软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(6):1588−1601 [doi: 10.3724/SP.J.1001.2012.04110]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

不确定集值数据的高效相似查询*

陈 珂, 洪银杰, 陈 刚+

(浙江大学 计算机科学与技术系,浙江 杭州 310027)

Efficient Processing of Similarity Search on Uncertain Set-Valued Data

CHEN Ke, HONG Yin-Jie, CHEN Gang+

(Department of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China)

+ Corresponding author: E-mail: cg@zju.edu.cn

Chen K, Hong YJ, Chen G. Efficient processing of similarity search on uncertain set-valued data. *Journal of Software*, 2012,23(6):1588−1601. http://www.jos.org.cn/1000-9825/4110.htm

Abstract: Setting similarity search on possible worlds is semantically and computationally different from the traditional technique for sets of certain data. Considering the uncertainty of items of set, i.e. there is a certain probability for an item appearing in a set, the traditional technique used for processing sets is not applicable. This paper brings forward the formulas to measure the expected similarity of the sets based on possible worlds' semantics. In the expected contexts, if the expected similarity of a pair of sets (X,Y) is larger than a given threshold value $\tau\in(0,1)$, this pair could be called as similar set pair. In the normal algorithm, the complexity of the expected similarity of uncertain sets based on possible worlds is of exponential order. This paper has provided new algorithms to calculate expected similarity by dynamic programming. The complexity of these algorithms is of polynomial order and they reduce execution time greatly. The final experiments have indicated the usability and the high performance of the new algorithms.

Key words: similarity search; expected similarity; dynamic programming; uncertain set

摘 要: 基于可能世界的不确定集合的相似查询,从语义上或者从计算方法的角度来看,都有别于传统的确定型集合上的技术.由于集合中的项存在不确定性,即一个项出现在集合中是有一定概率的,使得传统处理集合的技术不再适用.提出了一个基于可能世界的集合期望相似度的度量公式.在期望的度量公式中,如果一对集合(X,Y)的期望相似度大于给定的阈值$\tau\in(0,1)$,则被称为相似集合对.一般的算法,在基于可能世界的情况下计算不确定集合的期望相似度,其复杂度是指数级的.提出了利用动态规划来计算集合期望相似度的算法,该算法的复杂度是多项式级别,极大地减少了计算时间.实验结果表明了基于该算法查询的可用性和高性能.

关键词: 相似查询;期望相似度;动态规划;不确定集值

中图法分类号: TP311 **文献标识码**: A

集值相似查询[1−6]是数据去重、整合、挖掘的一个重要工具,其主要目的是找出所有近似的记录对[3,7].

给定记录(集值数据)集 R 和查询集值数据 q 以及相似度的阈值$\tau\in(0,1]$,集合的相似查询就是返回相似度不

* 基金项目: 国家自然科学基金(60803003, 60970124)
收稿时间: 2011-02-24; 修改时间: 2011-05-18; 定稿时间: 2011-08-31

低于τ的所有记录对$\{(r,q)|sim(r,q)\geq\tau,r\in R\}$.其中,$sim(*,*)$是两个集合的相似度量函数.

现在已有不少的这方面的研究,但是主要关注在确定性集值数据.其实在很多应用中,记录里面的项可能并不是确定的[9−14],例如:1) 来自传感器收集的数据含有噪音;2) 人为的数据,如书写错误.研究表明,一般企业里约有 1%~5%错误的数据[15];3) 信息提取的正确率约在 70%~90%[16],比如对地址“52-A Goregaon West Mumbai 400 076”进行提取之后,发现城市可能是“West Mumbai”,也可能是“Mumbai”.

最新的不确定集合的相似查询是文献[14],该文在可能世界的基础之上对两个不确定集合是否相似给出了定义:$p(sim(r,s)\geq\gamma)\geq\alpha$.即表示不确定集合 r 和 s,在各种可能世界里面,如果满足相似度$\geq\gamma$的概率为$\geq\alpha$,则集合 r 和 s 相似.这种相似度量里面,需要用户提供两个阈值来进行判断,而阈值的提供对用户来就说是件很困难的事情.如果我们选用期望相似度量($eSim$)来进行判断,即 $eSim(r,s)\geq\tau$,那么用户只需提供一个阈值τ,就可以进行相似判断;而这种模式在定义形式上很接近传统的集合相似度量 $sim(r,s)\geq\tau$.

而且上述两种不确定集合的相似度量具有不同的语义.$p(sim(r,s)\geq\gamma)$表示 r 和 s 相似度大于γ的概率越大,则两者越相似;而 $eSim(r,s)$表示 r 和 s 的期望相似度越大,说明两者越相似.并且对 $p(sim(r,s)\geq\gamma)$而言,还得根据γ的情况来决定相似度的情况.显然,$eSim(r,s)$的模式比较简单些.通过下面的例子,可以进一步进行了解.

表 1 中的记录 $r1=\{A:0.4,B:0.6,D:0.3\}$,$r2=\{A:1,C:0.6,D:0.4\}$,同时存在互斥规则$\{A,B\},\{C,D\}$,即表示 A 和 B 不能同时出现在一个世界里面,C和D也不能同时出现.通过枚举可以得到所有的可能世界,见表中,对于$r1$,存在可能世界为$\{\{A\},\{B\},\{A,D\},\{B,D\}\}$;对于 $r2$,存在可能世界为$\{\{A,C\},\{A,D\}\}$.对于 $r1$ 的可能世界$\{A\}$的概率计算,通过互斥规则我们知道,A 和 B 是不会同时出现的,那么出现$\{A\}$世界的概率就是出现 A(不会出现 B)同时不出现 D 的概率,即 $p(A)\times(1-p(D))$.表 1 列举了所有的可能世界以及对应的概率和相似度,相似度计算采用 $Jaccard$ 函数($Jaccard(x,y)=|x\cap y|/|x\cup y|$).通过对可能世界的相似度和概率的计算,我可以容易地得到期望相似度:$eSim(r1,r2)=0.236$;假设γ的取值分别是$[0.25,0.5,0.75]$,可以得到对应的概率分别为:$p(sim(r1,r2)\geq0.25)=0.472$,$p(sim(r1,r2)\geq0.5)=0.328$ 以及 $p(sim(r1,r2)\geq0.75)=0.048$.可以看出,两者具有不同的语义,适用于不同的应用.文献[14]对大于某一相似阈值的概率计算做了不少研究,本文将从另一个角度,即计算期望相似度,对不确定集合的相似度量进行研究.

