一种基于模式的实体解析算法

刘辉平 金澈清 周傲英
(华东师范大学软件学院数据科学与工程研究院 上海 200062)

摘 要 实体解析是数据融合和数据清洗的关键步骤,旨在从大量的数据集中找出描述相同实体的记录.当前主要有两种基本的解决思路,一种是穷尽式的实体解析,通过两两比较数据集中的所有记录,然后再合并相似的记录,从而找到描述某一个实体的若干记录集合.然而,该方法的计算复杂度比较高($O(n^2)$,其中 n 表示数据集合的规模),难以处理大型数据集合.另一种思路是基于分块的实体解析,它调用特定的分块函数(如哈希函数、滑动窗口技术等)将集合中较为相似的记录划分到同一个块中,再仅对属于同一块中的记录进行两两比较.这种方法显著降低了运行时间,但会损失部分精度,因为某些描述同一实体的记录可能没有被分到同一个块中.文中提出了一种基于模式的实体解析算法,通过将相似的记录合并成记录集合并尝试生成对应的记录模式,然后进行模式之间的两两比较来产生一个边界值,以确定对应的记录集合是否需要进行进一步的精确比较,从而判断是否属于同一个实体.与第一种方法相比,该方法可有效地过滤部分不可能相似的记录,从而避免了针对所有数据记录进行两两比较,显著地降低了时间复杂度;与第二种方法相比,该方法并不损失任何精度.基于真实和模拟数据集合的实验结果验证了新方法的执行效率和有效性.

关键词 数据融合;数据清洗;实体解析;编辑距离;字符串相似度
中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2015.01796

A Pattern-Based Entity Resolution Algorithm

LIU Hui-Ping JIN Che-Qing ZHOU Ao-Ying
(*Institute for Data Science and Engineering, Software Engineering Institute, East China Normal University, Shanghai* 200062)

Abstract As a critical step in data integration and data cleaning, entity resolution (ER) aims at identifying groups of records that refer to the same real-world entity. Currently, there mainly exist two typical methods to handle this issue. One is exhaustive entity resolution, which compares all record pairs to determine the entity they belong to. However, its complexity ($O(n^2)$, n stands for the size of dataset) is too high to handle big volume dataset. The other is blocking-based entity resolution, which maps similar records to the same block by a specific method (e.g., hash function, sliding window, etc). Then only the records in the same block need to be compared. This method improves the efficiency while sacrifices the effectiveness. Since some records refer to the same entity may not in the same block. In this paper we propose a pattern-based entity resolution, which represents the similar records by a record pattern, then we will generate a bound by comparing record patterns. With this bound, we can decide if the two patterns' corresponding records need to be precisely compared to verify whether they refer to the same entity. In this way, we can both dramatically accelerate the process of entity resolution by filtering dissimilar records and ensure its correctness. Experiments on real and synthetic dataset show the efficiency and effectiveness of our method.

Keywords data integration; data cleaning; entity resolution; edit distance; string similarity

收稿日期:2014-07-06;最终修改稿收到日期:2015-04-03.本课题得到国家"九七三"重点基础研究发展规划项目基金(2012CB316203)、国家自然科学基金(61370101,61321064)、上海市教委科研创新重点项目(14ZZ045)资助.**刘辉平**,男,1990 年生,博士研究生,主要研究方向为数据质量、基于位置的服务.E-mail:hpliu1990@gmail.com.**金澈清**(通信作者),男,1977 年生,博士,教授,博士生导师,主要研究领域为数据流管理技术、不确定数据管理、数据质量、基于位置的服务.E-mail:cqjin@sei.ecnu.edu.cn.**周傲英**,男,1965 年生,教授,博士生导师,主要从事数据管理及应用研究,研究兴趣主要包括 Web 数据管理、数据密集型计算、内存集群计算、大数据基准测试和性能优化.

1 引　言

对不同来源的数据进行融合时会导致数据冗余现象，这不仅影响了数据的质量，也对数据处理造成了障碍[1-3]. 如何解决此一问题是数据清洗中的重要研究课题. 实体解析(Entity Resolution)通过对数据集进行分析[1,4-6]，从而发现描述同一现实世界中的实体的若干数据记录(Record).

实体解析一般包含两种基本操作：匹配操作(Match)和合并操作(Merge)[7-10]. 匹配操作用于判断两条记录是否对应于同一个实体(Entity). 给定一个相似度度量和阈值 δ，若记录间的相似度大于 δ，则记录匹配，记为 $\approx$，否则为不匹配，记为 $\not\approx$. 合并操作则用于合并匹配的记录，使它们成为一个记录集合共同对应同一实体. 图 1 以人名为例展示了实体解析的一个完整过程. 它先对记录进行两两匹配(编辑距离小于 4 认为匹配)，然后利用传递规则(若 $a\approx b, b\approx c$，则将 a,b,c 合并)将匹配的记录进行合并生成两个记录集合. 实体解析完成后，为了最终消除冗余数据达到数据清洗的目的，可以进行实体生成，为每个记录集合生成一个唯一的实体. 图 1 中最后生成两个实体 E_1、E_2.

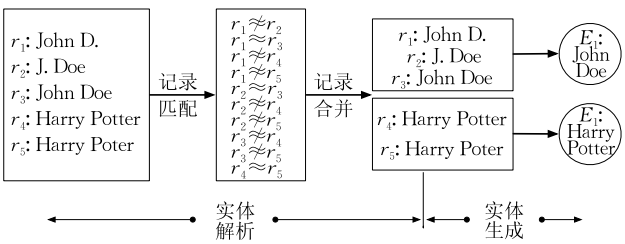

图 1　实体解析过程

由图 1 可以看出一个完整的实体解析需要进行 $\binom{n}{2}$ 次匹配操作(n 为记录条数)，其计算复杂度为 $O(n^2)$. 显然对于大的数据集，这种复杂度是不能容忍的. 为了减少匹配操作的次数，Benjelloun 等人[7]提出了 R-Swoosh 实体解析算法框架. 只要匹配操作和合并操作满足 ICAR 特性(幂等性、交换性、结合性、可被代表性)，该框架就能有效地降低匹配操作的次数，从而提高实体解析的效率. ICAR 特性中最重要的一点是可被代表性[11]，即若一条记录与某一记录集合中的任意一条记录匹配，则该条记录与该记录集合匹配，需要对它们进行合并. 为了方便，本文将没有进行合并的记录也看做记录集合，因此对于任意一个记录集合 R，它的基数(包含的记录个数)$|R|\geq 1$. 如图 1，利用可被代表性，如果第一次匹配之后 $r_1\approx r_3$，合并形成 $\{r_1,r_3\}$；第二次匹配之后 $r_2\approx r_3$，则 $r_2\approx\{r_1,r_3\}$，合并形成 $\{r_1,r_2,r_3\}$，这样避免了后面 r_2 与 r_1 间的匹配. 这种特性被广泛采用[2,8,12-13]，因为它可以避免记录与记录集合中的其他记录进行不必要的匹配. 但是如果记录集合间没有匹配的记录对，还是需要对集合中所有的记录对进行一一比较. 如图 2 所示，记录集合 R_1 和记录集合 R_2 需要进行完整的 $|R_1|\times|R_2|=3\times 2=6$ 次匹配，才能确定 R_1、R_2 不匹配. 事实上，由于大多数的记录之间都是不匹配的，所以这种不匹配的比较操作次数会占整个实体解析过程中匹配操作数的大部分.

