Hans Journal of Data Mining 数据挖掘, 2021, 11(4), 241-252
Published Online October 2021 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2021.114022

Hans汉斯

一种基于树模型的关联实体解析方法

王泽龙，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

收稿日期：2021年9月20日；录用日期：2021年10月22日；发布日期：2021年10月29日

摘　要

在大数据时代，Web数据呈现多样性和关联性，在实体解析(Entity Resolution)中体现为解析的数据集往往包含多个实体集，实体集之间具有关联关系。这种关系导致解析一个实体集的结果可以使另一个实体集的解析受益，这种具有关联关系的实体解析称为关联实体解析(Related Entity Resolution)。本文针对一对多类型关联实体的实体解析问题提出了关联树模型，并引申出相似节点、相似树、相似性传递等概念。我们提出了一种基于树的一对多关联实体解析方法。初始时依据关联实体的关联关系构建关联树；将本节点的属性相似度和关联子节点的部分属性相似度结合起来判断节点是否匹配；基于深度优先原则遍历关联树的每一个节点，依据节点的实体解析结果筛选出满足相似传递性的部分子节点，在遍历完叶子节点的过程中，生成部分相似子树，再对根节点的子节点集中节点进行相似匹配，寻找其他相似子树。本文提出一种相似树索引来表示关联树的匹配结果。用房地产大数据通过实验验证文中提出的关联树搜索算法比已有的关联实体识别算法在一对多关联实体上效率更高。

关键词

关联实体，关联树，相似节点，相似树，实体解析

A Related Entity Resolution Algorithm Based on Tree Model

Zelong Wang, Gui Li, Zhengyu Li, Ziyang Han, Keyan Cao

School of Information & Control Engineering, Shenyang Jianzhu University, Shenyang Liaoning

Received: Sep. 20th, 2021; accepted: Oct. 22nd, 2021; published: Oct. 29th, 2021

Abstract

In the era of big data, Web data is featured with obvious diversity and relevance. In Entity Resolu-

文章引用: 王泽龙, 李贵, 李征宇, 韩子扬, 曹科研. 一种基于树模型的关联实体解析方法[J]. 数据挖掘, 2021, 11(4): 241-252. DOI: 10.12677/hjdm.2021.114022

tion, it is reflected in the parsed data set that often contains multiple entity sets, and there is an association relationship between entities. Based on that relationship, the result of parsing one entity set can benefit the parsing of another entity set, and that kind of entity resolution with an associated relationship is called Related Entity Resolution. In this paper, focusing on the one-to-many types of related entities, the concept of relevance tree was proposed, and concepts such as similar nodes, similar trees, and similarity transfer were further derived. A relevance tree search algorithm was proposed in the study. Initially, a relevance tree was constructed according to the association relationship of the associated entities. Then, based on the depth-first principle, it traversed each node of the relevance tree, screened out some sub-nodes that meet the similarity transitivity based on the entity analysis results of the node, and continued to deepen the relevance tree until the leaf nodes were all traversed and subtrees of partial similarity were generated. After that, it matched the nodes in the sub-node set of the root node to find other similar subtrees. Based on the above, the similar tree index was proposed to represent the matching result of the relevance tree. It proposed a relevance tree merging algorithm, which merged nodes of the three types of relevance subtrees that may appear in the relevance tree based on the similarity tree index on the basis of maintaining the relationship between entities, and generated the "neat" relevance tree. Through experiments, it verified that compared with the existing related entity recognition algorithm, the relevance tree search algorithm proposed in the paper achieved higher efficiency on the one-to-many real estate related data set.

Keywords

Related Entity, Related Tree, Similar Nodes, Similar Tree, Entity Resolution

1. 引言

实体解析(Entity Resolution)是数据清洗的重要部分，对于数据挖掘和数据集成也至关重要[1] [2] [3] [4] [5]。在实践中，数据集成和数据挖掘常常涉及多个数据源，不同的数据提供方对同一个事物即实体(Entity)可能会有不同的描述。由于拼写错误、缩写方式不同、描述格式不同、属性值缺失、实体的某些属性值随着时间推移发生变化(比如年龄、居住地点、工作单位)等，描述同一实体的不同记录存在差异[6]。实体解析是从一个或多个数据源中匹配描述同一现实世界实体记录的过程。为了实现高质量的数据集成和数据挖掘，需要对数据进行实体解析。凭借数十年的研究，传统的实体解析拥有高效且有效的算法解决方案[7] [8] [9] [10]。但传统的实体解析仅针对单一类型的实体解析问题，匹配的记录之间是独立的。

大数据时代，数据呈现多样性和关联性，在实体解析中体现为数据集包含多个实体集，实体集之间具有关联关系，我们称为关联数据集。这种对关联数据集进行的实体解析，即关联实体解析(Relational Entity Resolution)。本文中，我们专注于对一对多关联实体进行实体解析。在关联实体解析中，关联数据集由多个实体集和它们之间的关系组成。在这种数据集中，某些实体的解析可能会影响到其他实体的解析。现有的关联实体解析[11] [12] [13] [14] [15] [16]中，很少有人专注于一对多关联实体的实体解析问题。比如 Dong 等人[14]通过关联关系将可能匹配的记录作为节点，构建依赖图来进行关联实体解析；Bhattacharya 等人[11]利用引文数据中的共同作者关系，进行基于关系的聚类，来实现关联实体解析，这