Table 1 Uncertain sets and corresponding possible worlds

表 1 不确定集合以及对应的可能世界

ID	Record
$r1$	$\{A:0.4,B:0.6,D:0.3\}$
$r2$	$\{A:1,C:0.6,D:0.4\}$

Rules:$\{A,B\},\{C,D\}$,互斥规则:$\{A,B\},\{C,D\}$

Possible worlds, $pw(\{r1,r2\})$	Appearance probability $Pr(pw(\{r1,r2\}))$	$sim(r1,r2)=Jaccard(r1,r2)$
$\{\{A\},\{A,C\}\}$	$0.4\times(1-0.3)\times1\times0.6=0.168$	1/2
$\{\{A\},\{A,D\}\}$	$0.4\times(1-0.3)\times1\times0.4=0.112$	1/2
$\{\{B\},\{A,C\}\}$	$0.6\times(1-0.3)\times1\times0.6=0.252$	0
$\{\{B\},\{A,D\}\}$	$0.6\times(1-0.3)\times1\times0.4=0.168$	0
$\{\{A,D\},\{A,C\}\}$	$0.4\times0.3\times1\times0.6=0.072$	1/3
$\{\{A,D\},\{A,D\}\}$	$0.4\times0.3\times1\times0.4=0.048$	1
$\{\{B,D\},\{A,C\}\}$	$0.6\times0.3\times1\times0.6=0.108$	0
$\{\{B,D\},\{A,D\}\}$	$0.6\times0.3\times1\times0.4=0.072$	1/3

Possible worlds space

为了描述数据的不确定性,本文用不确定的数据集合(比如从文本或网页对象的描述中抽取关键词组成集合)进行建模,集合的不确定性表示词在集合中出现的概率.我们用期望相似度($eSim$)来计算两个集合的相似度,并根据用户指定的相似阈值来确定是否相似.对网页对象描述中的关键词集合进行相似判断,可以帮助去掉重复的记录、对网页数据进行聚类、数据整合、数据过滤等,这些方面都已具有很多的应用了.

为了更快地计算集合对的期望相似度,本文提出了新的期望相似度度量方法——基于动态规划和转移规则的计算方法.通过该方法的计算,可以将指数复杂度(2^n)的计算时间降到多项式的计算时间.通过实验和分析,

说明了该期望相似度度量方法不仅适用于多种相似函数,而且还具备高性能的优点.

本文的主要贡献:

- 提出了不确定集合的期望相似度计算方法,并进行了详细的定义;
- 提出了动态规划算法,降低期望相似度计算的复杂度;
- 设计了相似度量的裁剪技术,有效地提高了算法的执行;
- 通过实验演示,说明算法的可用和高效性.

本文第 1 节对本文提出的问题进行详细说明和定义.第 2 节介绍基于动态规划的期望相似度度量方法.第 3 节给出多种剪枝技术.第 4 节考虑更广泛的应用场景,提出集合的连接查询的解决方案.第 5 节给出实验并进行分析.第 6 节是相关工作.第 7 节对本文进行总结.

1 问题定义

本节里,我们针对不确定集合的可能世界模型对期望的相似度进行了定义,并且根据用户指定的阈值τ对两个集合是否相似进行判定.

不确定数据模型见表 1,我们采用可能世界的语义[9,10,14]:1) 每个项在集合里面出现带有一定的概率或$p\in(0,1)$;2) 互斥规则(x-rule),表示项之间的相关,假定一个项只会出现在一个 x-Rule;如果没有 x-Rule,那么就相当于项之间都是独立的.

定义 1. 基本操作,集合交、并的定义.

对于确定性的集合,我们可以将其看做概率都为 1 的不确定集合.根据确定型集合交、并的定义,我们对不确定集合之间交、并操作进行如下定义[8]:

$$X\cap Y=\{(i,p)|p(i)=X.p(i)\times Y.p(i)\} \tag{1}$$

$$X\cup Y=\{(i,p)|p(i)=X.p(i)+Y.p(i)-X.p(i)\times Y.p(i)\} \tag{2}$$

以表 1 的数据为例:$r1\cap r2=\{A:0.4,D:0.12\}$,$r1\cup r2=\{A:1,B:0.6,C:0.6,D:0.58\}=r1+r2-r1\cap r2$.

定义 2. 可能世界概率定义.

$$p(w)=\prod_{x\in \text{x-rule}}\left(\prod_{i\in(w\cap x)}p(i)\times\chi(w\cap x\neq\varnothing)+\left(1-\sum_{i\in x}p(i)\right)\times\chi(w\cap x=\varnothing)\right) \tag{3}$$

其中,$\chi(\exp)=\begin{cases}1, & \exp=\text{ture}\\ 0, & \exp=\text{false}\end{cases}$,x-rule 表示互斥的规则.如果没有互斥的规则,那么有

$$p(r.w)=\prod_{i\in I(r)}(p(i)\times\chi(i\in w)+(1-p(i))\times\chi(i\notin w)) \tag{4}$$

其中,$\boldsymbol{I}(r)$表示 r 所有的项.

由于 $p(\{w1,w2\})=p(w1)\times p(w2)$,根据公式(3)对表 1 的可能世界$\{\{A\},\{A,C\}\}$进行计算可得

$$p(\{\{A\},\{A,C\}\})=p(\{A\})\times p(\{A,C\})=r1.p(A)\times(1-r1.p(D))\times r2.p(A)\times r2.p(C)=0.168.$$

定义 3. 不确定集合的期望相似度计算.

根据所有的可能世界进行相似度计算,得到集合对的期望相似度.

$$eSim(r1,r2)=\sum_{w1\in\{pw(r1)\}}\left\{\sum_{w2\in\{pw(r2)\}}(sim(w1,w2)\times p(\{w1,w2\}))\right\} \tag{5}$$

采用相似函数 *Jaccard*,对表 1 可以得到见表 2 所有可能世界对应的 *Jaccard* 相似度.

定义 4. 用户指定相似度阈值为τ,如果两个不确定集合 $r1$ 和 $r2$ 是相似的,那么必须满足 $eSim(r1,r2)\geqslant\tau$.

根据定义 3,从文本或网页对象(对象看做是一个由关键字组成的集合)的数据集 R 里面找出与查询对象 q(q 也可以被看作一个由关键字组成的集合)相似的所有对象,也就是找出所有与 q 相似度不低于τ的对象.即

$$\{r|eSim(q,r)\geqslant\tau,r\in R\}.$$

Table 2 Similarity of all possible worlds

表 2 所有可能世界对应的相似度

$Pw(r1)$	$Pw(r2)$	
	$p(\{A,C\})=0.6$	$p(\{A,D\})=0.4$
$p(\{A\})=0.28$	1/2	1/2
$P(\{B\})=0.42$	0	0
$P(\{A,D\})=0.12$	1/3	1
$p(\{B,D\})=0.18$	0	1/3

2 期望相似度的计算

首先回顾确定性集合的相似度量.一般一个相似函数对两个集合进行计算,返回一个相似度 $sim\in(0,1]$.不同的应用会采用不同的相似函数,常见的有以下几种定义:

- *Jaccard similarity* 函数: $JS(X,Y)=\dfrac{|X\cap Y|}{|X\cup Y|}$;
- *Cosine similarity* 函数: $CS(X,Y)=\dfrac{\vec{X}\cdot\vec{Y}}{\|\vec{X}\|\cdot\|\vec{Y}\|}=\dfrac{|X\cap Y|}{\sqrt{|X|\cdot|Y|}}$;
- *Dice similarity* 函数: $DS(X,Y)=\dfrac{2\times|X\cap Y|}{|X|+|Y|}$.

本文将重点关注 *Jaccard* 相似函数,其他一些也可以采用该技术进行处理.

现在,我们根据 *Jaccard* 相似函数对期望相似度量进行定义:

$$\begin{aligned} eJS(r1,r2)&=\sum_{w1\in\{pw(r1)\}}\left\{\sum_{w2\in\{pw(r2)\}}(JS(w1,w2)\times\Pr(\{w1,w2\}))\right\}\\ &=\sum_{w1\in\{pw(r1)\}}\left\{\sum_{w2\in\{pw(r2)\}}\left(\frac{|w1\cap w2|}{|w1\cup w2|}\times p(\{w1\})\times p(\{w2\})\right)\right\}\end{aligned}\tag{6}$$

这是一种最直观的计算方式,同样也是计算复杂度最高的.由于 $r1$ 和 $r2$ 的可能世界是指数级的,因此复杂度也是指数级的.假设没有互斥规则,且 $N(r)$表示 r 中项的个数,$Nc(r)$和 $Nu(r)$分别表示对于确定项(概率为 1 或者 0)的个数和不确定项(概率在(0,1)范围)的个数,那么其复杂度为 $2^{Nu(r1)+Nu(r2)}$.