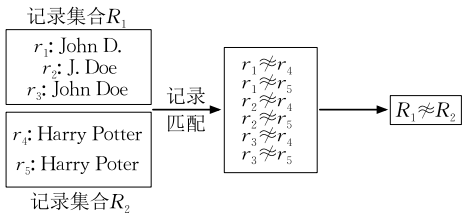

图 2　一般的记录匹配

为了解决这个问题，本文提出了一种基于模式(Pattern)的实体解析算法. 它用特定的模式来描述一个记录集合(见 3.3 节)，并采用模式相似度算法(见 3.2 节)计算模式相似度作为对应记录集合间相似度的边界值，根据相似度边界值与阈值间的比较以确定是否需要对记录集合间的记录进行精确的两两匹配. 图 3 描述了一个例子，记录集合 R_1 和记录集合 R_2，其对应的记录模式分别为 P_1、P_2. 由于 P_1、P_2 不匹配，所以其相应的记录集合 R_1、R_2 也不匹配. 通过这种方法，可以过滤掉那些显著地不可能相似的记录集合对，以避免对记录集合中的所有记录进行两两匹配，起到了很好的剪枝作用，从而在提升了算法运行效率的同时又不牺牲结果精度. 此外，还可利用所生成的模式生成实体(见 4.2 节).

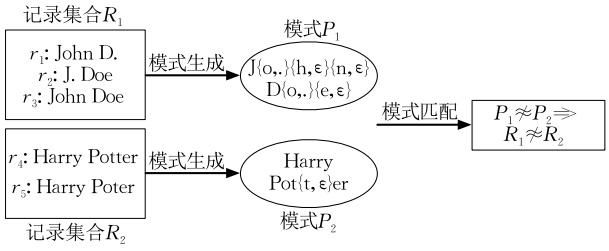

图 3　基于模式的匹配

总的来说，本文的贡献如下：

(1) 提出了一套基于模式的实体解析和实体生成的算法框架. 该框架在不影响准确度上提高了实

体解析的效率.

(2)基于编辑距离设计开发了模式生成和模式匹配的算法,使得模式间的相似度能够成为记录集合相似度的一个边界值,从而达到剪枝的目的.

(3)在真实数据集和模拟数据集上进行了充分的实验,实验结果表明本文提出的基于模式的算法框架在实体解析和实体生成阶段跟现有的方法相比具有良好的效果.

本文第 2 节介绍相关工作;第 3 节详述模式的相关概念和模式生成以及模式相似度计算算法;第 4 节介绍基于模式的实体解析框架和基于模式的实体生成算法;第 5 节通过实验来验证本文所提算法的有效性和高效性;第 6 节总结全文.

2 相关工作

2.1 实体解析

为了发现数据集合中表示现实世界中同一实体的数据记录,实体解析问题被大量研究.从准确度和效率两方面考虑,实体解析可以大致分为两类:穷尽式实体解析(Exhaustive ER)和基于分块的实体解析(Blocking-based ER)[12].

穷尽式的实体解析[7-8,13-14]需要遍历整个数据集来达到高准确度.所以穷尽式的实体解析结果一般也被当做标准答案来使用[12,15].为了减少穷尽式实体解析中的匹配次数,文献[8,13]通过选取记录集合中的一条记录作为记录集合的代表记录.通过代表记录间的匹配来决定其对应的记录集合间是否匹配.但是由于代表记录选取的随机性和不稳定性,会对实体解析的结果准确性造成影响.文献[7]提出 R-Swoosh 框架利用 ICAR 属性可以获得高精确度,但是对于不匹配的记录,还是需要进行完全的两两比较.本文通过基于模式的方法很好地解决了这个问题.

另一方面,为了提高实体解析的效率,各种基于分块的实体解析算法被提出[2,9-10,16-17].文献[9]利用哈希函数将记录映射到不同的块中,然后对相同块中的记录进行匹配操作.文献[2,10]先将记录按某个属性进行排序,然后对排序的记录利用滑动窗口进行扫描,只有在同一窗口中的记录才进行两两匹配.文献[16,18]将记录属性变成若干个 gram,然后将 gram 作为分块的依据.文献[17]先通过简单高效的距离度量将记录粗略地分到各个子集中,然后在每个子集里用精确的距离度量对记录进行两两匹配.上述分块算法都大大提高了实体解析的效率,但是由于匹配的记录可能没有被分到同一个块中,所以都在某种程度上降低了实体解析的准确度.本文提出的算法在提高实体解析效率的同时能够保证实体解析的准确度.

2.2 相似度计算

实体解析的匹配操作需要利用相似度度量计算记录间的相似度从而判断它们是否属于同一个实体.常见的相似度(或者距离)度量有编辑距离、Jaccard 相似度、余弦相似度、欧式距离等.其中编辑距离[19]是计算两条给定的字符串之间距离的经典度量,广泛应用于实体解析的记录匹配[2,4,10,20].其采用删除、插入和替换三种基本操作来把一条字符串转换为另一条字符串,用最少操作次数来表示该两条字符串之间的距离.距离越小表示字符串越相似.本文用 $d(s_1,s_2)$ 来表示字符串 s_1 和 s_2 之间的编辑距离,显然 $d(s_1,s_2)\leq\max(|s_1|,|s_2|)$.编辑距离一般采用动态规划(Dynamic Programming)来计算,其递推公式如下:

$$d[i,j]=\min\begin{cases}d[i,j-1]+1\\ d[i-1,j]+1\\ d[i-1,j-1]+c(s_1[i],s_2[j])\end{cases},$$

其中

$$c(s_1[i],s_2[j])=\begin{cases}1, & s_1[i]\neq s_2[j]\\ 0, & s_1[i]=s_2[j]\end{cases},$$

这里 $s[i]$ 表示字符串 s 的第 i 个字符.而且 $d[i,j]=i+j$ 当 $i\times j=0$.基于这种距离通过对它进行归一化可以得到字符串间的编辑距离相似度 $d_{\text{sim}}(s_1,s_2)$.其计算方式如下:

$$d_{\text{sim}}(s_1,s_2)=1-\frac{d(s_1,s_2)}{\max(|s_1|,|s_2|)}.$$

显然 $d_{\text{sim}}(s_1,s_2)\in[0,1]$.本文采用编辑距离计算记录相似度且基于编辑距离设计模式的生成和相似度计算算法.

3 记录模式

3.1 记录模式的相关概念

注意到每一个记录集合中的记录之间都是有一定相似性的,因为它们都是通过匹配之后合并到一起成为一个记录集合,对应于现实世界中的同一个实体.基于这种特性,可以发现和抽取记录集合中记录的共同特征,对它们相同的部分进行统一,不同的部分进行保留,从而形成一个特定的模式.那么对于

一个记录集合的处理就可以直接对该记录集合所对应的模式进行处理. 下面对模式进行形式化定义.

定义 1. 令 Σ 表示字母表，ε 表示空字符. 一个模式：$P=P[1]P[2]\cdots P[n]$. 其中对于任意的 $i=1,\cdots,n$，元素 $P[i]=\{c_{i,1},\cdots,c_{i,n_i}\}$（对于 $j=1,\cdots,n_i$，有 $c_{i,j}\in\Sigma\cup\{\varepsilon\}$）. 其中 $P[i]$ 的长度 $|P[i]|=n_i$，模式 P 的长度 $|P|=n$.

由以上定义可知任一模式能够构成的字符串的个数为 $\prod_{i=1}^{n}|P[i]|$. 这些可能构成的字符串称为模式的实例. 例如模式 P：h{e,a}llo wor{l,ε}d，其实例集为{hallo world, hello word, hallo word, hello world}.

定义 2. 给定记录集合 R 和模式 P，P 的实例集为 Q. 若 $\forall r_i\in R$，都有 $r_i\in Q$. 则称 P 为 R 的一个模式.

根据上面的定义容易看出对于只包含一条记录的记录集合，其模式就是该条记录本身. 但是对于包含多条记录的记录集合，情况就要复杂得多. 例如记录集合 R：{hallo word, hello world}的对应模式可以是 P：h{e,a}llo wor{l,ε}d. 这时引入了两个不相关实例：hello word 和 hallo world. 本文的 3.3 节将给出基于编辑距离的模式生成算法.

现在可以用模式代替记录解决实体解析问题. 同样我们需要利用模式相似度对模式进行匹配操作，若其相似度超过阈值，则模式匹配，需要对模式所对应的记录集合进行合并操作并更新对应新记录集合的模式. 为了利用模式进行剪枝，我们定义模式间的相似度是其对应实例间相似度的最大值. 本文的 3.2 节将给出基于编辑距离的模式相似度算法.