些关联实体解析方法都将实体间的关联关系假定为平等的，这类关联实体解析方法在一对多关联实体的实体解析上效果不是很理想。因此为了高效地解决一对多关联实体的实体解析问题，本文提出一种关联树模型，利用树的层次特性来描述关联实体之间的一对多关联关系。基于关联树模型，我们提出了一种关联树搜索算法，在树中高效有序地搜索相似匹配的节点对，并提出一种相似树索引来表示关联树的匹配结果，以便于后续的关联树合并与节点合并。传统的属性相似度算法无法发掘关联实体的关联关系进行实体解析，因此本文基于属性相似度提出一种关联实体匹配算法，将本节点的属性相似度和关联子节点的部分属性相似度结合起来判断节点是否匹配。本文的主要贡献如下：

1) 以集合划分为基础，基于基本的关联实体解析模型，提出基于一对多关联关系的关联树实体解析模型，并引申出相似节点、相似树、相似性传递等概念。

2) 为了查找相似的关联实体，本文提出一种关联树搜索算法，该算法基于一对多关联实体的相似性传递性质来进行深度优先搜索。为了表示关联树的匹配结果，便于后续的关联树合并，本文提出一种相似树索引，映射关联树中的相似节点。

3) 针对一对多关联实体解析问题，本文提出一种节点匹配算法，将本节点的属性相似度和关联子节点的部分属性相似度结合起来判断节点是否匹配。

4) 在房地产大数据的真实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关联树模型的实体解析方法与现有的关联实体解析方法相比具有良好效果。

2. 关联树模型

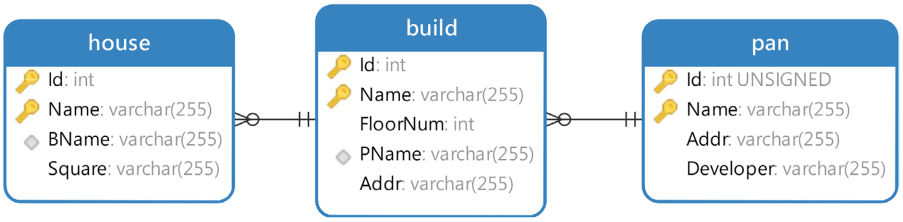
2.1. 关联实体解析模型

实体类型指的是实体的关系模式类型，比如盘是一种实体类型而栋是另一种实体类型。每一个记录 r 都描述真实世界的一个实体，即都属于一个实体类型。由于数据常常源于多个数据源，因此记录集中可能存在多个记录描述同一真实世界的实体。记录包括多个属性，分为简单属性和引用属性两类：简单属性的值是字符串、整型、浮点型等简单类型；引用属性的值是对另外一个实体类型的记录的引用。图 1(c)中，栋名是栋实体的简单属性；盘名是栋实体记录指向盘体的引用，是引用属性。

我们将数据集划分为由一个个簇组成的集合，每个簇都代表一个真实世界的实体。因此我们将簇 c 定义为一组记录。给定实体集 $R=\{r_1,\cdots,r_n\}$，实体集 R 即为簇集 $P=\{c_1,\cdots,c_m\}$，其中 $c_1\cup\cdots\cup c_m=R$，$c_i\in P$ 且 $i\neq j$，$c_i\cap c_j=\varnothing$。

传统实体解析算法解析单个实体集 R。实体解析算法接收 R 的一个划分 P_R 作为输入，并返回 R 的另一个划分 P_R'。划分是表示实体解析算法输出的一种通用方法。比如当前只考虑图 1(c)中的栋实体集 $B=\{b_1,b_2,b_3,b_4,b_5\}$，$B$ 的实体解析问题就是对 B 的划分问题。输入的实体集 B 可以看作单个划分 $P_B=\{\{b_1\},\{b_2\},\{b_3\},\{b_4\},\{b_5\},\{b_6\},\{b_7\}\}$，实体解析算法输出的为划分 $P_B'=\{\{b_1,b_4\},\{b_2,b_5\},\{b_3,b_6\},\{b_7\}\}$。如果我们在 R 的一个划分 P_R 运行实体解析算法产生 R 的另一个划分 P_R'，这一过程称为传统实体解析[17]。

关联实体解析不同于传统的实体解析，要对多类型的，关联的实体集同时进行实体解析，实体解析过程中要考虑不同类别的记录间的关联关系。接下来举例说明，图 1(b)~(d)分别为房地产数据集种的三个实体集，图 1(a)为三个实体集的关联关系，用 ER 图的形式表示，其中包括盘与栋之间的一对多关系，栋与户之间的一对多关系；在已知房地产数据集种各实体集的关联关系(图 1(a))情况下，如何判断三种类型实体集的重复记录就是一个关联实体解析问题。关联实体解析利用实体间关联关系优化不同类型实体的实体解析顺序提高实体解析的效率。比如 b_1 和 b_4 匹配后，因为栋实体集 B 与户实体集 H 具有关联关系，则与 $\{h_1,h_2,h_3\}$ 匹配的记录可能在 $\{h_7,h_8\}$ 中，应该优先对这些实体子集进行匹配。



（a）ER图

Id	Name	Addr	Developer
p_1	银座公寓二期	港盛路北侧	大连万科
p_2	新银座二期	港盛路130号	大连万科
p_3	新银座公寓	香周路东侧，港盛路北侧	万科

（b）盘实体集

Id	Name	PName	FloorNum	Addr
b_1	2号楼	银座公寓二期	13	港盛路3-2
b_2	C3号楼	银座公寓二期	13	港盛路3-3
b_3	4号楼	银座公寓二期	13	港盛路3-4
b_4	2号楼	新银座二期	13	中山区港盛路2号
b_5	3号楼	新银座二期	13	中山区港盛路3号
b_6	4号楼	新银座二期	13	港盛路4号
b_7	1~3号楼	新银座公寓	22	港盛路北A区

