Hans Journal of Data Mining 数据挖掘, 2020, 10(1), 1-15
Published Online January 2020 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2020.101001

Hans汉斯

Semi-Structured Entity Resolution Algorithm

Hailang Wei, Gui Li, Zhengyu Li, Ziyang Han, Keyan Cao

School of Information and Control Engineering, Shenyang Jianzhu University, Shenyang Liaoning
Email: 357488751@qq.com

Received: Nov. 21st, 2019; accepted: Dec. 4th, 2019; published: Dec. 11th, 2019

Abstract

Entity resolution is the identification of similar or identical records in one or more datasets. In this paper, an entity resolution algorithm based on string similarity is proposed for semi-structured data with unknown patterns. The records are divided into several substrings, and the correlation between substrings is calculated by editing similarity. On this basis, the maximum weighted matching algorithm of binary graph is introduced to measure the correlation between records. Due to the computing time complexity of this method is higher, for Web entity resolution large data sets, the time cost is larger; therefore, this article also puts forward a kind of entity resolution algorithm based on set similarity, considering record as a collection of all the property values, each attribute value as the elements in the collection, using an array of tag to represent each element, according to these tags array for each record to create a signature, to find other similar records match the signature. The optimized maximum matching algorithm is used to select the truly similar records. Finally, this paper uses the actual data set to verify that the above method is more effective than the traditional method.

Keywords

Entity Resolution, Edit Similarity, Set Similarity, Maximum Weighted Matching of Binary Graph

半结构化实体解析算法

韦海浪，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: 357488751@qq.com

收稿日期：2019年11月21日；录用日期：2019年12月4日；发布日期：2019年12月11日

摘　要

实体解析是指识别一个或多个数据集中的相似或相同的记录。该文主要针对模式未知的半结构化数据，

文章引用: 韦海浪, 李贵, 李征宇, 韩子扬, 曹科研. 半结构化实体解析算法[J]. 数据挖掘, 2020, 10(1): 1-15.
DOI: 10.12677/hjdm.2020.101001

提出了一种基于字符串相似度的实体解析算法，将记录分成多个子字符串，采用编辑相似度计算子字符串之间关联度，在此基础上引入二分图最大加权匹配算法度量记录之间的关联度。由于该方法的计算时间复杂度比较高，对于Web大数据集实体解析来说，所需的时间成本较大，因此，该文还提出了一种基于集合相似度的实体解析算法，将记录看作所有属性值的集合，每个属性值作为集合中的元素，用一个标记数组来表示每个元素，根据这些标记数组为每个记录创建一个签名，找出与签名相匹配的其他相似记录。并且采用优化后的最大匹配算法来选出真正相似的记录。最后，该文采用实际数据集进行实验验证了上述方法比传统方法更有效。

关键词

实体解析，编辑相似度，集合相似度，二分图最大加权匹配

1. 引言

随着互联网技术的发展和应用，数据量越来越大，越来越多的数据分析师需要将不同 Web 数据源中指向现实世界同一实体的记录识别出来。然而这些数据往往是以模式未知的半结构化数据形式存储的。这些数据虽然存在一定的结构，但是结构不完全一致，往往存在着数据位移，属性缺失、属性值交错存储等问题，因此对数据集成造成了很大的困难。如表 1 所示的模式未知的两条记录，没有一个属性是完全匹配的，但实际上指的是同一个实体。

Table 1. Training data set
表 1. 训练数据集

记录	地址 1	地址 2	地址 3
r_1	华北路东、西南路北 AB 区、东规划路、西华北路、南西南路、北规划路	121.57、38.94	沙河口区春风街 1 号、西南路 905-1 号~905-7 号、春风街 5-1 号
r_2	沙河口区春风街 1 号、西南路北侧 A、B 区、西南路 905-1 号	华北路东、东至规划路、春风街 5-1 号、西南路 905-7 号	121.575、39.945、南西南路、西华北路、北规划路

为了解决这个问题。本文提出下面的方法。首先，引入并改进了基于字符串相似度识别相似记录的算法。将每个记录的所有属性值作为一个长字符串，采用分隔符分词的方法将字符串分为多个子字符串，再对每个子字符串进行相似度计算，利用二分图最大加权匹配[1]算法计算出相似记录。由于在 Web 大数据集中该算法的最坏的运行时间复杂度是 $O(n^3m^2)$，其中 n 是子字符串的数量，m 是记录的数量[1]。因此，本文还引入并改进了基于集合相关性识别相关记录的算法来降低 m 的数量。将记录看作所有属性值的集合，每个属性值作为集合中的元素，用一个标记数组来表示每个元素，根据这些标记数组为每个记录创建一个签名，找出与签名相匹配的其他相似记录。最后采用基于三角不等式优化了的最大二分图匹配的方法来选出真正相似的记录。这里将两个集合作为二分图的两边(每个元素假设为一个顶点)，用相似性函数来测量元素之间的相似度。

本文的主要贡献如下：

1) 提出一种不考虑模式、类型的记录相似度定义，能够有效的表达半结构化记录的相似性。

2) 在本文定义的字符串相似度基础上，引入最大二分图加权匹配算法(Kuhn-Munkres 算法)有效计算记录之间的相关性。

3) 为了减少候选记录之间的比较次数，引入并改进集合相似性，为记录创建有效签名空间，以及生成最优签名的步骤，将签名和倒排索引结合，对相关记录进行初步选择。

4) 为了降低最大匹配算法的时间复杂度，采用两种新的过滤器，减少不相关的候选集，同时采用三角不等式来优化最大匹配方法。

论文的其余章节组织如下。本文在第 2 节介绍了目前阶段对于半结构化实体解析的一些相关工作，第 3 节介绍了集合相关性和字符串相似度的相关理论，在第 4 节和第 5 节给出了基于字符串相似度和集合相似度识别相关记录的模型，提出相应的算法，最后在第 6 节对算法进行实验验证。第 7 节进行了总结。

