基于非主属性值的实体匹配

杨 强[1)] 李直旭[1)] 蒋 俊[1)] 赵朋朋[1)] 刘冠峰[1)] 刘 安[1)] 周晓方[1),2)]

[1)](苏州大学计算机科学与技术学院 江苏 苏州 215006)

[2)](昆士兰大学信息技术与电子工程学院 布里斯班 澳大利亚 4067)

摘 要 实体匹配旨在找出不同数据源中指代同一实体的实例.已有的实体匹配方法大都基于实体主属性值的相似度进行匹配,而很少有工作考虑到使用实体的非主属性值来辅助实体匹配.然而,当两条指代同一实体的主属性值差异较大的时候,这两个实体可能不会被认为是匹配的实体.另一方面,这两个实体很可能共享一些特别的非主属性值,而这些非主属性值恰好可以反映出两个实体的匹配关系.基于这种思想,文中提出了一种新颖的基于非主属性值的实体匹配算法.该算法以类似于决策树的结构为基础,通过使用这种结构,不仅可以解决噪声值和空缺值带来的问题,而且可以极大地提高发现匹配记录以及尽可能早地排除不匹配记录的效率.多个数据集上的实验结果表明我们的方法比现有的实体匹配方法具有更高的准确率和召回率.此外,使用我们提出的基于决策树的匹配算法等有关技术较 Baseline 匹配算法在匹配效率上高出 10 倍多.

关键词 实体匹配;非主属性;数据质量;性能;算法

中图法分类号 TP392 **DOI 号** 10.11897/SP.J.1016.2016.02075

Record Matching with Non-Key Attribute Values

YANG Qiang[1)] LI Zhi-Xu[1)] JIANG Jun[1)] ZHAO Peng-Peng[1)] LIU Guan-Feng[1)]
LIU An[1)] ZHOU Xiao-Fang[1),2)]

[1)](*School of Computer Science and Technology, Soochow University, Suzhou, Jiangsu* 215006)

[2)](*School of Information Technology and Electrical Engineering, University of Queensland, Brisbane* 4067, *Australia*)

Abstract Record Matching (RM) finds out instances referring to the same entity between different data sources. Existing work mainly uses the similarity between the key attribute values of instances for RM, while seldom work employs non-key attribute values. As a result, when two instances referring to the same entity do not have similar key attribute values, they might be missed as a matching pair. On the other hand, some particular non-key attribute values shared by the two instances might reflect the relationship between them. Based on the intuition, we propose a novel RM method based on non-key attribute values. Compared to key attribute, non-key attributes can be more noisy and inconsistent. Besides, there are usually a lot more non-key attributes than key attributes, thus RM based on non-key attributes faces a significant efficiency problem. To deal with these challenges, we propose a rule-based algorithm based on a tree-like structure. With this tree-like structure, we can not only deal with noisy and missing values, but also greatly improve the efficiency of the method by finding out matched instances or filtering unmatched instances as

收稿日期:2015-07-28;在线出版日期:2016-01-28.本课题得到国家自然科学基金(61402313,61472263,61303019,61572336)、江苏省博士后科研基金(1501090B)、中国博士后第 58 批面上基金(2015M581859)和江苏软件新技术与产业化协同创新中心的资助.**杨 强**,男,1990 年生,硕士,主要研究方向为数据清洗、数据融合、机器学习. E-mail: 20144227003@stu.suda.edu.cn. **李直旭**(通信作者),男,1983 年生,博士,副教授,主要研究方向为数据清洗、数据融合、数据挖掘、机器学习、众包数据管理. E-mail: zhixuli@suda.edu.cn. **蒋 俊**,男,1991 年生,硕士,主要研究方向为数据清洗、数据融合、机器学习. **赵朋朋**,男,1980 年生,博士,副教授,主要研究方向为 Deep Web、数据集成、信息检索、数据挖掘、管理信息系统. **刘冠峰**,男,1982 年生,博士,副教授,主要研究方向为可信计算、海量数据分析、社会网络信息挖掘、服务计算. **刘 安**,男,1981 年生,博士,副教授,主要研究方向为社交网络、信息安全、空间数据库. **周晓方**,男,博士,教授,主要研究领域为空间数据库、多媒体数据库、高性能查询处理、Web 信息系统、数据挖掘、生物信息、网络调研.

early as possible. The experimental results based on several data sets demonstrate that our method outperforms existing RM methods by reaching a higher precision and recall. Besides, the proposed techniques can greatly improve the efficiency of a baseline algorithm beyond 10 times.

Keywords record matching; non-key attribute; data quality; performance; algorithm

1 引 言

随着信息时代数据量级的剧增,数据之间的不一致和冲突问题日益凸显[1].为了将不同来源的不一致数据进行融合,前人在"实体匹配"方面做了大量研究工作,旨在发现不同数据库中表示同一实体的实例[2].

目前,大多数的实体匹配方法都是借助前缀过滤或 Q-gram 等方法通过度量各实例主属性值之间的字符串相似度,从而根据预定义的相似度阈值做出是否匹配的决策[3-4].然而,含义相同的主属性值形式上可能千差万别,含义不同的主属性值也可能表示同一实体.因此,没有任何一种相似度度量方法可以准确度量所有主属性值之间的相似度,且一个统一给定的相似度阈值也会极大影响实体匹配的准确率和召回率.

虽然有时匹配实体之间没有相似的主属性值,但是它们的部分非主属性值却可能很相似.例如,表 1 中的实体 Ascend M2 和表 2 的实体 Mate2,两者没有很高的主属性相似度,但是它们具有一些相同的非主属性值,如 Huawei,Bar,Android 4.3,且具有相同的发布时间 2014.03,从中我们可以发现两者很有可能表示同一实体.换句话说,非主属性值也可以帮助我们识别实体之间的关系.基于上述观察,本文提出一种基于非主属性实现实体匹配的方法.我们的方法是对目前已有的基于主属性的实体匹配方法的补充.与仅使用主属性的方法相比,我们主要关注如何使用非主属性以提高实体匹配的准确率和召回率.

表 1 从天猫网站收集的在售手机信息

	Product	Manufacturer	Size	RAM	Release	Type	OS	…
t1	w2013	SAMSUNG	3.70 inches	1GB	2013.04	Flip	Android 4	…
t2	8295	Coolpad	4.70 inches	1GB	2013.01	Bar	Android 4.1	…
t3	MXII	MeiZu	4.40 inches	2GB	2012.12	Bar	Flyme 2	…
t4	Ericsson U1	Sony	3.50 inches	512MB	2009.10	Bar	S60V5	…
t5	4S	Apple	3.50 inches	512MB	2011.01	Bar	IOS5	…
t6	A880	Lenove	6.00 inches	1GB	2013.04	Bar	Android 4.2	…
t7	Ascend M2	HuaWei	6.10 inches	2GB	2014.03	Bar	Android 4.3	…
t8	S930	Doov	2.80 inches	512MB	2010.09	Slide	—	…
t9	G9098	SAMSUNG	3.67 inches	2GB	2014.09	Bar	Android 4.3	…
t10	8730L	Collpad	5.50 inches	1GB	2012.05	Bar	Android 4.3	…

表 2 从中关村在线网站收集的在售手机信息

	Product	Manufacturer	Size	RAM	Release	Type	OS	…
s1	Galaxy w2013	SAMSUNG	3.7″	1GB	2013-04	Flip	Android 4.0	…
s2	8295	Coolpad	4.7″	—	2013-01	Bar	Android 4.1	…
s3	Meizu MX2	MeiZu	4.5″	2.0GB	2012-04	—	Flyme 2.0	…
s4	3s	XIAOMI	5.1″	3GB	—	Bar	MIUI V5	…
s5	IPhone 4s	Apple	—	512MB	2011-10	Bar	IOS 5.0	…
s6	A880	Lenove	6.0″	1GB	2013-04	Bar	Android 4.1	…
s7	Mate2	HuaWei	6.1″	2GB	2014-03	Bar	Android 4.3	…
s8	S930	Doov	2.8″	512MB	2010-09	Slide	—	…
s9	8730L	Coolpad	—	1GB	2012-05	Bar	Android 4.3	…