（c）栋体集

Id	Name	BName	Square
h_1	2-1-1(跃)	2号楼	75.13
h_2	2-12-4	2号楼	82.59
h_3	2-5-4	2号楼	75.13
h_4	3-1-2（商）	3号楼	82.59
h_5	3-10-4	3号楼	75.13
h_6	4-1-4(跃)	4号楼	57.39
h_7	2-5-4 (跃)	2号楼	75.13
h_8	2-10-4 (跃)	2号楼	57.39
h_9	3-11-10	3号楼	75.13
h_{10}	4-1-4	4号楼	57.39
h_{11}	3-20-2	1~3号楼	146.96

（d）户体集(H)

Figure 1. Real estate data set D
图 1. 房地产数据集 D

2.2. 关联树模型

现实世界的实体联系一般都可以转换为一对多的联系，而这种一对多实体的实例的对应关系可以表示为树结构，比如房地产领域的楼盘、楼栋和户信息之间的关系可以表示为：

定义 1：实体：表示为 $E=\left(A_1,A_2,\cdots,A_i,\cdots,A_n\right)$，其中，$A_i$ 表示实体 E 的 i 个属性，$Dom\left(A_i\right)$ 为属性 A_i 的定义域。

定义 2：记录：表示为 $e=\left(A_1:V_1,A_2:V_2,\cdots,A_i:V_i,\cdots,A_n:V_n\right)$ 形式的属性和属性值对的集合。

定义 3：关联实体：在实体联系(*E-R*)中的具有一对多联系的实体称为关联实体，实体集 R,S 中表现出一对多的实体联系，即为 $R(1)\to S(n)$。

定义 4：关联属性：在一对多联系的两个实体的一方的关键字属性或多方实体的外键属性称为关联属性。

定义 5：关联子树：给定一个一对多的实体关系 $R(1)\to S(n)$，则关联子树 T_r 定义为如图 2 所示的二层树，表示两类实体的实例之间对应关系，其中根节点为 r，子节点集为 S_r， $S_r=\{s_1,s_2,s_3\}$。

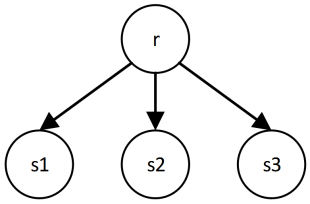


Figure 2. Relevance subtree
图 2. 关联子树

定义 6：关联树 T 由无数个关联数据集 D 中一对多关系的关联实体构成的关联子树 T_i 组成，$T=T_1\bigcap T_2\bigcap\cdots\bigcap T_i$。

关联树 T 为一个有向无环连通图，节点 $N(T)$ 和边 $E(T)\subseteq N(T)\times N(T)$。实体 r_i,s_j 为关联数据集 D 内实体集 R,S 的实体，即 $r_i\in R$， $s_j\in S$，则创建节点 v,w，若实体 r_i 与实体 s_j 为关联实体，其中 r_i 为一方，s_j 为多方，则为节点 v,w 创建边 (v,w)，其中，节点 v 为父节点，节点 w 为子节点，即 $w\in p.childrens$；如果 $x\in v.childrens$ 则 x 一定是节点 v 的子节点，v 一定是 x 的父节点。关联树的根节点 v_{root} 为伪节点，无意义(无输入边的节点是根节点)。每个节点 v 都有一个标签 $lbl(v)$，它存储着节点的数据；标签不一定是唯一的。

2.3. 关联树的概念与性质

在一对多关联实体数据集 D 中，我们将数据集 D 构建为关联树 T_D。因此对数据集 D 中关联实体的实体解析问题转为对关联树中 T_D 中节点的实体解析问题。比如房地产关联数据集 D 的盘实体集，栋实体集，户实体集可以依据一对多关联关系构建为关联树 T_D，如图 3(a)所示。

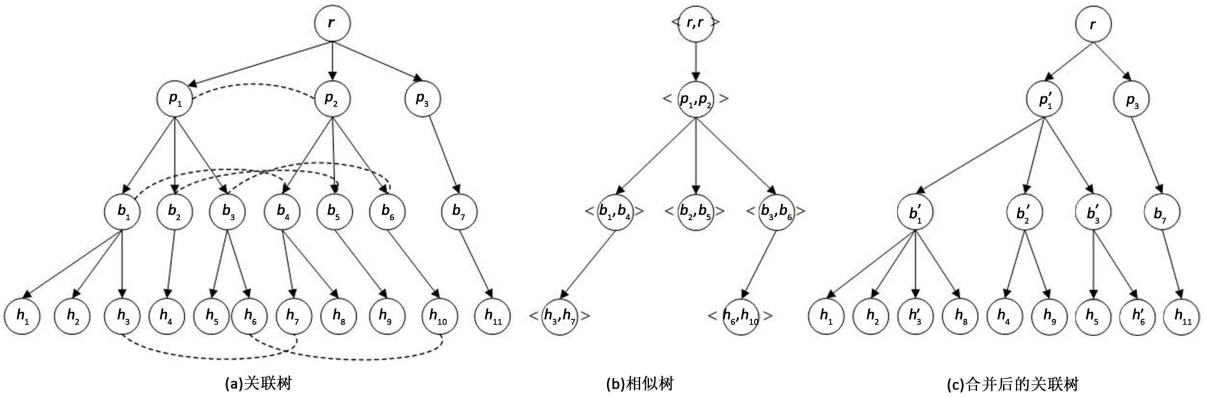


Figure 3. Relevance tree entity resolution
图 3. 关联树实体解析

定义 7：节点相似度：关联树的任意一个节点都表示为现实世界中的一个具体实体，关联树中任意

两个节点 v 和 w 的相似度表示为：

$S(v,w)=f_{sim}(v,w)$，其中 f_{sim} 是两个节点的相似度计算函数。

定义 8：相似节点：如果两个节点 v,w 的相似度 $S(v,w)$ 大于规定的阈值 θ，即 $v\approx w$，则认为这两个节点是相似的，记为相似节点对 $\langle v,w\rangle$。相似节点对分为两种情况：