2. 相关工作

实体解析又被称为记录链接、对象识别、重复检测等。经过近几年的发展，有一系列实体解析技术问世；文献[2]是对早期实体解析工作的综述；文献[3] [4] [5]是国内对实体解析技术的研究综述；文献[6]用记录的相关字段构成键，按照键对记录进行排序，然后按顺序对记录进行相似度检测。文献[7]把每条记录看成字符串，对记录进行排序，然后使用固定大小的优先队列按顺序扫描已排序的记录，并将它们聚类；文献[8]采用 N-gram 方法对记录进行聚类；文献[9]将每个字段值分解成多个 tokens，对 tokens 进行排序，进而对记录进行排序，使用滑动窗口的方法来比较临近范围的记录。文献[10]提出了一个近似的 Jaccard 相似性度量来评估两个字符串之间的相似性。文献[1] [11]提出了用于集合相似性搜索和连接的近似算法。文献[12]提出了使用前缀过滤器来解决集合相似性连接问题。这些工作都未考虑到这种模式未知的半结构化数据，不适用于处理 Web 数据源大数据集中的实体解析。

3. 相关理论

3.1. 字符串相似度

本文将每条记录的所有属性值作为一个长字符串，用标点符号作为分隔符分词法将字符串分为多个子字符串，这些子字符串之间的相关性用编辑相似度计算。编辑相似度公式为：

$$NEds(x,y)=1-\frac{ED(x,y)}{\max(|x|,|y|)}$$

其中 $|x|$ 和 $|y|$ 为 x 和 y 的长度。$ED(x,y)$ 表示字符串 x 和 y 之间的编辑距离(Edit Distance)。字符串编辑距离是建立在一系列编辑操作(字符串中单个字符的插入、删除和替换)的基础之上的，通常每个编辑操作会被分配一个代价来表示其发生的可能性。给定一个编辑操作的集合和它们的代价，两个字符串 x 和 y 之间的编辑距离 $ED(x,y)$ 被定义为：把字符串 x 转换成字符串 y 所需要采取的编辑操作序列的最小代价。下面给出编辑操作，编辑操作的代价和编辑距离的形式化定义：

定义 1：令 A 是字符的有限集合，A^*是 A 上所有串的集合，ε 表示空字符，$|\varepsilon|=0$，$|x|$ 表示字符串 x 的长度。

编辑操作指：

替换：$a\to b$ (如果 $a=b$，则被称为一致替换，否则为非一致替换)；

删除：$a\to\varepsilon$；

插入：$\varepsilon\to a$；

定义 2：代价函数指的是每个编辑操作指定一个[0，1]之间的非负实数为代价的函数。令 $C(a \to b)$，$C(a \to \varepsilon)$ 和 $C(\varepsilon \to a)$，分别表示替换，删除和插入操作的代价，对于一个编辑操作的序列 $S = e_1, e_2, \cdots, e_k$，其代价被定义为公式：

$$C(S) = \sum_{i=1}^{k} C(e_i)$$

其中规定任何一致替换的代价都是 0。

定义 3：两个字符串 x 和 y 之间的编辑距离 $ED(x, y)$ 被定义为：把字符串 x 转换成字符串 y 所需要采取的编辑操作序列的最小代价。即：

$$ED(x, y) = \min\{C(S)\}$$

在确定这些子字符串之间的相似度后，为了描述记录之间的相关性，结合上述子字符串之间的编辑相似度，引入了第 4.2 节中的二分图最大加权匹配算法[1]，求出记录之间的最大加权匹配分数作为记录之间的相似度 $Sim(r_i, r_j)$。

3.2. 集合相关性

首先将两个集合 R 和 S 的相关性概念形式化。在这里主要采用的是 Jaccard 相似度函数来识别相似度较高的元素对。假设用相似度函数 $\Phi(x, y)$ 来表示两个元素 x 和 y 之间的 Jaccard 相似度，取值范围在[0，1]之间，1 代表完全相似，0 代表完全不同。为此提供了相似度阈值 α，相似函数 Φ 的定义如下：

$$\Phi_\alpha(x, y) = \begin{cases} \Phi(x, y) & \text{if } \Phi(x, y) \geq \alpha \\ 0 & \text{if } \Phi(x, y) \leq \alpha \end{cases}$$

从上述定义中可以看出，相似度大于或等于阈值 α 的元素视为匹配元素。

给出一个相似函数 Φ 为两个集合 R 和 S 构建加权二分图。图中每个顶点代表 R 或 S 中的元素，使用 Φ 为连接顶点的边加权。然后求出该图的最大加权二分图匹配[1]，其中一个数据集中的每个顶点恰好与另一个数据集中的一个顶点相连接，使边的权值之和达到最大值。最大匹配的分数就是边权重之和。本文用 $\left|R \tilde{\cap}_{\Phi_\alpha} S\right|$ [13] [14]表示最大匹配分数。定义 4 给出了基于最大匹配分数的集合相关性定义：

定义 4：假设两个集合 R 和 S 中含有相似的元素则称两个集合相关，用 $related_{\varnothing}$ 来表示，相似函数为 $\varnothing_\alpha$。将集合相关性分为下面两种情况：

1) SET-SIMILARITY：检查两个集合是否相似。则 SET-SIMILARITY 计算公式为：

$$related_\Phi(R, S) = similar_{\Phi_\alpha}(R, S) = \frac{\left|R \tilde{\cap}_{\Phi_\alpha} S\right|}{|R| + |S| - \left|R \tilde{\cap}_{\Phi_\alpha} S\right|}$$

2) SET-CONTAINMENT：检查一个集合是否是另一个集合的子集。其中 $|R| \leq |S|$，则 SET-CONTAINMENT 的计算公式为：

$$related_\Phi(R, S) = contain_{\Phi_\alpha}(R, S) = \frac{\left|R \tilde{\cap}_{\Phi_\alpha} S\right|}{|R|}$$