从期望相似度的计算过程,可以发现枚举所有的可能世界,然后计算各种可能世界下的相似度和对应的概率,最后累加获得期望值.不同的可能世界可能对应相同的相似度,而且不同的相似度个数明显少于可能世界的个数,约为不确定项个数的平方级别.如果能获得所有不同相似度对应的概率,那么同样也能够得到期望相似度.根据表 1 和表 2 的情况,有不同的相似度见表 3.

Table 3 Different similarity and corresponding probability

表 3 不同的相似度以及对应的概率

Similarity value	Appearance probablity
0	0.252+0.168+0.108=0.528
1/2	0.168+0.112=0.28
1/3	0.072+0.072=0.144
1	0.048

$eJS(r1,r2)=0\times0.528+1/2\times0.28+1/3\times0.144+1\times0.048=0.236$.结果与之前的计算相同.

2.1 独立情况下的计算

我们首先讨论项之间是独立情况的计算方法.

用 $\boldsymbol{I}$ 表示有序的项全集,$\boldsymbol{I}_k$ 表示第 k 个项,S_k 表示计算 k 项之后的结果(或者说是运行状态),记录了当前各种相似度的值(n/m)和对应的概率($P_{n,m}(Sk)$).当考虑 $\boldsymbol{I}_{k+1}$ 的时候,可能有 3 种情况:

- 项 $\boldsymbol{I}_{k+1}$ 同时出现在集合 $r1$ 和 $r2$;
- 项 $\boldsymbol{I}_{k+1}$ 仅出现在集合 $r1$ 或 $r2$;

- 项 $\boldsymbol{I}_{k+1}$ 都不出现在集合 $r1$ 和 $r2$.

在考虑 $\boldsymbol{I}_{k+1}$ 的时候,状态 S_k 也需要作相应的转化,变成状态 S_{k+1},具体见表 4.

Table 4 Transformation rules of similarity n/m in S_k

表 4 状态 S_k 里面的相似度 n/m 转变规则

Rule No.	Next state	Occurrence condition	Occurrence probability
Rule 1	$(n+1)/(m+1)$	Appearance in set $r1$ and $r2$ at the same time	$r1.p(I_{k+1})\times r2.p(I_{k+1})$
Rule 2	$n/(m+1)$	Appearance in set $r1$ or $r2$ only	$(1-r1.p(I_{k+1}))\times r2.p(I_{k+1})+r1.p(I_{k+1})\times(1-r2.p(I_{k+1}))$
Rule 3	n/m	No appearance in set $r1$ and $r2$	$(1-r1.p(I_{k+1}))\times(1-r2.p(I_{k+1}))$

在初始状态 S_0 里面,$P_{0,0}(S_0)=1$,其他的概率均为 0.并且在任何一个状态里面,都应该满足 $\sum_{p\in S_k} p=1$.

图 1 是状态转化的动态规划计算示意图.

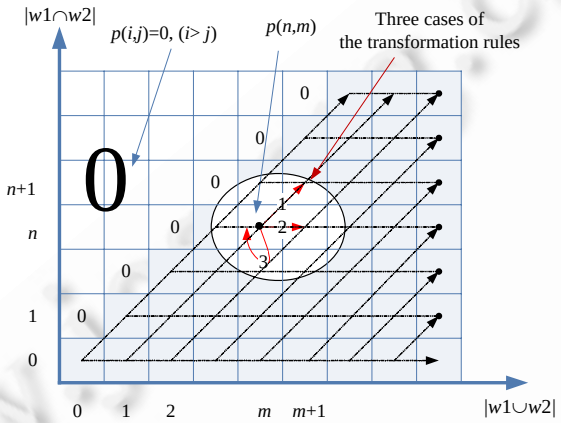

Fig.1 Dynamic computation scheme

图 1 动态规划计算

这个动态规划计算更加有利于说明状态 S_k 到 S_{k+1} 的转化.从上述转化规则中可以看出,$P_{n,m}(S_k)$可能转化为 $P_{n+1,m+1}(S_{k+1})$,$P_{n,m+1}(S_{k+1})$,$P_{n,m}(S_{k+1})$.也就是说,$P_{n,m}(S_{k+1})$可能由 3 部分组成:$P_{n-1,m-1}(S_k)$,$P_{n,m-1}(S_k)$和 $P_{n,m}(S_k)$.我们可以得到下面的动态计算方法:

$$P_{n,m}(S_{k+1})=P_{n,m}(S_k)\times r1.\tilde{p}(I_{k+1})\times r2.\tilde{p}(I_{k+1})+P_{n,m-1}(S_k)\times(r1.\tilde{p}(I_{k+1})\times r2.p(I_{k+1})+ r1.p(I_{k+1})\times r2.\tilde{p}(I_{k+1}))+P_{n-1,m-1}(S_k)\times r1.p(I_{k+1})\times r2.p(I_{k+1}) \tag{7}$$

其中, $\tilde{p}(I_{k+1})=1-p(I_{k+1})$;在 $P_{n,m}(S_k)$里面,n,m 的取值范围是$[0,k]$,且 $n<=m$ 的情况下值为$[0,1]$,其他情况均为 0.

已知初始状态 S_0 里面 $P_{0,0}(S_0)=1$,其他的概率均为 0.根据转换规则和 $\boldsymbol{I}_1$ 的情况,可以得到 S_1 的状态($P_{0,0}(S_1)$, $P_{0,1}(S_1)$,$P_{1,1}(S_1)$).利用此动态规划计算,可以计算得到最终的状态 $S_n(n=|I(r1\cup r2)|)$.根据 S_n 的状态信息,我们可以得到对应的期望相似度:

$$eJS(r1,r2)=\sum_{j=1}^{n}\sum_{i=1}^{j}(p_{i,j}(S_n)\times(i/j)) \tag{8}$$

例如,以 $r1=\{A:1.0,B:0.2,C:0.5\}$,$r2=\{A:0.4,C:0.7\}$为例,那么$|I(r1\cup r2)|=|\{A,B,C\}|=3$,对应的状态转换如图 2 所示.计算过程:

1. 初始状态,如图 2 所示中的 S_0,只有 $P_{0,0}(S_0)=1$.
2. 计算 A 的转变规则以及对应的状态 S_1.$r1$ 中 A 的概率是 1.0,$r2$ 中 A 的概率是 0.4,根据表 4 中的转变规则,可以得到 $P_{0,0}(S_1)=(1-1.0)\times(1-0.4)=0$,$P_{1,1}(S_1)=1.0\times0.4=0.4$,$P_{0,1}(S_1)=1.0\times(1-0.4)+(1-1.0)\times0.4=0.6$. 同样,通过对 B 和 C 的计算得到了 S_2 和 S_3.
3. 对最终的状态 S_3 进行计算,得到期望的相似度

$eSim(r1,r2)=\sum_{i>0}\sum_{j>0}(p_{i,j}(S)\times(i/j))$ =0.048×(1/1)+0.34×(1/2)+0.082×(1/3)+0.112×(2/2)+0.028×(2/3)=0.376.