然而,使用非主属性进行实体匹配并非易事.与主属性相比,非主属性中会存在更多的噪声值,而且数据间的不一致问题更为严重.此外,通常情况下数据表中非主属性个数比主属性个数多,因此基于非主属性的实体匹配中存在着严重的效率问题.使用非主属性进行实体匹配至少面临以下 3 个挑战:(1)属性选择.面对诸多的非主属性,如何评估每一个非主属性识别匹配实体的能力,从而做出较优选

择进行匹配；(2) 噪声数据和丢失数据. 非主属性中可能存在噪声数据和丢失数据. 在实体匹配过程中噪声数据和丢失数据会影响匹配的效果，因此处理噪声数据和丢失数据也是一个很大的挑战；(3) 效率问题. 基于非主属性的实体匹配方法与只使用主属性的实体匹配方法相比前者时间开销要多出很多倍，因为前者需要对实例之间更多的属性值进行比较.

针对上述三方面的挑战，我们提出了一种基于非主属性值的实体匹配方法. 具体来说，我们首先考察并衡量每一个非主属性识别匹配实体的能力和过滤不匹配实体的能力. 并基于考察结果以及非主属性之间的关联关系，我们动态建立一种特别的基于规则的概率决策树(Probabilistic Rule-based DecisionTree，PRTree). 通过该概率决策树，我们能够尽早发现匹配实例对或尽早排除不匹配实例对.

与我们将要在文中描述的 Baseline 算法相比，基于 PRTree 的实体匹配方法极大地提高了实体匹配效率，且不易受到噪声数据和丢失数据的影响. 此外，在文章的第 3 节，我们将对结合使用主属性和非主属性进行实体匹配的方法进行探讨.

我们在本文中的贡献如下：

(1) 不同于以往基于主属性的实体匹配方法，我们创造性地提出使用非主属性进行实体匹配.

(2) 通过考察各非主属性对于实体匹配的相关特性及它们之间的关联关系，我们动态建立了一棵基于规则的概率决策树，并基于此树设计实现了高效准确的实体匹配算法.

我们在 3 个数据集上对我们的算法进行了验证. 结果表明我们提出的基于规则的概率决策树算法不仅能达到较好的匹配准确率和召回率，且节省了超过 80%的匹配开销.

2　问题定义

给定两个关系表 $T_1=\{t_1,t_2,\cdots,t_n\}$ 和 $T_2=\{s_1,s_2,\cdots,s_m\}$，$T_1$ 中的每一个实例 $t_i\in T_1(1\leq i\leq n)$，$T_2$ 中的每一个实例 $s_j\in T_2(1\leq j\leq m)$，实体匹配旨在发现存在于两个表间表示同一实体的实例对 (t_i,s_j). 以往的实体匹配方法主要是使用主属性值，依赖于一些字符串相似度函数，如编辑距离(Edit distance)以及一个相似度阈值决定两个表中的实体是否匹配. 其形式化表示如下：设定 A_0 是表 T_1 和表 T_2 的主属性，给定一个字符串相似度函数 $sim(\cdot,\cdot)$ 以及一个相似度阈值 θ，对于实例对 (t_i,s_j)，如果满足 $sim(t_i[A_0],s_j[A_0])\geq\theta$，则 t_i,s_j 为同一实体，其中 $t_i\in T_1(1\leq i\leq n)$，$s_j\in T_2(1\leq j\leq m)$.

然而，仅仅使用主属性值判定实例对是否匹配是不够的，因为主属性值通常具有多种表现形式，如 Li Hua 和 Hua Li 实则表示同一个人. 又或是主属性值直接无法使用，如丢失数据和噪声数据. 本文中，我们使用非主属性值进行实体匹配. 形式化表示如下，假定表 T_1 和表 T_2 具有相同的非主属性集合 $S_{NK}=\{A_1,A_2,\cdots,A_p\}$，使用非主属性进行实体匹配的定义如下.

定义 1.　非主属性的实体匹配(RM with Non-Key Attributes，NokeaRM). 给定两个数据表 $T_1=\{t_1,t_2,\cdots,t_n\}$ 和 $T_2=\{s_1,s_2,\cdots,s_m\}$，两者具有相同的非主属性集合 $S_{NK}=\{A_1,A_2,\cdots,A_p\}$，NokeaRM 问题旨在找到一个基于 S_{NK} 的函数 $F(t_i,s_j)$ 和一个相似度阈值 τ，使得 $\forall t_i\in T_1(1\leq i\leq n)$，$\forall s_j\in T_2(1\leq j\leq m)$，$(t_i,s_j)$ 为指向同一实体的实例，当且仅当它们满足

(1) $F(t_i,s_j)\geq\tau$；

(2) $\forall s_k\in T_2,F(t_i,s_j)\geq F(t_i,s_k)$.

3　使用非主属性进行实体匹配

我们首先在 3.1 节展示一个 Baseline 算法，接下来在 3.2 节介绍基于规则的概率决策树的算法.

3.1　Baseline：基于 DifScore 的实体匹配算法

Baseline 算法主要使用非主属性区分某一实体不同于其他实体的能力进行实体匹配，这一能力被定义为属性的区分度得分(Differentiation Degree Score，difScore).

属性的 difScore 大致上反应了该属性下属性值分布情况. 例如，在没有重复记录的情形下数据表中所有的主属性值都是不同的，因此主属性具有最高的 difScore 也即是 1.0. 在表 1 中属性“Size”也具有较高的 difScore，因为其值的分布比较多样，而属性“Type”则具有一个相对较低的 difScore. 其具体形式如下所示：在关系表 T 中，在所有的实例当中假定属性 $A_i(1\leq i\leq p)$ 有 $\mathrm{distinct}(A_i,T)$ 个不同的属性值，则属性 A_i 的 difScore 计算方法为

$$\mathrm{difScore}(A_i,T)=\frac{\mathrm{distinct}(A_i,T)}{|T|}\qquad(1)$$

其中：$|T|$表示关系表中实体的总数目；distinct(A_i, T)表示属性 A_i 下属性值不同的个数.

在获得关系表中每个属性的 difScore 的基础上，我们计算实例 $t\in T_1$ 和 $s\in T_2$ 之间的相似度，其计算公式如式(2)所示：

$$F_{\text{baseline}}(t,s)=\frac{\sum_{A\in S_{NK}}(\text{difScore}(A)\times sim(t[A],s[A]))}{\sum_{A\in S_{NK}}\text{difScore}(A)} \tag{2}$$

其中：$t[A]$，$s[A]$分别表示实体 t 和 s 在属性 A 下的属性值；$sim(t[A],s[A])$表示属性值 $t[A]$，$s[A]$之间的相似度；S_{NK} 表示两表中共同的非主属性集合.

然而通过实验发现，Baseline 算法只能识别出一半左右的匹配实体(大约 45%～55%的召回率)，准确率也只达到 80%左右. 其原因在于：(1) difScore 并不能完全反应属性的区分度，因为其并未考虑属性之间的组合关系；(2) 丢失数据干扰实例对相似度的计算. 更为重要的一方面是 Baseline 算法的复杂度为 $O(pnm)$，其中 $|T_1|=n$，$|T_2|=m$，$|S_{NK}|=p$.