实体解析中，对于任意两个记录集合，若它们不存在一对匹配记录，则显然该两个记录集合不可能匹配. 同样对于模式而言，若模式的实例集中不存在一对匹配的实例，则模式之间也不可能匹配. 基于此我们得到以下实体解析的剪枝规则(pruning rule)：

剪枝规则. 对于两个记录集合 R_1 和 R_2，它们对应的模式分别为 P_1 和 P_2. 则

(1) 若 $P_1\approx P_2$，则可能 $R_1\approx R_2$.

(2) 若 $P_1\not\approx P_2$，则 $R_1\not\approx R_2$.

设 P_1、P_2 对应的实例集分别为 Q_1 和 Q_2. 对于(1)，因为 $P_1\approx P_2$，故 Q_1、Q_2 间存在匹配的实例. 由于 $R_1\subseteq Q_1$，$R_2\subseteq Q_2$，所以若匹配的实例(记录)对存在于 R_1、R_2，则可能有 $R_1\approx R_2$，否则 $R_1\not\approx R_2$. 对于(2)，由于 $P_1\not\approx P_2$，故 Q_1、Q_2 中不存在匹配的实例，由定义 2 知 R_1、R_2 中也不存在匹配的记录，故有 $R_1\not\approx R_2$.

由以上剪枝规则可知，模式间的相似度可以看成记录集合间相似度的一个上界. 对于剪枝规则中的(2)通过对模式进行一次匹配，如果 $P_1\not\approx P_2$，即实例间相似度的最大值小于阈值 δ，则可判断记录集合的匹配情况为 $R_1\not\approx R_2$. 因为 R_1、R_2 中不可能存在相似度高于阈值 δ 的记录对. 此时不需要再对 R_1、R_2 中的记录进行两两匹配. 可以大大节省实体解析的运行时间，极大提高实体解析的效率. 而对于(1)，通过 $P_1\approx P_2$，不能直接得到 $R_1\approx R_2$. 因此为了不对结果的准确性造成影响，需要做进一步的验证. 这种情况下浪费了一次模式匹配，但是模式匹配的复杂度相当于一次记录间的匹配，因此对实体解析的影响并不是很大. 而且通常情况下，大部分记录都是不匹配的，因此(2)出现的情况远远高于(1)出现的情况，所以(2)对实体解析效率的提升远远大于(1)的消耗. 正是由于(2)对实体解析中匹配操作的大量剪枝，使得整体上基于模式的实体解析在不影响结果准确性的同时能对实体解析的效率进行显著地提升. 第 5 节的实验将进行非常好的说明和验证.

3.2 模式相似度

现在可以将记录间的匹配操作转换成其对应的模式之间的匹配操作. 通过模式间的匹配进而确定对应记录集合是否匹配. 本文设计一种基于编辑距离的相似度算法来对模式进行相似度计算，使得模式间的相似度是其对应实例间相似度的最大值，从而利用剪枝规则.

由于模式中的元素可能有多个字符而且还有空字符 ε 的存在，所以不能直接将字符串的编辑距离用于模式之间的距离. 为了计算模式之间的距离，需要根据模式的特点对编辑距离进行改编. 为了解决多字符和空字符的问题，本文采用如下设计原则：

(1) 对于多字符. 如果两个模式元素中含有相同的字符(包括空字符)，则该两个模式元素匹配.

(2) 对于空字符. 如果不匹配，则包含空字符的模式元素这时取空字符，此时其不占字符长度，对编辑距离没有影响.

通过这种处理方式可以找出两个模式中编辑距离最小的实例对. 用这个最小距离作为模式的编辑距离. 模式编辑距离 $\boldsymbol{D}(P_1,P_2)$ 的递推公式如下：

$$\boldsymbol{D}[i,j]=\min\begin{cases}\boldsymbol{D}[i,j-1]+A(P_2[j])\\ \boldsymbol{D}[i-1,j]+A(P_1[i])\\ \boldsymbol{D}[i-1,j-1]+C(P_1[i],P_2[j])\end{cases},$$

其中

$$A(P[i])=\begin{cases}1, & \varepsilon\notin P[i]\\ 0, & \varepsilon\in P[i]\end{cases},$$

$$C(P_1[i],P_2[j])=\begin{cases}1, & P_1[i]\bigcap P_2[j]=\varnothing\\ 0, & P_1[i]\bigcap P_2[j]\neq\varnothing\end{cases},$$

这里 $P[i]$ 表示模式 P 的第 i 个元素. 且初始条件为

$$\boldsymbol{D}[i,j]=\min\begin{cases}0, & i=0,j=0\\ \boldsymbol{D}[i,j-1]+A(P_2[j]), & i=0\\ \boldsymbol{D}[i-1,j]+A(P_1[i]), & j=0\end{cases}.$$

上述递推公式中我们采用了两个惩罚函数:A 函数和 C 函数,来解决模式中的多字符和空字符问题. 其中 C 函数对应设计原则(1),只要模式元素间含有相同字符,则认为匹配. 这样若记录对应的字符串间编辑距离某一位置字符匹配,则对应的模式元素间该位置必定也会匹配. A 函数对应设计原则(2),若模式元素中含空字符,则可以直接将其忽略,不影响整个编辑距离操作,保证编辑距离维持最小. 最后两模式的编辑距离即为 $\boldsymbol{D}[|P_1|,|P_2|]$. 特别地,对于两个只包含一条记录的记录集合,其模式的编辑距离等于记录的编辑距离.

下面证明模式的编辑距离不大于对应记录集合的记录间的最小编辑距离. 即证:

定理 1. 给定两个记录集合 R_1 和 R_2,它们对应的模式分别为 P_1 和 P_2. 则

$$\boldsymbol{D}(P_1,P_2)\leq\min_{r_i\in R_1,r_j\in R_2}d(r_i,r_j).$$

证明. 对于初始情况 $i\cdot j=0$ 时,根据模式的定义和模式编辑距离的初始化规则,有 $\forall r_i\in R_1,r_j\in R_2$,$\boldsymbol{D}[i,j]\leq d[i,j]$. 对于 $i\cdot j\neq 0$,由于 $A(P[i])\leq 1$ 且 $C(P_1[i],P_2[j])\leq c(s_1[i],s_2[j])$. 通过归纳推导,可以得到对于任意 $r_i\in R_1,r_j\in R_2$,有 $D(P_1,P_2)\leq d(r_i,r_j)$. 证毕.

对模式编辑距离归一化可以得到模式的编辑距离相似度,简称模式相似度 $D_{\text{sim}}(P_1,P_2)$:

$$D_{\text{sim}}(P_1,P_2)=1-\frac{\boldsymbol{D}(P_1,P_2)}{\max(|P_1|,|P_2|)}.$$

与字符串编辑距离相似度类似,模式相似度 $D_{\text{sim}}(P_1,P_2)\in[0,1]$. 当两模式的实例集中存在相同的实例,则其模式编辑距离为 0,此时模式相似度最大为 1. 反之当两模式的实例集完全不同时,其模式相似度最小可以为 0. 通过模式相似度,利用定理 1,可以得到下面的结论.

引理 1. 模式编辑距离相似度满足剪枝规则.