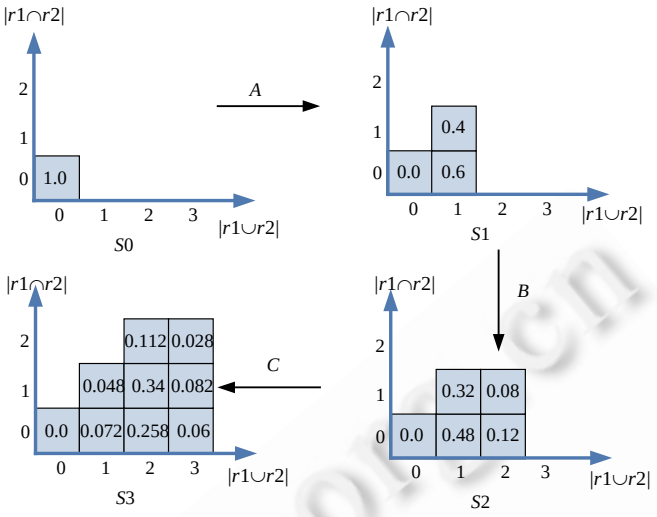

Fig.2 Examples of status transformation

图 2 状态转换的例子

从动态规划的计算过程,我们只需要保存当前状态的信息即可,即空间代价为 $O(n^2)$,其中,$n=|I(r1\cup r2)|$.在每个转换的步骤中,计算的次数为 $O(k^2)$,一共需要 n 步计算,因此总的计算代价为

$$Cost=\sum_{k=1}^{n}O(k^2)=O(n^3) \tag{9}$$

下面是独立情况下,*Jaccard* 的期望相似度计算的伪代码.

算法 1. *eJS*(*r*1,*r*2).

Input: *r*1 and *r*2 are uncertain sets;

Output: *Jaccard* expected similarity of *r*1 and *r*2.

1. *initial*(*S*); //*S.n*=0, $P_{n,m}(S)$=0
2. $P_{0,0}(S)$=1;
3. Foreach item $i\in I(r1\cup r2)$ do
4. $S\leftarrow transform(S,r1.p(i),r2.p(i))$;
5. End Foreach
6. $eJS(r1,r2)=\sum_{j=1}^{n}\sum_{i=1}^{j}(p_{i,j}(S_n)\times(i/j))$
7. return *eSim*;

算法 2. *transform* (*pS*,*p*1,*p*2).

Input: *pS* the previous State of records' Pair;

Input: *p*1, *p*2 are the probability of the next item in records' Pair;

Output: *nS* the new State of records' Pair.

1. *initial*(*nS*); //$P_{n,m}(nS)$=0
2 *nS.n*←*pS.n*+1
3. FOR *j*=0 to *pS.n*
4. FOR *i*=0 to *j*
5. IF $P_{i,j}(pS)\neq 0$

6. $p_{i+1,j+1}(nS) \leftarrow p_{i+1,j+1}(nS)+p_{i,j}(pS)\times p1\times p2$; //rule 1(对应表 4 的 rule)
7. $p_{i,j+1}(nS) \leftarrow p_{i,j+1}(nS)+p_{i,j}(pS)\times(\tilde{p}1\times p2+p1\times\tilde{p}2)$; //rule 2
8. $p_{i,j}(nS) \leftarrow p_{i,j}(nS)+p_{i,j}(pS)\times\tilde{p}1\times\tilde{p}2$; //rule 3
9. ENDIF
10. ENDFOR
11. ENDFOR
12. return *nS*;

2.2 互斥情况下的计算

接下来,我们来讨论项之间是互斥情况的计算.

用 R 表示有序的互斥规则全集,R 里面是互斥的项,R_k 表示第 k 个互斥规则,S_k 表示计算 k 项之后的结果(或者说是运行状态),记录了当前各种相似度的值(n/m)和对应的概率($P_{n,m}(Sk)$).当考虑 R_{k+1} 时,可能有 4 种情况:

- 项 R_{k+1} 同时出现在集合 $r1$ 和 $r2$:
  出现在集合 $r1$ 和 $r2$ 里面的项相同;
  出现在集合 $r1$ 和 $r2$ 里面的项不同;
- 项 R_{k+1} 仅出现在集合 $r1$ 或 $r2$;
- 项 R_{k+1} 都不出现在集合 $r1$ 和 $r2$.

在考虑 R_{k+1} 的时候,状态 S_k 也需要作相应的转化,变成状态 S_{k+1},具体见表 5.

Table 5 Transformation rules of similarity n/m in S_k

表 5 状态 S_k 里面的相似度 n/m 转变规则

Rule No.	Next state	Occurrence condition	Occurrence probability
Rule 1′	$(n+1)/(m+1)$	Appearance in set $r1$ and $r2$ are same	$\sum_{i\in R_{k+1}} r1.p(i)\times r2.p(i)$
Rule 2′	$n/(m+2)$	The items appearance in set $r1$ and $r2$ are different	$\sum_{i\in R_{k+1}} r1.p(i)\times\sum_{i\in R_{k+1}} r2.p(i)-\sum_{i\in R_{k+1}}(r1.p(i)\times r2.p(i))$
Rule 3′	$n/(m+1)$	Appearance in set $r1$ or $r2$ only	$\left(1-\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(\sum_{i\in R_{k+1}} r2.p(i)\right)+\left(\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(1-\sum_{i\in R_{k+1}} r2.p(i)\right)$
Rule 4′	n/m	No appearance in set $r1$ and $r2$	$\left(1-\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(1-\sum_{i\in R_{k+1}} r2.p(i)\right)$

从上述转化规则中可以看出,$P_{n,m}(S_k)$可能转化为 $P_{n+1,m+1}(S_{k+1})$,$P_{n,m+2}(S_{k+1})$,$P_{n,m+1}(S_{k+1})$,$P_{n,m}(S_{k+1})$.也就是说,$P_{n,m}(S_{k+1})$可能由 4 部分组成:$P_{n-1,m-1}(S_k)$,$P_{n,m-2}(S_k)$,$P_{n,m-1}(S_k)$和 $P_{n,m}(S_k)$.我们可以得到下面的动态计算方法.

$$\begin{aligned}P_{n,m}(S_{k+1}) = {} & P_{n,m}(S_k)\times\left[\left(1-\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(1-\sum_{i\in R_{k+1}} r2.p(i)\right)\right]+\\ & P_{n,m-1}(S_k)\times\left[\left(1-\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(\sum_{i\in R_{k+1}} r2.p(i)\right)+\left(\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(1-\sum_{i\in R_{k+1}} r2.p(i)\right)\right]+\\ & P_{n-1,m-1}(S_k)\times\left[\left(\sum_{i\in R_{k+1}}(r1.p(i)\times r2.p(i))\right)\right]+\\ & P_{n,m-2}(S_k)\times\left[\left(\sum_{i\in R_{k+1}} r1.p(i)\right)\times\left(\sum_{i\in R_{k+1}} r2.p(i)\right)-\left(\sum_{i\in R_{k+1}}(r1.p(i)\times r2.p(i))\right)\right]\end{aligned} \tag{10}$$

从动态规划的计算过程,我们只需要保存当前状态的信息即可,即空间代价为 $O(n^2)$,其中,$n=|R(r1\cup r2)|$,$|r1\cup r2|_{\max}=n$.在每个转换的步骤中,计算的次数为 $O(k^2)$,一共需要 n 步计算,因此总的计算代价:

$$Cost=\sum_{k=1}^{n}O(k^2)=O(n^3) \tag{11}$$

从上述对独立情况的分析以及对互斥情况的分析可以发现,两者具有类似的计算过程.为了阐述方便,本文将重点关注独立情况下的计算,因此,下文将只针对独立的情况进行分析和实验验证,互斥情况下的计算可以通过类似的步骤实施.