3.2 基于规则的概率决策树算法

我们提出了一个更为可靠的 NokeaRM 算法，该算法通过使用非主属性建立的基于规则的概率决策树进行实体匹配. 众所周知，决策树是一种提高效率的方法，因为树中的每一个节点都有机会对输入的数据做出最终的决策. 因而，通过使用决策树这种结构可以在尽可能短的时间内获取匹配实例对或者排除不匹配实例对. 此外，我们提出了一种特殊的决策树可以克服 Baseline 算法的两个缺陷，既考虑到了属性之间的组合关系，又不易受缺失值的影响. 接下来，我们首先给出该算法的基本思路，然后介绍如何建立基于规则的概率决策树，最后我们将展示基于此决策树的 NokeaRM 算法.

3.2.1 算法基本思路

首先，给定的实体对 $t_i\in T_1$，$s_j\in T_2$ ($1\leq i\leq n$，$1\leq j\leq m$)匹配与否是由其各个属性下的值匹配与否所决定的. 然而，鉴于不同的属性在唯一确定实体的能力上有所不同，我们选择一组最具有决策能力的属性集进行匹配；然而，为了尽可能地减少决策的开销，我们期望尽早发现匹配实体对或尽早排除不匹配实体对，因此我们决定使用类似于决策树的结构进行实体匹配，即将各个属性放入决策树中对实例对是否匹配进行逐层联合判定；此外，属性集中的各个属性在决策实体匹配时并不是相互独立的而是相互依赖共同作用的，因而建树过程以及通过树做匹配计算时，还需要考虑各属性间的相互依赖度.

如图 1 所示，实体对的匹配结果 R 是由属性集 $S=\{A_1,A_2,\cdots,A_5\}$ 决定的，以路线 1 为例，A_2 的结果是在 A_1 的条件下得到的，而 A_4 的结果是在 A_2 的条件下得到的，以此类推，最终的结果是由以上各步的结果综合作用得到的.

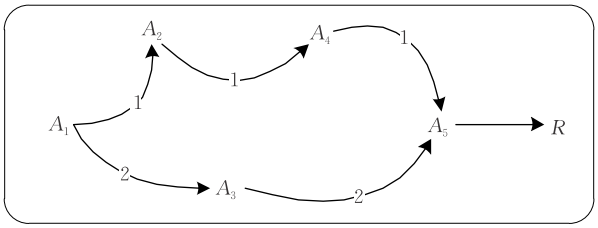


图 1 属性间的相关性

实体 t_i 与 s_j 其匹配结果可形式化表示为

$$R=\begin{cases}1, & \text{如果 } t_i=s_j\\ 0, & \text{否则}\end{cases} \tag{3}$$

另外，我们从属性的角度上考虑匹配结果，用 M_k 代表实体所在属性组合下的第 k 个属性对于 R 的一个贡献，如果两者值相等则是一个正的贡献，否则是一个负的贡献. 具体如下：

$$M_k=\begin{cases}1, & t_i[A_k]=s_j[A_k]\\ 0, & \text{其他}\end{cases},\ 1\leq k\leq p \tag{4}$$

对于关系表中的非主属性的集合 $S_{NK}=\{A_1,A_2,\cdots,A_p\}$，实体间的匹配结果可通过条件概率进行量化，其概率可通过贝叶斯条件概率进行评估，其公式如式(5)所示：

$$p(S_{NK})=\Pr\{R=1|S_{NK}\}=\frac{\Pr\{S_{NK}|R=1\}\Pr\{R=1\}}{\Pr\{S_{NK}|R=1\}\Pr\{R=1\}+\Pr\{S_{NK}|R=0\}\Pr\{R=0\}} \tag{5}$$

3.2.2 建立基于规则的概率决策树

基于上一小节的分析，我们考虑使用非主属性建立基于规则的概率决策树(Probabilistic Rule-based decision Tree，PRTree). PRTree 的理论依据如下：(1) 决策树有利于减少实体匹配的时间开销；(2) 贝叶斯概率决策树有助于反应具有一定决策能力的属性之间的依赖关系；(3) 不同的属性具有不同的决策能力，具体表现为属性的两个重要性质，即决策匹配的充分性(Sufficiency)和决策匹配的必要性(Necessity)，其定义如下.

定义 2. 决策匹配的充分性和必要性(Sufficiency and Necessity). 对于给定的两个关系表 $T_1=$

$\{t_1,t_2,\cdots,t_n\}$和 $T_2=\{s_1,s_2,\cdots,s_m\}$，假设两者具有相同的属性集合 S，属性 $A_k\in S(1\leq k\leq p)$的充分性概率是指当满足 $t_i[A_k]=s_j[A_k]$时，实体 $t_i\in T_1(1\leq i\leq n)$和 $s_j\in T_2(1\leq j\leq m)$为同一实体的概率，而属性 A_k的必要性概率是指当 t_i和 s_j为同一实体时，也即当 $t_i=s_j$时，$t_i[A_k]=s_j[A_k]$的概率.

属性的充分性特性和必要性特性在决定PRTree的结构上起着重要作用. 其中属性的充分性反应了属性识别匹配实体的能力，即当属性的值相同时，能够在多大概率上反应其所对应的实体为同一实体；而属性的必要性反应了属性排除不匹配实体的能力，即当实例对表示为同一实体时，其在某一属性上也具备相同属性值的概率.

根据定义 2，属性的充分性概率和必要性概率可通过训练得到，其中训练集中已知两个数据表中匹配的实例对，其定义为($\boldsymbol{Tr}_1$，$\boldsymbol{Tr}_2$). 假定训练集中的数据分布接近于实体匹配的两个真实数据表，则我们可以得到其计算公式如式(6)和(7)：

$$suf(A_k)=\sum_{\{(t_i,s_j)|\boldsymbol{Tr}_c\}}(B(t_i=s_j|t_i[A]=s_j[A]))/(B(t_i=s_j|t_i[A]=s_j[A])+B(t_i\neq s_j|t_i[A]=s_j[A])) \tag{6}$$

$$nec(A_k)=\sum_{\{(t_i,s_j)|\boldsymbol{Tr}_c\}}(B(t_i[A]=s_j[A]|t_i=s_j))/(B(t_i[A]=s_j[A]|t_i=s_j)+B(t_i[A]\neq s_j[A]|t_i=s_j)) \tag{7}$$

其中，$B(bool1|bool2)$为布尔函数，其表示如式(8)：

$$B(bool1|bool2)=\begin{cases}1,\text{如果 } bool2=\text{TRUE 并且 } bool1=\text{TRUE}\\1,\text{如果 } bool2=\text{TRUE 并且 } bool1=\text{FALSE}\\0,\text{如果 } bool2=\text{FALSE}\end{cases} \tag{8}$$

其中，$\boldsymbol{Tr}_c$是一个在建树不同阶段中动态改变的限制集，初始化为 $\boldsymbol{Tr}_c=\{(t_i,s_j)|t_i\in\boldsymbol{Tr}_1,s_j\in\boldsymbol{Tr}_2\}$.

下面我们讨论如何建立 PRTree 决策树. 鉴于实例对间的匹配度计算需要考虑属性间的相互依赖关系，我们并不是一次找出 top-k 个具有较大 difScore 的属性元素作为概率决策树上的属性节点，而是采用贪心算法，每次只在上一步的决策结果的条件之下计算出未使用属性的条件充分性概率和必要性概率，从而选择出当前条件最具有决策力的属性放入树中的对应位置. 此建树过程将迭代进行直到达到一定的停止条件.

基于上述思路，接下来我们将从两方面介绍如何建立 PRTree：其一为属性的选择；其二为停止条件.

(1) 属性选择. 根据当前待选节点的类别(包括根节点、非叶节点和叶节点)，在已经确定的候选节点集的基础上，每次选择具有最大充分性或必要性的节点作为当前节点，并加入到候选节点集中. 通过每次选择具有最大充分性概率或最大必要性概率的属性，可以尽可能早地发现匹配实体，并且排除不匹配实体. 具有较大充分性概率的属性说明其识别实例对匹配的能力较强；具有较大必要性概率的属性表明其在排除实例对为不匹配实体的能力较强.

