ISSN 1000-1239/CN 11-1777/TP
49(10): 2189-2197, 2012

基于滑动窗口的 Top-*K* 概率频繁项查询算法研究

王 爽[1,2] 王国仁[2]
[1](东北大学软件学院 沈阳 110819)
[2](东北大学信息科学与工程学院 沈阳 110819)
(wangsh@mail.neu.edu.cn)

Sliding Window Top-*K* Frequent Item Query on Uncertain Stream

Wang Shuang[1,2] and Wang Guoren[2]
[1](*Software College*, *Northeastern University*, *Shenyang* 110819)
[2](*School of Information Science and Engineering*, *Northeastern University*, *Shenyang* 110819)

Abstract Frequent items detection has play an important role in many applications, such as network monitor, network intrusion detection, association rules mining, and so on. While is has been studied intensively in the field of static uncertain data, frequent items detection is still novel in the emerging uncertain data stream field. In this paper, based on the sliding window model, an efficient algorithm (sTop*K*-UFI) is proposed to detect frequent items on uncertain data stream. In order to reduce the computation cost, the algorithm incrementally updates the exact top-*K* results upon each window slide and thus eliminates the need for complete recomputation from scratch. On the other hand, according to the current data in the sliding window, items to be frequent in the future are predicated according to the Poisson distribution. By computing the upper and lower bound of the probability, the pruning rules are proposed, which can reduce the candidate set and improve the query efficiently. Extensive experiments on real and synthetic datasets show the sTop*K*-UFI algorithm is an effective way to solve the problem of frequent items detection on uncertain data stream; it could significantly reduce the candidate set, save the memory space, improve the execution time and meet the requirements of practical applications.

Key words Top-*K*; frequent item; uncertain data ;data stream; sliding window

摘 要 频繁项查询在网络监控、网络入侵检测、关联规则挖掘等方面是一项非常重要的技术.该技术在静态的不确定数据中已经得到了深入的研究.但随着数据流特征和不确定性表现的日益明显,在不确定数据流环境下的查询已经成为一项新的研究课题.因此基于数据流普遍采用的滑动窗口模型,提出了一种高效的概率 Top-*K* 频繁项查询算法 sTop*K*-UFI.该算法避免了每次窗口更新都重新计算查询答案,而是利用现有的计算结果进行增量更新,从而减少查询代价.另外,该算法基于窗口中的现有数据对未来可能成为频繁项的元素进行预测,并利用泊松分布计算元素成为频繁项的概率上下界,提出相应的过滤策略,可以显著减少检测数据的数量,提高查询效率.实验结果表明,所提出算法可以有效地减少候选集、降低搜索空间、改善在不确定数据流上的查询性能.

关键词 Top-*K*;频繁项;不确定数据;数据流;滑动窗口

中图法分类号 TP311.13

收稿日期:2012-06-05;**修回日期**:2012-07-19

基金项目:国家自然科学基金重点项目(60933001);国家杰出青年科学基金项目(61025007)

在网络监控、网络入侵检测、蠕虫病毒探测等方面,频繁项的查询是一项非常重要的技术.目前,现有的 Top-K 频繁项查询算法都是基于确定的数据,即数据的存在性和精确性均确凿无疑.然而,在许多现实的应用中,例如经济、军事、物流、金融、电信、传感器网络等领域,数据的不确定性普遍存在,不确定性数据扮演关键角色[1].传统的数据管理技术无法有效管理不确定性数据,因此本文提出了针对不确定数据流的 Top-K 频繁项查询算法,具有一定的意义和价值.

由于数据的不确定性,需要对不确定数据建模,可能世界模型则是应用最广泛的数据模型.在该模型中,各元组的任一合法组合均构成一个可能世界实例(instance),实例的概率值可以通过相关元组的概率计算得到,如表 1 所示:

Table 1 Uncertain Database

表 1 不确定数据

ID	*Data*	*Pr*
t_1	8	0.6
t_2	8	0.4
t_3	3	0.8
t_4	8	0.3
t_5	8	0.5
t_6	3	0.2

表 1 是一个不确定性数据库,概率字段表示该元组的发生概率.元组之间独立也可能存在依赖关系(但本文仅考虑元组独立的情况).若各元组之间独立(假设采用基于元组的滑动窗口模型,窗口包含的元组数为 4,当前窗口数据为 t_1-t_4),则共有 $2^4=16$ 个可能世界实例,各实例的概率等于实例内元组的概率乘积与实例外元组的不发生概率的乘积,如表 2 所示.例如,可能世界实例$\{t_1,t_2\}$的发生概率为 $0.6\times0.4\times(1-0.8)\times(1-0.3)=0.0336$.

Table 2 Possible Worlds of Table 1

表 2 当前窗口可能世界实例

Possible World Instance	*Pr*	Possible World Instance	*Pr*
$W_0=\{\}$	0.0336	$W_8=\{t_1,t_2\}$	0.0336
$W_1=\{t_1\}$	0.0504	$W_9=\{t_2,t_4\}$	0.0096
$W_2=\{t_2\}$	0.0224	$W_{10}=\{t_1,t_4\}$	0.0216
$W_3=\{t_3\}$	0.1344	$W_{11}=\{t_1,t_2,t_3\}$	0.1344
$W_4=\{t_4\}$	0.0144	$W_{12}=\{t_1,t_2,t_4\}$	0.0144
$W_5=\{t_2,t_3\}$	0.0896	$W_{13}=\{t_2,t_3,t_4\}$	0.0384
$W_6=\{t_1,t_3\}$	0.2016	$W_{14}=\{t_1,t_3,t_4\}$	0.0864
$W_7=\{t_3,t_4\}$	0.0576	$W_{15}=\{t_1,t_2,t_3,t_4\}$	0.0576

本文基于上述不确定数据模型,采用数据流滑动窗口模型,提出了一种高效的不确定 Top-K 频繁项查询算法(Top-K uncertain frequent item, Top$K$-UFI).该算法避免了每次窗口更新都重新计算查询答案,而是利用现有的计算结果进行增量更新,从而减少查询代价.另外,本文提出的剪枝策略可以显著减少检测数据的数量,提高查询效率.

