Hans Journal of Data Mining 数据挖掘, 2022, 12(3), 280-296
Published Online July 2022 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2022.123028

Hans汉斯

基于局部敏感哈希及模糊连接的实体解析算法

樊沁怿，李　贵，李征宇

沈阳建筑大学，信息与控制工程学院，辽宁 沈阳

收稿日期：2022年6月20日；录用日期：2022年7月21日；发布日期：2022年7月28日

摘　要

随着互联网技术的发展和应用，Web数据量越来越大，在Web数据集成中，实体解析作为其中的重要环节，其主要任务是将不同Web数据源中指向现实世界同一实体的记录识别出来。然而这些数据往往都来自于不同的数据源，存在着数据重复等问题。为了解决特定领域的实体解析问题通常采用自定义模糊连接的方式来解决。但是目前较为先进的模糊连接技术诸如前缀过滤技术等均不支持转换规则的定制，并且表现出较差的性能和可扩展性。为了解决特定领域的数据重复问题，提升实体解析算法的可扩展性，本文引入了一种基于自定义转换规则的模糊连接技术来提升算法的可扩展性；采用基于局部敏感哈希映射的签名方案来获得签名，与前缀过滤相比，通过局部敏感哈希映射产生的签名更具有代表性，能够对局部敏感哈希映射中高频出现的签名进行剪枝来显著地减少需要匹配的次数；最后通过集合相似性来判断是否为重复实体，并利用实际地产领域数据集验证了算法的有效性。

关键词

模糊连接，实体解析，集合相似性，局部敏感哈希

Entity Resolution Algorithm Based on Locality Sensitive Hash and Fuzzy Join

Qinyi Fan, Gui Li, Zhengyu Li

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

Received: Jun. 20th, 2022; accepted: Jul. 21st, 2022; published: Jul. 28th, 2022

Abstract

With the development and application of Internet technology, the amount of Web data is increas-

文章引用: 樊沁怿, 李贵, 李征宇. 基于局部敏感哈希及模糊连接的实体解析算法[J]. 数据挖掘, 2022, 12(3): 280-296.
DOI: 10.12677/hjdm.2022.123028

ing. In Web data integration, entity resolution is an important link. Its main task is to identify records from different Web data sources that point to the same entity in the real world. However, these data often come from different data sources, and there are problems such as data duplication. In order to solve the problem of entity resolution in a specific domain, a custom fuzzy join is usually used to solve it. However, the more advanced fuzzy join technologies such as prefix filtering technology do not support the customization of transformation rules, and show poor performance and scalability. In order to solve the problem of data duplication in specific fields, improve the scalability of entity resolution algorithms, in this paper, a fuzzy join technology based on custom conversion rules is introduced to improve the scalability of the algorithm; a signature scheme based on locality-sensitive hash mapping is used to obtain signatures. The signature is more representative, and can prune the signatures that appear frequently in the local sensitive hash map to significantly reduce the number of matching; finally, it is determined whether it is a duplicate entity through the similarity of the set, and the actual real estate field data set is used to verify the effectiveness of the algorithm.

Keywords

Fuzzy Join, Entity Resolution, Set Similarity, LSH (Locality-Sensitive Hashing)

1. 引言

记录连接[1]，也称为记录匹配，是实体解析中的一项基本操作。基本思想是：通过计算属性值之间的相似度，并结合属性级阈值和记录级阈值来决定两条记录是否匹配。记录连接与合并操作共同构成了实体解析。记录连接操作用于判断两条记录是否为同一个实体，合并操作是将匹配的记录进行合并，使它们成为同一个记录集合来对应同一个实体。图 1 以楼盘信息为例，展示了实体解析的完整过程，表 1 是抓取的某一网站关于某个楼盘的信息，表 2 是抓取的另一网站关于某个楼盘的信息。记录连接操作用于判断抓取的不同网站的楼盘信息是否对应的同一个楼盘，合并操作则用于合并匹配的楼盘记录，使他们成为一个记录的集合来对应相应的楼盘，生成的实体如表 3 所示。模糊连接(也称为集合相似连接或模糊匹配)是记录匹配中使用的一种强大的运算，它可以根据给定的相似性函数有效地识别彼此相似的记录对。给定参考表 R 和输入表 S，对于每个记录 $s \in S$，进行模糊连接运算后返回所有记录 $r \in R$，使得 sim (s, r) $\geq \theta$，其中 sim 是相似性函数，θ 是用户指定的阈值。常用的相似度函数包括 Soundex、Levenshtein 距离(编辑距离)、Hamming 距离、余弦相似度、Jaro-Winkler 相似度、Jaccard 相似度等。