(2) 停止条件. 当所有待考察属性在当前位置的条件充分性概率或条件必要性概率都低于某一值(τ_{stop})时，此时应停止创建更多的节点，将上一次刚加入候选节点集的属性作为叶子节点.

我们在算法 1 中给出了建立 PRTree 的流程，具体描述如下：

(1) 根节点(Root Node). 在计算出每个属性的充分性和必要性后，我们选择充分性和必要性得分最大的属性作为根节点. 而根节点有 3 个分支，分别是匹配(Matched(Y))，不匹配(Unmatched(N))和无效(Invalid(Null)). 如果根节点是具有最高充分性得分的属性，则节点的匹配分支是叶节点并将以一定的置信度输出“Matched”，而不匹配分支和无效分支则是非叶节点. 相反地，如果根节点为最大的必要性得分的属性，节点的不匹配分支则成为叶节点并以一定的置信度输出“Unmatched”，而匹配分支和无效分支都应是非叶节点. 根节点的计算如式(9)所示：

$$P_{\text{root}}=\max_{A\in S_{NK}}(suf(A),nec(A)) \tag{9}$$

(2) 非叶节点(Non-Leaf Node). PRTree 中的非叶节点代表了该节点的充分性概率或必要性概率不能直接判断出匹配的结果，需要结合其他属性，直至能够做出判断，也即是到达 PRTree 的叶节点. 我们以目前已确定的候选节点集为基础，当计算余下的其他属性的充分性得分和必要性得分时，必须在满足候选节点集中所有元素的条件下进行.

为了构建 PRTree 的非叶节点，我们在目前已有的所有祖先节点集的条件下，计算余下属性的充分性和必要性，选择具有最大条件充分性或条件必要性得分的属性作为当前节点.

具体地，属性的条件充分性和条件必要性可通过式(6)和(7)并将 $\boldsymbol{Tr}_c$改变为一个更小的训练集结

合计算而得到,其中所有的实例都应满足祖先节点的条件.非叶节点的计算方法如式(10)所示:

$$P_{\text{nonleaf}}=P_{A|\text{Candit(PRTree)}}\max_{A\in(S_{NK}-\text{Candit(PRTree)})}(suf(A),nec(A)) \quad (10)$$

其中 Candit(PRTree)为已选中的祖先节点构成的集合.

(3)叶节点(Leaf Node).PRTree 的叶节点表明了从根节点到当前节点构成的属性集合可以判断给定的实例对的匹配结果或者目前已有的数据无法给出较为明确的匹配结果.PRTree 的每个叶节点都会输出"Matched"或"Unmatched",通过输出结果可以判断实例对是否匹配.当属性的 $suf(A)$ 和 $nec(A)$ 一直很低的时候,也即低于一个合理的阈值 τ_{stop} 时,我们将停止获取更多的属性用以创建 PRTree的节点.然后,我们会创建一个"Exit(Not Decided)"节点作为最后的节点,其表明我们无法判断实例对是否匹配.叶节点的计算如式(11)所示:

$$P_{\text{leaf}}=\max_{A\in(S_{NK}-\text{Candit(PRTree)}),suf(A)\approx\tau_{\text{stop}},nec(A)\approx\tau_{\text{stop}}}(suf(A),nec(A)) \quad (11)$$

根据算法 1,通过对表 1 和表 2 所在的大量训练数据的训练,我们建立了基于非主属性的 PRTree,如图 2(a)所示.在接下来的部分中,我们将介绍如何使用 PRTree 进行实体匹配.

算法 1. 建立 PRTree.

输入:具有属性集 S 的训练表 $\boldsymbol{Tr}_1$ 和 $\boldsymbol{Tr}_2$,充分性阈值 τ_s 和必要性阈值 τ_n 以及不再继续创建 PRTree 的阈值 τ_{stop}

输出:PRTree 的根节点的引用 *root*

root=NULL, *cur*=*root*, *AncsCond*=∅

WHILE S≠∅ DO

FOR ALL $A\in S$ DO

根据式(6)和(7)计算或更新 $suf(A|AncsCond)$ 和 $nec(A|AncsCond)$

IF $A.suf<\tau_s$ THEN $A.suf=0$;

ELSE

IF $A.nec<\tau_n$ THEN $A.nec=0$;

END IF

END IF

END FOR

在所有的非主属性中将具有最大值的 $suf(A|AncsCond)$ 或 $nec(A|AncsCond)$ 属性 A 赋给 *cur*,*cur*=A

cur.conf=Max($suf(A|AncsCond)$,$nec(A|AncsCond)$);

IF *cur.conf*<τ_{stop} THEN

cur="Exit(Not Decided)"

Breaks

END IF

cur. NullChild=*BuildPRTree*($\boldsymbol{Tr}_1$,$\boldsymbol{Tr}_2$,$S=S-\{A\}$, τ_s,τ_n,τ_{stop})

IF *cur.conf*=$suf(A|AncsCond)$ THEN

cur.YChild="Matched"

AncsCond=*AncdCond*+{(A,*unmatched*,*cur.suf*)}

cur=*cur.NChild*

ELSE

cur.NChild="Unmatched"

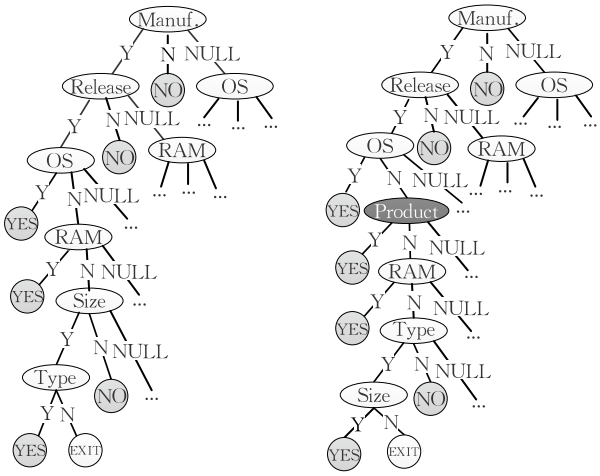
AncsCond=*AncdCond*+{(A,*matched*,*cur.nec*)}

cur=*cur.YChild*

END IF

END WHILE

RERUTN *root*



(a) 不含主属性的PRTree (b) 含主属性的PRTree

图 2 针对表 1 和表 2 的 PRTree

3.2.3 基于 PRTree 的 NokeaRM 算法

通过在训练集上不断地筛选符合上述要求的属性,创建了如图 2(a)所示的 PRTree.我们在算法 2 中给出如何借助 PRTree 执行 NokeaRM 算法.对于测试集数据表中的每一个实例对(t,s),从 PRTree 的根节点开始访问,并在到达叶子节点时停止访问.每次当实例对(t,s)到达属性 A_k 这个节点时,我们检查实例对在当前节点是否具有相同的属性值.如果相同,则访问当前节点的"Matched"子节点,否则访问当前节点的"Unmatched"子节点.但是,如果实例对在当前节点的属性值为空时,则应访问该 PRTree 的"Invalid"子节点,表明待比较的实例对在该属性下出现缺失值.当实例对访问叶节点时,我们会根据叶节点的性质从而决定输出"Matched" 或者"Unmatched".决策的置信度是由实例对所经过的从根节点到当前节点组成的所有节点共同决定的,

除根节点外，每一个节点结果的计算并不是独立于其他节点而得到的，而是在满足该节点所经历祖先节点的条件下所得. 此外，PRTree 中处于不同高度的节点，在计算置信度时应给予不同的权重分配，因为越是靠近根节点的属性，其受其他属性的影响越小，而越是远离根节点的属性，其受其他属性的影响越大. 因而，我们采取了一定的手段保证属性权重的合理分配. 综合以上的考虑我们采用如式(12)计算实例对之间的相似度

$$F_{\mathrm{PRTree}}(t,s)=\prod_{1\le i\le H(A)}(\mathrm{Conf}(A_i)\times sim(t[A_i],s[A_i]))^{\frac{1}{i}} \tag{12}$$

其中，$H(A)$代表当前节点在 PRTree 中的高度. 我们使用$(\cdot)^{1/i}$调整第 i 个节点的影响，因为越接近当前节点的节点，对当前节点所做出的决策影响越大. 当节点所处的层次越低，其削弱的力度越大；当节点所处的层次越高，其削弱的力度越小. 因为高层次的节点是在低层次节点的基础上建立的，也即高层次的节点依赖于低层次的节点，其对整棵树的影响较低层次的节点影响小，故应给予较低的削弱力度；对于高层次的节点其原理类似. 我们使用 $Conf(A_i)$表示沿着路径节点 A_i的置信度，其计算方法如式(13)所示：

$$Conf(A_i)=\begin{cases}suf(A_i), & \text{如果 } A_i \text{ 为“Matched”节点}\\ 1-nec(A_i), & \text{如果 } A_i \text{ 为“Unmatched”节点}\\ 1, & \text{如果 } A_i \text{ 为“Invalid”节点}\end{cases} \tag{13}$$

算法 2. 使用 PRTree 实现实例对的 NokeaRM 算法.