1 相关工作

由于不确定数据普遍存在,目前已有很多不确定数据频繁项查询方面的工作.按照频繁项的定义,研究内容主要分为两类:基于期望的方式[2]和基于概率的方式[3].

定义 1[2]. $esup^t(D)$. 元素 t 期望支持度为

$$esup^t(D)=\sum_{t\in T_i}p_i, \tag{1}$$

T_i 为不确定数据库 D 的元组,p_i 为该元组出现的概率,则 $esup^t(D)$为所有包含元素 t 的元组概率之和.在表 1 中,在当前窗口(t_1-t_4)中元素 8 出现次数的期望为 $esup^8(D)=0.6+0.4+0.3=1.3$.

定义 2[2]. 期望频繁项(expected frequent item, EFI).若元素 t 为期望频繁项,则满足不等式(2):

$$esup^t(D)>eminsup. \tag{2}$$

如果元素 t 期望支持度大于阈值 $eminsup$,则认为元素 t 为频繁项,否则为非频繁项.虽然采用期望的方式可以容易地查找到频繁项,但是期望仅仅是一个统计值,并不能真正完整地反映频次分布的概率特性.针对上述问题,Zhang 等人提出了一种基于概率的频繁项定义[3].

定义 3[3]. 概率频繁项(probability frequent item, PFI).若元素 t 为概率频繁项,则满足不等式(3):

$$\sum_{\omega\in PW,C^t(\omega)\geqslant minsup}P[\omega]>minprob, \tag{3}$$

其中 $minsup$ 为频次阈值,$minprob$ 为概率阈值,$C^t(\omega)$为元素 t 在 w 实例空间中出现的次数.如果 t 在每个可能实例中成为频繁项的概率之和超过了给定概率阈值 $minprob$,则 t 为概率频繁项.定义 3 在每一个可能实例中分别考察元素 t 成为频繁项的概率,比期望的方式更能反映实际情况.

令 $minsup=2$,$minprob=0.3$. 对于表 1 的不确定数据库,要判断 8 是否为频繁项,需要考察表 2 的可能世界模型,在后 8 个实例中,8 的个数均不小于 $minsup$,概率和为 $0.396>minprob$. 所以 8 是频繁项.

定义 4. $sup^t(D)$. 元素 t 概率频次分布.

元素 t 在确定数据库中其出现频次为确定值. 但是在不确定数据库中其出现频次为随机变量,用 $sup^t(D)$表示.

表 3 为元素 8 在当前窗口(t_1-t_4)出现频次的概率分布,其中 $Pr\{sup^8(D)=2\}=0.436$,表示元素 8 出现 2 次的概率为 0.436.

Table 3　"8" Frequency Distribution

表 3　元素 8 频次分布

Frequency	Pr
0	0.168
1	0.324
2	0.436
3	0.072

根据定义 3、定义 4 可知,元素 t 成为频繁项的概率为 $Pr\{sup^t(D)\geqslant minsup\}$. 如果 $\sum_{i=minsup}^{num^t(D)} sup^t(D)>minprob$,则元素 t 为频繁项,其中,$num^t(D)$为数据库 D 中 t 出现的总次数. 文献[3]提出了一种基于动态规划的方法计算频繁项概率,将原先的指数级复杂度降低为多项式级的复杂度. 计算公式如(4)所示:

$$P_{i,j}^t=\begin{cases}p_{i-1,j-1}^t\times p_i+p_{i-1,j}^t\times(1-p_i), & v_i=t,\\ p_{i-1,j}^t, & v_i\neq t,\end{cases} \tag{4}$$

P_i 为第 i 个元组出现的概率,$P_{i,j}^t$为在前 i 个元组中元素 t 出现 j 次的概率,该概率计算可以分为两部分:1)在前 $i-1$ 个元组中元素 t 仅出现 $j-1$ 次,则在第 i 个元组中元素 t 应该出现;2)在前 $i-1$ 个元组中元素 t 已经出现了 j 次,则在第 i 个元组中元素 t 不应该出现.