这里我们将属性级阈值与记录级阈值都设置为 0.6。由图 1 可以看出一个完整的实体解析过程需要进行$(n*m)^2$次匹配操作(假设 R 表中由 n_1 条记录，每条记录有 m_1 个属性；S 表中有 n_2 条记录，每条记录有 m_2 个属性)计算复杂度为 o $(nm)^2$，对于数据量较为庞大的数据集来说采用图 1 的原始方法是不可行的。因此大多数现有的技术[2] [3] [4]都采用的图 2 所示的过滤验证体系结构。过滤步骤使用基于签名的算法为 S 和 R 中的每个字符串都生成一组签名，如果 sim (s, r) $\geq \theta$，则 S 和 R 至少共享一个公共签名。由于集合重叠可以通过等连接来测试，因此可以使用大数据引擎或关系数据库引擎来评估这一步骤花费的时间成本。在验证步骤中，为每个未被过滤的候选对(s, r)调用相似性函数，并且仅输出相似性超过给定阈值的那些对。

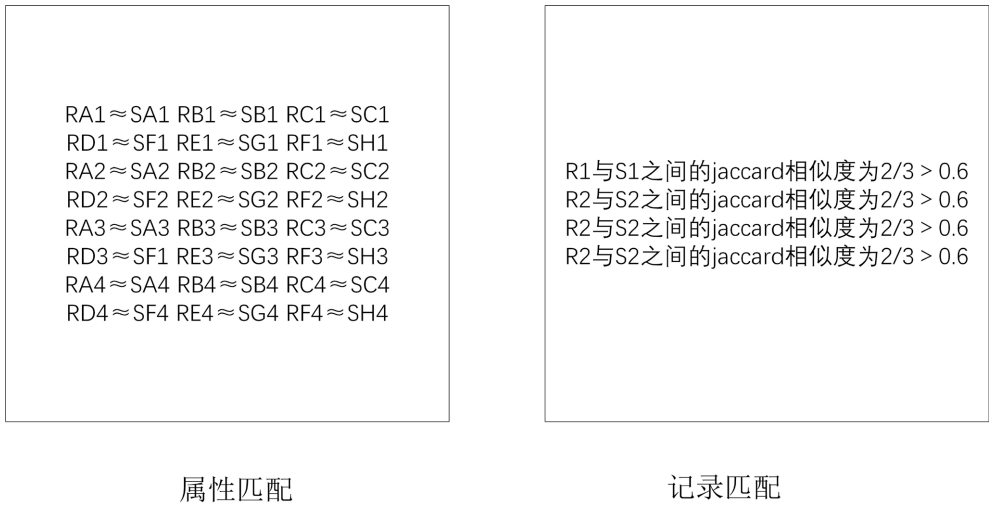

Figure 1. Entity resolution process
图 1. 实体解析过程

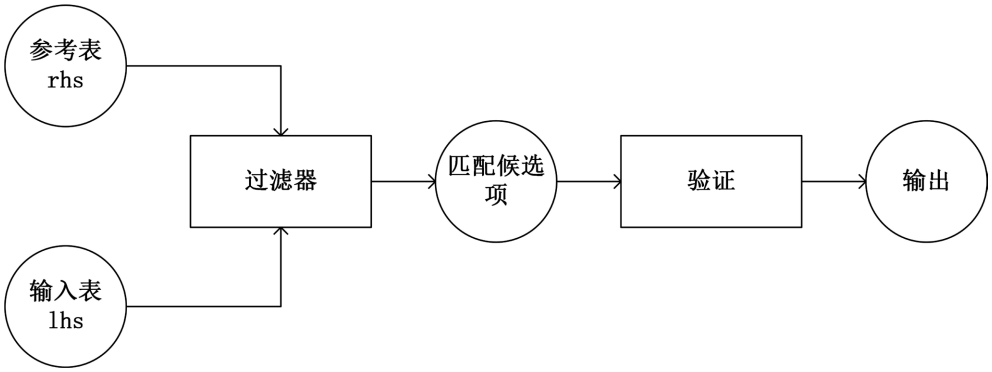

Figure 2. Fuzzy join filtering verification architecture
图 2. 模糊连接过滤验证体系结构

Table 1. Information about four real estates crawled by a website
表 1. 某网站抓取的关于四个楼盘的信息