根据用户需求取阈值 δ，当且仅当 $related_\Phi(R, S) \geq \delta$ 时，两个集合就被称为相关的。

4. 基于字符串相似度的实体解析算法

4.1. 数据分词

由于半结构化数据常常表现为模式未知或属性值为文本型字符串的形式存储(表 1)，很难直接判断出

记录之间的关联性。因此首先指定以下分词规则对数据进行处理：

将半结构化数据集中的每条记录的所有属性值作为一个长字符串(如表 2)，并采用分隔符将每个记录分隔成多个子字符串。例如将表 2 中数据采用顿号分隔符分为表 3 的多个子字符串。

Table 2. Training data set string representation
表 2. 训练数据集字符串表示

r_1	沙河口区春风街 1 号、西南路 905-1 号~905-7 号、春风街 5-1 号、华至北路东、西南路北 A B 区、东至规划路、西至华北路、南至西南路、北至规划路、121、39
r_2	春风街 1 号、西南路北侧、西南路 905-1 号、华北路东、东至规划路、春风街 5-1 号、西南路 905-7 号、121.575、39.945、南西南路、西华北路、北规划路

Table 3. Data segmentation results
表 3. 数据分词结果

r_1	r_2
沙河口区春风街 1 号	春风街 1 号
西南路 905-1 号~905-7 号	西南路北侧
春风街 5-1 号	西南路 905-1 号
华北路东	华北路东侧
西南路北 A、 B 区	东规划路
东至规划路	春风街 5-1 号
西至华北路	西南路 905-7 号
南至西南路	121.575、39.945
北至规划路	南西南路
121、39	西华北路
	北规划路

4.2. 记录相关性计算

为了衡量记录之间的相关性，首先结合 4.1 节中的分词结果，计算出这些子字符串之间的编辑相似度。根据基于子字符串的相似度来定义记录的相似度。提出了一种基于二分图最优匹配的相似度定义，方法如下：

根据待比较的两条记录 r_i 和 r_j 生成一个完全有权二分图 $B=\{V_1 \bigcup V_2, E, W\}$，其中 V_1 中每个节点 u 代表 r_i 的一个子字符串 u_i，V_2 中的每个节点 v 代表 r_j 的一个子字符串 v_j，$E=V_1 \times V_2$，权函数 W 定义为 3.1 节中两个子字符串之间的相似度 $W(u,v)=Sim(u_i,v_j)$。$M=(V',E')$ 是 B 的最优匹配，其中 $V' \subseteq V_1 \bigcup V_2$，$E' \subset E$。则记录 r_i 和 r_j 的相似度定义为：$Sim(r_i,r_j)=\frac{\sum_{e \in E'} W(e)}{\max\{m,n\}}$ 其中 m、n 分别代表 r_i 和 r_j 的子字符串个数。

本文采用 Kuhn-Munkres 算法(KM 算法)来计算记录的相似度，下面给出具体的算法流程：

1) 由算法 1 构造两条记录的完全二分图；

2) 算法 2 计算两条记录相似性，其中 $similarity(r_1[i], r_2[j])$ 返回 r_1 的第 i 个字符串和 r_2 的第 j 个字符

串之间的相似性， $optimal\_match(B,w,m)$ 指的是二分图的最优匹配，返回最优匹配中所有权值之和，其中 B 表示完全二分图，w 是 B 的权值矩阵，m 是 w 的阶；

3) 根据用户需求给定阈值 δ，用算法3将记录之间相似度大于该阈值的记录选择出来作为相似记录对。

算法 1：从记录中抽取带权的完全二分图

输入：两条记录 r_1，r_2

输出：带权完全二分图

1) $m \leftarrow r_1$ 的子字符串个数

2) $n \leftarrow r_2$ 的子字符串个数

3) *Foreach i,j do*

4) $w[i,j] \leftarrow 0$ /*W 是二分图的权值矩阵*/

5) *Foreach i,j do*

6) $w[i,j] \leftarrow similarity(r_1[i], r_2[j])$

7) $X \leftarrow r_1$ 的所有子字符串， $Y \leftarrow r_2$ 的所有子字符串

8) *If m < n then* 向 X 中加入 n-m 个点

9) *Else* 向 Y 中加入 m-n 个点

10) $E \leftarrow X$ 中的每个点到 Y 中所有点均有边

11) $B \leftarrow (X \cup Y, E)$

12) $record\_to\_bipartite(r_1,r_2) \leftarrow (w, B, max\{m,n\})$

算法 2：计算记录相似度

输入： $record\_to\_bipartite(r_1,r_2)$

输出： $Sim(r_1,r_2)$

1) $(w,B,m) \leftarrow record\_to\_bipartite(r_1,r_2)$

2) $sum \leftarrow optimal\_match(B,w,m)$

3) $Sim(r_1,r_2) \leftarrow sum/m$

4) return $Sim(r_1,r_2)$

算法 3：划分相似记录

输入：所有记录之间的相似度 $Sim(r_i,r_j)$、阈值 δ

输出：最终的相似记录对集合 C

1) $C \leftarrow \varnothing$ /*初始化相似记录对集合*/

2) foreach $Sim(r_i,r_j)$ do

3) if $Sim(r_i,r_j) \geq \delta$

4) return $C \leftarrow (r_i,r_j)$

5) else delete (r_i,r_j)

6) return C

5. 基于集合相关性的实体解析算法

第 4 节描述的方法虽然能快速的找到相似记录，但是其算法时间复杂度为 $O(n^3m^2)$，其中 n 是子字符串的数量，m 是记录的数量。本节在第 4 节算法的基础上进行了改进，减少 m 的数量。