同构相似，指的是两个节点的属性结构相同，属性值的相似度大于给定的阈值；

异构相似，指的是两个节点的属性结构有相交部分，且它们的属性值相似度大于给定的阈值；

引理 1：关联树种每层节点都代表一类实体，实体解析中只有实体类别相同才需判断记录是否指向同一实体。因此执行相似度匹配函数的节点 v,w 必须为同层节点才有意义，即 $v.label=w.label$ 。

引理 2：关联树中实体间关系为一对多关联关系。在一对多关系中，当且仅当一方为相似节点时，对多方进行相似节点匹配才有意义。比如图 1(b)中的房地产数据集 D 中，只有盘记录集中的 p_1 和 p_2 相似后，表明两个记录为同一个现实世界中的楼盘实体，因此 p_1 关联的子节点集 $B_{p_1}=\{b_1,b_2,b_3\}$ 和 p_2 关联的子节点集 $B_{p_2}=\{b_4,b_5,b_6,b_7\}$ 才可能包含相似的关联栋节点，有比较的解释意义。即，当两个关联子树的父节点 v,w，若 $v\approx w$ 则比较其子节点集 S_v 与 S_w 中的节点才有意义(包含相似节点)。

根据引理 2，我们可以引申出：

引理 3：若关联树 T_D 中节点 v,w 指向同一现实实体，为相似节点，则其父节点 $p(v),p(w)$ 必然也为相似节点，即 $p(v)\approx p(w)$，直至根节点。

由引理 3 可知，所有相似节点的父节点相似，则关联树中相似节点对具有树形结构，我们将这种具有树形结构的相似节点对集合称为相似树。

定义 9：相似子树：如果关联子树对 T_i,T_j 的根节点对 $\langle i,j\rangle$ 为相似节点，则我们称关联子树对 T_i,T_j 为相似子树，其子节点集 S_i,S_j 可能包含相似节点 $\langle v,w\rangle$，其中 $v\in S_i,w\in S_j,p(v)=i,p(w)=j$ 。比如图 3(a)中关联子树 T_{p_1} 与 T_{p_2} 的根节点 p_1 和 p_2 相似，依据引理 2，关联子树 T_{p_1} 与 T_{p_2} 的子节点集可能包含相似节点，即 T_{p_1} 与 T_{p_2} 为相似子树。

定义 10：相似树：关联树 T_D 中相似节点对 $\langle v,w\rangle$ 具有树型结构，即关联子树对 $\langle T_v,T_w\rangle$ 中相似子节点构成相似树 T_{sin}，相似树中节点表示为相似节点对，边为相似节点对之间的关联关系。相似树 T_{sin} 为关联子树对 $\langle T_v,T_w\rangle$ 的子树，分为三种情况：

1) 同构相似，指的是两个根节点及两个根节点下的所有子树的根节点都是同构相似；

2) 异构相似，指的是两个根节点是异构相似，两个根节点下可能存在部分子树的根节点是异构相似；

3) 混合相似，指的是两个根节点存在同构或异构相似，两个根节点下可能部分子树的根节点存在同构或异构相似。

比如图 3(a)的关联树 T_D，我们对其中的关联子树进行解析后，组成的似树 T_{sin} 如图 3(b)所示。

结合引理 1、定义 6~10 可知在一对多关联实体数据集 D 中，对关联树中 T_D 中节点的实体解析问题可以转换为对关联树 T_D 进行搜索相似树 T_{sin} 的问题。

3. 基于树的一对多关联实体解析方法

本节介绍基于树的一对多关联实体识别方法(Tree-Based Relational Entity Resolution, TB-RER)，它借助关联实体间的关联关系实现了关联树模型。TB-RER 由两部分组成：关联树搜索算法和节点匹配算法。3.1 节介绍关联树搜索算法，3.2 节介绍节点匹配算法。

3.1. 关联树搜索算法

一对多关联实体解析不同于传统实体解析，对多类型关联数据进行实体解析。冗余匹配的情况也更

为复杂。在一对多关联子树中，我们将匹配可能出现的结果分为以下四种类型：

1) 子节点对相似，根节点对不相似(False Positive)。

2) 子节点对不相似，根节点对不相似(True Negative)。

3) 子节点对相似，根节点对相似(True Positive)。

4) 子节点对不相似，根节点对相似(True Negative)。

其中类型 1)、2)和 4)为一对多关联实体解析的冗余匹配。

我们提出一个基于深度优先原则的关联树搜索算法 $TMatch_{DPS}(T,\theta)$，该算法依据节点的相似性传递(引理 2)解决冗余匹配类型 1)和 2)。$TMatch(T,\theta)$ 包括三个主要功能模块：节点匹配模块、候选对搜索模块和相似树索引生成模块。涉及两个数据结构：一个相似树索引 T_{sim} 和一个候选集 Q。

