在任意文档中的概率较低
  - 如果文档集由邮件组成，那么选择k=5，应该比较合适
  - 对研究论文类的大文档，选择k=9则比较安全

基于词的shingle

- 对于新闻报道的近似重复检测来说，将shingle定义为一个停用词，加上后续的两个词，可形成一个有用的shingle集。
  - 在进行Web网页表示时，这种做法的优势在于，新闻文本比周边元素提供了更多的shingle集。
  - 上述做法在表示时，更偏向新闻文本中的shingle集

例子

- 一则广告新闻报道可能是
  - "A spokesperson for the Audi Corporation revealed today that studies have shown it is good for people to buy Audi products."
- 停用词标成红色，包含9个基于停用词，加上后续两个词，构建的shingle，前三个是:
  - *A spokesperson for*
  - *for the Audi*
  - *the Audi Corporation*
- 但含义相同的简文广告"Buy Audi."报道中，一个shingle都没有

保持相似度的集合摘要表示

- Shingle集合非常大
  - 即使将每个shingle都哈希为4个字节，一篇文档的shingle集合所需要空间，仍然大概是该文档所需空间的4倍。
  - 如果有数百万文档，很可能不能将这些文档的shingle集合都放入内存中。
- 可将大集合替换成小规模的“签名”表示。
  - 对于签名而言，所需要的重要特性，是能够仅仅通过比较两篇文档的签名集合，就可以估计实际shingle集合之间的Jaccard相似度。

与真实值的差异

- 5万字节文档的shingle，可能会映射为2万字节的哈希结果，然后替换成1000字节大小的签名集合
- 基于最终签名集合得到的原始文档，Jaccard相似度的估计值，与真实值的差异也就在几个百分点之内。

谢 谢