5.1. 属性值规范化标记分词及创建倒排索引表

首先将这些字段分为字符型和数值型并对其标记化分词处理：

数值型字段：例如表示地图坐标的字段(121.575509, 38.945373)由于不同数据源中表示的方式不同，在这里，统一将数值型字段精确到小数点后两位。

字符型字段：采用基于领域词典(基于城市，地区等划分的词典)的分词法，例如将“东至规划路”和“东规划路”统一规范化为“规划路”。

Table 4. Semi-structured tag array collection representation
表 4. 半结构化标记数组集合表示

R
r_1: (t_1 = 西南路， t_5 = A、B区， t_6 =沙河口区， t_4 =春风街， t_7 = 1号， t_8 = 905-1号) r_2: (t_1 = 西南路， t_2 = 华北路， t_3 = 规划路， t_4 = 春风街， t_9 = 5-1号， t_{10} = 905-7号) r_3: (t_1 = 西南路， t_2 = 华北路， t_3 = 规划路， t_{11} = 121.57， t_{12} = 38.94)

S_1	S_2	S_3	S_4
$s_1^1:(t_2\ t_5\ t_3\ t_7\ t_4)$	$s_1^2:(t_1\ t_7\ t_8)$	$s_1^3:(t_1\ t_5\ t_6\ t_7\ t_8)$	$s_1^4:(t_1\ t_5\ t_6\ t_8)$
$s_2^1:(t_1\ t_6\ t_2\ t_3\ t_4)$	$s_2^2:(t_1\ t_2\ t_3\ t_7\ t_4)$	$s_2^3:(t_5\ t_6\ t_{11}\ t_{12})$	$s_2^4:(t_2\ t_3\ t_4\ t_9\ t_{10})$
$s_3^1:(t_1\ t_6\ t_3\ t_2\ t_4)$	$s_3^2:(t_1\ t_5\ t_2\ t_4\ t_9)$	$s_3^3:(t_1\ t_5\ t_6\ t_3)$	$s_3^4:(t_1\ t_2\ t_3\ t_7\ t_9)$

表 4 中，R 是一个参考集，对应于一个实体记录，S 是用于搜索相关集合的记录集合。将表 1 中的记录划分为表 4 中记录 R 的 12 个字段，在这里用符号 t_i 代替标记，按照这些分词频率递减的顺序作为标记化后的下标。此外将记录中的每个文本属性都看作一个元素。每个元素都用一组标记(例如 $r_1=\{t_1,t_5,t_6,t_4,t_7,t_8\}$，$t_1$ = 西南路，t_5 = AB 区，t_6 = 沙河口区，t_4 = 春风街，t_7 = 1 号，t_8 = 905-1 号)来表示。对于 S 数据集，使用上述生成的标记为该数据集按照标记在各个元素中出现的频率创建一个倒排索引表。

5.2. 有效签名生成

本节将签名和倒排索引一起使用来识别相关的集合对。该算法从 5.1 节中生成的标记集合中找到一个子集来生成一个有效签名。通过这个签名来寻找相关的候选集，从而删除不相关的候选集。

5.2.1. 有效签名

在集合 R 中，假设每个元素 $r \in R$ 都是一个标记数组。则定义所有标记的集合为 $R^T = \bigcup_{r\in R} r$。根据标记集合，定义一个签名如下：

定义 5 (签名)：给定一个集合 R，则标记集合 R^T 的任何子集都是 R 的签名。

定义 6 (有效签名)：给定一个集合 R 和一个相关性阈值 δ，如果任何可能匹配的集合 S 满足 $related_{\varnothing}(R,S)\geq\delta$，同时在 R^T 中有一个签名 K_R^T，使 $S^T\cap K_R^T\neq\varnothing$，那么该签名 K_R^T 为有效签名。

如果给定集合 $R=\{r_1,r_2,\cdots,r_n\}$ 的一个签名 K_R^T，那么将有效签名定义为 $K_R=\{k_1,k_2,\cdots,k_n\}$，其中 $k_i=r_i\cap K_R^T$ (即，k_i 是 r_i 的一组签名标记)。

例 1：对于表 4 中的集合 R，有 $R^T=r_1\cup r_2\cup r_3=\{t_1,t_2,t_3,\cdots,t_{12}\}$，假设子集 $K_R^T=\{t_1,t_2,t_3,t_4\}$ 是集合 R 的一个可能的签名。相应的有效签名就是 $K_R=\{\{t_1,t_4\},\{t_1,t_2,t_3,t_4\},\{t_1,t_2,t_3\}\}$。

5.2.2. 加权签名方案

最大匹配阈值：在处理签名时，定义一个相关量 θ，作为最大匹配阈值。最大匹配阈值是基于 $similar_{\varnothing}(R,S)$ 和 $contain_{\varnothing}(R,S)$ 定义的。

对于 $contain_{\varnothing}(R,S)$，要判断 R 和 S 的关联性则需要满足：

$$contain_{\varnothing}(R,S)=\frac{\left|R\tilde{\cap}_{\varnothing}S\right|}{|R|}\geq\delta$$

即 $\left|R\tilde{\cap}_{\varnothing}S\right|\geq\delta|R|$。因此本文为 $contain_{\varnothing}$ 定义了最大匹配阈值 $\theta=\delta|R|$。

对于 $contain_{\varnothing}(R,S)$ 要想判断 R 与 S 相关联则需要满足公式：

$$similar_{\varnothing}(R,S)=\frac{\left|R\tilde{\cap}_{\varnothing}S\right|}{|R|+|S|-\left|R\tilde{\cap}_{\varnothing}S\right|}\geq\delta$$

鉴于 $|S|\geq\left|R\tilde{\cap}_{\varnothing_\alpha}S\right|$，则有：

$$\delta\leq\frac{\left|R\tilde{\cap}_{\varnothing}S\right|}{|R|+|S|-\left|R\tilde{\cap}_{\varnothing}S\right|}\leq\frac{\left|R\tilde{\cap}_{\varnothing}S\right|}{|R|+|S|-|S|}=\frac{\left|R\tilde{\cap}_{\varnothing}S\right|}{|R|}$$