Chui 等人[4]采用类似于 Aprio 方法解决了期望频繁项集的挖掘问题,而 Leung 等人[5]基于 FP-Growth 的方法解决了相同问题,但其算法性能更加优化,然而他们都没有解决不确定数据流环境下的频繁项集挖掘问题. Zhang 和 Carson 等人[6-8]对不确定数据流中频繁项集的问题进行了研究,但是这些文章均采用期望频繁项定义. 根据上述分析可知,采用期望定义的方式会丢失相关信息,不能更好地反映真实情况. 因此,文献[9-13]采用了概率频繁项定义. 文献[9]采用 DP(dynamic programming-based Apriori algorithm)方法,使用式(4)计算概率,并利用 Apriori 的方法查找所有的概率频繁项集. 文献[10]采用 DC (divide-and-conquer-based Apriori algorithm)方法,递归地对数据库进行折半划分,直到仅剩余一个元组时进行概率计算. 通过实验表明,DC 算法性能较 DP 更加优化. 文献[11]使用泊松分布公式计算近似概率,效率较高但无法得到精确的结果. 由于频繁项集数量巨大,文献[12-13]分别提出了不同的闭合频繁项集的定义与挖掘算法将结果进行压缩. 但这些文章并没有解决滑动窗口模型下的不确定流挖掘算法. 文献[14]提出了滑动窗口下的概率聚集查询问题,但是该文解决的是近似查询问题,不能得到精确的查询结果. 文献[15]同样提出了滑动窗口下的不确定 Top-K 查询算法,但与本文解决的问题不同,本文解决的是 Top-K 频繁项查询问题,其提出的算法对本文问题并不适用. 本文提出了一种新的基于滑动窗口模型的不确定 Top-K 频繁项查询算法,可以精确地得到查询结果,有效地减少检测数据的数量,提高查询效率.

2　问题描述

将连续到达的元组视为数据流. 一个滑动窗口是数据流上固定个数的元组序列,并随着时间的推进不断向前滑动.

定义 5. (KUFI)Top-K 不确定频繁项.

$T=\{t^1,t^2,\cdots,t^K\}$, $\forall t_i\in T$, 并且 $\forall t_j\notin T$, 满足不等式(5):

$$\sum_{\omega\in PW(t_i),C^i(\omega)\geqslant minsup} P[\omega]\geqslant \sum_{\omega\in PW(t_j),C^j(\omega)\geqslant minsup} P[\omega]. \tag{5}$$

定义 5 是基于概率的不确定 Top-K 频繁项定义. 式(5)判断可能世界中的每一个可能实例,将元素 t 出现频次不小于 $minsup$ 的概率值相加,概率之和最大的 K 个元素即为概率 Top-K 频繁项.

定义 6. (SKUFI)不确定数据流 Top-K 频繁项.

不确定数据流 $DS=\{e_1,e_2,\cdots,e_n\}$,滑动窗口大小为 win. 对于每一个滑动窗口 $DS[i:i+win-1]=\{e_i,e_{i+1},\cdots,e_{i+win-1}\}$,其中 DS 表示滑动窗口,i 是当前窗口的窗口号(即第 i 个滑动窗口),在 DS_i 窗口中满足定义 5 的元组即为不确定数据流 Top-K 频繁项.

为了提高滑动窗口处理效率,滑动窗口被均匀地划分为同样大小的一些子窗口,即基本窗口.我们以基本窗口作为更新处理单元.其中 $DS[i]$ 表示第 i 个滑动窗口.令 BW_L 是基本窗口的长度,也就是基本窗口中包含元组的个数.W_b 为滑动窗口的宽度,即滑动窗口中含有基本窗口的个数,则滑动窗口的长度 $W_L=BW_L\times W_b$.令 $B[i],B[i+1],\cdots,B[i+W_b-1]$ 是当前窗口内的一系列基本窗口,$B[i]$ 是即将过期的基本窗口,$B[i+W_b]$ 是即将到来的基本窗口.表 4 为本文使用符号的相关说明:

Table 4 List of Symbols

表 4 相关符号定义

Symbol	Definition
D	Uncertain database D
$minsup$	Frequency Threshold
K	Parameter K
PW	Possible world
w	Possible world instance
$C^t(w)$	Count of element t in w
$p_j(t)$	The probability of element t in tuple T_j
$sup^t(D)$	Probability distribution of element t in D
$num^t(D)$	Count of element t in $D(p_j(t)>0)$
$esup^t(D)$	Expected count of element t in D
$DS[i]$	ith sliding window
$B[i+j]$	jth basic window in ith sliding window
BW_L	Length of basic window
W_b	Length of sliding window

显然,在每一个滑动窗口中,采用现有的静态不确定频繁项查询算法可以得到结果,但是该算法每次窗口更新都需要重新计算频繁项,没有利用各个窗口之间的关联数据,计算代价大.因此,本文提出一个新的滑动窗口算法(sliding window Top-K uncertain frequent item mining algorithm,sTop$K$-UFI),随着概率数据流中不确定元组的过期和到达,增量地维护频繁项.

3 滑动窗口 Top-K 频繁项查询基本算法

本节针对定义 6,提出了基于滑动窗口模型下的不确定频繁项查询基本算法,该算法总体上分为 2 步:滑动窗口初始化阶段;窗口滑动阶段,即数据过期与数据插入的处理.

3.1 滑动窗口初始化阶段

每到达一个新的基本窗口 $B[i+j]$,对该基本窗口中的所有不同元素计算其频次分布 $sup^t(B[i+j])$,计算频次分布公式如式(4)所示.当数据不断流入,直至当前滑动窗口装满,依次计算了每个基本窗口 $B[i+j]$ 中所有元素的频次分布,最终在进行频繁项查询时,需要将当前滑动窗口中所有基本窗口的计算结果进行合并,从而计算总的频次分布 $sup^t(DS[i])$,最终判定是否为概率频繁项.