算法 1： $TMatch_{DPS}(T,\theta)$

输入：关联树 T，阈值 θ

输出：相似树索引 T_{sim}

T_{sim}：空相似树索引

Q：空候选集

初始化

1 Q， T_{sim} 中加入 $\langle T.root, T.root\rangle$

2 当 Q 不为空：

3 for each $\langle i,j\rangle \in Q$

4 if $f_{sim-r}(i,j) > \theta$ then，

5 添加 $\langle i,j\rangle$ 到 T_{sim} 中

6 $Q \cup LookUp(\langle v,w\rangle)$ 其中 $p(v)=i, p(w)=j$

7 return T_{sim}

我们对候选集 Q 中关联子树对 T_i, T_j 的根节点对 $\langle i,j\rangle$ 进行相似节点匹配 f_{sim} (第 4 行)，节点匹配模块 f_{sim} 将在 3.2 节中详细说明，如果 $f_{sim}(i,j) \geq \theta$，即 $i \approx j$，则认为关联子树对 T_i, T_j 为相似子树，其子节点集 S_i, S_j 满足引理 2，包含的节点对的匹配类型为(3)或(4)，即关联子树对 T_i, T_j 为相似子树。将其子节点集 S_i, S_j 中的节点对 $\langle v,w\rangle$ 加入到候选集 Q 中，其中 $\langle v,w\rangle$ 满足引理 1，$v.level = w.level$ 。如果 $f_{sim}(i,j) < \theta$，即根节点对 $\langle i,j\rangle$ 不相似，其其子节点集 S_i, S_j 不满足引理 2，包含节点对的匹配类型只包含(1)或(2)，不具有匹配意义，阈值 θ 可以通过领域专家给出或者通过实验得到。

我们举例来简单说明这个过程：

以图 3(a)的关联树 T 中的关联子树 $T_{b_1}, T_{b_2}, T_{b_4}$ 为例，其中关联子树对 T_{b_2}, T_{b_4} 的根节点 b_2, b_4 不匹配，则其子节点集 $H_{b_2} = \{h_4, h_5\}$， $H_{b_4} = \{h_7, h_8\}$ 不满足引理 2，我们对关联子树对 T_{b_2}, T_{b_4} 进行剪枝，跳过匹配关联子树对 T_{b_2}, T_{b_4} 的子节点集 H_{b_2}, H_{b_4}。虽然 h_4 和 h_7 的户名都为 103 但 h_4 和 h_7 所关联的栋实体不同，h_4 为 3 号楼 103， h_7 为 2 号楼 103，不具备匹配意义，为冗余匹配。关联子树对 T_{b_1}, T_{b_4} 的栋节点 b_1, b_4 匹配，则其子节点集 $H_{b_1} = \{h_1, h_2, h_3\}$， $H_{b_4} = \{h_7, h_8\}$ 满足引理 2，继续对其子节点集的关联子树进行搜索。

若相似节点对 $\langle i,j\rangle$ 中节点 i, j 为非叶子节点，那么相似节点对 $\langle i,j\rangle$ 的子节点集 S_i, S_j 中，需要匹配的子节点对有 $|S_i| \cdot |S_j|$ 个，即我们需要执行 $|S_i| \cdot |S_j|$ 次节点匹配函数，这是不现实的。函数 *LookUp* (第 6 行)输入相似节点对 $\langle i,j\rangle$ 的子集 S_i, S_j ，输出节点对 $\langle i,j\rangle$ 子集 S_i, S_j 的相似关联子树候选对。我们使用 G. Papadakis 的分块技术[5]对节点对 $\langle i,j\rangle$ 的子集 S_i, S_j 中的对象分块，最后生成候选对，加入到候选集 Q 中。

我们的候选集 Q 是一个优先队列，用来维持待匹配节点对的解析顺序，Q 的每个节点存储一对匹配节点对和相应的优先级。将初始的根节点对输入到 *LookUp* 中，将返回的子节点候选匹配对依次插入到候选集 Q 中，并给予相同的、较低的优先级，节点每深入一层，对 *LookUp* 返回的候选匹配节点的优先级提高一级，以便关联树匹配算法能及时的解析出关联树的部分子树，提高解析效率。比如图 3(a)的关联树 T 中的为例，我们先解析候选集 Q 中的节点对 $\langle r,r\rangle$，优先级为 0，$\langle r,r\rangle$，必相似，我们将节点对 S_r 输入到函数 *LookUp* 中，将返回的子节点候选对 $\langle p_1,p_2\rangle,\langle p_2,p_3\rangle$，加入到候选集 Q 中，优先级为 I，依次匹配 $\langle p_1,p_2\rangle$ 和 $\langle p_2,p_3\rangle$，当 $\langle p_1,p_2\rangle$ 相似后，我们基于深度优先原则，将其子集 S_{p_1},S_{p_2} 输入到 *LookUp* 中，过滤生成候选节点对 $\langle b_1,b_4\rangle,\langle b_2,b_5\rangle,\langle b_3,b_6\rangle$ 等，加入到候选集 Q 中，优先级为 II，先匹配节点对 $\langle b_1,b_4\rangle$，该节点对相似，我们继续深入匹配其子节点集 S_{b_1},S_{b_4}，S_{b_1},S_{b_4} 输入到 *LookUp* 中过滤生成候选节点对 $\langle h_3,h_7\rangle$，$\langle h_3,h_7\rangle$ 为叶子节点，无子节点，不再深入，匹配其相邻节点，按优先级依次匹配直至候选队列为空。

为了对实体解析顺序的优化，我们在候选集对匹配(第 4 行)和候选对搜索(第 6 行)之间对实体解析顺序进行了优化。当节点对 $\langle i,j\rangle$ 成功匹配后，会搜索候选集中包含 i,j 的其他候选对并执行匹配操作。当搜索完所有与 i,j 相关的候选对后，我们再添加相似节点对至相似树索引中并对其子节点集进行候选对搜索。比如，当 $\langle p_1,p_2\rangle$ 匹配成功后，在搜索其子节点前先匹配 $\langle p_2,p_3\rangle$，若 $\langle p_2,p_3\rangle$ 也成功匹配，则将 $\langle p_1,p_2,p_3\rangle$ 整体作为相似节点索引插入到相似索引树中，作为根节点 $\langle r,r\rangle$ 的子节点，再去搜索 $\langle p_1,p_2,p_3\rangle$ 的子节点集的候选对。