	SA	SB	SC	SD	SE	SF	SG	SH
1	华润时代之城	普通住宅，洋房	沈海热电厂原址	30%	2.00	沈阳润投房地产开发有限公司	1080	1000 个左右
2	华润置地九悦	大平层	黄河北大街三台子地铁口 B 口	35%	2.50	沈阳润嘉房地产有限公司	570	520
3	金地檀溪	住宅	浑南新区浑南中路 18 号	35%	2.10	沈阳金地长青房地产开发有限公司	3773	
4	保利云上	普通住宅，洋房	浑南区祝科街与浑南大道交汇处	35%	1.8	沈阳林海房地产开发有限公司	1230	1070

Table 2. Information about four real estates crawled by a website
表 2. 某网站抓取的关于四个楼盘的信息

	RA	RB	RC	RD	RE	RF	RG	RH
1	华润时代之城	普通住宅，洋房	沈海热电厂原址	沈阳润投房地产开发有限公司	1080	1000	高层 3.5 元/m^2/月，洋房 3.8 元/m^2/月	35 栋
2	华润置地九悦	大平层	黄河北大街三台子地铁口 B 口	沈阳润嘉房地产有限公司	570	520	住宅 3.6 元/m^2/月	21 栋
3	金地檀溪	住宅	浑南新区浑南中路 18 号	沈阳金地长青房地产开发有限公司	3773		高层 2 元/m^2/月，洋房 2.4 元/m^2/月	49 栋
4	保利云上	普通住宅，洋房	浑南区祝科街与浑南大道交汇处	沈阳林海房地产开发有限公司	1230	1070	高层 3 元/m^2/月，洋房 3.6 元/m^2/月	8 栋小高层，21 栋洋房

Table 3. Generate entity
表 3. 生成实体

1	华润时代之城	普通住宅，洋房	沈海热电厂原址	30%	2.00	沈阳润投房地产开发有限公司	1080	1000 个左右	高层 3.5 元/m^2/月，洋房 3.8 元/m^2/月	35 栋
2	华润置地九悦	大平层	黄河北大街三台子地铁口 B 口	35%	2.50	沈阳润嘉房地产有限公司	570	520	住宅 3.6 元/m^2/月	21 栋
3	金地檀溪	住宅	浑南新区浑南中路 18 号	35%	2.10	沈阳金地长青房地产开发有限公司	3773		高层 2 元/m^2/月，洋房 2.4 元/m^2/月	49 栋
4	保利云上	普通住宅，洋房	浑南区祝科街与浑南大道交汇处	35%	1.8	沈阳林海房地产开发有限公司	1230	1070	高层 3 元/m^2/月，洋房 3.6 元/m^2/月	8 栋小高层，21 栋洋房

2. 相关工作

Arasu 等人开发了一个基于转换的模糊连接技术[5]。为了获得良好的性能，他们使用局部敏感哈希(LSH)来为每个记录生成签名，并在参考表 R 上创建签名索引。与精确的前缀过滤方法不同，LSH 算法[6]通过生成多个签名来保证较高的准确率。用于签名生成的 LSH 方法显著改进了以前的技术，提高了性能。这种方法还可以利用多个 CPU 进行并行处理，因为对不同 S 记录的查找可以针对索引并行进行。

如果参考表 R 过大的情况下，单节点的解决方案九不在使用。Fier 等人[7]最近的一项实验研究比较了几种模糊连接技术。基于这项实验的结果，当前的这些模糊连接技术都存在着些许不足之处。首先，

对于频繁出现的令牌比较敏感。例如，上述实验中综合得分最高的技术，Vernica [8]等人使用的一种流行签名方案的变体，称为前缀过滤[9]。这个方法就对频繁出现的令牌十分敏感，这通常会导致任务的失败或者超时。此外，前缀过滤技术不是非常具有选择性，除非相似性阈值 θ 非常高(例如 0.95)。因此，验证步骤的时间成本可能非常昂贵。实际上，为了在记录中获得所需的准确率－召回率的平衡，一般选择 0.8 左右的阈值。