3 剪枝策略

事实上,我们并不需要计算那些在$r1$和$r2$都是确定的项($p(i)$=1或$p(i)$=0的项).如果某个项i满足$r1.p(i)$=1.0和$r2.p(i)$=1.0,那么计算该项的时候,转移规则里面的$P_{n,m}(S_k)$都转变成$P_{n+1,m+1}(S_{k+1})$,即向右上移动一格;如果某个项i满足$r1.p(i)$=1.0和$r2.p(i)$=0,那么计算该项的时候,转移规则里面的$P_{n,m}(S_k)$都转变成$P_{n,m+1}(S_{k+1})$,即向右移动一格.我们将$r1$和$r2$都是确定的项选出来,用$c(r1)$和$c(r2)$表示;剩下的部分称为不确定的部分,用$u(r1)$和$u(r2)$表示.

那么,期望相似度的计算可以使用下面的公式:

$$eJS(r1,r2)=\sum_{w1\in\{pw(u(r1))\}}\sum_{w2\in\{pw(u(r2))\}}\left(\frac{|w1\cap w2|+|c(r1)\cap c(r2)|}{|w1\cup w2|+|c(r1)\cup c(r2)|}\times p(\{w1\})\times p(\{w2\})\right) \tag{12}$$

因此,复杂度为$O(n^3)$,其中,n=$|I(u(r1)\cup u(r2))|$.

下面还有几个性质,可以用来估计相似度的上限,以帮助更有效地进行过滤.

性质. 在独立情况下:

1) 如果$i\in I(r1\cup r2)$,则有$eJS(r1,r2)\leqslant eJS(r1\cup\{i:1.0\},r2\cup\{i:1.0\})$;

2) 如果$i\in I(r1\cup r2)\backslash I(r1\cap r2)$,则有$eJS(r1,r2)\leqslant eJS(r1\backslash\{i\},r2\backslash\{i\})$.

证明:对于$r1$和$r2$里面任何项i,我们考虑状态$P_{n,m}(S_k)$在计算项i后对期望相似度的贡献:

1. 对$eJS(r1,r2)$,根据状态S_k的转变规则,可以得到下面的值

$$value1=n/m\times P_{n,m}(S_k)\times(1-r1.p(i))\times(1-r2.p(i))+(n+1)/(m+1)\times P_{n,m}(S_k)\times(r1.p(i)\times r2.p(i))+$$
$$n/(m+1)\times P_{n,m}(S_k)\times((1-r1.p(i))\times r2.p(i)+r1.p(i)\times(1-r2.p(i))).$$

2. 对$eJS(r1\cup\{i:1.0\},r2\cup\{i:1.0\})$,根据状态$S_k$的转变规则,可以得到下面的值(相当于向右上移动一格)

$$value2=(n+1)/(m+1)\times P_{n,m}(S_k)\times(1\times 1).$$

因为有$0\leqslant n\leqslant m(0<m)$,因此$n/(m+1)\leqslant n/m\leqslant(n+1)/(m+1)$.可以得到下面不等式

$$value1\leqslant(n+1)/(m+1)\times P_{n,m}(S_k)\times[(1-r1.p(i))\times(1-r2.p(i))+r1.p(i)\times r2.p(i)+(1-r1.p(i))\times r2.p(i)+r1.p(i)\times(1-r2.p(i))]=$$
$$(n+1)/(m+1)\times P_{n,m}(S_k)\times 1=value2.$$

每个$P_{n,m}(S_k)$对$eJS(r1,r2)$的贡献都小于$eJS(r1\cup\{i:1.0\},r2\cup\{i:1.0\})$,因此可以得到:

$$eJS(r1,r2)\leqslant eJS(r1\cup\{i:1.0\},r2\cup\{i:1.0\}).$$

对于$r1$和$r2$的非共有项$i(i\in I(r1\cup r2)\backslash I(r1\cap r2))$,我们同样可以采用类似的方法来计算$P_{n,m}(S_k)$对期望相似度的贡献:

1. 在$eJS(r1,r2)$,根据状态S_k的转变规则,可以得到下面的值

$$value1=n/m\times P_{n,m}(S_k)\times(1-r1.p(i))\times(1-r2.p(i))+n/(m+1)\times P_{n,m}(S_k)\times((1-r1.p(i))\times r2.p(i)+r1.p(i)\times(1-r2.p(i)))$$
$$(r1.p(i)\times r2.p(i)=0).$$

2. 在$eJS(r1\backslash\{i\},r2\backslash\{i\})$,根据状态$S_k$的转变规则,可以得到下面的值(相当于向右上移动一格)

$$value2=n/m\times P_{n,m}(S_k)\times 1.$$

因为有$n/(m+1)\leqslant n/m$,可以得到下面不等式

$$value1\leqslant n/(m+1)\times P_{n,m}(S_k)\times[(1-r1.p(i))\times(1-r2.p(i))+(1-r1.p(i))\times r2.p(i)+r1.p(i)\times(1-r2.p(i))]=$$
$$n/(m+1)\times P_{n,m}(S_k)\times 1=value2.$$

因此,同样可以得到$eJS(r1,r2)\leqslant eJS(r1\backslash\{i\},r2\backslash\{i\})$.

以图 2 为例,假设 τ=0.75,判断 $r1$ 和 $r2$ 是否相似.计算 $r1$ 和 $r2$ 的相似度可以得到 $eJS(r1,r2)$=0.376,那么 $r1$,$r2$ 不属于相似结果,但是这样就需要计算完所有的项.利用上述的计算方法,可以更加有效地对非相似结果进行过滤.当计算完{A}之后,分析{B,C}我们发现,{B}属于 2)的情况,而{C}属 1)的情况.因此可以估计 $eJS(r1,r2)$的上限

$eJS(r1,r2)\leqslant eJS(r1\backslash\{B\},r2\backslash\{B\})\leqslant eJS(r1\backslash\{B\}\cup\{C{:}1.0\},r2\backslash\{B\}\cup\{C{:}1.0\})=eJS(\{A{:}1.0,C{:}1.0\},\{A{:}0.4,C{:}1.0\})$.

根据图 2 中的 $S1$ 以及公式(12),可以得到 $eJS(r1,r2)\leqslant eJS(\{A{:}1.0,C{:}1.0\},\{A{:}0.4,C{:}1.0\})=0.7<\tau$.只需计算一项,就可以判定 $r1$ 和 $r2$ 不是相似结果.

4 集合的连接查询

之前已经对一般的查询提出了解决方案,但是考虑到更广泛的应用场景,需要对一般的查询进行扩展.比如,在对网页的数据过滤、聚类、整合的过程中,往往需要将已知的若干网页(特征数据)作为查询条件,对抓取的网页数据流进行处理.我们将这类问题归于连接查询.本节将基于上述相似度度量,通过计算上界以避免相似度计算,来解决高效的不确定集合的连接查询问题.