输入：PRTree 的根节点 *root*，实例对(t,s)
输出：实例对(t,s)的相似度：$F_{\mathrm{PRTree}}(t,s)$
cur=*root*; *level*=1; *sim*=1;
WHILE TRUE DO
  IF *cur.output*=“Exit” THEN
    RETURN −1
  END IF
  IF *cur* is a *leafNode* THEN
    RETURN *sim*
  END IF
  IF $t[cur]$=NULL OR $s[cur]$ =NULL THEN
cur=*cur.NullChild*
  ELSE
    $sim=sim\times(cur.conf\times sim(t[cur],s[cur]))^{1/i}$
  END IF
  IF $t[cur]=s[cur]$ THEN
cur=*cur.YChild*
  ELSE
cur=*cur.NChild*
  END IF
level++
END WHILE
RETURN *sim*

通过上述的步骤结合式(6)～(13)可以计算出给定实例对的相似性，并通过与相似度阈值的比较判断给定的实例对是否表示为同一实体.

例 1.　我们以表 1 中实体 t_1和表 2 中的实体 s_1为例介绍如何使用基于 PRTree 的 NokeaRM 进行实体的匹配.

(1) Step1. 我们在训练集中根据式(6)～(13)通过不断地迭代运算找出每一次迭代过程中符合要求的候选属性. 从图中可以看出 Manufacturer 具有较大的必要性概率，也即对于手机而言当 Manufacturer 不同时，两个手机实体为同一实体的概率基本为 0；又如当 Manufacturer 和 Release 一样时，OS 具有较大的充分性概率. 如果 OS 一样，则可判定这两个手机实体为同一实体，如果 OS 不一样则需要从 OS 的“N”分支继续寻找符合要求的其他属性. 最终，我们得到了如图 2(a)所示的 PRTree.

(2) Step2. 在 Step1 完成之后，根据 Step1 计算所得结果，我们得到了图 2(a)中节点的 *Conf* 值，如表 3 所示.

表 3　图 2(a)中节点的 *Conf* 值

Node	Manufacturer	Release	OS	RAM	Size	Type
Confidence	0.94	0.89	0.92	0.81	0.83	0.76

(3) Step3. 对于给定的表 1 中的实体 t_1和表 2 中的实体 s_1，我们根据算法 2 中介绍如何使用图 2(a)中的 PRTree 进行实体的匹配，设定属性值的编辑距离相似度阈值为 0.7，实体相似度阈值为 0.55. 我们首先从根节点“Manufacturer”开始遍历，由于 $sim(t_1[\mathrm{Manuf.}],s_1[\mathrm{Manuf.}])=1>0.7$，所以我们遍历“Manufacturer”的“Matched”子节点“Release”；因为 $sim(t_1[\mathrm{Release}],s_1[\mathrm{Release}])=0.86>0.7$，所以我们遍历“Release”的子节点“OS”；接下来我们继续计算 $sim(t_1[\mathrm{OS}],s_1[\mathrm{OS}])=0.86>0.7$，所以我们到达“OS”的“YES”节点也即是叶子节点；最终，通过式(12)结合表 3 中各节点的 *Conf* 值，得到表 1 中的实体 t_1和表 2 中的实体 s_1的相似度 $F_{\mathrm{PRTree}}(t_1,s_1)=1\times0.94^{\frac{1}{1}}\times0.89\times0.86^{\frac{1}{2}}\times0.92\times0.82^{\frac{1}{3}}\approx0.748>0.55$，所以表 1 中的实体 t_1和表 2 中的实体 s_1两者是匹配的，也即 t_1和 s_1为同一实体.

3.2.4 基于 PRTree 的 NokeaRM 算法复杂度分析

基于 PRTree 的 NokeaRM 算法较 Baseline 算法而言极大地减少了时间上的开销，具体体现在实例对之间属性的比较次数的减少，因为前者使用尽可能少的属性做出最终的决策，而后者使用全部的属性才能做出决策. 算法 2 的时间复杂度为 $O(qnm)$，其中 q 是在匹配过程中每个实例对所到达的平均高度. 注意到 $q \ll p$，因为我们可以排除一大部分不匹配的实例对，并在尽可能短的时间内找到匹配的实例对.

鉴于主属性在实体识别上的功效(唯一标识某一实体)，其能够在一定程度上辅助非主属性进行实体匹配，因而我们在本小节讨论如何将基于 PRTree 的 NokeaRM 算法与基于主属性的实体匹配方法(keyRM)相结合. 一种可能的方式就是分别从 keyRM 和 NokeaRM 中获取实例对的相似度权重之和，但是此方式的问题是没有一个合适的权重能够满足所有的情形. 另外一种方式是先执行 KeyRM 或者 NokeaRM 并限定一个严格的准确率阈值，接下来执行另外一种 RM 算法用以提高召回率. 然而，通过我们的观察发现，这种方式也不能达到一个较高的准确率和召回率. 最终，我们发现在建立 PRTree 的过程中使主属性参与进去，这种结合方式可以达到最好的性能. 如图 2(b)中所示，新的 PRTree 将主属性“Product”作为其一个节点存在. 我们在接下来的实验部分对这种结合方式的效果加以评估.

4 实验部分

我们在两个真实数据集和一个合成数据集上验证了我们的方法.

手机数据集(Mobile). 我们分别从天猫网(Tmall)和太平洋电脑网(PConline)搜集了在售的手机信息. Tmall 数据表中含有 4k 个元组，53 个属性，PConline 数据表中含有 5.6k 个元组，46 个属性. 这两个数据表共有的实例个数为 3.8k，共有的属性个数为 38，其中共有的属性如 Release Date，Operation System，RAM，Screen Size，Type 等.

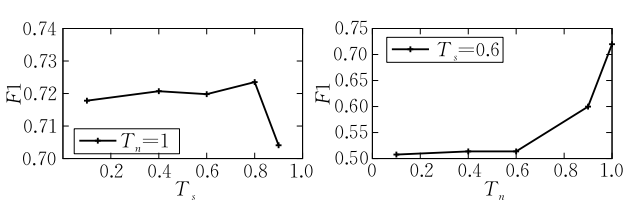
相机数据集(Camera). 我们分别从天极网(Yesky)和太平洋电脑网(PConline)搜集了在售的数码相机的信息. Yesky 数据表中含有 2.5k 个元组，50 个属性，PConline 数据表中含有 3.4k 个元组，44 个属性. 这两个数据表共有的实例个数为 2.5k，共有的属性个数为 31，其中共有的属性如 Type，Pixels，Panel，Wifi，Manufacturer 等.