在每一个基本窗口中,可以计算元素 t 的频次分布函数 $sup^t(B[i+j])$,则元素 t 的总的频次分布函数如式(6)所示:

$$sup^t(DS[i])=\prod_{j=0}^{W_b-1}sup^t(B[i+j]). \tag{6}$$

这里的乘法并不是简单的算术乘积,而是卷积计算.例如,表 5 和表 6 分别显示元素 a 在基本窗口 1 和基本窗口 2 的分布,表 7 表示元素 a 在两个基本窗口的总体分布:

Table 5 $sup^a(B[1])$

表 5 "a"在 $B[1]$频次分布

Frequency	Pr
0	1/4
1	1/2
2	1/4

Table 6 $sup^a(B[2])$

表 6 "a"在 $B[2]$频次分布

Frequency	Pr
0	1/2
1	1/2

Table 7 $sup^a(B[1+2])$

表 7 元素"a"在基本窗口 1 和 2 出现频次的分布

Frequency	Pr
0	1/8
1	3/8
2	3/8
3	1/8

从上述计算可以看出,计算多个基本窗口的总的频次分布与多项式乘积等价.元素 a 在基本窗口 1 的分布函数可以写成 $f(a)=\frac{1}{4}+\frac{1}{2}x+\frac{1}{4}x^2$,$x$ 的项数表示元素 a 出现的次数,x 的系数表示元素 a 出现该次数的概率值.例如 $\frac{1}{4}x^2$ 表示元素 a 出现

的 2 次的概率值为 1/4. 同理元素 a 在窗口 2 的分布函数可以写成 $f(a)=\frac{1}{2}+\frac{1}{2}x$. 则元素 a 在两个基本窗口的分布函数可以写成 $f_1(a)\times f_2(a)=\frac{1}{8}+\frac{3}{8}x+\frac{3}{8}x^2+\frac{1}{8}x^3$.

$sup^t(B[i+j])$为元素 t 在第 i 个滑动窗口中的第 j 个基本窗口的频次分布,则总的分布等价为所有基本窗口分布的多项式乘积. 计算两个多项式乘积的时间复杂度为 $O(m^2)$,m 为多项式中项的个数. 计算 W_b 个多项式乘积的时间复杂度为$O(W_b m^2)$. 本文采用快速傅里叶变换方法计算多项式乘积,该方法可以使两个多项式乘积的时间复杂度提高为 $O(m\log m)$,则 W_b 个多项式乘积的复杂度可以提升为 $O(W_b m\log m)$.

3.2　窗口滑动阶段

当窗口滑动时,当前窗口中的最老基本窗口数据过期,新基本窗口插入,对于其中每一个元素 t,需要考虑过期数据及新插入数据对其概率的影响. 按照式(6)进行概率计算,当仅有新数据到达,不考虑窗口元素过期的情况,依据式(6),利用现有的频次分布函数 $sup^t(D)$进行增量的更新,可以很容易地得到某元素成为频繁项的概率. 但是当有数据过期的情况下,删除数据会影响现有的 $sup^t(D)$值,需要重新计算 $sup^t(D')$($sup^t(D')$为删除过期数据后的概率),重新计算 $sup^t(D')$的时间复杂度为 $O(W_b^2 m\log m)$. 为了避免重新计算,提高效率,本文提出了一种新的计算方法,可以将时间复杂度提升为 $O(W_b m\log m)$.

对于每一个元素 t,本文设计了一种 sup-list 的数据结构来保存不同窗口的频次分布信息. 数据结构如图 1 所示:

$$\langle t, i, sup^t[0], sup^t[1], \cdots, sup^t[W_b-2], sup^t[W_b-1]\rangle$$

Fig. 1　sup-list data structure.

图 1　sup-list 数据结构示意图

t 为元素,i 为滑动窗口标号,$sup^t[j]$表示从第 j 个基本窗口到第 W_b-1 个基本窗口中元素 t 的总的频次分布. 即将当前 W_b-1 基本窗口的数据与 W_b-2 基本窗口数据合并,得到 $sup^t[W_b-2]$. 然后将 $sup^t[W_b-2]$结果与 W_b-3 基本窗口的数据合并,得到 $sup^t[W_b-3]$,以此类推,直到计算到整个滑动窗口中的频次分布 $sup^t[0]$.

尽管在当前窗口中需要保存 W_b-1 个频次分布函数,但是 $sup^t[j]$的计算结果可以通过 $sup^t[j+1]$的结果增量计算得到,所以计算的复杂度与初始化阶段时间复杂度相同,均为 $O(W_b m\log m)$. 算法的空间复杂度为 $O\left(\frac{W_b(W_b+1)}{2}m\right)$.

当有基本窗口过期时,仅删除当前窗口中最老的计算结果 $sup^t[0]$值即可,不需要重复计算,从而提高计算效率. 当新基本窗口插入,首先计算新基本窗口 $B[new]$中所有不同元素的频次分布,然后与现有的计算结果 $sup^t[j]$按照式(6)依次进行合并,从而得到更新后的不同窗口的频次分布.

3.3　基本算法 bs-TopKUFI

基于滑动窗口的概率频繁项查询的基本算法如算法 bs-TopKUFI(basic sliding window top-K uncertain frequent item mining algorithm)所示:

算法 1. bs-TopKUFI.

输入:DS——不确定数据流;$minsup$——最小支持度阈值 K;BW——基本窗口尺寸;W_b——滑动窗口尺寸;

输出:Top-K 概率频繁项集合 FK.