定义 5. 定义两个不确定记录的数据集 R_1 和 R_2(如果 R_1=R_2,那么称为集合的自连接),从中找出所有的记录对,满足相似度不低于用户预设的阈值 τ,即$\{(r_1,r_2)|eSim(r_1,r_2)\geqslant\tau,r_1\in R_1,r_2\in R_2\}$.

算法 3. *FindSimPair*(R_1,R_2,τ).

Input: R_1, R_2 are the datasets of uncertain records;

Input: τ is the user-defined threshold of similarity;

Output: All *pairs*(r_1,r_2) that satisfied the *eSim* not less than τ.

```
1.  S←NULL
2.  FOR r1 in R1
3.      FOR r2 in R2
4.          IF upper_bound(r1,r2)<τ
5.              Continue;            //upper bound filter
6.          ENDIF
7.          IF eSim(r1,r2)>=τ
8.              S←S+(r1,r2);         //add to result set
9.          ENDIF
10.     ENDFOR
11. ENDFOR
12. return S;
```

算法 4. *upper_bound*(r_1,r_2).

Input: r_1 and r_2 are uncertain sets;

Output: Upper bound similarity of r_1 and r_2.

```
1. Union←I(r1∩r2)
2. r1' ←Union+c(r1);
3. r2' ←Union+c(r2)
4. upperbond←sim( r1',r2' )
5. return upperbond;
```

算法 3 考虑了从两个集合里面找出所有满足条件的记录对.在实际应用中,一般是一个较小的集合作为查询条件(常驻内存)对一个较大的集合(记录流)做查询.对记录流的处理一般作为外循环(R_1),较小的集合常驻内存作为 R_2.每个记录对先进行上届过滤,采用剪枝策略过滤那些明显不能成为结果的记录对.再对剩下的记录对进行相似度的计算,以判断是否符合用户指定的条件.

算法 4 主要在计算两个不确定记录之前,考虑第 3 节中的剪枝策略,预测相似值的上届.如果该值都无法满足用户预设的阈值,那么就不需要进行相似度的计算.根据剪枝策略中的性质 1,可以先将两个记录的共有元素的概率都增加到 1,即算法中的第 1 行.同时,根据性质 2 对相似度进一步的放大,可以将不确定(<1)的非共有元素去掉,这样计算的结果肯定是最终相似度的上届,即算法中的第 2 行和第 3 行.之后的计算就是两个确定型记录之间是计算.如果相似函数是 *Jaccard*,那么算法中第 1 行~第 4 行的复杂度都是 $O(n)$,那么该算法的复杂度就是 $O(n)$.

5 实 验

对于其他的相似函数(*Dice*,*Cosine*),我们也可以用动态规划计算和状态转移来实现.表 6 列举了几种常用的相似函数,包括定义、空间复杂度和时间复杂度.

Table 6 Definition and complexity of several similar functions

表 6 各种相似函数的定义及复杂度

Similar function	Definition	Spatial complexity	Time complexity
Jaccard	$\sum_{w1\in\{pw(u(r1))\}}\left\{\sum_{w2\in\{pw(u(r2))\}}\left(\frac{\lvert w1\cap w2\rvert+\lvert c(r1)\cap c(r2)\rvert}{\lvert w1\cup w2\rvert+\lvert c(r1)\cup c(r2)\rvert}\times p(\{w1\})\times p(\{w2\})\right)\right\}$	$O(n^2)$	$O(n^3)$
Cosine	$\sum_{w1\in\{pw(u(r1))\}}\left\{\sum_{w2\in\{pw(u(r2))\}}\left(\frac{\lvert w1\cap w2\rvert+\lvert c(r1)\cap c(r2)\rvert}{\sqrt{(\lvert w1\rvert+\lvert c(r1)\rvert)\times(\lvert w2\rvert+\lvert c(r2)\rvert)}}\times p(\{w1\})\times p(\{w2\})\right)\right\}$	$O(n^3)$	$O(n^4)$
Dice	$\sum_{w1\in\{pw(u(r1))\}}\left\{\sum_{w2\in\{pw(u(r2))\}}\left(\frac{\lvert w1\cap w2\rvert+\lvert c(r1)\cap c(r2)\rvert}{\lvert w1\rvert+\lvert w2\rvert+\lvert c(r1)\rvert+\lvert c(r2)\rvert}\times p(\{w1\})\times p(\{w2\})\right)\right\}$	$O(n^2)$	$O(n^3)$

从表 6 可以看出,复杂度依赖于计算公式分子和分母的各种情况.比如 *Jaccard*,依赖于各种可能世界的交和并的值,因此只需二维数组来存储各种情况,空间复杂度为 $O(n^2)$,时间复杂度为 $O(n^3)$;而 *Cosine* 依赖于可能世界的交以及各自的长度大小,因此需三维数组来存储各种情况,空间复杂度为 $O(n^3)$,时间复杂度为 $O(n^4)$.

本节将对不确定集合的查询做性能分析,在不同的相似函数以及不同的实验数据环境中做性能比对测试.

5.1 实验数据

我们从 DBLP 数据集里面抽取 5K 的记录(题目+作者)作为实验数据,将数据记录进行分词(token),每个分词作为记录(集合)的一个项,见表 7.

Table 7 Description of experimental data

表 7 实验数据说明

Data set	N	Avg_len	$\lvert I\rvert$	Max_len
DBLP	5k	10	9 976	43

元素的概率有均匀分布和高斯分布两种(uniform or Gaussian distribution),分别称为 U-DBLP 和 G-DBLP. 均匀分布的范围在[0.1,0.9]内;高斯分布采用 μ=0.5,σ=1,并且取值在[0.1,0.9]内.每次生成数据的时候,在记录里面选取 θ 比例的元素作为不确定的部分,其余部分为确定元素(即概率=1).表 8 是实验的参数设置,其中黑体部分表示缺省值.

Table 8 Setting of parameters

表 8 参数设置

Parameters	Values
τ	0.3, 0.4, **0.5**, 0.6, 0.7
θ	0.3, 0.4, **0.5**, 0.6, 0.7
N	1k, 2k, **3k**, 4k, 5k

5.2 实验分析

为了阐述 *eSim*/*eSim*+算法的性能,我们主要测试算法的执行时间.即在各个实验中,各种算法的总执行时间.目前还没有与 *eSim*/*eSim*+同类的算法,因此我们选择公式(6)定义的基本算法(Naïve 算法)作为基准测试;在第 2 节中提到的算法,我们称为基本的动态计算方法,用 *eSim* 来表示;在第 3 节中提到的算法,我们称为改进的动态计算方法,用 *eSim*+来表示.文中的算法用 Java 实现,所有的实验在 PC 机上进行:双核 CPU(2.00GHz),3G 内存.

本段将对实验结果进行分析,参数设置如上所述.主要对 Naive 算法、*eSim* 以及 *eSim*+进行比对,结果如图 3 所示,分 3 种情况进行测试:相似度阈值(τ)对执行时间的影响、不确定项的比例(θ)对执行时间的影响、数据规模(N)对执行时间的影响.各种算法分别在 U-DBLP 和 G-DBLP 数据集实验,测试的相似函数选取 *Jaccard* 和 *Cosine*.下面的实验将以算法 3 的连接查询进行.其中:一个集合较小(只有若干记录——预设置的查询条件),10 条记录,作为查询条件集合;另一个集合较大,模拟数据流,规模在 1K~5K.