因此我们采用一个可以扩展的模糊连接算法，它能够支持通过转换规则对原来输入表中的数据进行替换，通过对转换规则的自定义，能够实现模糊连接的扩展。对于输入表 S 很大但是参考表 R 足够小使得索引能够适合单个节点作为主内存的情况，我们采用广播模糊连接技术，其中 R 上的索引呗广播(即复制)到多个工作节点，并且输入表 S 在这些节点之中被分区。更具挑战性的情况是当 R 上的索引不适合单个节点的主内存时。在这种情况下，我们开发了一种混洗模糊连接技术，其中 S 和 R 都跨节点划分。与以前使用的前缀过滤作为签名的很想扩展方法相比，我们采用 LSH 来生成签名，使用 LSH 的好处时：在实践中，除了少数签名外，所有签名的频率都非常低。虽然一些签名实际上出现的频率十分频繁，但是由于每行都有足够的冗余签名，因此可以对这些高频数显的签名进行删减，然而对召回率的影响可以忽略不计。由于在一个或两个表中非常频繁出现的签名不需要被连接，这种修剪导致数据混洗成本的显著降低。

本文的主要贡献如下：

1) 针对特定领域的实体解析问题，引入了一种基于自定义转换规则的模糊连接技术，提高了算法的可扩展性。

2) 采用 LSH 生成签名来寻找相似的记录，该方法的本质是将高维数据降维至低维数据，已达到高效处理海量高维数据的近似最近邻问题。并对可以通过高频出现的签名进行剪枝进一步提高效率。

3) 采用基于IDF逆文档频率策略来对参考表中的令牌进行加权。认为越少出现的令牌区分能力更强。并使用集合属性相似度，来为记录进行匹配。

4) 提出了两种模糊连接方法，在参考表 R 较小的情况下采用广播模糊连接，当参考表 R 较大的情况下采用混洗模糊连接。

3. 基于集合相似性的模糊连接

从形式上来说，模糊连接是用相似性函数 sim 和阈值 θ 参数化之后的算式，它将两个实体 S 和 R 作为输入，并为 S 中的每一条记录都返回 R 中相似度高于指定阈值的所有记录。

$$\mathrm{FJ}_{\mathrm{sim}\theta}(\mathrm{S},\mathrm{R})=\left\{\langle \mathrm{s},\mathrm{r}\rangle \middle| \mathrm{s}\subseteq \mathrm{S},\mathrm{r}\subseteq \mathrm{R},\mathrm{sim}(\mathrm{s},\mathrm{r})\geq\theta\right\}$$

虽然特定的相似性函数(例如编辑距离)可以处理一种数据类型的相似问题，但是没有一个单独的相似性函数可以处理大量不同数据类型的相似问题，其中一些问题可能是领域或应用所特有的的。我们观察到几个例子。例如，在地产决策系统中需要将用户查询与国内许多城市的楼栋信息参考表进行匹配。因此模糊连接必须能够处理存在数据位移(辽宁省浑南区沈阳市↔辽宁省沈阳实浑南区)，数据不完整(浑南中路 25 号↔浑南区浑南中路 25 号)，数据交错(浑南区 25 号浑南中路↔浑南区浑南中路 25 号)等数据质量情况下的数据重复问题。

3.1. 核心相似性函数

我们使用加权 Jaccard 作为核心相似性函数，并通过对基于自定义转换规则的模糊连接来处理如数据位移、属性缺失、属性值交错等问题。

给定两个字符串 a 和 b，我们使用令牌化函数将它们转换成两个令牌序列[a_1, a_2, ⋯, a_m]和[b_1, b_2, ⋯, b_n]。通过加权函数 w 为令牌分配权重。通常情况下，我们使用反向文档频率(idf)权重[10]，它认为具有较低频率的令牌在确定相似性时应该具有更大的权重。

$$\mathrm{Wi}(\mathrm{token}) = \log\left(\frac{\text{total\_rows}}{\text{rows\_cotaining\_token\_in\_column\_i}}\right)$$

A 和 B 的相似性定义为：

$$\mathrm{sim}\left(\left[a_1, a_2, \cdots, a_m\right], \left[b_1, b_2, \cdots, b_n\right]\right) = \frac{\sum_{x \in A \cap B} w(x)}{\sum_{x \in A \cup B} w(x)}$$

其中集合 $A = \{a_1, a_2, \cdots, a_m\}$ 和 $B = \{b_1, b_2, \cdots, b_n\}$，因为我们使用了集合相似性，所以令牌的顺序并不影响相似性。

3.2. 自定义的转换规则

我们将字符串转换规则定义为<lhs → rhs>，其中 lhs 和 rhs 是令牌序列，lhs 可以为空。在令牌序列 s 上应用转换规则，将 s 中的每个匹配的 lhs 序列替换为序列 rhs。