① FK=NULL; FPK=0 /* FPK 为集合 FK 中排在第 K 位元素的频繁概率值 */
② for each t in $B[i+j]$
③　按照式(4)计算 $sup^t(B[i+j])$
④　/* 初始化 */
⑤　if (当前滑动窗口未满)
⑥　for each t in $B[i+j]$
⑦　按照式(6)计算 $sup^t[j]$ ($j=0,\cdots,w_b-1$)
⑧　else
⑨　/* 窗口滑动 */
⑩　for each item t
⑪　按照 $sup^t[0]$值计算频繁概率 $fp(t)$
⑫　if $fp(t)>FPK$
⑬　将元素 t 以$\langle t, fp(t)\rangle$形式加入 FK 集合中
⑭　for each item t in expiring basic window /* 每个过期基本窗口中的元素 t */
⑮　删除 $sup^t[0]$值
⑯　for each item t in new basic window /* 每个新到达基本窗口中的元素 t */
⑰　按照式(6)更新 $sup^t[j]$值($j=0,\cdots,w_b-1$)

对于每个到达的基本窗口,计算该基本窗口中所有元素的频次分布(行①～③),将该基本窗口的计算结果与前面每一个基本窗口数据合并,分别计算从第 i 个到当前第 j 个基本窗口总的频次分布函数(行⑥～⑦).当窗口滑动,依据当前窗口的频次分布函数,计算 Top-K 概率频繁项并输出结果(行⑪～⑬).删除数据的处理非常简单,仅将当前窗口中最老的频次分布函数的计算结果删除即可(行⑭～⑮).插入数据时,将最新基本窗口中的计算结果插入到当前的频次分布函数列表中,并将新基本窗口的频次分布函数与当前窗口的分布函数合并,对其进行更新,得到最新的频次分布函数(行⑰).

4 滑动窗口 Top-K 频繁项查询优化算法

由第 3 节可知,要确定不确定数据库中的频繁项,需要对当前滑动窗口中的所有元素计算其成为频繁项的概率,才能最终判定.但是由于检测元素的数量巨大,算法的效率不高.因此,在计算过程中,利用不确定数据的特殊性找到过滤方法,进一步减少检测数据的数目,仅维护潜在的有效元素,从而有效地降低算法的时间和空间复杂度是十分必要的.

改进算法的基本思路是采用基于泊松分布函数的方法对检测元素逐步进行过滤,随着数据的不断到达,在现有数据的基础上计算元素在当前滑动窗口中成为频繁项的概率上界,利用该上界过滤掉最终不可能成为 Top-K 频繁项的元素,从而提高查询的效率.

4.1 基于泊松分布的过滤方法

随着基本窗口的不断到达,依据现有数据逐步构造候选集.但是这些数据仅仅是滑动窗口的部分数据,还有一部分数据在当前时刻无从得知,但是可以预测.最坏的情况为,在后续数据中与元素 t 取值相同的元素全部不出现;最好的情况为新插入的元素全部与 t 取值相同,且概率为 1.依据该方法可以推测元素 t 成为频繁项概率的下界和上界,从而对元素进行过滤.本文利用泊松分布方法进行过滤.

设 $X_1,X_2,\cdots,X_n$ 为 n 次独立的泊松试验序列,满足 $Pr(X_i=1)=p_i$,设 $X=\sum_{i=1}^{n}X_i$,表示 n 次实验成功的次数,变量 X 服从泊松二项分布.精确地 X 分布可以通过迭代的方法计算.如果 n 很大、p_i 很小时,X 近似地服从参数为 μ 的泊松分布,如式(7)所示:

$$Pr(X=j)=\frac{e^{-\mu}\mu^{j}}{j!},\tag{7}$$

其中,$\mu=E(X)=\sum_{i=1}^{n}p_i$.通过泊松实验的描述,很显然元素 t 出现的次数满足泊松实验的定义,可以使用式(8)近似计算 t 成为频繁项的概率:

$$Pr\{sup^{t}(D)\geqslant minsup\}\approx 1-\sum_{j=0}^{minsup-1}\frac{e^{-\mu}\mu^{j}}{j!}.\tag{8}$$

由文献[11]可知,使用泊松分布计算的近似概率与精确概率之间的误差为 $\delta=\min(u^{-1},1)\sum_{i=1}^{n}p_i^2$,因此元素 t 成为频繁项概率的取值范围为 $[1-\sum_{j=0}^{minsup-1}\frac{e^{-\mu}\mu^{j}}{j!}-\delta,1-\sum_{j=0}^{minsup-1}\frac{e^{-\mu}\mu^{j}}{j!}+\delta]$.

定理 1. $Pr\{sup^{t}(D)\geqslant minsup\}$ 近似值随期望单调递增.证明略.

由定理 1 可知,式(8)随期望 μ 单调递增.因此可以通过计算元素 t 的期望上界和下界计算频繁概率上界和下界.期望上界、下界的计算公式如式(9)(10)所示:

$$ue^{t}(i+j)=\sum_{m=0}^{j}esup^{t}(BW[m]))+BW_{L}\times(W_{b}-j-1),\tag{9}$$

$$le^{t}(i+j)=\sum_{m=0}^{j}esup^{t}(BW[m])),\tag{10}$$

$ue^{t}(i+j)$,$le^{t}(i+j)$ 分别表示第 i 个滑动窗口中当第 j 个基本窗口中数据到达时元素 t 的期望上界、下界.$ue^{t}(i+j)$ 的计算由两部分组成:1)从 $BW[0]$ 到 $BW[j]$ 个基本窗口,可以计算元素 t 的精确期望,同时也是期望的下界;2)从 j 到 W_b-1 个基本窗口,因为数据不可知,因此需要预测,即假设后续数据部分全部为元素 t 且存在概率为 1,则这部分期望值为 $BW_L\times(W_b-j-1)$.