证明. 任设两个记录集合 R_1 和 R_2,它们对应的模式分别为 P_1 和 P_2. 再设给定相似度阈值为 $\delta\in[0,1]$,R_1、R_2 中相似度最高的记录对为 r_i、r_j. 则有

$$d_{\text{sim}}(r_i,r_j)=1-\frac{d(r_i,r_j)}{\max(|r_i|,|r_j|)},$$

$$D_{\text{sim}}(P_1,P_2)=1-\frac{\boldsymbol{D}(P_1,P_2)}{\max(|P_1|,|P_2|)}.$$

由定理 1 有

$$\boldsymbol{D}(P_1,P_2)\leq d(r_i,r_j).$$

再由模式的定义有

$$\max(|P_1|,|P_2|)\geq\max(|r_i|,|r_j|).$$

这里的 $|r_i|,|r_j|$ 分别表示 r_i,r_j 对应字符串的长度. 所以得到

$$\frac{\boldsymbol{D}(P_1,P_2)}{\max(|P_1|,|P_2|)}\leq\frac{d(r_i,r_j)}{\max(|r_i|,|r_j|)},$$

因此

$$D_{\text{sim}}(P_1,P_2)\geq d_{\text{sim}}(r_i,r_j).$$

故有

(1)若 $P_1\approx P_2$,即 $D_{\text{sim}}(P_1,P_2)\geq\delta$. 此时 $d_{\text{sim}}(r_i,r_j)$ 与 δ 关系不确定,故可能有 $R_1\approx R_2$ 或 $R_1\not\approx R_2$.

(2)若 $P_1\not\approx P_2$,即 $D_{\text{sim}}(P_1,P_2)<\delta$,有 $\delta>D_{\text{sim}}(P_1,P_2)\geq d_{\text{sim}}(r_i,r_j)$. 故 $r_i\not\approx r_j$,所以 $R_1\not\approx R_2$.

因此模式编辑距离相似度满足剪枝规则. 证毕.

由上述证明可知模式间的相似度是记录集合相似度的一个上界. 事实上,可以继续对这个上界进行压缩,使得其变得更加紧凑,这样其剪枝效果也会变得更好. 由模式的定义很容易有

$$|P|\geq\max_{r'\in R}|r'|,$$

其中 R 为记录集合,P 为 R 对应的模式,$|r'|$ 为 R 中任一记录的字符串长度. 因此根据引理 1 的证明可以得到

$$\begin{aligned}\max(|P_1|,|P_2|)&\geq\max(|r'|,|r''|)\\&\geq\max(|r_i|,|r_j|),\end{aligned}$$

其中 $r'\in R_1,r''\in R_2$. 不妨设

$$D'_{\text{sim}}(P_1,P_2)=1-\frac{\boldsymbol{D}(P_1,P_2)}{\max(|r'|,|r''|)}.$$

则

$$D_{\text{sim}}(P_1,P_2)\geq D'_{\text{sim}}(P_1,P_2)\geq d_{\text{sim}}(r_i,r_j).$$

此时可以将记录集合相似度的上界从 $D_{\text{sim}}(P_1,P_2)$ 缩小到 $D'_{\text{sim}}(P_1,P_2)$. 第 5 节的实验采用的是 $D'_{\text{sim}}(P_1,P_2)$.

3.3 模式生成

由上文可知,对于只包含一条记录的记录集合,其模式为其记录本身. 而对于包含多条记录的记录集合,由于其是由多个记录集合合并而成,直接对记录集合的记录字符取并集生成模式并不是一种非常好的选择. 如记录集合 R:{hallo word, hello world},

直接取记录字符的并集生成的模式 P'：h{e,a}llo wor{l,d}{d,ε}，其可能的实例数为 2×2×2=8. 引入了 6 个不相关实例. 显然更好的模式应该为 P：h{e,a}llo wor{l,ε}d，其可能的实例数为 2×2=4，引入了 2 个不相关实例. 相比于 P'，P 在覆盖记录集合 R 的同时引入的不相关实例更少. 基于此，本文并不直接对记录集合生成模式，而是利用其子记录集合进行模式的编辑距离匹配后生成的距离矩阵进行回溯来得到该记录集合的模式. 这样通过距离矩阵考虑记录字符间的对应关系可以减少生成模式的可能实例数，尽可能少地引入不相关的实例. 例如 R_1 和 R_2 对应的模式分别为 P_1 和 P_2. 若 $R_1 \approx R_2$，此时需要对 R_1、R_2 进行合并操作. 设合并后的新记录集合为 R，则 R 的模式 P 为 P_1、P_2 的共同模式，也即模式的模式.

通过模式的编辑距离生成的距离矩阵 $\boldsymbol{D}$，从 $\boldsymbol{D}[|P_1|,|P_2|]$ 出发根据最短距离进行回溯到 $\boldsymbol{D}[0,0]$，即可得到模式的模式. 其具体的生成规则如下：

$$P[k]=\begin{cases} P_1[i]\cup\varepsilon, & \boldsymbol{D}[i,j]=\boldsymbol{D}[i,j-1]+A(P_2[j]) \\ P_2[j]\cup\varepsilon, & \boldsymbol{D}[i,j]=\boldsymbol{D}[i-1,j]+A(P_1[i]) \\ P_1[i]\cup P_2[j], & \boldsymbol{D}[i,j]=\boldsymbol{D}[i-1,j-1]+C(P_1[i],P_2[j]) \end{cases}.$$

特别地

$$P[k]=\begin{cases} P_1[i]\cup\varepsilon, & j=0 \\ P_2[j]\cup\varepsilon, & i=0 \end{cases},$$

其中 k 为回溯中的某个位置，直到 $i=0$，$j=0$ 时回溯完毕. 这时 P_1、P_2 的模式 $P=\cdots P[k]\cdots P[0]$（k 按回溯顺序倒序排列）. 特殊情况下，当生成规则中的 3 个条件同时满足时，C 函数的优先级高于 A 函数. 这样有助于相同字符的合并，从而减少引入不相关的实例.

下面我们证明以上生成规则得到的模式 P 是新记录集合 R 的模式.

证明.　由生成规则可知，共有两种情况：

(1) $P_1[i]$ 和 $P_2[j]$ 对齐（匹配），则同时保留 $P_1[i]$ 和 $P_2[j]$ 作为该位置的模式元素.

(2) $P_1[i]$ 和 $P_2[j]$ 不对齐（不匹配），此时我们用空字符与 $P_1[i]$ 或 $P_2[j]$ 匹配，这样既没有引入脏字符又可以保留 $P_1[i]$ 或 $P_2[j]$ 的必要字符.

故而新模式 P 能够有效保留 P_1、P_2 的必要单元. 设 P、P_1、P_2 的实例集分别为 Q、Q_1、Q_2，则 $(Q_1\cup Q_2)\subseteq Q$. 因为 P_1、P_2 分别为 R_1 和 R_2 的模式，所以有 $R_1\subseteq Q_1$，$R_2\subseteq Q_2$，$(R_1\cup R_2)\subseteq Q$. 又 $R=R_1\cup R_2$，故有 $R\subseteq Q$. 所以 P 为 R 的模式.　　证毕.

下面我们用一个例子来说明模式相似度的计算和模式生成的过程. 设记录集合 R_1：{hallo word}，R_2：{hello world}，显然其对应的模式分别为 P_1：hallo word，P_2：hello world. 根据模式相似度的计算算法，P_1、P_2 模式编辑距离的距离矩阵如表 1 所示（为了方便，表中省略了单词间的空格）. 其中矩阵中的每个值 $\boldsymbol{D}[i,j]$ 表示 P_1 的前 i 个元素与 P_2 的前 j 个元素间的编辑距离. 由于此时的模式为记录本身，所以模式间的编辑距离等于其对应记录间的编辑距离. 故 P_1、P_2 模式编辑距离为 $\boldsymbol{D}[9,10]=2$. 其模式相似度为 $1-2/10=0.8$. 假设阈值 $\delta=0.7$，则 R_1、R_2 匹配，此时需要将 R_1、R_2 合并，形成新的记录集合 R：{hallo word, hello world}. 与此同时，需要产生 R 对应的模式 P. 如表 1，根据模式的生成规则，在模式距离矩阵 $\boldsymbol{D}$ 的基础上从 $\boldsymbol{D}[9,10]$ 开始回溯. 由于 $\boldsymbol{D}[9,10]=\boldsymbol{D}[9,9]+A(P_1[9])=\boldsymbol{D}[8,9]+C(P_1[9],P_2[10])=2$，根据 C 函数优先原则，此时的 $P[k]=\{d\}\cup\{d\}=\{d\}$. 因此下一位置从 $\boldsymbol{D}[8,9]$ 开始回溯直到 $\boldsymbol{D}[0,0]$ 结束，最后形成的模式 P 为 h{e,a}llo wor{l,ε}d.