相似索引树生成：若候选集 Q 中节点对 $\langle i,j\rangle$ 相似节点匹配(第 4 行)匹配成功后，我们对匹配成功的节点对进行相似树索引构建(第 5 行)，若关联树匹配基于深度优先原则遍历节点对进行匹配，则相似树也基于深度优先顺序进行构建。若关联树匹配基于广度优先原则遍历节点对进行匹配，则相似树也基于广度优先顺序进行构建。

3.2. 节点匹配算法

3.2.1. 传统的基于属性相似度计算

属性相似度计算

设 $R[A_1,A_2,\cdots,A_n]$ 和 $S[A'_1,A'_2,\cdots,A'_n]$ 是两个实体，具有 n 个对齐属性集合 A_i 和 A'_i，$i\in[1,n]$，我们假定两个实体之间的属性已对齐并作为输入提供。当然，我们的方法也适用于同一实体。让 r,s 为 R,S 中的记录，并且 $r[A_i]$ 和 $s[A'_i]$ 是属性 A_i 和 A' 的值。

相似函数 $f(r[A_i],s[A'_i])$ 计算相似度分数，分数的取值范围为[0,1]，比如编辑距离和 Jaccard 相似度。分数越接近 1 越意味着 $r[A_i]$ 和 $s[A'_i]$ 具有更高的相似性。

属性匹配规则是一个三元组 $\approx(i,f,\theta)$ 表示为 $f(r[A_i],s[A'_i])\geq\theta$ 的布尔值，其中 $i\in[1,n]$ 是一个属性索引，f 是一个相似度函数，$\theta\in[0,1]$ 是相似度阈值。属性匹配规则 $\approx(i,f,\theta)$ 返回为 true 意味着 $r[A_i]$ 与 $s[A'_i]$ 相对于特定相似函数 f 和阈值 θ 匹配。

我们将 $r[A_i]\approx_{(f,\theta)}s[A'_i]$ 作为相似函数 f 和阈值 θ 的属性匹配规则。在下文中，我们将其简写成 $r[A_i]\approx s[A'_i]$。

记录相似度计算：

记录相似度计算是一组对不同属性的属性相似度计算的结合。我们通常根据属性的重要程度不同为不同的属性 A_i 分配不同的权重 λ_i，以综合的角度判断记录是否匹配，为了减少其求解的搜索空间，记录相似度计算时，通常将每个属性都采用相同的相似度算法和阈值来进行比较。

当节点为同构节点时，节点与节点同为一类实体，属性一致。当节点为异构节点时，节点与节点为不同实体，属性不一致。将属性对齐后为：

$$f_{sim}(r,s)=\sum_{i=1}^{n}\left(\lambda_i f_{A_i}\left(r[A_i],s[A_i']\right)\right) \tag{1}$$

其中 A_i 与 A_i' 为对齐后的属性。

3.2.2. 基于关联属性的相似度算法

在我们进行传统的属性相似度方法进行相似度计算时，仅计算当前节点属性的相似度判断该节点是否相似，但在关联数据集中，多个实体集之间的具有关联关系，其关联关系由外键属性构成，如图 1(a)所示，栋实体集具有盘实体集的共同属性，盘名；户实体集具有栋实体集的共同属性，栋名；在关联实体集中，不仅有当前实体集的属性可以判断是否为同一实体，其关联实体的部分属性也可以判断其是否为同一实体。因此我们将本节点的属性相似度和关联子节点的部分属性相似度结合起来判断当前节点是否匹配。

定义一个关联属性相似度算法 f_{sim-r}，其中 $p(s_i)=r_i, p(s_j)=r_j$：

$$f_{sim-r}(r_i,r_j)=\delta f_{sim}(r_i,r_j)+(1-\delta) f_{sim-related}(s_i,s_j) \tag{2}$$

$$f_{sim-related}=\sum_{j=1}^{k}\left(\lambda_j f_{A_j}\left(s_i[A_j],s_j[A_j']\right)\right) \tag{3}$$

其中，f_{sim} 是节点 r_i,r_j 自身的基于属性的相似度，采用已有的基于属性的相似度算法[18]，$f_{sim-related}$ 是节点 r_i,r_j 的关联节点部分属性的相似度，δ 是自身属性相似度和关联节点属性相似度的权值分配系数，可由用户根据关联节点属性相似度的重要性进行设定。$f_{sim-related}$ 采用关联属性进行相似度匹配。K 为关联节点属性选取个数，由领域专家给出或者通过实验得到。

算法 2：$f_{sim-r}(e_1,e_2,\delta)$

输入：节点 v,w ,权值 δ

输出：节点相似度 sim

1 初始化 sim 为 0

2 foreach $s.A_i\in S, i\in[0,k]$

3 foreach $s_{rel,1}\in S_1, s_{rel,2}\in S_2$

4 $sim=sim+\delta f_{sim-related}(s_{rel,1}.A_i, s_{rel,2}.A_i)$

4 foreach $e.A_i\in E, i\in[1,n]$

5 $sim=sim+(1-\delta) f_{sim}(e_1.A_i, e_2.A_i)$

6 return sim

算法 2 描述了基于关联属性的相似度算法，该算法输入节点 v,w 和权值 δ，通过计算关联节点的部分属性和自身属性的相似度来计算节点 v,w 的综合节点相似度 sim，并将综合相似度 sim 作为输出返回，权值 δ 用于判断关联节点属性相似度与自身节点属性相似度的比重。我们循环搜索节点 v,w 的关联节点(第 2 行)并计算关联节点的部分属性的关联相似度 $f_{sim-related}$ (第 3 行)，遍历 v,w 的属性(第 4 行)，计算节点 v,w 的节点相似度 f_{sim} (第 5 行)，最终返回综合相似度 sim。