根据分析可知元素 t 成为频繁项的概率上界,下界为式(11)和式(12)所示:

$$up^{t}(i+j)=1-\sum_{j=0}^{minsup-1}\frac{e^{-ue^{t}(i+j)}(ue^{t}(i+j))^{j}}{j!}+\delta;\tag{11}$$

$$lp^{t}(i+j)=1-\sum_{j=0}^{minsup-1}\frac{e^{-le^{t}(i+j)}(le^{t}(i+j))^{j}}{j!}-\delta.\tag{12}$$

通过计算概率上界和下界,可以得到 Top-K 候选集 $cTop\text{-}K$.该候选集中元素的数目远远小于不确定数据库中不同元素的个数.Top-K 候选集中的元素满足定理 2.

定理 2. $t \notin cTop\text{-}K, up^t(i+j) < lp^{t_K}(i+j)$.

定理 2 说明,如果元素 t 成为频繁项概率的上界比当前 Top-K 候选集中排在第 K 的元素下界还要小,则其肯定不能成为最终结果,可以将其过滤掉.

根据定理 2,随着数据的不断到达,可以逐步精化元素的 $esup(t)$值,从而更精确地得到频繁概率上界和下界,过滤掉更多的元素,而不用采用基于动态规划的方法进行概率计算,降低了算法的开销.实验结果表明,该方法可以过滤掉大量的元素.

4.2　候选集上界过滤

依据 4.1 节的剪枝规则可以得到 Top-K 候选集,大大减少了需要计算的元素数目.但是候选集中的元素的概率计算顺序如何判定,计算顺序是否与查询效率相关,是否需要对候选集中的所有元素进行判断,本节对该问题进行了阐述.

由定理 1 可知,元素成为频繁项概率的近似值随其期望单调递增.因此,元素的计算顺序可以按照其期望递减排序.对于候选集中的元素,按照其期望降序排列,利用式(6)计算其精确概率,最终得到真正的 Top-K 频繁项.但是,该方法需要对候选集中的所有元素进行计算,如果在计算的过程中就能够判断剩余的元素肯定不能成为 Top-K 的答案,则计算即可终止.

用 λ^+ 表示已经计算过的元素中(按期望降序排列)排在第 K 位的元素的概率值,即已知的 Top-K 频繁项的概率下界.λ^- 表示没有计算的元素成为频繁项概率的上界.如果已知 $\lambda^+ \geqslant \lambda^-$,计算就可以终止.关键的问题是如何计算 λ^-.

由于 $c\text{-}TopK$ 中所有的元素按照期望降序排列,即 $esup(t_1) < esup(t_2) < \cdots < esup(t_n)$,其中 $1 \cdots l$ 为已经计算过的元素,$l+1 \cdots n$ 为没有计算过的元素.现在想要判断 $l+1, \cdots, n$ 个元素成为频繁项的概率上界 λ^-,上界可以采用 chernoff 不等式和 Markov 不等式计算得到,如式(13)所示:

$$\lambda^- = \begin{cases} \dfrac{-(minsup - 1 - esup(t_{i+1}))^2}{4eup(t_{i+1})}, \\ 0 < \dfrac{minsup - 1 - esup(t_{i+1})}{esup(t_{i+1})} < 2e-1, \\ 2^{1+esup(t_{i+1})-minsup}, \\ \dfrac{minsup - 1 - esup(t_{i+1})}{esup(t_{i+1})} \geqslant 2e-1, \\ \dfrac{esup(t_{i+1})}{minsup}, \text{ other.} \end{cases} \tag{13}$$

式(13)给出了不同条件下 λ^- 的值,证明略.

定理 3. λ^- 随 $esup(t_{l+1})$的值单调递增.

证明略.

由于元素按照期望降序排列,λ^- 是当前没有计算过的所有元素成为频繁项概率的最大值.如果 $\lambda^+ \geqslant \lambda^-$,计算就可以终止.否则,继续计算下一个元素,直到该条件满足为止.

4.3　优化算法 is-TopKUFI

本节给出滑动窗口概率频繁项查询改进算法 is-TopKUFI (improved sliding window top-K uncertain frequent item mining algorithm)的详细描述.该算法与基本算法类似,区别有两处:1)在不同基本窗口计算结果合并过程中,可以过滤掉某些不能成为最终结果的元素;2)窗口滑动计算 Top-K 结果时,通过采用对候选元素按照期望降序排列,并在计算过程中计算成为结果的概率上界方法,仅对候选集中的部分进行判断.

算法 2. is-TopKUFI.

输入:DS——不确定数据流;$minsup$——最小支持度阈值 K;BW——基本窗口尺寸;W_b——滑动窗口尺寸;

输出:Top-K 概率频繁项集合 FK.

① FK =NULL; C=NULL;FPK=0;
　　/* FPK 为候选集合 C 中排在第 K 位元素的频繁概率值 */
② for each t in $B[i+j]$
③　　将 t 加入候选集 C 中
④　计算 $esup^t[j]$, $ue^t(j)$, $le^t(j)$
⑤　计算概率上界 $up^t(j)$和下界 $lp^t(j)$)
⑥　　if $up^t(j) < lp^{t_K}(j)$
⑦　　　从候选集 C 中删除 t
⑧　　else
⑨　　　计算 $sup^t[j]$
⑩　　/* 滑动窗口 */
⑪　将候选集 C 中的所有元素按照 $esup(t)$值降序排列
⑫　　for each t in C
⑬　　　根据 $sup^t[0]$计算 $fp(t)$
⑭　　　if $fp(t) > FPK$
⑮　　　将元素 t 以〈t, $fp(t)$〉形式加入 FK 集合中
⑯　计算 λ^-
⑰　if ($FPK \geqslant \lambda^-$)
⑱　　break;
⑲　　for each item t in expiring basic window
　　/* 每个过期基本窗口中的元素 t */
⑳　　　删除 $sup^t[0]$

㉑ for each item t in new basic window
/ * 每个新到达基本窗口中的元素 t * /
㉒ 按照式(6)更新 $sup^t[j]$值($j=0,\cdots,w_b-1$)

5 实验结果与分析

由于目前尚没有采用滑动窗口模型的 Top-K 概率频繁项的查询算法,本文两种算法(基本算法和优化算法)进行比较.