所以对于 $similar_{\varnothing}$ 来说最大匹配分数也是 $\theta=\delta|R|$。

加权签名方案：加权签名方案是在未加权签名方案[13]中的所有签名基础上，为集合创建一个相似度上限更严格的签名。下面给出该签名方案的方法：

给出 r 中任何元素与 s 之间的 jaccard 相似度计算公式为：$\frac{|r\cap s|}{|r\cup s|}$，对于任何元素 r，假设有一个元素 s 只与 r 共享一个标记；它会有一个相似的分数 $\frac{1}{|r|+|s|-1}\leq\frac{1}{|r|}$。那么当一个元素 s 与 r 共享 x 个标记时，其相似度得分为：$\frac{x}{|r|+|s|-x}\leq\frac{x}{|r|}$。这个上界 $\frac{x}{|r|}$ 为 r 中的每个标记赋予权值 $\frac{1}{|r|}$，表示每个标记对总体最大匹配得分的贡献的上限。

下面形式化定义了加权签名方案。

定义 7 (加权签名方案)给定一个集合 $R=\{r_1,\cdots,r_n\}$ 和一个关联阈值 δ，加权签名方案是所有含有满足 $\sum_{i=1}^{n}\frac{|r_i|-|k_i|}{|r_i|}<\delta|R|$ 的有效签名集合 $K_R=\{k_1,\cdots,k_n\}$ 的总和签名 K_R^T。

例 3：以表 4 中的集合 R 为例，假设 $\delta = 0.75$。拐点 $\theta = |R|\delta = 3 \times 0.75 = 2.25$。$K_R^T = \{t_8, t_9, t_{10}, t_{11}, t_{12}\}$ 在加权签名方案中是图 1 中的一个有效签名 $K_R = \{\{t_8\}, \{t_9, t_{10}\}, \{t_{11}, t_{12}\}\}$ 的有效签名，其中

$$\sum_{i=1}^{n} \frac{|r_i| - |k_i|}{|r_i|} = \frac{6-1}{6} + \frac{6-2}{6} + \frac{5-2}{5} = 2.1 < \theta \text{。}$$

5.2.3. 最优签名选择

在上文中，选择倒排索引表中集合的并集作为相关候选集。因此，如何最大限度减少这个并集的规模，选择最优签名是一个主要问题。在本文中，采用与该并集大小成正比的倒排索引表的总长度作为优化目标。

问题(最优有效签名选择)：给定一个集合 R 和一个关联阈值 δ，找到一个有效签名 K_R^T 满足可以使 $\sum_{t \in K_R^T} |I[t]|$ 最小化，$|I[t]|$ 指的是倒排索引表中标记的个数。

本文中，将这个问题看做是一个背包问题，使用贪心近似算法来解决。下面给出该方法步骤：

1) 给出一个集合 $R = \{r_1, r_2, \cdots, r_n\}$，对于每一个在 R^T 中的 t，给它赋值 value $= \sum_{r_i | t \in r_i} \frac{1}{|r_i|}$ 以及一个代价 cost $= |I[t]|$；

2) 为了最小化倒排序列的大小，在 R^T 中按 cost/value 对所有标记以递增次序进行排序；

3) 依次选择标记，直到定义 7 中的条件得到满足为止，得到有效签名，其中所选的标记是 K_R^T 的签名标记；

例 4：考虑表 4，图 2 中表示了这个倒排索引，costs (即：索引列表的长度)为在 R^T 中的 12 个 tokens $t_1, \cdots, t_{12}$ 分别为 9，7，7，6，6，6，5，3，3，1，1，1。Values 对于 t_1 值为 8/15，t_4 的值为 1/3 每个 t_2, t_3 值为 11/30，$t_5, t_6, t_7, t_8, t_9, t_{10}$ 为 1/6，t_{11}, t_{12} 为 1/5。我们可以基于 cost/value 值按照递增序列排序并对他们进行选择。首先选 t_{12}，$|k_3| = 1$ 以及 $|k_1| = |k_2| = 0$。因此，有 $\sum_{i=1}^{n} \frac{|r_i| - |k_i|}{|r_i|} = 2.8$ 比值 $\theta = 2.4$ 大。然后继续选择 t_{11}, t_{10}, t_9；如果选择 t_8，就有 $|k_1| = 1$ 以及 $|k_2| = |k_3| = 2$。则有 $\sum_{i=1}^{n} \frac{|r_i| - |k_i|}{|r_i|} = 2.1$，比 $\theta = 2.25$ 小，停止继续选择，则有效签名为 $K_R^T = \{t_{12}, t_{11}, t_{10}, t_9, t_8\}$。

5.3. 候选集过滤

尽管上述方法删除了许多不相关的集合，但仍然会产生许多不相关的候选集。为了进一步删除误报的候选集，采用检查过滤器和最近邻过滤器来删除误报候选集。

5.3.1. 检查过滤器

通过上述有效签名的创建，已经确定了与 k_i 共享标记的元素 s，那么可以近似的计算出 $\Phi(r_i, s)$。如果证明所有 r 中的共享标记 k_i 和 s 仍然满足 $\Phi(r_i, s) \le \frac{|r_i| - |k_i|}{|r_i|}$，那么集合 S 将作为候选集(即使 S 与 K_R^T 只共享一个标记)。该方法被称为检查过滤器。