4. 实验与分析

4.1. 实验设置

本文实验采用真实数据来对算法性能进行评测。其中真实数据采用房地产数据集，其数据来源于“面向房地产领域的 Web 数据抽取”项目所抽取的多个数据源的 5 个不同时段的房地产关联数据信息，共含

有约 80 万个记录，每个时段的房地产数据集中都包含盘记录表、栋记录表、户信息表。其中盘记录表中约含有 3 千个记录，包含地址、项目名称、容积率等 60 余种属性。栋记录表中约含有 2 万个记录，包含栋名称、楼型等 30 余种属性。户信息表中约含有 15 万个记录，包含门牌号、楼层、建筑面积等四十余种属性。我们分别将这五个房地产数据集{d1, d2, d3, d4, d5}分为五个不同大小数据集 D1，D2，D3，D4，D5，D1 = {d1}，D2 = {d1, d2}……D5 = {d1, d2, ..., d5}。我们将使用数据集 D1 用于对比实验，再从 D1 逐渐增加到 D5 用作可扩展评估。

本文实验代码采用 Java 编写，运行在配置 3.7 GHz AMD Ryzen7 2700X 处理器，16 GB 内存的 64 位 Windows10 操作系统中。

我们使用 F1-Score 来比较其性能，F1-Score 是准确率和召回率的调和均值，其中准确率是所有匹配实体中正确匹配的实体所占比例；召回率是所有应该匹配的实体中正确匹配的实例所占比例；我们使用算法运行时间来测试该算法的可扩展性。

4.2. 方法比较

本文采用基于关联属性的相似度算法来进行节点匹配，该算法基于属性相似度，为了减小其求解的搜索空间，我们对每个属性都采用相同的相似度算法和阈值来进行比较。对于不同的相似度算法，我们经实验选择 Jaccard 相似度算法来进行实验，经实验测得，我们将 3 个关联栋实体的属性加入盘实体的匹配中；将 2 个关联户实体的属性加入栋实体的匹配中，根据实验测得相似度阈值 θ 为 0.80，经实验测得权值分配系数 δ 为 0.83。

利用房地产数据集，采用上述有效性评测标准，我们将本文提出的关联树搜索算法、传统属性匹配算法和 DepGraph 算法进行性能比较。如图 4 所示，分别表示数据集总记录条数在 18 万，盘实体集 3000 条，栋实体集 2 万条，户实体集 16 万条时，两算法在房地产数据集中的 3 类实体集的准确性上，仅在盘实体集上与传统方法持平，比 DepGraph 算法略低，随着关联树的加深，栋实体集和户实体集的解析准确性也优于传统方法和 DepGraph 算法。

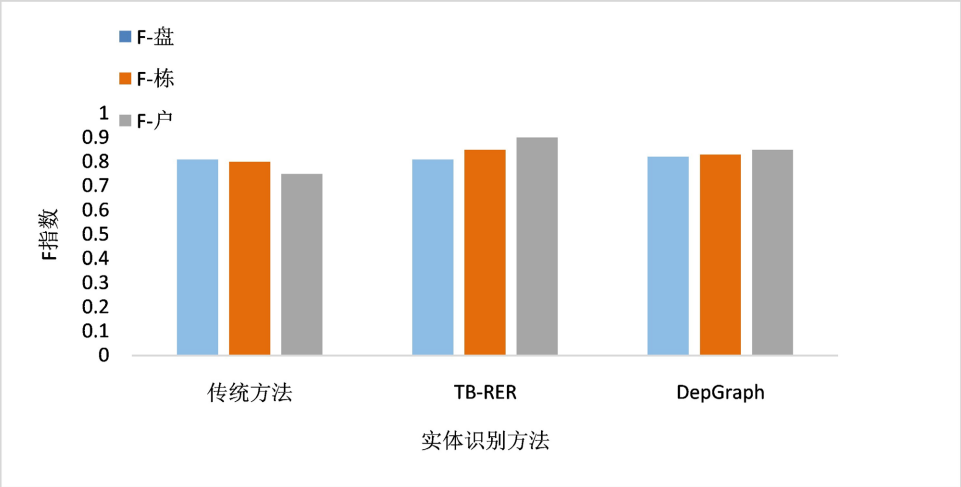


Figure 4. Comparison of TB-RER method and existing methods
图 4. TB-RER 方法与已有方法对比

4.3. 可扩展评估

为了测试关联树搜索算法的可扩展性，将 TB-RER 算法分别在规模 D1 逐渐增加到 D5 的房地产数据

集上进行实验，相邻数据集规模相差大约 20 w。观察图 5，TB-RER 随着数据集规模的增长，其时间开销的增长接近于线性，这得益于关联树模型的相似度传递，使树的匹配层数加深，所需要匹配的节点集远远少于其层的总节点集，并使用分块技术生成候选匹配对，引导实体解析在数据集减少的情况下又以高效的匹配顺序进行，避免了平方级的增长。

