Hans Journal of Data Mining 数据挖掘, 2021, 11(2), 121-133
Published Online April 2021 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2021.112012

Hans汉斯

基于布尔匹配规则的实体解析方法

褚良旭，李　贵，李征宇，韩子扬，曹科研

沈阳建筑大学，信息与控制工程学院，辽宁 沈阳
Email: 11468880@qq.com, ligui21c@sina.com

收稿日期：2021年3月20日；录用日期：2021年4月21日；发布日期：2021年4月28日

摘　要

实体解析(ER)是数据集成和数据清洗的一个重要步骤。判断记录是否相似可以通过记录的属性(属性值)是否相似来判断。基于规则的实体解析方法，通过制定规则来将每个属性(属性值)的相似度都进行比较(属性匹配规则)，为了减小其求解的搜索空间，属性匹配规则将每个属性都采用相同的相似度算法和阈值来进行比较，这导致实体解析的精度不高。为了提高精度，本文提出一种基于布尔匹配规则的改进的实体解析规则生成算法，与传统的基于属性匹配规则和机器学习的实体解析方法相比，改进的实体匹配规则算法精度更高。本文首先提出一种基于语法约束的布尔匹配规则。在此基础上，本文提出了一种规则合成(Rule Evolution)算法，他可以根据输入的实例验证规则，并自动合成对整个数据集有效的ER规则。在真实数据集和合成数据集上的实验结果表明，我们的方法具有很高的准确性，本文提出的规则在有效性上优于其他可解释规则(如低深度的决策树，其他基于规则的实体解析)。

关键词

实体解析，布尔匹配规则，属性匹配规则，数据集成

Entity Resolution Based on Boolean Matching Rules

Liangxu Chu, Gui Li, Zhengyu Li, Ziyang Han, Keyan Cao

School of Information & Control Engineering, Shenyang Jianzhu University, Shenyang Liaoning
Email: 11468880@qq.com, ligui21c@sina.com

Received: Mar. 20th, 2021; accepted: Apr. 21st, 2021; published: Apr. 28th, 2021

Abstract

Entity resolution (ER) is an important step in data integration and data cleaning. Judging whether

文章引用: 褚良旭, 李贵, 李征宇, 韩子扬, 曹科研. 基于布尔匹配规则的实体解析方法[J]. 数据挖掘, 2021, 11(2): 121-133. DOI: 10.12677/hjdm.2021.112012

the records are similar can be judged by whether the attributes (attribute values) of the records are similar. The rule-based entity analysis method compares the similarity of each attribute (attribute value) by formulating rules (attribute matching rules). In order to reduce the search space for its solution, the attribute matching rules adopt the same for each attribute. The similarity algorithm and threshold are compared, which leads to low accuracy of entity analysis. In order to improve the accuracy, this paper proposes an improved entity parsing rule generation algorithm based on Boolean matching rules. Compared with the traditional entity parsing methods based on attribute matching rules and machine learning, the improved entity matching rule algorithm has higher accuracy. This article first proposes a Boolean matching rule based on grammatical constraints. Then, based on the proposed Boolean formula rules, this paper proposes a Rule Evolution algorithm, which can verify the rules according to the input examples and automatically synthesize the effective ER rules for the entire data set. Experimental results on real data sets and synthetic data sets show that our method has high accuracy. The rules proposed in this paper are better than other interpretable rules (such as low-depth decision trees, other rule-based Entity resolution).

Keywords

Entity Resolution, Boolean Matching Rules, Attribute Matching Rules, Data Integration

1. 引言

实体解析[1] [2] [3] (也称为记录链接或重复记录检测)是识别代表同一真实世界实体的记录的过程。例如，在房地产数据集中，合并两个楼盘表。可能希望合并它们所有的记录。在这种情况下，同一个楼盘可能由多个记录表示，因此必须标识并组合这些匹配的记录。

当今主流的实体解析中基于机器学习(ML)的解决方案通常是首选，当前大多数解决方案是Fellegi-Sunter 模型的变体[4]，其中实体解析被视为分类问题。这些方法包括基于 SVM 的方法[5]，基于决策树的方法[6]，基于聚类的技术[7]和基于马尔可夫逻辑的模型[8]。但是，使用 ML 方法的系统(如支持向量机[5]或模糊匹配[4])不支持可解释性，因为它们的模型由权重和功能参数组成，即使对于专业人员也很难解释。

随着 Amazon Mechanical Turk 社区中众包工作者的涌现，最近的研究重点已转移到利用人类智慧来帮助验证不确定的记录对。由于众包平台的普及，人们一直在努力利用众包工作者解决实体匹配问题[9] [10]。

最近基于规则的实体解析应用通常希望使用声明性 EM 规则。这样的规则在数据库社区中也很流行，因为它们为提高执行时的性能提供了巨大的机会，例如[11]中研究的那些规则。但是，这些方法通常假设 ER 规则是由领域专家给出的[12]，通过假设给定的 DNFEMrule 结构来发现相似性函数及其关联的阈值。然而，由于手写实体解析规则非常耗时和容易出错，本文研究是能否能通过学习正反实例自动生成可解释的 ER 规则。

布尔匹配规则通常在基于规则的系统中设计和实现。布尔匹配规则对在比较步骤中生成的比较属性的相似性值给出 true 或 false (是或否)决定。因为比较组件通常是 ER 过程中计算上最昂贵的组件，所以