算法 4 描述了检查过滤器。对签名中的每一个标记，检索与签名匹配的倒排索引列表。如果集合通过 5~6 行中的步骤，则将集合添加到候选集合中。

算法 4：候选集选择及检查过滤

输入：参考集 R，签名 K_R^T，倒排索引表 I

输出：候选集 C

1) C←{}　　/*初始化候选集为空集*/

2) foreach $1 \le i \le n$ do

3)　foreach $t \in k_i$ do/*对签名中的标记进行遍历*/

4)　　foreach $\langle S, s \rangle \in I[t]$ do

5)　　if $\Phi(r_i, s) \ge \frac{|r_i| - |k_i|}{|r_i|}$ then

6)　　if $S \notin C$ then $C[S] \leftarrow \varnothing$

7)　　$C[S] \leftarrow C[S] \cup r_i$

8) return C

例 5：以表 4 为例。假设 $\delta = 0.75$，有含 $K_R = \{\{t_8\}, \{t_9, t_{10}\}, \{t_{11}, t_{12}\}\}$ 的有效签名集合 K_R^T。访问相应的倒排列表，并在三个候选集 S_2、S_3 和 S_4 上测试检查过滤器。S_2 没有通过检查过滤器 $\mathrm{Jac}(r_1, s_1^2) = 0.5 < \frac{|r_1| - |k_1|}{|r_1|} = 5/6$，$\mathrm{Jac}(r_2, s_3^2) = 4/7 < \frac{|r_2| - |k_2|}{|r_2|} = 2/3$。$S_3$ 和 S_4 通过检查过滤器 $\mathrm{Jac}(r_1, s_1^3) = \frac{5}{6} \ge \frac{|r_1| - |k_1|}{|r_1|} = 5/6$ 和 $\mathrm{Jac}(r_2, s_2^4) = 5/6 > \frac{|r_2| - |k_2|}{|r_2|} = 2/3$。

5.3.2. 最近邻过滤器

最近邻过滤器，即 R 和 S 之间的最大匹配分数值不会超过 R 中每个元素与其在 S 中最相似的元素(最近邻)的相似性总和：

$$\left| R \tilde{\cap}_{\varnothing} S \right| \le \sum_{r \in R} \max_{s \in S} \Phi(r, s)$$

所以，如果有一个候选集 S 的最近邻相似度之和小于 θ，那么就可以删除这个候选集。本文采用下面几种技术来优化该过滤器。

1) 搜索有效最近邻：遍历每个标记 $t \in r$，对于每一个标记 t，使用倒排索引表 $I[t]$获取 S 中包含标记 t 的所有元素 s。计算得到 r 和每个 s 之间的相似性；相似度得分最大的 s 被认为是最近邻。

2) 计算重用：在检查过滤器中，已经计算了 r 和 $s \in S$ 中包含 r 的签名标记的所有元素之间的实际相似度。由于这些超过 $\frac{|r| - |k|}{|r|}$ 的最大值之外的的其他不包含 r 签名标记的元素其相似度分数一定不会超过 $\frac{|r| - |k|}{|r|}$ 界限，所以这个相似度最大值一定是最近邻相似度。

3) 提前终止：对于 $r \in R$ 中签名标记出现在候选集合 S 中的元素，其最近邻相似度并不能保证 $\Phi(r, s) \le \frac{|r| - |k|}{|r|}$，因此要么重用检查过滤器中的计算公式，要么直接进行最近邻搜索。而对于签名标记没有出现在 S 集合中的元素 $r' \in R$，对所有 $s \in S$ 边界 $\frac{|r'| - |k'|}{|r'|}$ 仍然保留。该算法首先通过对匹配元素 r

的最近邻相似性进行求和来推断出一个总的估计值，并使用 $\frac{|r'|-|k'|}{|r'|}$ 估计非匹配元素 r'。然后，遍历每个 r'，使用 r' 的最近邻相似性来更新总估计值。如果总估计低于 θ，则删除该候选集。

算法 5 描述了最近邻过滤器。NNSearch 指最近邻搜索步骤。在第 3 步构建一个总估计值，4~5 行计算匹配元素的最近邻相似度，6~9 行计算不匹配元素的最近邻相似度与提前终止的最近邻相似度。

算法 5：最近邻过滤器

输入：参考集 R，阈值 δ，倒排索引 I，候选集：C

输出：最终候选集 C'

1) $C' \leftarrow \varnothing$ /*初始化最终候选集*/

2) foreach $S \in C$ do

3) total $\leftarrow \sum_{i=1}^{n} \frac{|r_i|-|k_i|}{|r_i|}$

4) foreach $r \in C[S]$ do

5) total $\leftarrow$ total + NNSearch$(r,S,I) - \frac{|r|-|k|}{|r|}$

6) foreach $r \in (R/C[S])$ do

7) total $\leftarrow$ total + NNSearch$(r,S,I) - \frac{|r|-|k|}{|r|}$

8) if total < θ then

9) goto 3

10) $C' \leftarrow C' \cup S$

11) return C'

例 6：以表 4 为例，假设 $\delta = 0.75$。对于候选集 S_3。当 r_1 和 s_1^3 共享签名标记。r_1 的最近邻是 s_1^3，他们的相似分数为 5/6。对于 r_2，给它一个相似分数上界 $\frac{|r_2|-|k_2|}{|r_2|} = 2/3$。对于 r_3，尽管 s_2^3 与签名 r_3 相似，但是它没有通过检查过滤器，因此也可以给它一个上界 $\frac{|r_3|-|k_3|}{|r_3|} = 0.6$。然后为 r_2 做最近邻搜索，发现 s_3^3 的相似度为 0.25。因此 r_2 的相似度分数上界更新为 0.25。则总估计值为 $\frac{5}{6}+0.6+0.25 < \theta = 2.25$，因此提前终止最近的邻居过滤器并删除候选集 S_3。

5.4. 最大匹配验证

在本节中，将对最大匹配验证进行优化。假设 $\psi(r,s) = 1-\phi(r,s)$ 为相似函数 ϕ 的对偶距离函数。可以看出，如果距离函数满足一定的约束条件(例如：满足三角不等式)，则最大匹配中一定存在相同的元素。假设有一对相同的元素 r 和 s 与其他元素 r' 和 s' 的关系有：

$$\phi(r,s')+\phi(r',s) = 1-\psi(r,s')+1-\psi(r',s) \quad (1)$$

$$= 2-\psi(r,s')-\psi(r',r) \quad (2)$$

$$\le 2-\psi\left(r',s'\right) \tag{3}$$

$$=1+1-\psi\left(r',s'\right) \tag{4}$$

$$=\phi\left(r,s\right)+\phi\left(r',s'\right) \tag{5}$$

在(1)中，将相似函数转换为它们的对偶距离函数。在(2)中使用 $r = s$，并在(3)中应用三角形不等式，从(4)到(5)，由于 $r = s$ 则使用 $\Phi\left(r,s\right)=1$ 来替换，在(5)中，可以看出连接 r 和 s 比连接 r 到 s' 和 r' 到 s 更好。因此，r 和 s 一定存在于最大匹配中。