表 1　模式相似度的计算和模式生成矩阵

		h	e	l	l	o	w	o	r	l	d
	0	1	2	3	4	5	6	7	8	9	10
h	1	0	1	2	3	4	5	6	7	8	9
a	2	1	1	2	3	4	5	6	7	8	9
l	3	2	2	1	2	3	4	5	6	7	8
l	4	3	3	2	1	2	3	4	5	6	7
o	5	4	4	3	2	1	2	3	4	5	6
w	6	5	5	4	3	2	1	2	3	4	5
o	7	6	6	5	4	3	2	1	2	3	4
r	8	7	7	6	5	4	3	2	1	2	3
d	9	8	8	7	6	5	4	3	2	1	2

3.4　模式的时空复杂度

对于模式的相似度计算，其中计算 A 函数和 C 函数时由于需要对模式元素中的每个字符进行匹配，所以其计算复杂度为 $O(\sum_{i=1}^{n}|P_1[i]|\sum_{i=1}^{m}|P_2[i]|)$，而其空间复杂度与字符串的编辑距离类似为 $O(|P_1||P_2|)$. 而对于模式的生成，其可以直接利用模式相似度计算所生成的距离矩阵进行线性回溯，所以其时空复杂度均为 $O(|P|)$，其中 P 为生成的模式.

4 基于模式的实体解析框架

通过第 3 节的介绍,本文完整地给出了记录所对应模式的生成和模式相似度的计算算法.并且证明了这两种算法的有效性.本节我们介绍基于上述模式算法的实体解析框架,该框架能够很好地利用模式的剪枝规则通过迭代式的计算在不影响准确率的情况下使得运行时间大大降低.

4.1 基本框架

一个有效的实体解析框架能够利用匹配和合并两种基本操作产生出实体解析结果集,使得该结果集满足:

(1)对源数据集的一个完整划分;

(2)对于任意记录只能属于唯一记录集合;

(3)结果集中不存在能够匹配的记录集合,使得每一个记录集合都对应一个不同的实体.

基于模式我们设计了一个迭代式的实体解析算法 IterER.其伪代码见算法 1.

算法 1. IterER:产生实体解析的输出结果集.

输入:记录集合 I,阈值 δ

输出:满足条件的结果集 I'

```
1. FOR 记录 r IN I DO
2.     r.pattern←r;
3.     r.round←0;
4. cRound←0;
5. WHILE TRUE DO
6.     T←∅;
7.     FOR 记录 r IN I DO
8.         从 I 中移除 r;
9.         matchingRecords←∅;
10.        FOR 记录 r′ IN I DO
11.            IF r.round≥cRound OR r′.round≥cRound
               THEN
12.                计算 r.pattern 和 r′.pattern 相似度 e;
13.                IF e≥δ AND r≈r′ THEN
14.                    产生 r.pattern 和 r′.pattern 的模式 p;
15.                    r←合并 r,r′;
16.                    r.pattern←p;
17.                    r.round←cRound+1;
18.                    添加 r′到 matchingRecords;
19.        FOR 记录 r′ IN matchingRecords DO
20.            从 I 中移除 r′;
21.        添加 r 到 T;
22.    cRound←cRound+1;
23.    IF T 的集合大小不变 THEN
24.        BREAK;
25.    ELSE I←T;
26. I′←T;
27. RETURN I′;
```

该算法的输入为需要进行实体解析的记录集合以及相似度的阈值 δ,输出为满足上述条件的实体解析结果集.对于给定的记录集 I,首先对其中的每个记录进行初始化,将每个记录看成包含一条记录的记录集合,其初始模式为其本身.为了叙述方便,下文将记录集合也看做记录.由于本文的算法是迭代式的,为了避免每一轮迭代中相同的记录对重复比较,我们为每一个记录添加了 *round* 属性,其初始值为 0(1~3 行).对于每一轮迭代,我们首先取 I 中的任一记录 r 与 I 中其他记录进行比较.为了避免重复比较,只有记录对中有一个记录的 *round* 值不小于当前的迭代轮数(*cRound*),该对记录才有可能进行匹配操作.因为 *round* 值小于当前的迭代轮数的记录对会在前几轮迭代中进行比较.对于一对记录,先计算它们的模式相似度,然后利用剪枝规则进行剪枝.若模式相似度超过阈值 δ,才对它们进行精确的匹配操作,以判断该记录对是否匹配.若该记录对匹配,需要对它们进行合并操作生成一个新记录,然后通过模式生成算法更新新记录的模式.同时新纪录的 *round* 属性也要更新为当前迭代轮数加 1,这样可以保证新纪录在下一轮迭代中能够和所有其他记录进行匹配.另外跟 r 相匹配的记录需要把它们保存在匹配集合中,当 r 跟 I 中所有其他的记录匹配完毕时,需要把匹配集合中的所有记录从 I 中删除.这样每一次匹配合并后 I 中的记录就会减少,保证该算法最后可以收敛终止,同时也可以保证每一个记录只属于一个集合,不会再跟其他的记录进行合并成新的记录.当 r 与 I 中所有其他记录进行匹配操作后,将 r 加入临时记录集和 T.当 I 中的所有记录完成与 r 相同的操作之后,一轮迭代完成,而该轮迭代的结果也被保存在了临时记录集合 T 中(7~21 行).一轮迭代后,首先要将当前迭代轮数自增 1.如果临时记录集合的大小不变,则记录集合中已经不存在相似的记录对,此时迭代终止(22~25 行).最后返回最新记录结果集 I'(26~27 行).图 4 展示了该算法的运行过程.括号中的数字表示每个记录的 *round* 值.为了方便,这里规定如果两记录的编辑距离小于 4 则认为匹配.如图共有 5 条记录,第一轮迭代中 r_1、r_3 合并,r_4、r_5 合并形成新的记录集合且

其 *round* 值加 1，此时共有 3 个记录集合. 第二轮迭代中 r_2 与 r_1、r_3 合并形成新的记录集合且其 *round* 值相应加 1，此时共有 2 个记录集合. 最后一轮迭代中，两记录集合的模式相似度小于阈值 δ，直接判定两记录集合不匹配，此时结果集依然只有 2 个记录集合，迭代结束，输出最终结果.

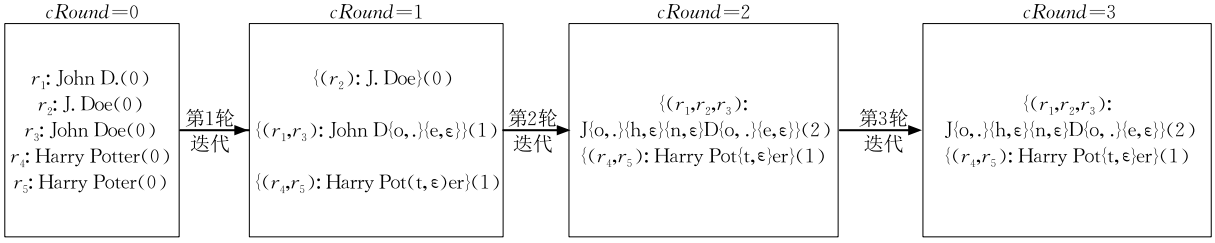

图 4　IterER 算法的运行过程

假设输入数据集中不存在匹配的记录，则每两个记录之间都需要进行比较且不存在记录合并，此时本文的实体匹配算法与两两比较方法的复杂度一致，为 $O(n^2)$. 事实上，实际运行过程中进行的比较次数远远小于理论比较次数，第 5 节的实验可以很好地体现.

4.2　实体生成

采用以上框架可以得到若干个记录集合，每个记录集合对应现实世界中的唯一实体. 但是实际情况中往往需要生成最终的真实实体，使得其可以唯一表示该集合. 这样一些冗余的记录就可以得到消除使得整个数据集得到清洗.