以前的研究通过开发算法来减少必须比较的候选记录对的数量，同时找到所有可能的匹配项。本文为了生成更精确规则，本文提出下面的规则生成方法。首先，引入并构建了基于语法约束的布尔匹配规则。将布尔匹配规则所需的属性匹配规则和布尔匹配规则建模成语法，将所需的实例建模成语法所需的约束。并在求解器中进行合成。为了使布尔匹配规则在求解器中，本文提出了新的 ER 合成规则算法来自动生成 ER 规则。此外本文还提出了一种自定义求解器的算法来将自定义求解器发现数值阈值，将通过合成器为元素选择的正示例(E')和反示例(E)分隔开。本文的主要贡献如下：

1) 为了生成简洁可解释的规则，本文提出了一个新的基于语法约束的布尔匹配规则。(第 3 节)

2) 为了能在求解器中自动合成规则，本文设计了一种新的合成算法 Rule Evolution。采用反例引导归纳综合的思想，从少量实例中进行规则合成。(第 4 节)

3) 反例引导归纳综合使用的合成器(程序)不能进行复杂的数值函数的推理，本文提出了一种自定义求解器算法将自定义求解器发现数值阈值，将通过合成器为元素选择的正示例(E')和反示例(E)分隔开。(第 4 节)

4) 通过实验验证了本文的系统在匹配精度明显优于其他可解释模型，但是我们的规则有更少的子句。在精度方面，它也可以与其他不可解释的模型相比较。(第 5 节)

2. 相关理论

2.1. 相关定义

定义 1. 属性：实体是根据其特征来描述的，称为属性[13]。这些属性的值提供有关特定实体的信息。重要属性是指将一个实体与另一个实体区分开来的属性。ER 中的匹配规则对这些属性起作用。

定义 2. 比较器：一种用于确定两个属性值之间相似度的算法。比较器也称为相似性函数[14]。

定义 3. 项：由比较器，一个属性的两个值以及一个显式或隐式阈值组成的逻辑命题。如果比较器确定的两个值之间的相似度上升到阈值设置的水平，则该项为真。

定义 4. 规则：规则是由单个项或由两个或多个项[15]的逻辑结合(与运算)形成的逻辑命题。

定义 5. 规则集：规则集是由单个规则或两个或多个规则的逻辑与(或运算)形成的逻辑命题。

在 ER 系统中，两个记录是否链接是由两个记录中的属性值分别使规则集命题为真还是假来确定的。根据上面的定义，这意味着两条记录中的属性值必须在规则集的至少一条规则中使所有项为真。不链接的两个记录则不满足任何规则。

2.2. 匹配规则

$R\left[A_1, A_2, \cdots, A_n\right]$ 和 $S\left[A_1', A_2', \cdots, A_n'\right]$ 是带有 n 个对应属性 A_i 和 $A_i'\left(i \in (1,n)\right)$ 的两个关系。假设两个关系之间的属性已经对齐并作为输入提供。设 r,s 为 R,S 中的记录，并且 $r\left[A_i\right], s\left[A_i'\right]$ 为记录 r,s 中属性 A_i, A_i' 的值。相似度函数 $f\left(r\left[A_i\right], s\left[A_i'\right]\right)$ 计算阈值[0,1]的相似度值，例如编辑距离和 Jaccard 相似度。得分越高，说明 $r\left[A_i\right], s\left[A_i'\right]$ 的相似度越高。但在现实的数据集中，每个记录都包含很多的属性，每个属性需要的相似度函数和阈值都不同。

定义 6. 属性匹配规则属性匹配规则是一个三元组 $\approx(i, f, \theta)$ 带值表示布尔函数 $f\left(r\left[A_i\right], s\left[A_i'\right]\right) \geq \theta$，其中 $i \in [1,n]$ 是一个索引，f 是相似度函数，$\theta \in [0,1]$ 是一个阈值。属性匹配规则 $\approx(i, f, \theta)$ 估计为 ture,意味着对于特定的相似度函数和阈值，$r\left[A_i\right], s\left[A_i'\right]$ 匹配。

本文用 $r\left[A_i\right] \approx (f, \theta) s\left[A_i'\right]$ 代表相似度 f 和阈值 θ 的属性匹配规则。在后文中将其简写为 $r\left[A_i\right] \approx s\left[A_i'\right]$。

定义 7. 记录匹配规则。记录匹配规则是对不同属性的一组属性匹配规则的结合。简单来说，如果集合中所有属性匹配规则的值都为真，这两条记录 r 和 s 匹配。

定义 8. 析取匹配规则。析取匹配规则是对一组记录匹配规则的析取。如果记录 r 和 s 被至少一个该规则的记录匹配规则匹配，则该规则将匹配它们。

定义 9. 布尔公式匹配规则。布尔公式匹配规则是将属性规则作为其变量和连接项的任意布尔公式。

3. 生成简洁可解释的 ER 匹配规则

制定 ER 规则存在许多挑战。实体解析的精确度结果直接取决于 ER 规则。基于规则的实体解析通常采用采用所有属性匹配规则的方法来比较两个记录是否指向同一实体。例如：给定一个属于表 1 的记录 r 和属于表 2 的记录 s，一个简单的实体匹配规则如下：

$$\varphi_1 : r[name] \approx \theta s[name] \wedge r[address] \approx \sigma s[address] \wedge r[city] \approx s[district] \wedge r[tel] \approx s[tel]$$

$\approx\theta$ 和 $\approx\sigma$ 是不同的相似度函数，并且如果它们所有对应的属性具有相似的或等效的值，则规则 φ_1 就认为两个元组是指向同一实体。