1) 实验环境:实验硬件环境为 Pentium3.0 GHz CPU;4 GB 内存;操作系统为 Windows XP;并采用 Microsoft VC++6.0 编程环境开发了模拟测试程序.

2) 数据集:实验采用两组数据集:一组是真实数据集 accidents(http://fimi.cs.helsinki.fi/),截取其中的部分元组,缺省的元组数为 10 万个;另一组数据使用 IBM Data Generator 发生器生成 10 万个元组,该组数据分布相对稀疏.由于这两个数据产生的都是频繁项集,本文对其简单修改,只保留其中一组属性的值构造频繁项.对于第 1 组真实数据集,通过分析数据取值的分布情况,按照相应的分布为每个元组分配概率值.第 2 组数据采用平均分布.其他参数的缺省值是 $minsup=1\,000$,$K=100$,滑动窗口元组数为 10 000,基本窗口更新的元组数为 100.

图 2 显示参数了 W 对时间的影响.W 表示滑动窗口中总的元组个数,取值范围为 1 000～10 000.

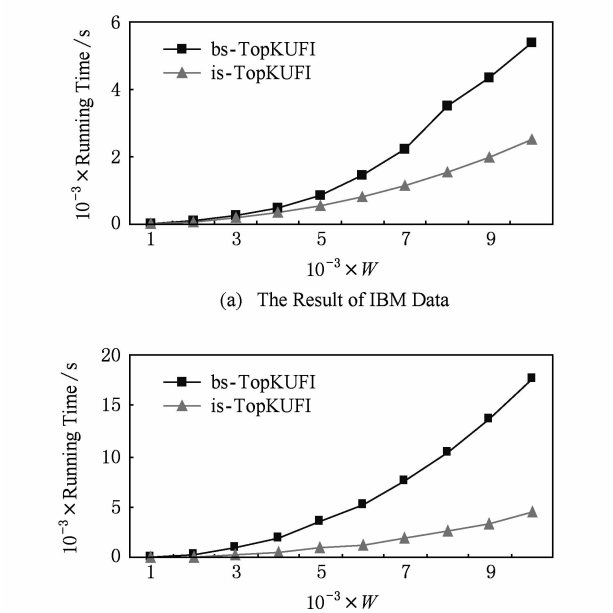


Fig. 2 Effect of W on time.
图 2 参数 W 对时间的影响

图 2(a)是 IBM 数据的结果,图 2(b)是在真实数据集上的结果.如图 2 所示,随着数据量增加,算法运行时间也会增加,但是增加相对平缓.is-TopKUFI 算法的性能优于 bs-TopKUFI 算法.因为 is-TopKUFI 采用泊松分布进行过滤,仅需要计算期望即可过滤掉大量元素,而不用进行频繁概率的计算,因此计算时间更快.

图 3 显示了参数 $minsup$ 对时间的影响,取值范围为 100～1 000.如图 3 所示,随着数据量增加,bs-TopKUFI 算法的时间并不改变.因为在基本算法中,需要对所有元素进行频次分布函数的计算,计算元素的数目与 $minsup$ 的并不相关,因此时间不变.而 is-TopKUFI 算法采用泊松分布进行过滤,$minsup$ 值越大过滤能力越强,需要计算的元素越少,计算时间越短.

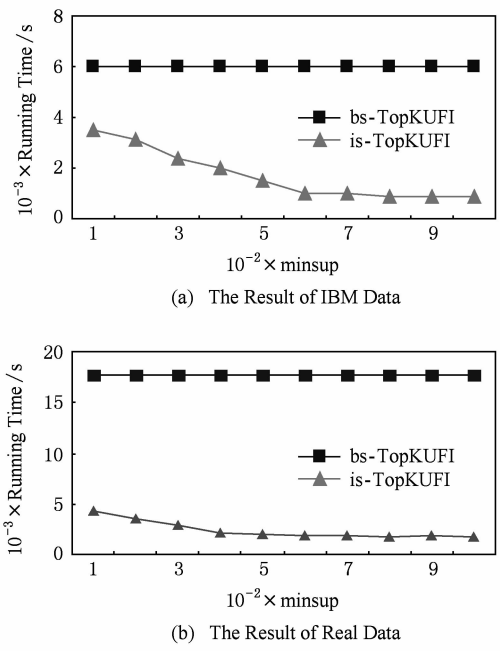
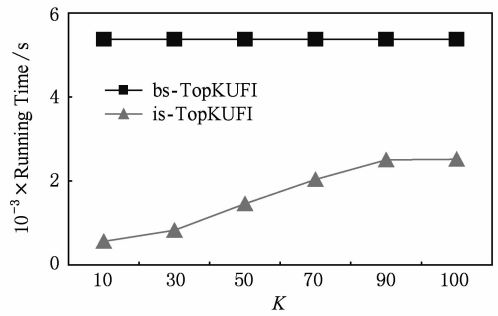
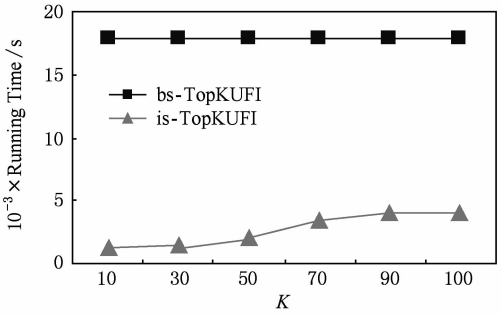