一般生成实体的方法是从一个集合中找到一个代表记录[8,13]，使得该记录可以代表整个集合唯一对应现实世界中的一个真实实体. 但是这种代表记录的选取不但需要遍历整个集合，而且选取的记录随选取方法和数据集特征的变化而变化，稳定性较差. 特别地，如果真实实体不存在于集合之中，那么这种方法就会变得无效.

利用本文的记录模式，可以从另一个思路解决这个问题. 通过上述的实体解析算法，可以得到若干记录集合和其所对应的模式. 对于一个集合所对应的模式，可以在模式生成过程中记录每一个模式元素中字符的出现频数. 利用这种带频数的记录模式，可以在每个模式元素中选取出现频数最大的字符作为该模式元素的值，最后可以生成一个出现概率最高的字符串作为该集合的代表，使其对应于一个真实实体. 例如图 4 中记录集合 $\{r_1, r_2, r_3\}$ 对应的带频数的模式为 {J:3}{o:2,.:1}{h:2,ε:1}{n:2,ε:1}{ :3}{D:3}{o:2,ε:1}{e:2,ε:1}，故其最可能的字符串为 John Doe，与真实实体一致. 显然利用记录模式生成实体可以直接扫描一遍模式快速得到最可能的记录，而不需要遍历集合中的所有记录从而提高实体生成的效率. 其次模式根据概率生成实体具有一定的容错能力，可以过滤掉一些噪声记录. 例如一些由于拼写、排版、习惯等原因造成的错误字符，其出现的频数必然要少于正常的字符，所以取频数最大的字符可以过滤那些错误字符.

5　实验结果与分析

5.1　实验设置

本文实验采用真实数据和模拟数据来对算法性能进行评测. 其中真实数据采用 CiteSeer 论文数据库，其包含近 1400 万条文献引用，470 多万个真实实体. 其文献属性包括作者、标题、日期、页码、卷号、出版社等. 为了方便，实验采用标题属性作为一条文献记录. 模拟数据采用 DBGenerator① 数据生成器在实体和记录比为 1∶20 且满足均匀分布情况下所生成的虚拟个人信息数据，其生成的数据属性包括姓、名、中间名以及住址. 本实验采用所有属性的串连作为一个记录.

本文实验代码采用 C# 编写，运行在配置 3.2GHz Intel Core i5-3470 处理器，8GB 内存的 64 位 Windows 8 操作系统中. 采用的匹配规则为：若记录集合间有一对记录相似则匹配. 合并操作为取记录的并集. 由于原始的两两比较方法极为耗时，为了进行性能对比，本文只与 R-Swoosh 做比较. R-Swoosh 也是一种迭代的实体解析算法，每当有新记录生成时，都要将新记录与已经解析好的记录重新进行匹配，直到没有新记录产生为止.

本文主要对实体解析的两个阶段进行评测：(1) 实体解析的结果；(2) 真实实体的生成. 为了验证结果的有效性，本文采用准确率(Precision)和召

① http://www.cs.utexas.edu/users/ml/riddle/data/dbgen.tar.gz

回率(Recall),并用 F_1-Score 作为评测标准[15]. 其计算公式如下:

$$\text{准确率}=\frac{\text{正确的记录匹配对数}}{\text{预测的记录匹配对数}},$$

$$\text{召回率}=\frac{\text{正确的记录匹配对数}}{\text{真实的记录匹配对数}},$$

$$F_1\text{-Score}=\frac{2\times\text{准确率}\times\text{召回率}}{\text{准确率}+\text{召回率}}.$$

效率方面,通过算法执行的记录比较次数和运行时间来比较其性能.

5.2 实体解析结果

利用真实数据集 CiteSeer 和模拟数据集 DBGen,采用上述有效性评测标准,我们得到本文提出的算法 IterER 与 R-Swoosh 的性能比较. 如图 5、图 6,分别表示记录条数为 5000 条时,两算法在 CiteSeer 和 DBGen 数据集上有效性随阈值的变化情况. 其中 IterER 为本文提出的基于模式的实体解析算法. 可以看到两算法的有效性(准确率、召回率、F_1-Score)曲线完全重合,表现出一致的有效性. 这也说明本文的算法并不影响结果的准确性. 从实验图中可以看出随着阈值越来越低,不匹配的记录越来越容易被合并,使得召回率不断升高. 同时负例的个数不断增加,使得准确率不断降低. 可以看到 CiteSeer 和 DBGen 的阈值分别取 0.5 和 0.6 时 F_1-Score 值最高;而当 CiteSeer 和 DBGen 的阈值分别取 0.3 和 0.4 时,其召回率几乎为 1,而准确率变得非常低,使得整体上的 F_1-Score 值最低. 容易推测当阈值降到最低接近 0 时,F_1-Score 值会变得最低,而此时的实体解析效果也会变得最差. 所以选定一个合适的阈值是影响算法有效性的关键. 不同的数据集中,两算法的召回率、准确率和 F_1-Score 最高都能达到 0.95 以上,进一步说明两算法在处理实体解析问题时都是非常有效的.

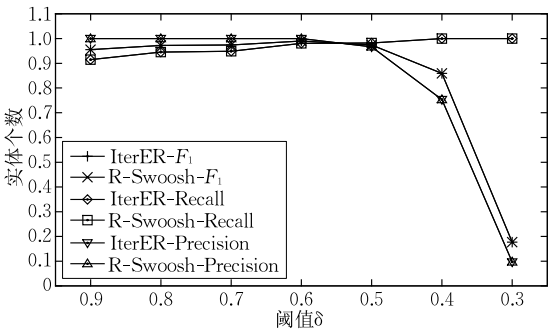

图 5 CiteSeer 记录为 5000 条时,有效性随阈值的变化情况

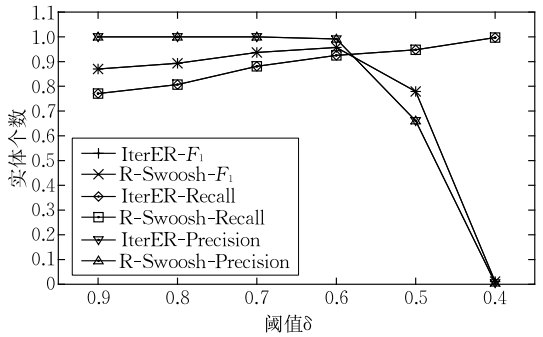

图 6 DBGen 记录为 5000 条时,有效性随阈值的变化情况

同样的数据集下,两算法随阈值变化而变化的运行时间如图 7、图 8. 可以看到在 CiteSeer 数据集中 R-Swoosh 的运行时间随着阈值的减小而降低. 这是因为随着阈值的减小,记录的合并操作会增加,使得形成的记录集合数目减少,导致记录的比较次数降低,所以 R-Swoosh 在 CiteSeer 数据集下的运行时间会随阈值减小而降低. 而对于本文提出的 IterER 算法,随着阈值的减少,运行时间会出现一个波动. 这是由于随着阈值的降低,每轮迭代合并的记录增多,使得整体的记录集合数变少. 而记录集合数目减少意味着每个集合中的记录个数的增加,所以其对应的模式会相应地变繁琐,导致模式间的相似度计算变得复杂. 而且这时模式会引入较多的不相关实例,使得模式间相似度变高,形成的边界值变松,过滤效果变差,记录的比较次数增多(见图 9、