然而，实际上，上面的规则 φ_1 可能会导致非常低的召回率，因为真实世界的数据可能包含多个问题，比如拼写错误、格式不同和丢失值。

Table 1. A instance of record R
表 1. 记录实例 R 表

记录 R	详细地址(address)	联系方式(tel)	城市地区 city	小区名 cname
r_1	和平长白 263-22 号 9 栋 3 号	123456	沈阳和平区	金地名悦
r_2	和平仙岛南路与长白二街交汇处 263-22 号 9 栋 3 号	123456	沈阳和平区	金地名悦
r_3	浑南区浑南东路 18 号 9 栋 23 号	888888	沈阳浑南区	金地滨河

Table 2. A instance of record S
表 2. 记录实例 S 表

记录 S	详细地址(address)	联系方式(tel)	城市地区 district	小区名 name
s_1	和平仙岛南路与长白二街交汇处 263-22 号 9 栋 3 号	123456	沈阳和平区	金地名悦
s_2	浑南区浑南东路 18 号 9 栋 23 号	888888	沈阳浑南区	金地滨河

为了提高识别精度，采用析取匹配规则的实体解析方法开始流行。例如，有如下两条规则：

$$\varphi_2 : r[name] \approx \theta s[name] \wedge r[address] \approx \sigma s[apt] \wedge r[city] \approx s[district]$$

$$\varphi_3 : r[name] \approx \partial s[name] \wedge r[tel] \approx s[tel]$$

例如 $\varphi_2 \vee \varphi_3$，如果两者有一个保留，那么表 1 中的 r 和表 2 中的 s 匹配。但从用户的角度来看，可以采用更自然的方法是用逻辑来指定规则。例如，下面的表示可能更友好：

$$\begin{aligned}\varphi_4 : &\textbf{if } r[name] \approx \partial s[name] \wedge r[tel] \approx s[tel] \\ &\textbf{then } r[name] \approx \partial s[name] \wedge r[tel] \approx s[tel] \\ &\textbf{else } r[name] \approx \partial s[name] \wedge r[address] \approx s[apt] \wedge r[city] \approx s[district]\end{aligned}$$

这种布尔匹配规则 φ_4 提供了一种更灵活的方法来表示匹配规则。

3.1. 基于语法和关联约束的布尔公式匹配规则

为了生成简洁且更精确的规则，本文将布尔公式匹配建模为一个语法约束引导合成(SyGuS) [16]问

题。语法约束引导合成将问题抽象表示为一组表达式的语法 G，以及表达式的约束 C，然后通过语法约束进行合成规则。语法 G 是一组递归重写规则(或表达式)，用于在一组终端符号和非终端(递归)符号上生成表达式。表达式提供了一种方法，可以通过依次应用表达式来从指定的初始符号派生表达式。下面给出了一个带有语法和关联约束的 SyGuS 问题示例：

语法 $expr \to expr \vee expr(\text{bound}:\text{B})$

$expr \to x \mid y \mid \neg x \mid \neg y$

约束 $(x \Rightarrow \neg y = expr) \wedge (y \Rightarrow \neg x = expr)$

上面的语法有一个非结束符号(也是初始符号) $expr$，它表示变量 x、y 或它们的负例的析取。表达式的空间是不受限制，但参数 B 限定了规则的使用次数。

布尔匹配规则(BMR)语法化。为了在 SyGuS 框架中表达 BMR 问题，本文使用布尔公式语法(G_{BMR})，定义如下：

语法 1： $G_{attribute} \to r[A_i] \approx (f,\theta) s[A_i']$

$i \in [1,n]$； $f \in F$； $\theta \in [0,1]$

语法 2： $(bound:N_d)\begin{cases} G_{BMR} \to G_{attribute}(bound:N_a) \\ G_{BMR} \to \neg G_{BMR} \\ G_{BMR} \to G_{BMR} \wedge G_{BMR} \end{cases}$

其中语法 $G_{attribute}$ 和 G_{BMR} 分别代表属性匹配规则和布尔公式匹配规则。边界 N_a 和 N_d 分别通过布尔公式限定 G_{BMR} 中的属性匹配规则($G_{attribute}$)的数量和语法扩展的深度，以此来限定布尔表达式的搜索空间。

本文在下文利用实例进行规则合成来控制 BMR 的结构，并提供一个简洁的公式作为输出。因为本文的 BMR 是尽可能小的，所以可以避免提供的示例产生的过度拟合。上面的 G_{BMR} 不能处理 Null 值(空值)，因为如果一个记录有一个 Null 值 A，那么就无法知道两个记录是否在属性 A 上匹配。本文在 G_{BMR} 中指定了一个新的语法 3，用来处理属性的空值。

语法 3： $G_{BMR} \to$ if $(\approx (1, noNulls, 1.0))$

then φ_2 else φ_3

$i \in [1,n]$； $f \in F$； $\theta \in [0,1]$

这个规则表示，如果一对记录中的匹配属性中没有空值，那么应该使用一个 BMR；否则应该使用不同的 BMR。这使合成器快速找到规则类似例子 φ_4 (第三节布尔匹配规则示例)。此方法不影响语法的可表达性和精确度。替代语法纯粹是为了使语法 G_{BMR} 和合成过程对有大量 *null* 的数据库更有针对性的。

约束。从语法 G_{BMR} 中选择的候选对象可以解释为布尔公式。已知正(*M*)和反(*D*)的例子，给定正(*M*)例和负(*D*)例，将 SyGuS 约束作为判定该 BMR 所提供例子的评价：

约束 1 $G_{BMR}(r_m, s_m) = ture, \forall (r_m, s_m) \in M$

约束 2 $G_{BMR}(r_d, s_d) = false, \forall (r_d, s_d) \in D$

基于语法约束的布尔匹配规则在本文的问题的情况下，部分程序还包括了行为约束。这种基于部分程序、行为约束和可能代码片段的语法的合成风格称为语法引导合成(SyGuS)，它最近受到了极大关注[17]。下面将给出一个完整的布尔匹配规则流程：