图 3 是测试不同的τ对执行时间的影响.我们在 3K 的数据集里面找出所有相似度满足条件($\geq\tau$)的结果.从测试结果发现,τ对执行时间的影响很小.对 Naïve 算法,执行的过程中需要计算所有的可能世界以及对应的相似度,τ不影响可能世界的计算.*eSim* 需要利用动态规划来计算最终的状态,τ也不影响最终的状态的计算.*eSim*+中的上限计算与τ有关,但是影响并不明显(*eSim*+本身的执行时间已经很小了).

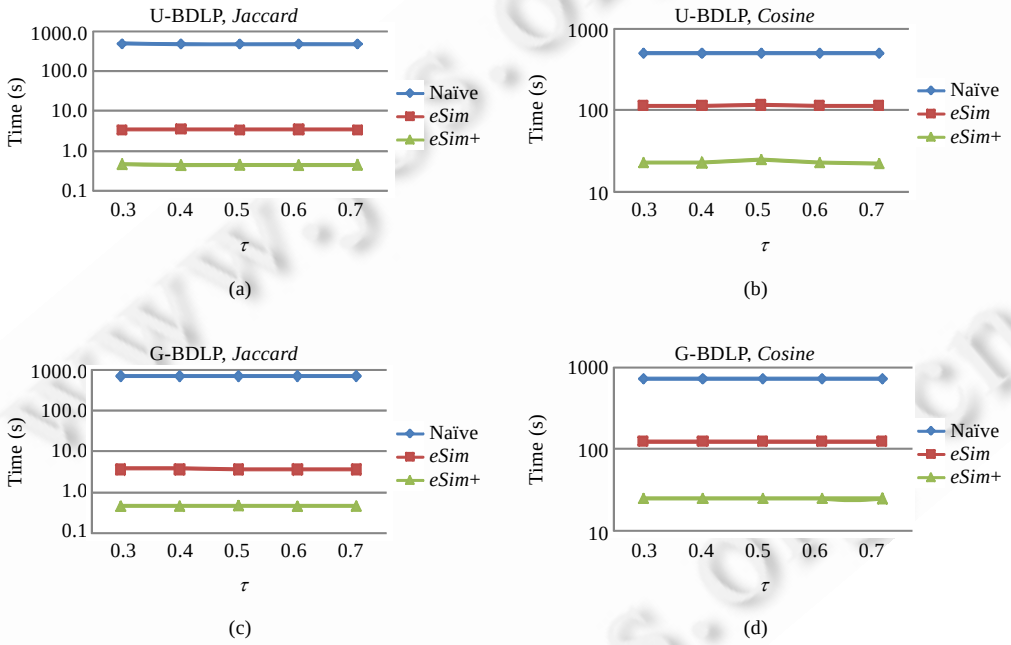

Fig.3 Experimental results of execution time vs similarity threshold (τ)

图 3 执行时间 vs.相似度(τ)的实验结果

图 4 是测试不同的θ对执行时间的影响,即测试记录中不确定项比例对性能的影响.从实验可以看出,θ的不同对性能影响很大.对 Naive 算法,执行的过程中需要计算所有的可能世界,θ越大,可能世界的数量就越多,计算复杂度就越大,趋势为指数级别.θ的不同对 *eSim* 和 *eSim*+的影响相对小多了.*eSim* 需要利用动态规划来计算最终的状态,θ对最终的状态的计算影响几乎可以忽略;*eSim*+中的计算过程与不确定项个数有关,因此影响比较明显,尤其是在 *Cosine* 的相似函数,其趋势为多项式级别.当θ很小的时候(比如θ=0.3),Naive 算法的执行时间反而比 *eSim* 小,这是因为在 Naive 计算可能世界与θ有关,而 *eSim* 算法里面的复杂度与集合大小相关,而不是与集合中不确定项的个数相关,*eSim* 的缺陷在算法 *eSim*+里面做了改进.

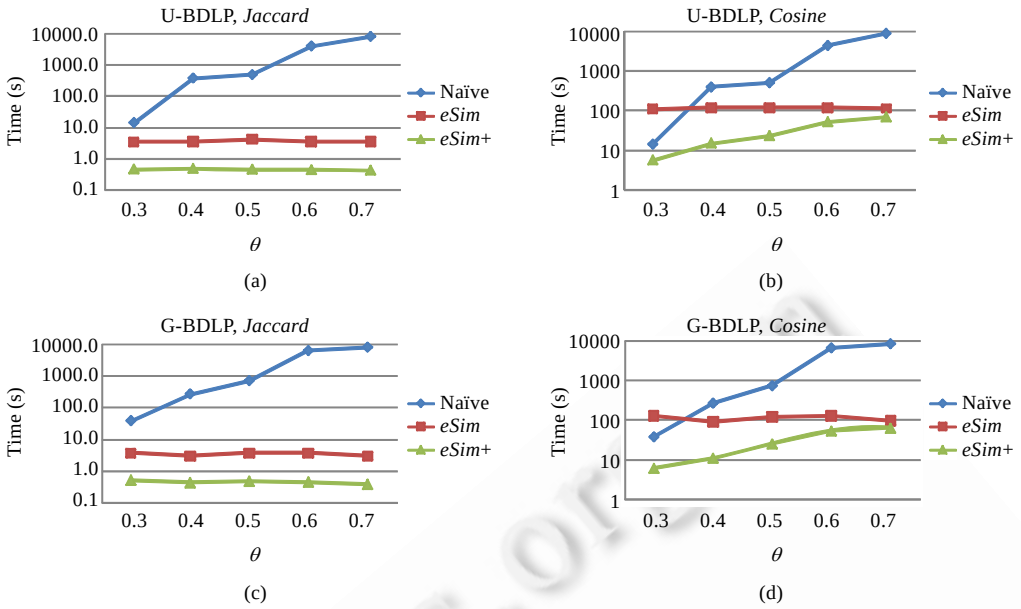

Fig.4 Experimental results of execution time vs uncertain proportion (θ)

图 4 执行时间与不确定比例(θ)的实验结果

图 5 是对不同数据集大小进行测试,各种算法的执行时间随数据集变大而递增,而且 Naïve 算法、*eSim* 和 *Sim*+的变化趋势基本一致.

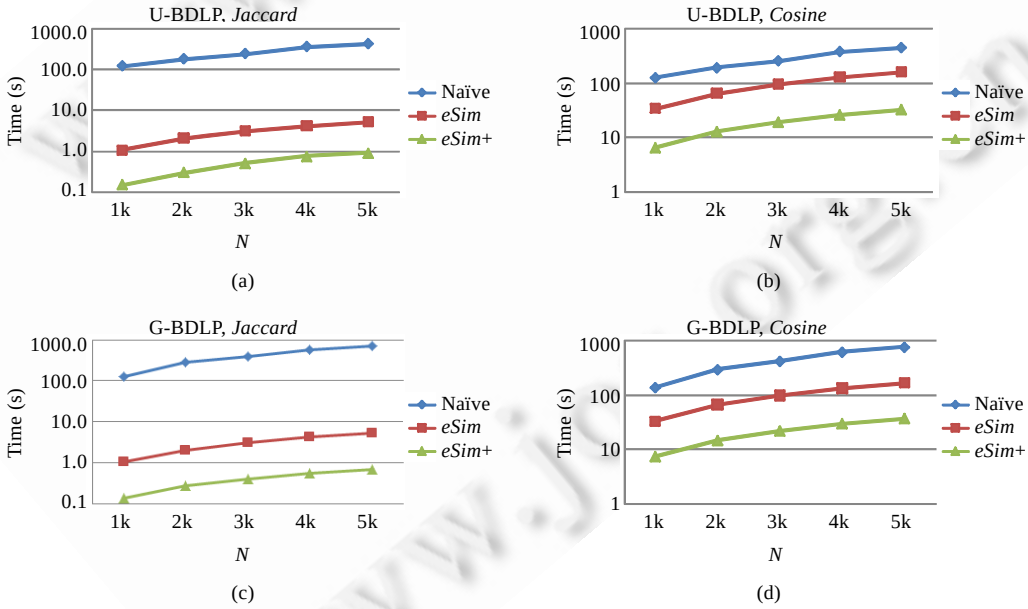

Fig.5 Experimental results of execution time vs scale of data (N)