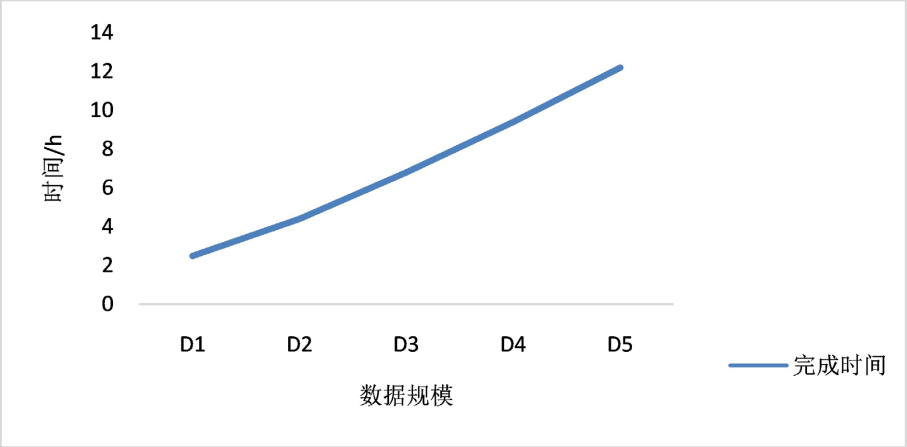


Figure 5. TB-RER scalable evaluation
图 5. TB-RER 可扩展评估

5. 结论

本文提出了一种基于关联树模型的关联实体解析方法。依据一对多关联实体的相似性传递性质，随着关联树层数加深，实体集的实体数也随之增大，但关联树中搜索的节点的量并没有随之增多，以此提高实体识别的效率。关联树匹配算法可以更快速、准确地查找潜在相似关联实体对；节点匹配算法可以更加精确地匹配相似节点对。实验结果验证了基于关联树模型的关联实体解析方法的有效性，随着树层数加深，效率逐渐远超传统方法，并且可以和现在主流的方法相媲美，甚至更好。

6. 结束语

实体解析是数据挖掘和数据集成的重要组成部分。大数据时代，数据呈现多样性，关联性以及时效性。如何对多类型的、不同时间抓取的、一对多关联关系的对象进行高效、准确的关联实体解析成为一个重要问题。针对已有的关联实体解析方法的不足，本文提出一种基于树模型的关联实体解析方法——TB-RER，该方法适合于一对多关联关系的数据。下一步工作将研究如何对一对多关联实体的解析结果进行实体生成。

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] Sun, Y. and Han, J. (2012) 2013 Mining Heterogeneous Information Networks: A Structural Analysis Approach. *ACM SIGKDD Explorations Newsletter*, **14**, 20-28. https://doi.org/10.1145/2481244.2481248

[3] Benjelloun, O., Garcia-Molina, H., Menestrina, D., Su, Q., Whang, S.E. and Widom, J. (2009) Swoosh: A Generic Approach to Entity Resolution. *The VLDB Journal*, **18**, 255-276. https://doi.org/10.1007/s00778-008-0098-x

[4] Efthymiou, V., Stefanidis, K. and Christophides, V. (2015) Big Data entity Resolution: From Highly to Somehow Similar Entity Descriptions in the Web. 2015 *IEEE International Conference on Big Data* (*Big Data*), Santa Clara, 29 October-1 November2015, 401-410. https://doi.org/10.1109/BigData.2015.7363781

[5] Papadakis, G., Ioannou, E., Niederée, C. and Fankhauser, P. (2011) Efficient Entity Resolution for Large Heterogeneous InformationSpaces. *Proceedings of the* 4*th ACM International Conference on Web Search and Data Mining* (*WSDM*), Hong Kong (China), 9-12 February 2011, 535-544. https://doi.org/10.1145/1935826.1935903

[6] Christen, P. (2012) A Survey of Indexing Techniques for Scalable Record Linkage and Deduplication. *IEEE Transactions on Knowledge and Data Engineering*, **24**, 1537-1555. https://doi.org/10.1109/TKDE.2011.127

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

[8] Galhardas, H., Florescu, D., Shasha, D., Simon, E. and Saita, C. (2001) Declarative Data Cleaning: Language, Model and Algorithms. *Proceedings of* 27*th International Conference on Very Large Data Bases*, Rome, 11-14 September 2001, 371-380.

[9] Kenig, B. and Gal, A. (2013) Mfiblocks: An Effective Blocking Algorithm for Entityresolution. *Information Systems*, **38**, 908-926. https://doi.org/10.1016/j.is.2012.11.008

[10] Papadakis, G., Ioannou, E., Palpanas, T., Niederée, C. and Nejdl, W. (2013) A Blocking Framework for Entity Resolution in Highly heterogeneous Information Spaces. *IEEE Transactions on Knowledge and Data Engineering*, **25**, 2665-2682. https://doi.org/10.1109/TKDE.2012.150

[11] Bhattacharya, I. and Getoor, L. (2007) Collective Entity Resolution in Relational Data. *ACM Transactions on Knowledge Discovery from Data*, 1, 5-es. https://doi.org/10.1145/1217299.1217304

[12] Arasu, A., Ré, C. and Suciu, D. (2009) Large-Scale Deduplication with Constraints Using Dedupalog. 2009 *IEEE* 25*th International Conference on Data Engineering*, Shanghai, 29 March-2 April 2009, 952-963. https://doi.org/10.1109/ICDE.2009.43

[13] Culotta, A. and Mccallum, A. (2005) A Conditional Model of Deduplication for Multi-Type Relational Data. Technical Report, University of Massachusetts, Amherst.

[14] Dong, X., Halevy, A.Y. and Madhavan, J. (2005) Reference Reconciliation in Complex Information Spaces. *Proceedings of the* 2005 *ACM SIGMOD International Conference on Management of Data*, Baltimore, 14-16 June 2005, 85-96. https://doi.org/10.1145/1066157.1066168

[15] Singla, P. and Domingos, P. (2004) Multi-Relational Record Linkage. *The KDD*-2004 *Workshop on Multi-Relational Data Mining*, Seattle, WA, August 2004, 31-48.

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

[17] Whang, S.E. and Garcia-Molina, H. (2012) Joint Entity Resolution. 2012 *IEEE* 28*th International Conference on Data Engineering*, Arlington, 1-5 April 2012, 294-305. https://doi.org/10.1109/ICDE.2012.119

[18] 褚良旭, 李贵, 李征宇, 韩子扬, 曹科研. 一种基于属性显著度的实体解析算法[J]. 数据挖掘, 2021, 11(2): 27-37. https://doi.org/10.12677/HJDM.2021.112004