应用这个三角不等式，本文从 R 和 S 中移除了所有相同的元素，并且在生成的集合上应用最大匹配算法。在最大匹配之后，将相同元素的数量添加到最大匹配分数中，即为最终的最大匹配分数，如表 4 中的数据，经验证 S_4 即为与 R 相关的记录。

6. 实验评估

6.1. 数据集

本文用 Python 对“面向领域的 Web 数据抽取”项目所抽取的多个数据源的房地产数据进行实验。抽取到的数据信息量如下表 5。这些数据记录都是以模式未知的半结构化数据存储的。可能存在的记录之间的差异有：数据物理存储顺序差异，数据丢失，数据类型差异以及属性交叉存储等。

Table 5. Statistics of information extracted from real estate
表 5. 房地产抽取信息量统计

城市	许可信息	楼盘	楼栋	户型
北京	7610	2212	12348	881294
上海	17404	5404	307450	3512295
广州	8927	1321	63231	1669550
深圳	2908	1561	10430	1140545
沈阳	6221	1567	30733	1945833
…	…	…	…	…
总计	160651	35926	772732	31019981

6.2. 实验方法

本文采用三种实体解析方法进行实验(表 6)。

Table 6. Description of experimental methods
表 6. 实验方法描述

Name	Description
传统方法	直接利用字符串相似度的方法，对模式未知的半结构化数据计算记录之间的相似度根据用户需求给定阈值 δ，大于该阈值的被认为是相似或相同记录。
基于字符串相似度的算法	将半结构化数据的每条记录看作字符串，每个分词看作子字符串，计算子字符串之间的编辑相似度，然后采用二分图最大匹配算法进行实体解析，最后根据用户需求给定阈值 δ，并找出关联分数大于阈值 δ 的记录，被认为是相似或重复记录。
基于集合相似度的实体解析算法	将半结构化数据的每条记录看作是属性的集合，为每条记录创建签名，采用签名和倒排索引表寻找相关的候选集，并且对候选集进行过滤，删除误报的候选记录，最后采用三角不等式优化二分图最大加权匹配，并对候选记录进行验证。

6.3. 实验结果及分析

为了使实验更具说服力本文分别从抽取到的房地产数据中随机抽取 500，000 条数据进行实验，其中有 1000 条重复数据。根据用户需求取记录相似度阈值 δ 分别为 0.70，0.75，0.80。来对上述三种方法进行实验分析。

评价指标：本文采用精确度(precision)、召回率(recall)、F_1-score，对实验结果进行评估分析。精确度(P)是分块正确的正例数量占所有分为正例的百分比，召回率(R)是分块正确的正例数量占实际正例数量的百分比，F_1-score 是它们的调和均值。同时采用运行时间来比较三种方法的运行速率。

从图 1 中可以看出，本文提出的两种模式未知的实体解析算法与传统的实体解析算法相比在精确率、召回率以及 F_1-score 上几乎相同，甚至能优于传统的方法。从图 2 中可以看出，本文提出的两种算法比传统的方法在运行速率上能提高不少。而基于集合相似度的算法更优于基于字符串相似度的算法。从图 3 中，比较了采用三种方法最终选择出的候选集个数，通过该表可以明显的看出基于集合相似性的方法大大减少了候选集 m 的个数。通过实验结果，可以看出本文提出的两种方法，在保证了与传统算法相同的结果的同时，提高了算法的运行速率，解决了模式未知的半结构化数据集实体解析的问题。