Fig. 3 Effect of minsup on time.
图 3 参数 minsup 对时间的影响

图 4 显示了参数 K 对时间的影响,取值范围为 10～100.可以发现,一般情况下由于查询结果显示页面的空间限制和用户操作习惯等原因,K 取值在 100 以下是最普遍的选择.图 4(a)是 IBM 数据结果,图 4(b)是在真实数据集上的结果.如图 4 所示,随着 K 增加,bs-TopKUFI 算法的时间并不改变.因为在基本算法中,需要对所有元素进行频次分布函数的计算,最终选取概率最大的 K 个元素,计算元素的数目与 K 并不相关,因此时间不变.而 is-TopKUFI 算法采用泊松分布进行过滤,K 值越大候选元素越多,因此计算时间越长.



(a) The Result of IBM Data



(b) The Result of Real Data

Fig. 4　Effect of K on time.

图 4　参数 K 对时间的影响

6　结　　论

不确定数据的频繁项检测是一个新兴的研究领域,其价值将在广泛的应用领域中得到重视.本文针对滑动窗口模型下的不确定性数据频繁项检测问题提出了基本算法与改进算法,并通过实验证明检测方法具有高效性和良好的可扩展性.未来工作包括支持更复杂的不确定数据模型,如具有互斥关系的不确定数据的检测.另外,基于网络监控方面的实际需求,对于分布式情况下的不确定频繁项检测还有待于进一步的研究.

参　考　文　献

[1] Zhou Aoying, Jin Cheqing, Wang Guoren, et al. A survey on the management of uncertain data[J]. Chinese Journal of Computers, 2009, 32(1): 1-16 (in chinese)
(周傲英,金澈清,王国仁,等. 不确定性数据管理技术研究综述[J]. 计算机学报, 2009, 32(1): 1-16)

[2] Cormode G, Garofalakis M. Sketching probabilistic data streams[C]// Proc of ACM SIGMOD'2007. New York: ACM, 2007: 281-292

[3] Zhang Qin, Li Feifei, Yi Ke. Finding frequent items in probabilistic data[C]// Proc of ACM SIGMOD'2008. New York: ACM, 2008: 810-832

[4] Chui C K, Kao B. A decremental approach for mining frequent itemsets from uncertain data[G]// LNCS 5012: Proc of PAKDD'2008. Berlin: Springer, 2008: 64-75

[5] Leung C K S, Mateo M A F, Brajczuk D A. A tree-based approach for frequent pattern mining from uncertain data [G]// LNCS 5012: Proc of PAKDD'2008. Berlin: Springer, 2008: 653-661

[6] Zhang Xiaojian, Peng Huili. A sliding-window approach for finding top-k frequent itemsets from uncertain streams[C]// LNCS 5446: Proc of APWeb/WAIM '09. Berlin: Springer, 2009: 597-603

[7] Carson K-S L, Fan Jiang. Frequent itemset mining of uncertain data streams using the damped window model[C]// Proc of SAC'11. New York: ACM, 2011: 950-955

[8] Carson K-S L, Boyu Hao. Mining of frequent itemsets from streams of uncertain data[C]// Proc of IEEE ICDE'2009. Piscataway, NJ: IEEE, 2009: 1663-1670.

[9] Thomas B, Hans P K, Matthias R, et al. Probabilistic frequent itemset mining in uncertain databases[C]// Proc of ACM SIGKDD'2009. New York: ACM, 2009: 119-128

[10] Sun L, Cheng R, Cheung D W, et al. Mining uncertain data with probabilistic guarantees [C]// Proc of ACM SIGKDD'2010. New York: ACM, 2010: 273-282

[11] Wang Liang, Cheng Reynold, Sau D L, et al. Accelerating probabilistic frequent itemset mining: A model-based approach[C]// Proc of ACM CIKM'2010. New York: ACM, 2010: 429-438

[12] Tong Y, Chen L, Ding B. Discovering threshold-based frequent closed itemsets over probabilistic data[C]// Proc of IEEE ICDE'2012. Piscataway, NJ: IEEE, 2012: 270-281

[13] Tang P, Peterson E A. Mining probabilistic frequent closed itemsets in uncertain databases[C]// Proc of ACM-SE '11. New York: ACM, 2011: 86-91

[14] Wang Qiutang, Wang Peng, Zhou Haofeng, et al. Estimating sliding window-based aggregate queries over probabilistic data streams[J]. Journal of Computer Research and Development, 2008, 45(Suppl): 169-174 (in Chinese)
(王秋棠,王鹏,周皓峰, 等. 基于滑动窗口的概率数据流上的聚集查询[J]. 计算机研究与发展, 2008, 45(增刊): 169-174)

[15] Jin Cheqing, Yi Ke, Chen Lei, et al. Sliding-window top-k queries on uncertain streams [C]// Proc of VLDB'2008. Trondheim, Norway: VLDB Endowment 2008: 301-312

Wang Shuang, born in 1980. PhD candidate. Her research interests include uncertain data management and data stream query.

Wang Guoren, born in 1966. Professor, PhD supervisor. His research interests include XML data management, bioinformatics, distributed database, and parallel computing.