算法 1：attributeRule(e)

输入：实例 e

相似度函数集 *F*

属性匹配规则的属性数 N_a：

输出：是否相似(true/false)

1 $n_a \leftarrow 1$

2 while $n_a < N_a$ do//attribute-matching rules loop

3 $f \in F$

4 $\theta \leftarrow$ customSynth (i,f)

5 if (evalSimFn(e, n_a, f)>=θ then

6 return true

算法 2 BMRule(e,n_a)

输入：实例 e

相似度函数集 F

N_a: Bound on attribute-matching rules

输出: true/false

1 if (??)then

2 n_a++

3 returnattributeRule(e)

4 else if (??)

5 return (BMRule(e,A))

6 else if (??)

7 return BMRule(e,n_a) &&BMRule(e,n_a)

8 else return BMRule(e,n_a)‖ BMRule (e,n_a)

算法 3matchingRule(e, N_a)

输入：实例 e，属性长度 N_a

输出：BMRule(e,n_a)

1 $n_a \leftarrow 0$

2 b←BMRule(e,n_a);

3 if $n_a < N_a$ then

4 return b

在上述基于语法约束的布尔匹配规则流程中，hole(??)用来表示算法的未知方面，由语法进行填充。它包含以下几个部分：

1) 算法 1 属性匹配规则。attributeRule 是一种语法函数。例如，假设比较的记录有 4 个对齐的属性，那么属性匹配规则的 n_a 的取值在 1 和 4 之间。同理，属性匹配规则对 12 个相似函数的候选空间 F 进行了范围界定。阈值的选择使用自定义的合成程序(详见 4.2 节)。函数 evalSimFn 符号表示函数 f 对例 e 中记录的属性 n_a 的评价(详见 4.2 节)。

2) 算法 2 布尔匹配规则。BMRule 是一个最多 N_a 倍的内联属性规则函数的语法函数 (由通过引用传递的变量强制执行)和对自身的多次递归调用，以指定语法的可能扩展。

3) 算法 3 匹配规则函数。用匹配规则函数(matchingRule)表示用于实体解析的布尔公式规则(BMR)。生成的规则由正反例约束验证，输出满足约束的布尔匹配规则。

3.2. 构建实例样本和评价指标

本文通过扩展语法指导的合成(SyGuS)框架来建模 BMR 问题。考虑问题的两个方面:精确问题和优化问题。前者对应从示例中找到满足所有约束的 BMR。但这样一个完美的 BMR 经常在实践中不存在，因为示例可能会有错误，或者语法表达不够，不能正确地分类所有的示例。后者通过发现基于最优度度量的约束部分满足的 BMR 来缓和条件。

优化 SyGuS 问题。给定语法和正负示例的约束，优化 SyGuS 问题就是在语法中找到一个候选 BMR，它满足约束的子集，使给定的最优度量最大化。

虽然 SyGuS 不能解决我们研究的问题，但它仍然可以作为我们算法的一个组成部分(第 4 节)。

本文构建实例样本来引导生成 BMR 规则，所选用的正实例样本由属性显著度选取，根据属性的显著度，对属性的显著程度进行排序。它首先通过小样本数据集估计每个属性的显著度：在相似的记录对中计算某些属性值匹配个数占总体匹配数的比例，其值越高意味着这些匹配对记录对相似的贡献度就越大；对应地，在不匹配的记录中计算某些属性值匹配的个数占总体不匹配数的比例，其值越高意味着这些属性对匹配的贡献度越低，进而用以惩罚这些属性的贡献度，最后，两者共同确定属性显著度。公式如下：

属性正证据。

$$PS_{A_m}=\frac{\left|\left\{\left(r_i,r_j\right)\mid r_i\approx_\sigma r_j,A_m\left(r_i\right)\approx_{\theta_{A_m}}A_m\left(r_j\right)\right\}\right|}{\left|\left\{\left(r_i,r_j\right)\mid r_i\approx_\sigma r_j\right\}\right|},0<m<M,0<i\neq j<N \tag{1}$$

其中分子$\left|\left\{\left(r_i,r_j\right)\mid r_i\approx_\sigma r_j,A_m\left(r_i\right)\approx_{\theta_{A_m}}A_m\left(r_j\right)\right\}\right|$表示达到相似度阈值$\sigma$的记录中，某个属性的匹配个数，分母$\left|\left\{\left(r_i,r_j\right)\mid r_i\approx_\sigma r_j\right\}\right|$表示达到相似度阈值$\sigma$的记录对个数(即匹配个数)。

属性负证据。

$$NS_{A_m}=\frac{\left|\left\{\left(r_i,r_j\right)\mid r_i\neq_\tau r_j,A_m\left(r_i\right)\approx_{\theta_{A_m}}A_m\left(r_j\right)\right\}\right|}{\left|\left\{\left(r_i,r_j\right)\mid r_i\neq_\tau r_j,\exists A_nA_n\left(r_i\right)\approx_{\theta_{A_n}}A_n\left(r_j\right)\right\}\right|},0<i\neq j<N,0<m,n<M \tag{2}$$

其中分子$\left|\left\{\left(r_i,r_j\right)\mid r_i\neq_\tau r_j,A_m\left(r_i\right)\approx_{\theta_{A_m}}A_m\left(r_j\right)\right\}\right|$表示低于阈值$\tau$ (即不相似)的记录中，某个属性匹配的个数，分母$\left|\left\{\left(r_i,r_j\right)\mid r_i\neq_\tau r_j,\exists A_nA_n\left(r_i\right)\approx_{\theta_{A_n}}A_n\left(r_j\right)\right\}\right|$所有低于阈值$\tau$ (即不相似)的记录对。