仿真数据集(Synthetic). 我们生成了两个仿真数据表，具有 100k 个元组，60 个共有属性. 我们使用了一定的规则使得数据的分布近似于真实数据集. 例如，匹配实体的属性值相似度统一地分布在 0 和 1 之间. 注意到真实数据集上有一些缺失的属性值，我们在生成合成数据集时对非主属性也随机产生一定数目的缺失值.

我们大致上使用 3 个度量标准评估方法的效果. 准确率($Precision$)：所有匹配的实体中正确匹配的实例所占的比例；召回率($Recall$)：所有应该匹配的实体中正确匹配的实体所占的比例；$F1$ Score：对准确率和召回率的综合考虑，计算方法如下所示：$F1=\dfrac{2\times precision\times recall}{precision+recall}$. 我们使用算法的时间开销评估方法的效率.

4.1 参数设定

在我们的算法当中有一些重要的参数需要设定，这些参数共同影响着算法的效果和效率. 在评估我们提出的方法之前，先对使用到的参数加以说明：(1)创建 PRTree 的阈值(Thresholds for Building the Tree)：图 3 展示了 $F1$ Score 随着两个阈值 τ_s 和 τ_n 在 0 和 1 之间变动时发生的改变. 通过观察发现，当 $\tau_s=0.8$，$\tau_n=1$ 时，$F1$ Score 达到最大值. 同时我们也使用一个阈值 τ_{stop} 控制树的高度以及质量，观察发现当 $\tau_{\text{stop}}\approx 0.6$ 时，树的效果达到最好. 注意到在图 3(a)中由于 τ_s 的值太高，一些比较重要的属性被排除而不能用于创建 PRTree，导致了 $F1$ Score 的急剧下降；(2)字符串相似度阈值(Threshold for String Similarity)：通过对 3 个数据集的实验发现最佳的字符串相似度阈值取值大约为 0.7；(3)匹配实例的阈值(Threshold for Matching Instances)：我们也需要一个用于判定实例对匹配结果的阈值 τ_{matched}. 通过对 3 个数据集上的实验发现 $\tau_{\text{matched}}=0.55$ 为一个比较合适的取值.



(a) T_s对$F1$的影响 (b) T_n对$F1$的影响

图 3 PRTree 的阈值对 NokeaRM 的影响

4.2　和以往方法的比较

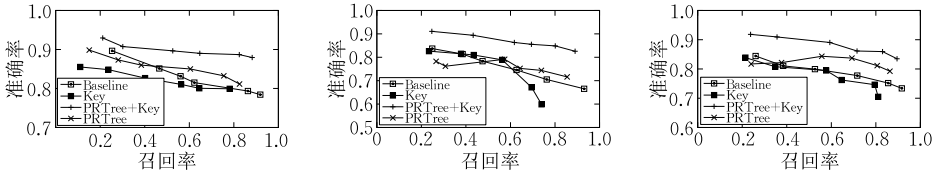
我们将本文中提出的实体匹配方法(Baseline 算法和 PRTree(Nokey)-based 算法)与现有的多种技术合成的最先进的 Key-based 实体匹配方法以及最先进的使用非主属性进行实体匹配的方法 Matching tree-based[5] 比较各自的准确率和召回率. 在我们算法中使用的字符串相似度函数统一选择 Edit distance.

(1) Key-based. 该方法合成了在多篇文章中提及的几种基于键值匹配的最新技术. 我们首先创建基于 Q-gram 的后缀数组,并将可能匹配的实体根据已知的可靠匹配实体中提取的"分块主键"对实体进行分块操作[6],此外,我们融入了 prefix-based filtering[7],Inverted Indices[7-8] 等多种优化过滤技术进行块内实体间的相似性计算,此方法提升了实体匹配的效率.

(2) Matching tree-based. 基于非主属性的匹配算法目前唯有 Dey 等人在文献[5]中提出的另外一种基于 0-1 分支决策树的匹配树算法 Matching tree-based. 简而言之,此方法在筛选匹配树的节点时,根据属性的匹配概率选择具有最大熵的属性,只是简单地根据属性值是否一致将其分为 0-1 分支从而建立 matching tree.

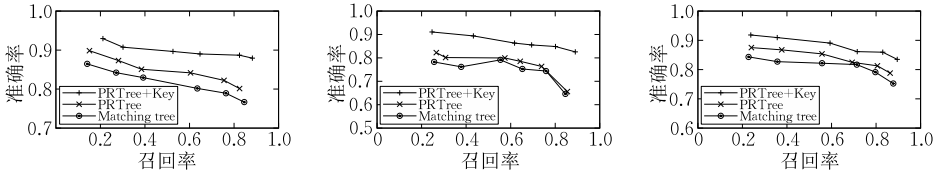
我们除了将上述两种方法与本文中提出的 Baseline 方法和无主属性的基于 PRTree 的 NokeaRM 方法作比较之外,也考虑了将基于 PRTree 的方法和 Key-based 的方法结合的方法,通过在 3 个数据集上的实验结果发现将主属性也用于创建 PRTree 时匹配效果达到最优. 因此,我们也使用了 PRTree(Key)-based 参与比较.

首先,我们将上述 5 种实体匹配方法在准确率和召回率方面做出评估,实验结果如图 4 所示,上述 5 种方法随着召回率地不断增大,其准确率整体上呈下降趋势. 不同的方法在下降的平稳程度上有所不同. 我们发现 Key-based 方法与我们提出的方法相比在准确率和召回率方面具有最差的效果,而 Baseline 方法效果优于 Key-based 方法,低于 Matching tree-based 方法,而 Matching tree-based 方法较 PRTree 方法相比效果略差,使用了主属性和非主属性的 PRTree 方法在准确率和召回率方面达到最优的效果. 没有使用主属性的 PRTree 方法也达到一个比较好的效果,之所以这种方法比使用主属性的 PRTree 方法效果略差是因为缺少了主属性的提供的信息. 我们对上述各种方法所达到的效果做出如下分析:Key-based 方法整体上波动较大,因为此方法易受数据的影响,当数据在表达方式上相差较大时,其准确率和召回率会受到极大的影响,导致本应该匹配的实体无法正确的匹配. 如图 4(b)所示,当 Key-based 方法的召回率超过 0.75 后,其准确率急剧下降. Baseline 方法在稳定性上优于 key-based 方法,因为 Baseline 方法使用了非主属性,受数据表达方式的影响较小. 如图 5 所示,Matching tree-based 方法和基于 PRTree 的方法的准确率随召回率稳定变化,但 Matching tree-based 方法的变化率比使用主属性的 PRTree 方法大. 因为一方面在数据集中有缺失值的存在,Matching tree-based 方法易受到缺失值的影响,而使用主属性的 PRTree 方法考虑了缺失值的情况;另一方面,Matching tree-based 方法忽略了不同层次的节点对匹配结果的影响,而使用主属性的 PRTree 方法考虑了不同层次节点的影响. 使用主属性的 PRTree 方法效果优于不使用属性的 PRTree 方法,原因在于使用主属性



(a) 手机　(b) 相机　(c) 仿真集

图 4　在准确率和召回率方面与以往方法的对比

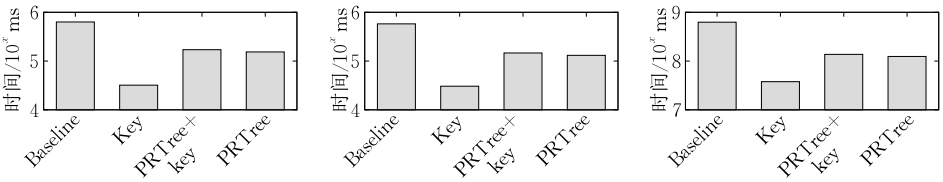


(a) 手机　(b) 相机　(c) 仿真集

图 5　在准确率和召回率方面与 Matching tree-based 方法的对比

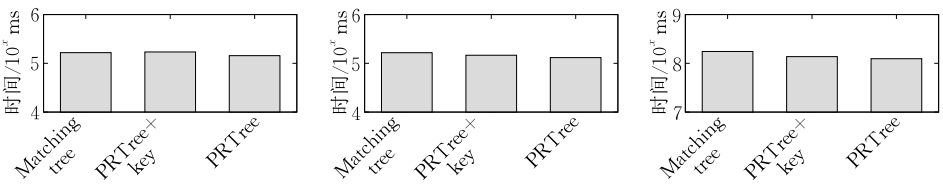
的 PRTree 方法通过使用主属性获取了匹配所需的重要信息,在一定程度上较少了影响的力度.大体上,在 3 个数据集上的实验结果表明,我们的方法在准确率方面提升了近 15%,召回率方面提升了 20% 左右.