例如当应用于令牌序列[辽宁省，沈洋市]时，发生<沈阳市→沈洋市>转变，导致序列变为[辽宁省，沈阳市]。如果要在一个令牌序列上应用多个转换，则转换的 lhs 匹配必须来自原始令牌序列，而不是来自使用另一个转换规则进行替换的令牌序列。

给定一组转换规则，应用与 s 相关的所有规则子集。应用每个规则子集生成 s 的变体。s 所有变体中与 r 的 Jaccard 系数最高的是。参考表中 s 和 r 的相似度是 s 所有的变体中与 r 的最高的 Jaccard 相似度。图 3 展示了在编辑转换规则的应用下如何定制相似性函数的例子。如果不应用编辑转换，原始记录之间的 Jaccard 相似度为 1/3。编辑转换的应用导致三个额外的变量。所有变体的最大 Jaccard 相似度提高到 2/3。

Figure 3. A matching instance when there is an edit transformation
图 3. 存在编辑转换时的匹配实例

4. 签名生成

签名方案决定了过滤步骤的选择性(见图 2)，这既影响从表 R 和表 S 中连接签名的时间成本，也影响验证步骤的时间成本，因为更具选择性的签名方案需要验证的记录对更少。在本节中，首先回顾两个众所周知的签名方案：前缀过滤和局部敏感哈希(LSH)。然后对这两种签名方案进行了比较。

4.1. 前缀过滤

前缀过滤根据权值对字符串 s 中的令牌进行排序(使用令牌 id 作为分隔符，因为前缀过滤要求所有令牌之间有一个稳定的全局总顺序)。让排序后的序列表示为$[s_1, s_2, \cdots, s_m]$。设 i 是满足下列条件的最小的值：

$$\frac{\sum_{k=1}^{i} W(S_k)}{\sum_{k=1}^{n} W(S_k)} \geq (1-\theta)$$

其中 θ 是相似性阈值。然后令牌$\{s_1, \cdots, s_i\}$是 s 的签名，任何加权 Jaccard 相似度大于 θ 的字符串都必须包含$\{s_1, \cdots, s_i\}$。

4.2. 局部敏感哈希映射

局部敏感哈希(LSH)是另一个用于集合相似连接的签名方案。它使用 k × m 个独立的哈希函数。设 h 为散列函数之一，令牌集为$\{s_1, \cdots, s_n\}$。然后定义 h 中的最小哈希令牌为 $\text{argmin}x \in \{s_1,\cdots,s_n\} h(x)$。设 y_i 为 h_i 下 $1 \leq i \leq (k \times m)$的最小哈希令牌。LSH 将它们分成 m 个组，每个组具有 k 个最小散列令牌，即 $((y_1,\cdots,y_k),(y_{k+1},\cdots,y_{2k}),\cdots)$，然后通过散列每个组生成 m 个签名，即 $\text{signature1} = h'(y_1, y_2, \cdots, y_k)$，$\text{signature2} = h'(y_{k+1}, y_{k+2}, \cdots, y_{2k})$ 等。其中 h'是不同的哈希函数。对于 Jaccard 相似度大于 θ 的任意两个字符串，上述签名方案具有大于 $1-(1-\theta^k)^m$ 的概率来生成$\{\text{signature}_1, \cdots, \text{signature}_m\}$签名集合。通常情况下我们使用 k = 4 和 m = 6 作为默认值，对于 $\theta \geq 0.8$，$1-(1-\theta^k)^m \geq 0.95$。

将上述 LSH 方案扩展到加权 Jaccard 相似性。设 si 为字符串 s 中的一个令牌，w(si)为该令牌的权重，h 是一个 hash 函数。它不是通过 $h(s_i)$作为哈希值来计算最小哈希值，而是首先将 h(si)统一映射为 0 到 1 之间的一个数字，表示为 h'，并使用 $\log(h')/w(s_i)$作为哈希值。生成签名的其余过程是相同的。

4.3. 比较

准确度与召回率：默认的设置是(k = 4, m = 6)，能够保证相似性阈值为 0.8 时准确率高于 0.95。然而在实际的情况中，实验表明实际召回率(≥0.999)比理论界限高得多。