那么单个属性综合显著度公式可以表示如下：

$$S_{A_m}=PS_{A_m}-NS_{A_m} \tag{3}$$

为解决PS_{A_m},NS_{A_m}可能不在同一量纲上的问题，可表示为

$$S_{A_m}=PS_{A_m}\cdot\left(1-NS_{A_m}\right) \tag{4}$$

评价指标本文想要生成一组最优的布尔匹配规则(BMR)，不用用户参与提供 BMR 的结构。为了评估 BMR，我们假设用户提供了一组示例，通过$\mathbf{E}=\mathbf{M}\cup\mathbf{D}$,其中 M 是正例，记录对代表相同的实体，D 是一种负例，即记录对代表不同的实体。对于 BMRΦ 和正例 M 负例 D。本文定义一个指标$\mu(\Phi,\mathrm{M},\mathrm{D})$，它得到一个 0 到 1 的值来评价 Φ 的性能。μ越大，Φ 越好。

$\mathbf{M}_\Phi\subset\boldsymbol{E}$是(r,s)所有例子的一个集合，是 r 和 s 通过 Φ 相匹配。最优度量的一些候选指标如下：

$$\mu_{\text{recall}} = \frac{\left|\mathbf{M}_{\Phi} \cap M\right|}{\left|M\right|}$$

$$\mu_{\text{precision}} = \frac{\left|\mathbf{M}_{\Phi} \cap M\right|}{\left|\mathbf{M}_{\Phi} \cap M\right| + \left|\mathbf{M}_{\Phi} \cap D\right|}$$

$$\mu_{\text{f1-score}} = \frac{2\text{precision} * \text{recall}}{\mu_{\text{recall}} + \mu_{\text{precision}}}$$

给定 R 和 S 两个关系，R 和 S 之间的对齐属性集合，以及正例 M 和负例 D,一个相似性函数 F 和一个评价指标 μ。ER-BMR 问题是发现一个 BMRΦ，使 μ 最大。

4. 规则合成算法

现有的 SyGuS 解决方案是为了解决精确 SyGuS 问题而设计的，而不是用来优化 SyGuS 问题，它发现一个 BMR，使给定的优化度量最大化。在本节中，先给出一个朴素方法解决优化 SyGuS 问题。然后，给出了新规则合成算法(4.2 节)。

4.1. 朴素方法和局限

事实上，给定一组示例、语法和约束，满足所有示例中所有约束的 BMR 可能不存在。因此，可以将目标转移到求满足示例子集的所有约束的 BMR。

朴素方法一般情况，一个简单的方法是从正例中选择多个随机的子集 S，并对 S 中的每个子集调用 SKETCH SyGuS 合成求解器[18]。然后可以选择性能最好的基于最优度度量的 BMR 对 E 中的所有实例进行了评价。

局限性。朴素方法有三个局限性。

(1) 从正例中选择多个随机的子集 S 可能导致生成的规则缺失某些高值匹配规则。因为随机选择具有概率性，不一定能完全找到所有的属性正例。

(2) 虽然选择了正例的子集，能够合成一个对示例集有良好覆盖的 BMR。但选择的正例并不一定肯定是正确的。因为我们生成的样本可能具有一定的错误。

(3) 符号求解器中，需要对数值相似函数和阈值进行推理，但现有求解器并不支持这样的推理。

4.2. 改进的规则合成算法(Rule Evolution)

下面介绍提出的新算法。首先在算法中从实例中正例中选择一个实例进行生成 BMR 规。

对于局限(1),对使用的正例集根据属性的显著度进行排序。每次根据属性显著度从高到低添加实例。

对于局限(2)，本文采用反例验证的思想，对生成的 BMR 规则，在实例上进行验证，如果反例的属性依然满足 BMR 规则，则 BMR 是合格的，否则将反例添加进实例集，重新生成 BMR，生成的 BMR 迭代直到没有反例满足 BMR 规则，将其输出。

对于局限(3)，通过添加了一个自定义合成器，用于在符号求解器中找到数值的阈值。

算法。在算法 1 中给出算法命名为 Rule Evolution，它有两个循环。外部循环(第 3~17 行)根据属性显著度抽取的实例样本来引导合成算法。在每次迭代中，给定一个样本 e_0 (第 5 行)，它从(第 6 行)开始 Synth，然后调用内部(CEGIS)循环(第 7~16 行)。在每次迭代中，它首先合成一个 BMR(第 8 行)，然后它将在反例集中验证发现反例(9 行)。如果没有反例，过程将终止(10~11 行),否则一个随机选择的反例将被添加到未来的 CEGIS 迭代中(12~14 行)。当前最好的 BMR 将回馈(15 行)。最后，算法将返回一个 BMR (第 18

行)。下文将解释该算法的各个部分。

算法 1：Rule Evolution

输入：实例集(包含正例反例)

相似度函数集 F

语法边界 $G_{BMR}\left(N_a, N_d\right)$

最优评价 μ

N_a: Bound on attribute-matching rules

K_{CEGIS}: Bound on CEGIS iterations

输出：Φ^*: A BMR from $G_{BMR}\left(N_a, N_d\right)$ maximizing μ

1 $n_a \leftarrow 1$

2 $\Phi^* \leftarrow 0$

3 while $n_a < N_a$ do //attribute-matching rules loop

4 $i \leftarrow 0$

5 $e_0 \leftarrow \text{sample}(\text{M})$

6 $\text{E}_{\text{syn}} \leftarrow \text{List}\left(e_0\right)$

7 while $i<K_{CEGIS}$ do // CEGIS loop

8 $\Phi_i \leftarrow \text{Synth}\left(G_{BMR}\left(N_a, N_d\right), \text{E}_{\text{SYN}}, f\right)$