其次,我们对上述 5 种方法在效率方面做出了对比,比较了这些方法在 3 个数据集上各自的时间开销.如图 6 所示,Key-based 方法的时间开销最少,因为此方法只使用了主属性进行实体匹配,而且使用了 block、prefix filtering 和 inverted indices 等技术减少匹配的开销;Baseline 方法的平均时间开销是 Key-based 方法的 30 倍左右,因为 Baseline 方法使用了所有的属性进行匹配,而基于 PRTree 的方法其时间开销比 Baseline 算法少 10 倍多,因为基于 PRTree 的方法筛选了充分性或必要性概率最大属性,并结合合理的停止建树阈值剔除一些影响因子小的属性,这样能够尽可能早的做出决策,极大地减少属性值的比较次数,从而提高匹配效率.从图 7 中可以看出,基于 PRTree 的方法在效率上与 Matching tree-based 方法相差不大,但在平均效率上看,Matching tree-based 方法稍低一些,主要因为 Matching tree-based 方法在创建 matching tree 时在属性的选择上开销较大,特别是在假阳性错误出现的时候.



(a) 手机 (b) 相机 (c) 仿真集

图 6 在效率方面与以往方法的对比

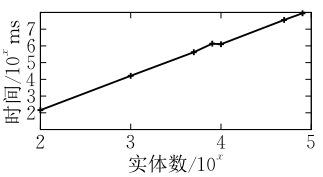


(a) 手机 (b) 相机 (c) 仿真集

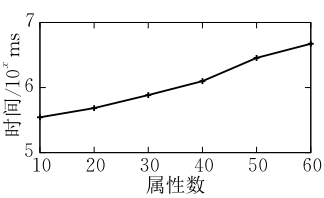
图 7 在效率方面与 Matching tree-based 方法的对比

4.3 可扩展性评估

如图 8 所示,我们在仿真数据集上对 PRTree 方法的可扩展性做出了评估.从图 8(a)中可以看出,随着记录数目从 100 变化至 10 W,其时间开销以一种近似于指数增长的方式缓慢地增加,也从另一方面证明了 PRTree 可以高效地减少比较时间.从图 8(b)中得出,随着属性数目从 10 增加至 60,其时间开销以近似于线性方式增加.



(a) 实体个数对效率的影响



(b) 属性个数对效率的影响

图 8 不同数目的记录数和属性数对 PRTree 在效率上的影响

5 相关工作

实体匹配近年来一直被广泛地研究[9-11],其应用领域涉及到医疗卫生[12-13]、国际安全[14]、信息检索[15-16]、商业数据管理[17]等.之前各种字符串相似性度量方法被提出,如基于字符的算法(如 edit distance[18]、Jaro distance[9]、Q-gram[19])和基于符号的算法(如 atomic string[20]、WHIRL[21]),也有一些混合的方法被提出[22-23],这些方法或是直接用于实体匹配或是间接辅助进行实体匹配,有的方法注重于匹配效率的提升,如 Q-gram 算法,而有的方法注重于提高匹配的质量,如 WHIRL.此外,文献[24]提出了两种可学习的文本字符串相似度度量方法,包括可学习的字符串编辑距离算法和使用 SVM 的向量空间度量算法,针对数据表不同的字段使用不

同的方法，在实体匹配过程中该方法具有较好匹配准确性. 近几十年来，一些基于分类[25]和语义网络[26]的方法也有被提出，然而存在灵活性差，时间开销大等问题.

实体匹配的一个重大问题是算法的时间复杂度. 为了减少实例对的比较次数，近年来有许多高效的方法被提出. 例如，一些工作将 Q-Gram 算法和倒排索引算法(Inverted Indices)[27]结合，通过这两种方法可以提高匹配效率，以尽可能少的时间开销为目的进行实体匹配，但此类方法的适用性存在一定的问题，如当参与匹配的实体存在缺失值，此类方法将不再适用. 通过使用一些提高效率的算法利用主属性进行实体匹配的方法[6-8]在进行实体匹配过程中易受数据的影响，如缺失值、错误值等，而且在数据表达方式不同时其匹配的结果也会受到严重的影响. 此外，还有一些基于前缀修剪(prefix-based pruning)的方法[28]、基于 top-k mode[29]的方法以及基于批量匹配(batch-based)的方法[30]，通过这些方法过滤那些“明显”不匹配的实例对，减少匹配过程中的比较次数.

文献[31]提出了一种使用正例和反例的方式结合相似度连接(similarity join)和结合(similarity unions)以及用户指定的相似度函数构建 SJU 操作树，查询某种适用于待匹配实体属性的相似性度量算法，达到每个属性以最合适的相似性算法进行匹配的目的，该方法不仅适用于小中型数据，而且在大型数据库上也适用. 此外，也有一些方法[32]将属性值映射到多维欧几里得空间，根据给定的合并规则结合属性的相似性选择一个属性的集合，应用多维相似性连接用以判断实例对的相似性.

文献[7]中提出了一种使用非主属性进行实体匹配的方法，该方法通过利用非主属性创建一棵类似于决策树的 Matching tree 进行匹配. 该文提出的 Matching tree 是一种 0-1 结构的二分树，通过一定的概率计算方式选择属性作为 Matching tree 的节点，然后从根节点到叶节点遍历进行匹配. 但是这种方法存在适用性问题，当属性值出现缺失时，匹配过程便无法进行. 而本文提出的 PRTree 却能很好地处理该问题，当遍历时出现缺失值 PRTree 能够产生独立的分支，即属性值为空时遍历该属性下的其它节点，进行后续的匹配过程. 此外，Matching tree-based 方法没有考虑 Matching tree 中不同层次的属性对匹配结果的影响度，粗糙地将靠近根节点的属性给予匹配结果较大的影响，将靠近叶子节点的属性给予匹配结果较小的影响，而实际上应该根据节点所在的层次给予合适的权重分配，而基于 PRTree 的方法使用了平衡函数合理地分配权重.

6　总结以及未来的工作

在本文中，我们研究了基于非主属性的实体匹配问题即 NokeaRM 问题. 我们首先提出了一种使用所有属性进行实体匹配的 Baseline 方法，该方法使用属性区分某一实体不同于其他实体的能力，综合考虑所有属性值的相似度对实例对进行实体匹配计算. 然而 Baseline 方法因为要比较很多的非主属性值对因而效率很差. 因此我们又提出了一个高效的基于概率决策树的 NokeaRM 方法. 其思想为在实体匹配的过程中，能够尽可能早地发现匹配的实体以及尽可能早地排除不匹配的实体，从而减少实体之间比较的时间开销，大大提高了基于众多非主属性值进行实体匹配的效率. 通过与现有的实体匹配方法相比，可以看出我们的方法达到了更高的准确率和召回率. 此外，我们扩展了基于非主属性的 PRTree 方法，将主属性也用于概率决策树的创建之中，进一步改善了实体匹配的精确度.

基于 PRTree 的方法也有需要改善的地方. 在未来的工作中，我们会考虑如何更好地解决错误属性值对匹配的影响. 我们可能考虑将数据纠错和实体匹配相结合，实现两者之间的交互，使两者相互促进. 更多匹配的实体有利于纠正更多的错误值，而更多错误值的纠正有利于发现更多匹配的实体. 此外，我们还考虑将众包技术(Crowdsourcing)融入到我们的方法之中，将一些机器无法做出的判断交给众包平台，从而进一步提升匹配精准度.