图 5 执行时间与数据规模(N)的实验结果

从图中可以发现,各个算法在不同分布(G-DBLP 和 U-DBLP)的数据集里面影响不大.同时,针对不同的算法

(*Jaccard* 和 *Cosine*),Naive 算法的结果基本一致;而 *eSim* 和 *Sim+*的结果差距比较明显,符合各种算法的复杂度分析结果($O(n^4)$或者 $O(n^3)$).影响实验性能最大的是算法本身,从整体来看,当相似函数为 *Jaccard* 的时候,*eSim* 的性能约为 Naïve 算法的 2~3 个数量级别,而 *eSim+*的性能约为 Naïve 算法的 3~4 个数量级别.

6 相关工作

最新的各种研究中,集合的相似度查询被广泛地用于各种应用[1−7].数据集里的记录是一个集合,利用相似函数计算记录对的相似度,根据相似度判断记录对是否相似.但是这些研究主要集中在确定型的集合,而很少有研究是在不确定型集合的相似度计算.最新的不确定集合的相似度量是文献[14],该文在可能世界的基础之上,对两个不确定集合是否相似给出了定义:$p(sim(r,s)\geqslant\gamma)\geqslant\alpha$.即表示不确定集合 r 和 s,在各种可能世界里面,如果满足相似度$\geqslant\gamma$的可能世界概率$\geqslant\alpha$,则集合 r 和 s 相似.这种相似度量里面,需要用户提供两个阈值来进行判断,而阀值的提供,本身对用户来说是件很困难的事情.

不确定型数据的处理已经有相当多的研究了,主要包括模糊集合与概率的统一研究[8].对模糊集合的定义可以用概率来解释,并且与传统的集合定义进行了统一.对不确定集合的处理主要是进行频繁集的挖掘[10,11].此外,还有很多研究对不确定型数据库进行 top-k 的查询[9,12,13].可见,不确定型数据库的研究越来越流行,并且被广泛应用.而本文对不确定集合的相似研究,与文献[14]很接近,但又有本质的区别.

动态规则计算在不确定型数据库里面已有应用,如文献[9,12,13].本文将动态规则计算应用于期望相似查询的计算过程,极大地提高了计算效率.

7 总 结

本文提出了各种针对不确定集合的相似函数定义,基于可能世界计算期望相似度.通常情况下,计算期望相似度的复杂度是指数级的.本文提出了动态规划计算和转移规则,将时间和空间的复杂度从指数级降低到多项式级.为了更加有效地对非相似集合对进行相似查询,提出了几种用于计算相似度上限的方法.最后的实验证明了动态规划计算和过滤方法的有效和高性能.

在未来的研究工作中,我们将关注不确定集合的索引以及相似连接.虽然文中已经提出了几条剪枝策略,适用于不确定集合的查询,但是未必适用于不确定集合的索引和相似连接.因此,还需要进一步研究适用广泛性能更佳的剪枝策略.

References:

[1] Arasu A, Ganti V, Kaushik R. Efficient exact set-similarity joins. In: Proc. of the VLDB. 2006. 918−929.

[2] Theobald M, Siddharth J, Paepcke A. SpotSigs: Robust and efficient near duplicate detection in large Web collections. In: Proc. of the SIGIR. 2008. 563−570. [doi: 10.1145/1390334.1390431]

[3] Bayardo RJ, Ma YM, Srikant R. Scaling up all pairs similarity search. In: Proc. of the WWW. 2007. 131−140. [doi: 10.1145/1242572.1242591]

[4] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning. In: Proc. of the ICDE. 2006. 5−17. [doi: 10.1109/ICDE.2006.9]

[5] Elmagarmid AK, Ipeirotis PG, Verykios VS. Duplicate record detection: A survey. IEEE Trans. on Knowledge and Data Engineering, 2007,19(1):1−16. [doi: 10.1109/TKDE.2007.9]

[6] Sarawagi S, Kirpal A. Effcient set joins on similarity predicates. In: Proc. of the SIGMOD Conf. 2004. 743−754. [doi: 10.1145/1007568.1007652]

[7] Xiao C, Wang W, Lin XM, Yu JX. Effcient similarity joins for near duplicate detection. In: Proc. of the WWW. 2008. 131−140. [doi: 10.1145/1367497.1367516]

[8] Gao QS, Gao XY, Hu Y. A uniform definition of fuzzy set theory and fundamental of probability theory. Journal of Dalian University of Technology, 2006,46(1):141−150 (in Chinese with English abstract).

[9] Soliman MA, Ilyas IF, Chang KC. Top-*k* query processing in uncertain databases. In: Proc. of the ICDE. 2007. 896−905. [doi: 10.1109/ICDE.2007.367935]

[10] Bernecker T, Kriegel HP, Renz M, Verhein F, Züfle A. Probabilistic frequent itemset mining in uncertain databases. In: Proc. of the KDD. 2009. 119−128. [doi: 10.1145/1557019.1557039]

[11] Chui CK, Kao B, Hung E. Mining frequent itemsets from uncertain data. In: Proc. of the PAKDD. 2007. 47−58.

[12] Yi K, Li FF, Kollios G, Srivastava D. Efficient processing of top-*k* queries in uncertain databases with *x*-relations. IEEE Trans. on Knowledge and Data Engineering, 2008,20(12):1669−1682. [doi: 10.1109/TKDE.2008.90]

[13] Yi K, Li FF, Kollios G, Srivastava D. Efficient processing of top-*k* queries in uncertain databases. In: Proc. of the ICDE. 2008. 1406−1408. [doi: 10.1109/TKDE.2008.90]

[14] Lian X, Chen L. Set similarity join on probabilistic data. In: Proc. of the VLDB. 2010. 650−659.

[15] Redman T. The impact of poor data quality on the typical enterprise. Communications of the ACM, 1998,41(2):79−82. [doi: 10.1145/269012.269025]

[16] Gupta R, Sarawagi S. Creating probabilistic databases from information extraction models. In: Proc. of the VLDB. 2006. 965−976.

附中文参考文献:

[8] 高庆狮,高小宇,胡月.概率论基本部分与模糊集合理论的统一定义.大连理工大学学报,2006,46(1):141−150.

陈珂(1977－),女,河南郑州人,博士,副研究员,CCF 会员,主要研究领域为数据库,信息安全.

陈刚(1973－),男,博士,教授,博士生导师,主要研究领域为数据库,信息安全,协同设计.

洪银杰(1981－),男,博士,主要研究领域为数据库,数据挖掘.