9 $\overline{E}_{\Phi_i} \leftarrow Verify\left(\Phi_i, E\right)$ // Counter-examples

10 if $\overline{E}_{\Phi_i} = \varnothing$ then

11 return Φ_i

12 else

13 $e_{i+1} \leftarrow \text{sample}\left(\overline{E}_{\Phi_i}\right)$

14 $\text{E}_{\text{SYN}} \leftarrow \text{E}_{\text{SYN}} * append\left(e_{i+1}\right)$

15 $\Phi^* = \arg\max_{\Phi \in \left\{\Phi^*, \Phi_i\right\}} \mu\left(\Phi, M, D\right)$

16 $i \leftarrow i+1$

17 $n_a \leftarrow n_a + 1$

18 return Φ^*

自定义的合成例程：我们从核心的合成例程(第 8 行)开始，它解决了 SyGuS 问题，即它从有界语法 $G_{BMR}\left(N_a, N_d\right)$ 中搜索满足 E_{SYN} 中示例所有约束条件的候选 BMR。Sketch 通过符号化地分析语法和约束的每个部分，并将搜索问题简化为一个布尔可满足性(SAT)问题。用 Sketch 直接解决搜索问题是不切实际的，因为它涉及到复杂的数值函数的推理。为了用 Sketch 解决这个问题，本文提出一个新的算法，允许 Sketch 与处理相似函数分析和综合阈值的定制求解器合作，而 Sketch 则为 BMR 做出离散决策。

Sketch 合成器做了以下工作：

1) 用多个元素或属性匹配规则扩展 G_{BMR} 语法；

2) 在 E_{SYN} 中选择示例为正(E_+)或反($E\_$)作为扩展后的 BMR 的元素；

3) 选择属性 $i \in [1, n]$ 和相似函数 $f \in F$ 用于这些元素或规则。

自定义求解器发现一个数值阈值，将通过 Sketch 为元素选择的正示例(E')和反示例(E)分隔开(如果存在的话)。此外，它要求 Sketch 求解器回溯并做出可选的离散决策。这个求解器将在 Sketch 中被多次调

用。算法 2 给出了该求解器的伪代码。

算法 2：自定义求解器

输入：f: Chosen similarity function

a: 匹配的属性 ID

E_+: Examples chosen to be positive

E_-: Examples chosen to be negative

输出: exists: Does a valid threshold exist

θ: A valid threshold separating E_+ & E_-

1 $\theta_{\text{atmost}} \leftarrow 1.0$

2 for $e \in E_+$ do

3 $\theta_{\text{atmost}} = \min\left(\theta_{\text{atmost}}, evallSIMFn\left(e,a,f\right)\right)$

4 $\theta_{\text{atleast}} \leftarrow 0.0$

5 for $e \in E_-$ do

6 $\theta_{\text{atleast}} = \max\left(\theta_{\text{atleast}}, evallSIMFn\left(e,a,f\right)\right)$

7 if $\theta_{\text{atleast}} < \theta_{\text{atmost}}$ then

8 exists←true

9 $\theta \leftarrow \dfrac{\theta_{\text{atmost}} + \theta_{\text{atleast}}}{2}$

10 else

11 exists←false

本文提出的算法 5 利用相似度函数的单调性进行来区分正反例。一般来说，单调性要求在至少一个相似度函数上，任何一对匹配记录都具有比非匹配对更高的相似度值。所以，从正例中选择某一属性的相似度值最小的与同一属性的反例中相似度值最大进行比较，如果某一属性的反例的相似度值小于正例的相似度值，那么这个属性合理，相似度函数可以使用，并且区分正反例的阈值为 $\theta = \dfrac{\theta_{\text{atmost}} + \theta_{\text{atleast}}}{2}$。

反例引导归纳合成(CEGIS)。本文使用反例引导归纳合成 GEGIS 的思想构建迭代算法，该算法有两个阶段：Synth (第 8 行)和 Verify (第 9 行)。其想法是迭代地合成一个适用于少量示例的 BMR，并通过仅添加合成程序目前没有正确处理的那些示例，以巧妙的方式扩展此集合。

假设算法选择了作为第一个正例，Synth 返回函数 $\Phi_0 = J\text{arrard}\left[name\right] \geq 0.6$。Verify 在 $\mathbf{M} \cup \mathbf{D}$ 中的所有例子上验证这个函数，并随机选取一个作为反例，即一个示例没有被函数正确匹配，因为对于示例，name 属性不完全相等。然后，它将把这个反例添加到 E_{SYN} 集合中，并开始下一个 CEGIS 迭代。在此迭代中，Synth 可以返回匹配所有正确的例子函数 $\Phi_1 = J\text{arrard}\left[name\right] \geq 0.8$。