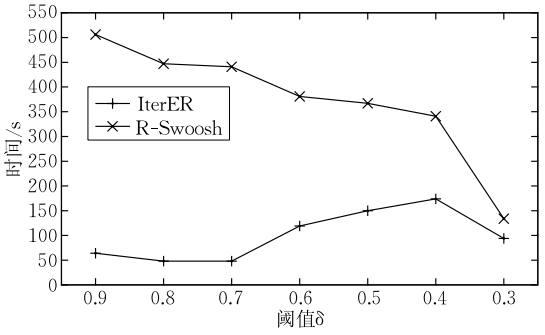

图 7 CiteSeer 记录为 5000 条时,运行时间随阈值的变化情况

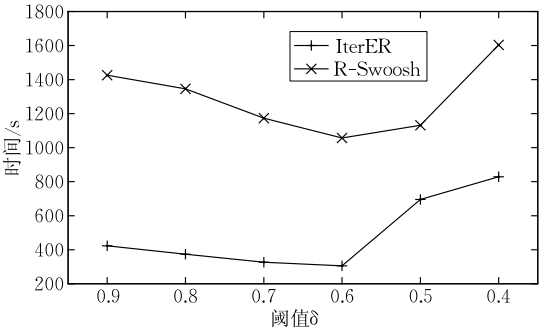

图 8 DBGen 记录为 5000 条时,运行时间随阈值的变化情况

图 10). 从而导致 IterER 算法的运行时间会随着阈值的减小有一个回升的过程. 从图 8 可以看出，对于 DBGen 数据集，R-Swoosh 的运行时间同样也会产生回升. 这是由于 DBGen 中的实体个数(250 个不同实体)比 CiteSeer 中的实体个数(108 个不同实体)要多很多，而且 DBGen 的数据满足均匀分布，使得记录的比较次数变多(见图 9、图 10)，所以 R-Swoosh 的运行时间也会有一个回升. 但是不论阈值如何设置，从实验图中可以看到本文算法的运行时间始终比 R-Swoosh 要少得多，其时间性能提升可以达到 2～10 倍. 这种巨大的性能提升源于本文提出的基于模式的实体解析算法可以过滤大量的不可能相似的记录从而节省大量的计算时间开销.

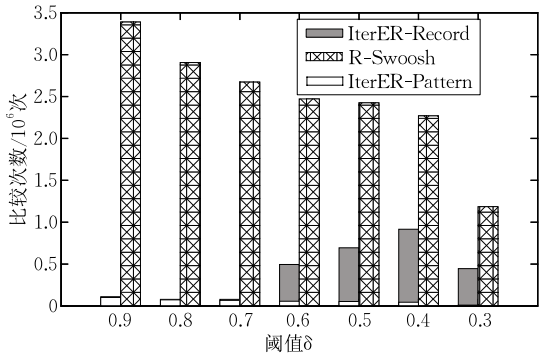

图 9　CiteSeer 记录为 5000 条时，比较次数随阈值的变化情况

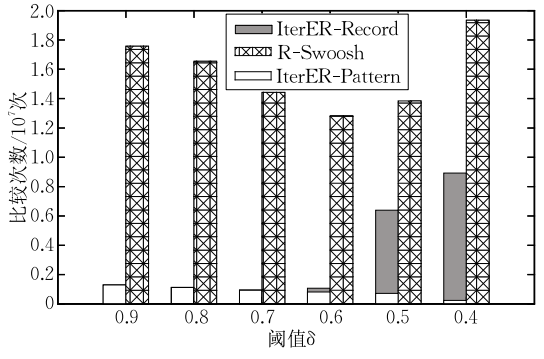

图 10　DBGen 记录为 5000 条时，比较次数随阈值的变化情况

为了解释 IterER 与 R-Swoosh 在运行时间上的巨大差异，我们在相同的数据集下统计了两算法在不同的阈值条件下记录比较次数的变化，如图 9、图 10. 其中 IterER-Pattern 表示 IterER 算法中模式的比较次数，IterER-Record 表示记录的比较次数. 可以看出本文算法的比较次数远远少于 R-Swoosh 的比较次数，而记录比较次数的变化跟运行时间的变化成正相关的关系，因此两算法的运行时间差异是由于算法进行的记录比较次数的差异造成的. 容易发现图 9、图 10 的比较次数走势与图 7、图 8 的运行时间走势是一致的. 此外图 9、图 10 显示在阈值较高的情况下，两算法的比较次数差异巨大. 这是由于高阈值下只有非常相似的记录才会合并，此时其生成的模式比较干净简洁，形成的边界值较紧，所以可以过滤掉绝大部分的不必要的记录比较. 更加说明基于模式过滤的有效性.

在相同的数据集下，通过增加数据集的记录条数，扩大数据量，选取每个数据量下最优的阈值(这里选取使 F_1-Score 值最高的阈值作为最优阈值)，我们得到在不同的数据量下，两算法的有效性和运行时间的差异比较，如图 11、图 12.

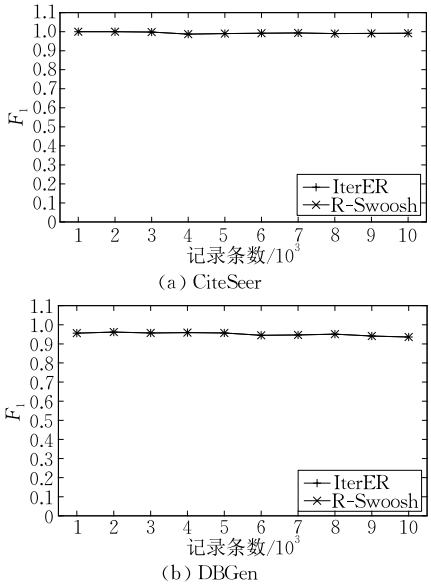

图 11　F_1-Score 随数据集大小的变化情况

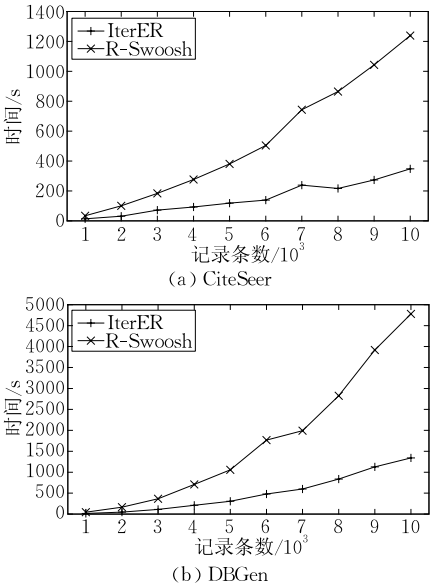

图 12　运行时间随数据集大小的变化情况

从图 11 中可以看出在不同的数据集中，随着数据集的大小不断变化，IterER 算法和 R-Swoosh 算法的有效性曲线(F_1-Score)始终重合且保持在较高的水平. 进一步说明两算法框架在解决 ER 问题上的重要意义. 此外图 12 反映无论数据集的大小如何变化，IterER 算法的运行时间始终远远小于 R-Swoosh 算法的运行时间，而且 IterER 的运行时间随数据集大小变化的增长速度远远小于 R-Swoosh 的增长速度. 这是因为随着数据量的增长，记录对之间的比较出现爆发式增长，使得 R-Swoosh 的运行时间呈现类似指数级的增长，而本文基于模式的 IterER 算法可以避免那些明显不相似的记录对间的比较，减少大量的比较次数，抑制记录对比较随数据量增加而形成的爆发式增长，从而节省大量的时间开销. 同时也说明 IterER 更加适合处理大数据量下的实体解析.

通过上述实验表明，本文基于模式的 ER 算法框架可以在确保实体解析结果准确性的同时能够有效地提升实体解析的效率.

5.3 实体生成

大部分实体解析研究工作都比较注重实体解析结果集的生成，而对最终的实体生成和表示关注较少. 本节选取一个代表性的方法进行对比，即最短距离法. 该方法从记录集合中选取一个记录做为该记录集合所对应的实体，其中该记录需满足其与记录集合中其他记录的编辑距离之和最短.