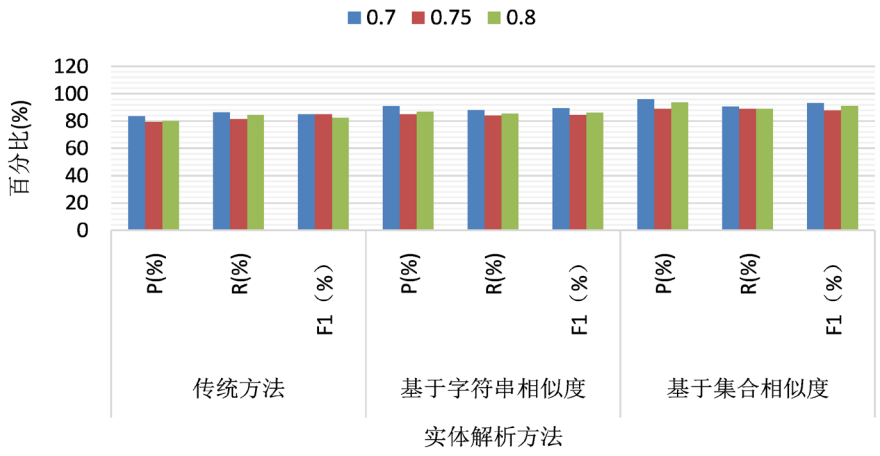

Figure 1. Experimental results of the three methods
图 1. 三种方法的实验结果

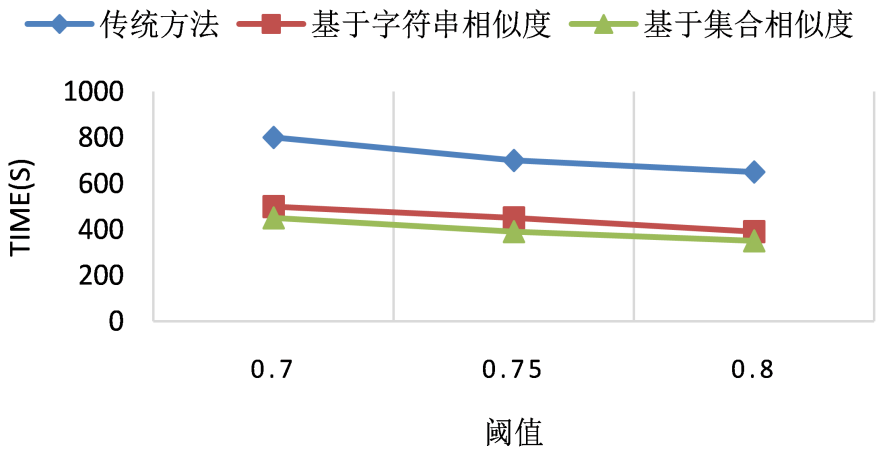

Figure 2. Running time comparison of three methods under different thresholds
图 2. 三种方法在不同阈值下的运行时间比较

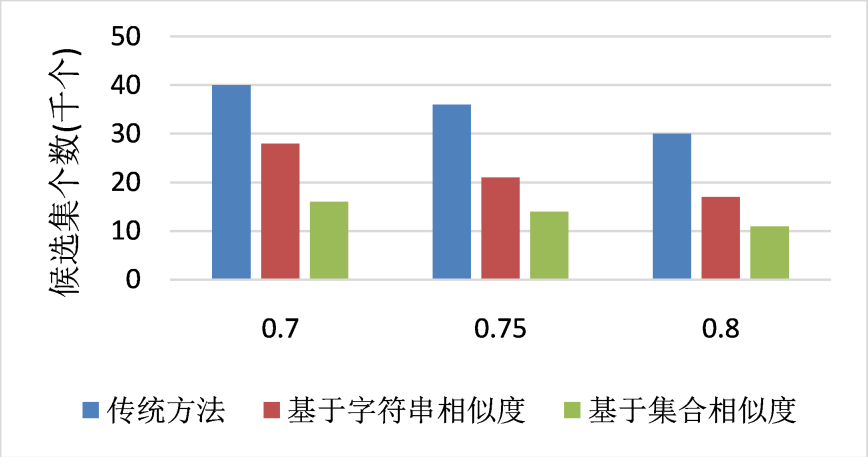

Figure 3. Comparison of the number of candidate sets of the three methods
图 3. 三种方法候选集个数的比较

7. 总结

本文研究了模式未知的半结构化数据实体解析问题，提出了一种基于字符串相似度的二分图最大加权匹配计算模式未知的记录相似度的方法，在此基础上，又提出了一种基于集合相似性的改进了的二分图最大加权匹配方法来解决实体解析问题。最后通过实验验证了在保证与传统方法实验结果相同的情况下，这两种方法提高了运行速率。理论分析和实验结果表明，这两种方法是正确且有效的。

基金项目

国家自然科学基金(No.61602323)；辽宁省博士启动基金(No.201601209)；住建部科学技术项目(No. 2017-K8-038)。

参考文献

[1] Galil, Z. (1986) Efficient Algorithms for Finding Maximum Matching in Graphs. *ACM Computing Surveys*, **18**, 23-38. https://doi.org/10.1145/6462.6502

[2] Elmagarmid, A.K. and Member, 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

[3] 高广尚. 面向实体解析的无监督聚类方法综述[J]. 计算机工程与应用, 2018(7): 11-19.

[4] 高广尚, 张智雄. 关于实体解析基本方法的研究和述评[J]. 数据分析与知识发现, 2019, 3(5): 27-40.

[5] 王宁, 李杰. 大数据环境下用于实体解析的两层相关性聚类方法[J]. 计算机研究与发展, 2014, 51(9): 2108-2116.

[6] Hernandez, M. and Stolfo, S. (1995) The Merge Purge Problem for Large Databases. ACM, New York. https://doi.org/10.1145/223784.223807

[7] Monge, A.E. and Elkan, C.E. (1997) An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. *Proceedings of Workshop on Research Issues on Data Mining and Knowledge Discovery*, Newport Beach, 14-17 August 1997, 23-29.

[8] Gravano, L. and Ipeirotis, P.G. (2001) Using Q-Grams in a DBMS for Approximate String Processing. *IEEE Data Engineering Bulletin*, **24**, 28-34.

[9] Ristad, E.S. and Yianilos, P.N. (1998) Learning String-Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **20**, 522-532. https://doi.org/10.1109/34.682181

[10] Deng, D., Fernandez, R.C., Abedjan, Z., Wang, S., Stonebraker, M., Elmagarmid, A., Ilyas, I.F., Madden, S., Ouzzani, M. and Tang, N. (2017) The Data Civilizer System. 8*th Biennial Conference on Innovative Data Systems Research*, Chaminade, CA, USA, 8-11 January 2017, 7 p.

[11] Deng, D., Li, G. and Feng, J. (2014) A Pivotal Prefix Based Filtering Algorithm for String Similarity Search. *ACM SIGMOD International Conference on Management of Data*, Snowbird, 22-27 June 2014, 673-684. https://doi.org/10.1145/2588555.2593675

[12] Fredman, M.L. and Tarjan, R.E. (1987) Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms. *Journal of the ACM*, **34**, 596-615. https://doi.org/10.1145/28869.28874

[13] Wang, J., Li, G. and Feng, J. (2011) Fast-Join: An Efficient Method for Fuzzy Token Matching Based String Similarity Join. 27*th International Conference on Data Engineering*, 11-16 April 2011, 458-469. https://doi.org/10.1109/ICDE.2011.5767865

[14] Wang, J., Li, G. and Feng, J. (2014) Extending String Similarity Join to Tolerant Fuzzy Token Matching. *ACM Transactions on Database Systems*, **39**, 7. https://doi.org/10.1145/2535628