在迭代 i 次，Synth 使用当前可用的例子 $\text{E}_{\text{SYN}} = \{e_1, e_2, \cdots\}$，并且用 Sketch 解决 SyGuS 问题,从边界语法 $G_{BMR}\left(N_a, N_d\right)$ 中来找到一个 BMRΦ_i，BMRΦ_i 可以正确处理所有的例子。另一方面，Verify 考虑了 $\text{E} = \mathbf{M} \cup \mathbf{D}$ 所有的例子，来识别了 Φ_i 未正确处理的实例。反例 e_{i+1} 从 Φ_i 中挑选出来添加到集合 E_{SYN} 中，以便在下一次 Synth 阶段中运用。这个过程将一直持续下去，直到 Synth 无法找到满足当前示例集的 BMR，或者直到它执行了 KCEGIS (CEGIS)迭代。如果验证不能找到任何反例(即 $\bar{E}_{\Phi_i} = \varnothing$)，算法终止并输出 Φ_i 作为最优 BMR，因为它可以正确地处理所有例子 E。

属性匹配引导合成。本文在 CEGIS 循环之上构建另一个循环，使用不同的初始示例(e0)多次重新启

动它。其思想是，按照属性显著度的排序将正例输入，在 CEGIS 循环中进行验证，验证没有反例后，再次选择新的属性正例进行迭代，如果重启的数量达到 N_a (属性匹配截止)，那么该算法终止并输出 CEGIS 和属性匹配迭代中最好的 BMRΦ^*。

5. 实验

5.1. 实验方法

为了验证本文提出的 Rule Evolution 的有效性，本文与可解释的实体解析和不可解释的实体解析进行比较。具体的对比方法如表 3 所示。

我们将 Rule Evolution 与决策树，SVM 进行了比较。所有 ML 方法都将 ER 转换为二进制分类问题。尽管 SVM 的输出缺乏逻辑可解释性，但是决策树可以解释为具有多个 DNF 子句的布尔公式，这些子句遍历路径导致正分类。

本文与基于规则的学习方法进行了比较。选择基于启发式的方法 SIFI 来与 Rule Evolution 对比，该方法根据专家提供的 DNF 语法搜索最佳相似性函数和阈值以进行属性比较。相比之下，Rule Evolution 会自动发现 BMR，而无需任何专家提供的规则结构。

Table 3. Comparison algorithm table
表 3. 对比算法表

算法种类	具体对比算法	
可解释的实体解析	SIFI (基于规则的算法)	低深度的决策树
不可解释的实体解析	SVM	高深度的决策树

5.2. 实验设计

本文在 Python 3 中实现了作为与 Sketch 合成工具(用 Java 和 C++编写)交互的 RuleSynth 脚本。并在 Python 中实现了 SVM。其他两种基准方法，即 SIFI 和决策树，分别来自[12]和[19]的作者。SIFI 是用 C++实现的。

对于决策树，我们分别显示深度 3，4 和 10 (Weka 中的默认配置)的结果。对于 SVM，本文将两个内核的结果分开。对于 SVM，我们使用平衡的类权重作为省力的配置，以优化 F 度量。我们还使用专家提供的默认配置和语法运行 SIFI。

本文提出的 Rule Evolution，为了实验做了如下的设计：

1) 设置语法深度 N_a 为 4；

2) CEGIS 迭代的最大界限为 $K_{CEGIS} = 800$，并且每次 CEGIS 迭代的超时时间为 15 分钟，以使其直到找到无法合成有效规则的示例集或超时并选择直到那时为止的最佳规则；

3) 属性匹配规则的数量 N_a：对于 RuleSynth，我们设置 N_a 为 8，和基于规则的数量一致。

5.3. 实验结果

本文对所使用的每个数据集进行了 5 倍交叉验证，将数据随机分为 5 个相等的分数(倍数)，并进行了 5 个实验。在每个实验中，20%是测试集，而其余的 80%为训练集。我们将测试集所有获得的平均 F 度量作为评估指标。对于我们比较的每种技术，我们都使用相同的数据集，对于每个实验，本文可能会为 ML 技术的参数找到不同的最佳值。实验结果如图 1 实验结果所示。

与可解释的决策树进行比较。本文将提出的算法 Rule Evolution 与可解释的决策树(深度为 3/4 的)进

行有效性评估(5 倍平均 F 度量)。图显示了不同可解释技术的平均 F 量度。我们观察到，对于所有数据集而言，Rule Evolution 的 F-measures 要比深度为 3 的决策树更高，与深度为 4 的决策树相比，在盘和栋的数据集上，Rule Evolution 的 F-measures 更高，在户的数据集上，深度为 4 的决策树的 F-measures 更高。

有效性与不可解释的方法。本文将 Rule Evolution 与两种不可解释的 ML 算法进行比较：1) 深度为 10 的决策树，2) SVM，图给出了不可解释方法的评估分数，可以看到 Rule Evolution 在不同的数据集上与不可解释的 ML 算法的平均 F-measure 值互有高低。但是，这些 ML 算法的有效性代价明显高于。且这些算法不可解释。

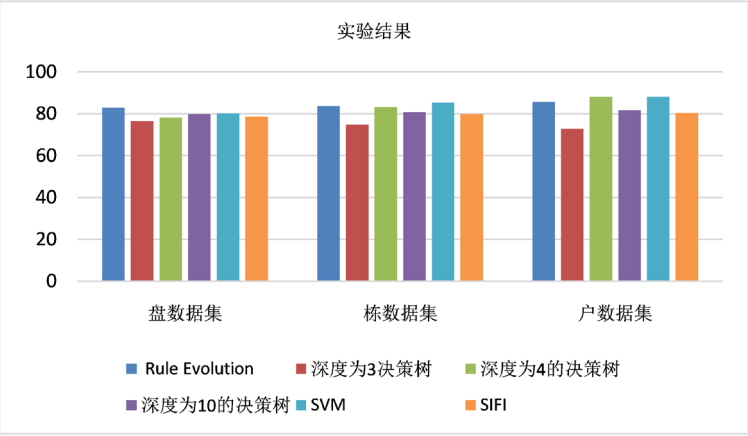

Figure 1. Experimental result
图 1. 实验结果

与基于规则的算法比较。最后将 Rule Evolution 与 SIFI 进行了比较。SIFI 要求专家提供的 DNF 模板作为输入，并完成该模板以生成规则。相反，Rule Evolution 自动发现规则，从而减少了专家的工作量。图 1 显示，对于所有数据集，Rule Evolution 性能均优于 SIFI。