利用上节中最优阈值所产生的结果集合进行实体生成，每个集合产生一个记录，对应一个现实世界中的真实实体. 由于 DBGen 是模拟数据，真实实体无从可知，所以本节实验采用 CiteSeer 数据集. 由于 CiteSeer 是论文数据库，我们通过谷歌搜索引擎对 CiteSeer 数据中真实对应的文献进行确认作为正确结果. 通过与模式生成的记录和最短距离方法生成的记录进行比较，得到如图 13 的结果. 图 13 显示了两种方法在不同数据集大小下的实体生成情况. Pattern 表示本文中的用模式生成实体的方法，MinDis 表示最短距离生成实体的方法. 由于实体解析结果的误差，所以生成实体的个数(集合数)与真实实体的个数存在偏差，这也直接影响实体的生成的准确性，因为错误的集合必将导致错误的实体生成. 由图中可以看出随着数据集大小的增大，实体的个数越来越多，准确性也越来越差. 但是基于模式的实体生成算法的准确率要好于一般基于最小距离的准确性. 由于噪声数据的干扰和数据本身的质量问题，使得一般的算法在容错方面的性能比基于模式的生成算法要弱，导致准确性要低. 效率方面，模式经过一次遍历可以生成实体，而一般方法则需要遍历整个记录集合，使得基于模式的实体生成在时间复杂度上的优势更明显.

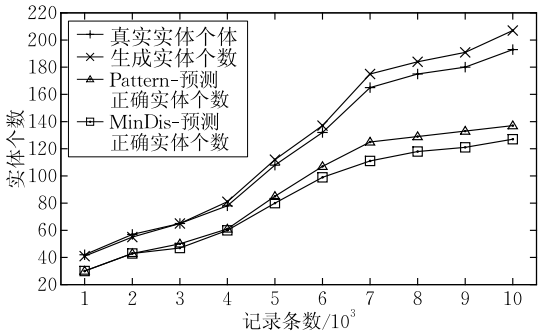

图 13 两种算法的实体生成对比

6 总结与展望

实体解析是数据融合和数据清洗过程中的重要步骤，实体解析的结果影响着数据质量. 特别是不同的数据源进行数据融合时需要进行实体解析找出对应同一实体的那些记录. 然而为了保证实体解析的精确度，现存的算法需要进行大量的比较计算，消耗大量的时间开销. 特别是在大数据环境下，这些计算是不能容忍的. 为了解决这一问题，本文提出了一整套从记录到实体的基于模式的实体解析的算法框架，旨在在现有方法的基础上提高实体解析的效率，适应大数据量的要求. 本文提出的实体解析算法框架利用记录模式可以有效地过滤掉一些不可能相似的记录，避免不必要的比较，从而大大地提高了实体解析的效率. 与现有的框架相比该框架在显著提升运行效率的基础上能够保证其准确性. 另外基于模式的实体生成由于具有良好的容错性，性能表现相比于一般算法也有优势.

虽然实体解析在数据融合与数据清洗中扮演着重要的角色，但是当前关于实体解析特别是实体生成的研究工作还是相对比较缺乏. 本文基于模式提出了一个比较初步的实体生成算法，但是其准确性依然有待提高. 为了完整解决这一难题，一些新技术和新成果，如机器学习、人工智能、自然语言处理等需要被考虑和应用，使得最终产生的实体能够更加贴近真实实体.

参 考 文 献

[1] Elmagarmid A K, Ipeirotis P G, Verykios V S. Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering, 2007, 19(1): 1-16

[2] Hernández M A, Stolfo S J. Real-world data is dirty: Data cleansing and the merge/purge problem. Data Mining and Knowledge Discovery, 1998, 2(1): 9-37

[3] Guo Zhi-Mao, Zhou Ao-Ying. Data quality and data cleaning: A survey. Journal of Software, 2002, 13(11): 2076-2082(in Chinese)
(郭志懋, 周傲英. 数据质量和数据清洗研究综述. 软件学报, 2002, 13(11): 2076-2082)

[4] Christen P. Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection. Berlin: Springer, 2012

[5] Jain A K, Murty M N, Flynn P J. Data clustering: A review. ACM Computing Surveys, 1999, 31(3): 264-323

[6] Winkler W E. Overview of Record Linkage and Current Research Directions. Washington: Statistical Research Division, 2006

[7] Benjelloun O, Garcia-Molina H, Menestrina D, et al. Swoosh: A generic approach to entity resolution. The International Journal on Very Large Data Bases, 2009, 18(1): 255-276

[8] Monge A E, Elkan C P. An efficient domain-independent algorithm for detecting approximately duplicate database records//Proceedings of the 2nd ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery. Vancouver, Canada, 1997: 23-29

[9] Fellegi I P, Sunter A B. A theory for record linkage. Journal of the American Statistical Association, 1969, 64(328): 1183-1210

[10] Hernández M A, Stolfo S J. The merge/purge problem for large databases//Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data. San Jose, USA, 1995, 24(2): 127-138

[11] Garcia-Molina H, Ullman J D, Widom J. Database System Implementation. Upper Saddle River, NJ: Prentice Hall, 2000

[12] Whang S E, Menestrina D, Koutrika G, et al. Entity resolution with iterative blocking//Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. Providence, USA, 2009: 219-232

[13] Jin L, Li C. Selectivity estimation for fuzzy string predicates in large data sets//Proceedings of the 31st International Conference on Very Large Data Bases. Trondheim, Norway, 2005: 397-408

[14] Bhattacharya I, Getoor L. Iterative record linkage for cleaning and integration//Proceedings of the 2004 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. Paris, France, 2004: 11-18

[15] Menestrina D, Whang S E, Garcia-Molina H. Evaluating entity resolution results//Proceedings of the 2010 International Conference on Very Large Data Bases. Singapore, 2010, 3(1): 208-219

[16] Baxter R, Christen P, Churches T. A comparison of fast blocking methods for record linkage//Proceedings of the 2003 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, USA, 2003, 3: 25-27

[17] McCallum A, Nigam K, Ungar L H. Efficient clustering of high-dimensional data sets with application to reference matching//Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Boston, USA, 2000: 169-178

[18] Chaudhuri S, Ganjam K, Ganti V, et al. Robust and efficient fuzzy match for online data cleaning//Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. San Diego, USA, 2003: 313-324

[19] Levenshtein V. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics-Doklady, 1966, 10(8): 707-710

[20] Chaudhuri S, Ganti V, Kaushik R. A primitive operator for similarity joins in data cleaning//Proceedings of the 22nd International Conference on Data Engineering. Atlanta, USA, 2006: 5-5

LIU Hui-Ping, born in 1990, Ph. D. candidate. His research interests include data quality and location-based services.

JIN Che-Qing, born in 1977, Ph. D., professor, Ph. D. supervisor. His research interests include data stream management and mining, uncertain data management, data quality and location-based services.

ZHOU Ao-Ying, born in 1965, professor, Ph. D. supervisor. His research interests focus on data management and applications, inclusive of Web data management, data intensive computing, in-memory cluster computing, big data benchmarking and performance optimization.

Background

Entity resolution is the problem of matching records that represent the same real-world entity and then merging the matching records. It is a critical process of data integration and data cleaning. Entity Resolution has been studied under various names including record linkage, merge/purge, deduplication, reference reconciliation, object identification, and others. The main task of entity resolution is comparing records and determining if they refer to the same entity.

There are two kinds of methods to resolve this problem, one is exhaustive entity resolution, which can achieve high precision, but consume lots of computing time. The other one is blocking-based entity resolution, it can generate a proper result with high efficiency. However, the result is affected more or less by blocks, witch makes it worse than exhaustive entity resolution in terms of effectiveness. To get a precise result efficiently, a general entity resolution framework, Swoosh, was proposed by Omar Benjelloun et al. Swoosh is an exhaustive entity resolution framework, which can reduce amount of comparisons of records when it holds the ICAR (Idempotence, Commutativity, Associativity and Representativity) properties. However, when there are no matching records between two record sets, all record pairs still need to be compared, witch makes it difficult to handle big volume dataset. To overcome this drawback of Swoosh, this paper proposes a noble method based on pattern, which represents the similar records by a record pattern, then we can generate a bound by comparing record patterns. With this bound, we can avoid comparing dissimilar records. In this way, we both accelerate the process of entity resolution and ensure its correctness.

This work is supported by the National Basic Research Program (973 Program) of China project (No. 2012CB316203), the National Natural Science Foundation of China under Grant (Nos. 61370101, 61321064), the Innovation Program of Shanghai Municipal Education Commission (14ZZ045).