参　考　文　献

[1] Scannapieco M. Object matching: New challenges for record linkage. The Philosophy of Information Quality, 2014, 358(38): 95-106

[2] Fan W, Jia X, Li J, Ma S. Reasoning about record matching rules. Proceedings of the VLDB Endowment, 2009, 2(1): 407-418

[3] Cheatham M, Hitzler P. String similarity metrics for ontology alignment//Proceedings of the 12th International Semantic Web Conference. Sydney, Australia, 2013: 294-309

[4] Li M, Chen X, Li X, et al. The similarity metric. IEEE Transactions on Information Theory, 2004, 50(12): 3250-3264

[5] Dey D, Mookerjee V S, Liu D. Efficient techniques for online record linkage. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(3): 373-387

[6] Aizawa A, Oyama K. A fast linkage detection scheme for multi- source information integration//Proceedings of the International Workshop on Challenges in Web Information Retrieval and Integration. Tokyo, Japan, 2005: 30-39

[7] Wang J, Li G, Feng J. Can we beat the prefix filtering?: An adaptive framework for similarity join and search//Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data. Scottsdale, USA, 2012: 85-96

[8] Christen P. A survey of indexing techniques for scalable record linkage and deduplication. IEEE Transactions on Knowledge and Data Engineering, 2012, 24(9): 1537-1555

[9] 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

[10] Lalitha L, Maheswari B, Karthik S. A survey on deduplication methods. International Journal of Research in Computer Engineering and Electronics, 2012, 31(3): 364-369

[11] Yan Z, Zheng N, Ives Z G, et al. Active learning in keyword search-based data integration. The International Journal on Very Large Data Bases, 2015, 24(5): 611-631

[12] Clark D. Practical introduction to record linkage for injury research. Injury Prevention, 2004, 10(3): 186-191

[13] Kelman C W, Bass A J, Holman C. Research use of linked health data—A best practice protocol. Australian and New Zealand Journal of Public Health, 2002, 26(3): 251-255

[14] Jonas J, Harper J. Effective counterterrorism and the limited role of predictive data mining. 1000 Massachusetts Ave, NW Washington, DC 20001-5403, Cato Institute: Technical Report No. 584, 2006

[15] Hajishirzi H, Yih W-T, Kolcz A. Adaptive near-duplicate detection via similarity learning//Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. Geneva, Switzerland, 2010: 419-426

[16] Su W, Wang J, Lochovsky F H. Record matching over query results from multiple web databases. IEEE Transactions on Knowledge and Data Engineering, 2010, 22(4): 578-589

[17] Bilenko M, Basil S, Sahami M. Adaptive product normalization: Using online learning for record linkage in comparison shopping //Proceedings of the 5th IEEE International Conference on Data Mining. Hilton head Island, South Carolina, 2005: 58-65

[18] Cohen W, Ravikumar P, Fienberg S. A comparison of string metrics for matching names and records//Proceedings of the KDD-2003 Workshop on Data Cleaning. Geneva, Switzerland, 2003, 1(1): 73-78

[19] Ukkonen E. Approximate string-matching with Q-grams and maximal matches. Theoretical Computer Science, 1992, 92(1): 191-211

[20] Monge A E, Elkan C, et al. The field matching problem: Algorithms and applications//Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining. Portland, Oregon, 1996: 267-270

[21] Cohen W W. Integration of heterogeneous databases without common domains using queries based on textual similarity. ACM SIGMOD Record, 1998, 27(2): 201-212

[22] Bayardo R J, Ma Y, Srikant R. Scaling up all pairs similarity search//Proceedings of the 16th International Conference on World Wide Web. Banff, Canada, 2007: 131-140

[23] Xiao C, Wang W, Lin X, et al. Efficient similarity joins for near-duplicate detection. ACM Transactions on Database Systems, 2011, 36(3): 15-24

[24] Bilenko M, Mooney R J. Adaptive duplicate detection using learnable string similarity measures//Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Washington, USA, 2003: 39-48

[25] Rahm E, Bernstein P A. A survey of approaches to automatic schema matching. The VLDB Journal, 2001, 10(4): 334-350

[26] Dhamankar R, Lee Y, Doan A, et al. iMAP: Discovering complex semantic matches between database schemas//Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data. Paris, France, 2004: 383-394

[27] Ukkonen E. Finding approximate patterns in strings. Journal of Algorithms, 1985, 6(1): 132-137

[28] Wang W, Xiao C, Lin X, Zhang C. Efficient approximate entity extraction with edit distance constraints//Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data. Providence, USA, 2009: 759-770

[29] Gal A. Managing uncertainty in schema matching with top-k schema mappings. Journal on Data Semantics VI, 2006, 4090(6): 90-114

[30] Chandel A, Nagesh P, Sarawagi S. Efficient batch top k search for dictionary-based entity recognition//Proceedings of the 22nd International Conference on Data Engineering. Atlanta, USA, 2006: 28-38

[31] Chaudhuri S, Chen B-C, Ganti V, Kaushik R. Example driven design of efficient record matching queries//Proceedings of the 33rd International Conference on Very Large Data Bases. University of Vienna, Austria, 2007: 327-338

[32] Jin L, Li C, Mehrotra S. Efficient record linkage in large data sets//Proceedings of the 8th International Conference on Database Systems for Advanced Applications. Kyoto, Japan, 2003: 137-146

YANG Qiang, born in 1990, M. S. His research interests include data cleaning, data integration and machine learning.

LI Zhi-Xu, born in 1983, Ph. D., associate professor. His research interests include data cleaning, data integration, data mining, machine learning and crowdsourcing data management.

JIANG Jun, born in 1991, M. S. His research interests include data cleaning, data integration and machine learning.

ZHAO Peng-Peng, born in 1980, Ph. D., associate professor. His research interests include Deep Web, data integration, information retrieval, data mining and MIS.

LIU Guan-Feng, born in 1982, Ph.D., associate professor. His research interests include trusted computing, massive data analysis, social network mining and services computing.

LIU An, born in 1981, Ph. D., associate professor. His research interests include social network, information security and spatial databases.

ZHOU Xiao-Fang, Ph.D., professor. His research interests include spatial and multimedia databases, high performance query processing, web information systems, data mining, bioinformatics and e-research.

Background

Record Matching (RM) finds out instances referring to the same entity between different data sources. Existing work mainly uses the similarity between the key attribute values of instances for RM, while seldom work employs non-key attribute values. As a result, when two instances referring to the same entity do not have similar key attribute values, they might be missed as a matching pair. On the other hand, some particular non-key attribute values shared by the two instances might reflect the relationship between them. Based on the intuition, we propose a novel RM method based on non-key attribute values. Compared to key attribute, non-key attributes can be more noisy and inconsistent. Besides, there are usually a lot more non-key attributes than key attributes, thus RM based on non-key attributes faces a significant efficiency problem. To deal with these challenges, we propose a rule-based algorithm based on a tree-like structure. With this tree-like structure, we can not only deal with noisy and missing values, but also greatly improve the efficiency of the method by finding out matched instances or filtering unmatched instances as early as possible. The experimental results based on several data sets demonstrate that our method outperforms existing RM methods by reaching a higher precision and recall. Besides, the proposed techniques can greatly improve the efficiency of a baseline algorithm.

This research is partially supported by the National Natural Science Foundation of China (Grant Nos. 61402313, 61472263, 61303019, and 61572336), the Postdoctoral scientific research funding of Jiangsu Province (No. 1501090B), the National 58 batch of Postdoctoral Funding (No. 2015M581859), and the Collaborative Innovation Center of Novel Software Technology and Industrialization, Jiangsu, China.