6. 结论

本文提出了一种基于布尔匹配规则的实体解析方法。本文给定高级规则规范作为语法和正反示例作为约束，为了在合成器上自动合成规则，本文提出了规则合成算法(Rule Evolution)来自动生成最优 BMR 规则。最后在真实数据集上进行了实验，实验结果验证了算法的有效性。实验结果表明其 F-measures 可以与传统方法和现在主流的方法相媲美，甚至更高。

参考文献

[1] Elmagarmid, A.K., Ipeirotis, P.G. and Verykios, V.S. (2007) Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, **19**, 1-16. https://doi.org/10.1109/TKDE.2007.250581

[2] Jain, A.K., Murty, M.N. and Flynn, P.J. (1999) Data Clustering: A Review. *ACM Computing Surveys*, **31**, 264-323. https://doi.org/10.1145/331499.331504

[3] Winkler, W. (2006) Overview of Record Linkage and Current Research Directions. Technical Report, Statistical Research Division, U.S. Bureau of the Census, Washington DC.

[4] Fellegi, I. and Sunter, A. (1969) A Theory for Record Linkage. *Journal of the American Statistical Association,* **64**, 1183-1210. https://doi.org/10.1080/01621459.1969.10501049

[5] Bilenko, M. and Mooney, R.J. (2003) Adaptive Duplicate Detection Using Learnable String Similarity Measures. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Washington DC, August 2003, 39-48. https://doi.org/10.1145/956750.956759

[6] Gokhale, C., Das, S., Doan, A., Naughton, J.F., Rampalli, N., Shavlik, J. and Zhu, X. (2014) Corleone: Hands-off Crowdsourcing for Entity Matching. *Proceedings of the* 2014 *ACM SIGMOD International Conference on Management of Data*, Snowbird, June 2014, 601-612. https://doi.org/10.1145/2588555.2588576

[7] Cohen, W.W. (2000) Data Integration Using Similarity Joins and a Word-Based Information Representation Language. *ACM Transactions on Information Systems*, **18**, 288-321. https://doi.org/10.1145/352595.352598

[8] Singla, P. and Domingos, P. (2006) Entity Resolution with Markov Logic. 6*th International Conference on Data Mining*, Hong Kong, 18-22 December 2006, 572-582. https://doi.org/10.1109/ICDM.2006.65

[9] Firmani, D., Saha, B. and Srivastava, D. (2016) Online Entity Resolution Using an Oracle. *Proceedings of the VLDB Endowment*, **9**, 384-395. https://doi.org/10.14778/2876473.2876474

[10] Wang, J., Kraska, T., Franklin, M.J. and Feng, J. (2012) Crowder: Crowd Sourcing Entity Resolution. *Proceedings of the VLDB Endowment*, **5**, 1483-1494. https://doi.org/10.14778/2350229.2350263

[11] Elmagarmid, A.K., Ilyas, I.F., Ouzzani, M., Quiané-Ruiz, J.-A., Tang, N. and Yin, S. (2014) NADEEF/ER: Generic and Interactive Entity Resolution. *Proceedings of the* 2014 *ACM SIGMOD International Conference on Management of Data*, Snowbird, June 2014, 1071-1074. https://doi.org/10.1145/2588555.2594511

[12] Wang, J., Li, G., Yu, J.X. and Feng, J. (2011) Entity Matching: How Similar Is Similar. *Proceedings of the VLDB Endowment*, **4**, 622-633. https://doi.org/10.14778/2021017.2021020

[13] Talburt, J. (2010) Entity Resolution and Information Quality. Morgan Kaufmann, Burlington. https://doi.org/10.1016/B978-0-12-381972-7.00003-8

[14] Naumann, F. and Herschel, M. (2010) An Introduction to Duplicate Detection. *Synthesis Lectures on Data Management*, **2**, 1-87. https://doi.org/10.2200/S00262ED1V01Y201003DTM003

[15] Whang, S. and Garcia-Molina, H. (2010) Entity Resolution with Evolving Rules. *Proceedings of the VLDB Endowment*, **3**, 1326-1337. https://doi.org/10.14778/1920841.1921004

[16] Alur, R., Bodik, R. and Dallal, E. (2015) Syntax-Guided Synthesis. *Dependable Software Systems Engineering*, **40**, 1-25.

[17] Solar-Lezama, A. (2009) The Sketching Approach to Program Synthesis. *Asian Symposium on Programming Languages and Systems*, Seoul, 14-16 December 2009, 4-13. https://doi.org/10.1007/978-3-642-10672-9_3

[18] Singh, R., Meduri, V., Elmagarmid, A., Madden, S. and Papotti, P. (2017) Synthesizing Entity Matching Rules by Examples. *Proceedings of the VLDB Endowment*, **11**, 189-202. https://doi.org/10.14778/3149193.3149199

[19] Ganzinger, H., Hagen, G., Nieuwenhuis, R., Oliveras, A. and Tinelli, C. (2004) Dpll(*T*): Fast Decision Procedures. *International Conference on Computer Aided Verification*, Boston, 13-17 July 2004, 175-188. https://doi.org/10.1007/978-3-540-27813-9_14