签名生成成本：前缀过滤通常情况下每行生成的签名比 LSH 少，计算量也比 LSH 少，LSH 要求每条记录调用几个散列函数。因此前缀过滤的签名生成成本较低。

签名频率的分布：如图 4 和图 5 所示，虽然前缀过滤每行生成的签名比 LSH 少，但前缀过滤生成的大多数签名通常比 LSH 生成的签名与表中更多的记录相关联。在 LSH (k = 4, m = 6)方案中，它生成的大多数签名具有更好的代表性，即，对于特定的签名 sig，表中以 sig 作为签名的行数非常少(百分之 99 的情况下都低于 10%)，因为签名实际上是来自该表的随机记录。除非参考表中存在相同或几乎相同的行，否则签名不太可能与大量行相关联。只有少量的“异常值”与大量的行相关联。相比之下，前缀过滤的选择性要差得多。例如，大约 20%的签名与 10 多行相关联，大约 5%的签名与 100 多行相关联。

两种签名方案的对立情况发生在原始数据行中有许多行仅包含高频出现的令牌。在这种情况下，前缀过滤使用这些高频出现令牌作为签名，并且这些签名与许多行相关联。另一方面，LSH 生成随机 k 个令牌的哈希值作为签名，这些签名通常与少得多的行相关联。只有在极少数情况下，LSH 才会生成与许多行相关联的签名。鉴于这些特性，我们在实验中使用 LSH 作为默认签名方案。

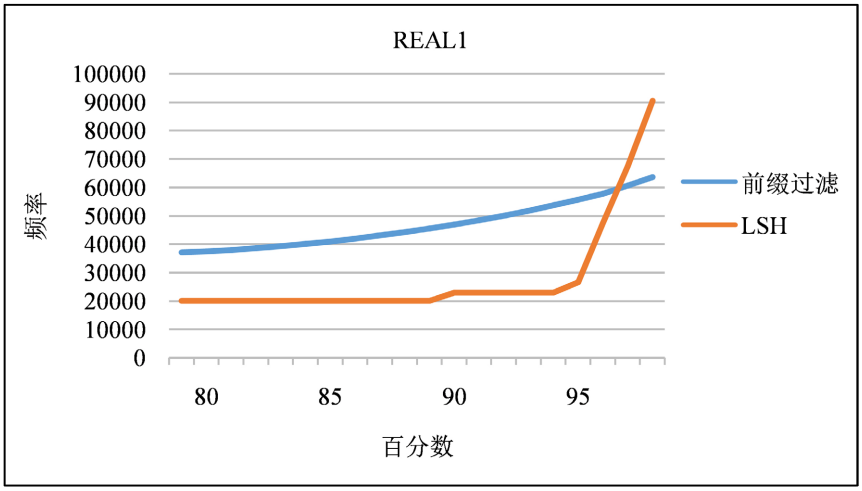

Figure 4. Frequency distribution of signatures in REAL1 dataset
图 4. REAL1 数据集中签名的频率分布

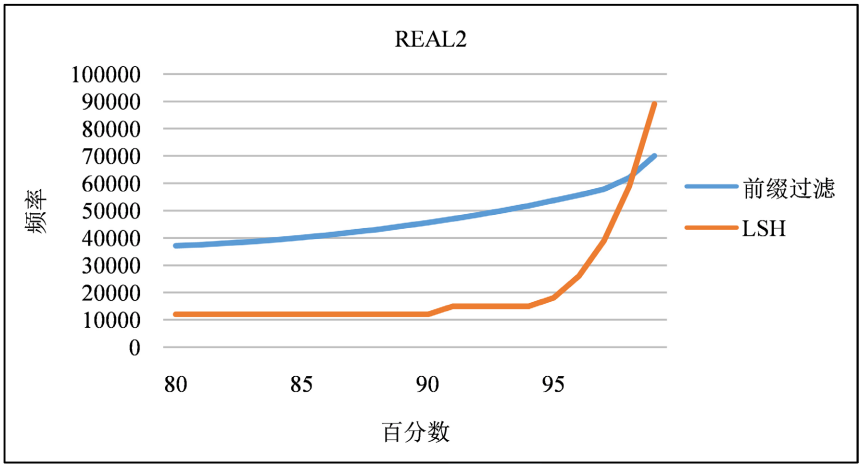

Figure 5. Frequency distribution of signatures in REAL2 dataset
图 5. REAL2 数据集中签名的频率分布

4.4. 签名修剪

签名修剪策略基于以下观察：首先，模糊连接操作的运行时间很大程度上取决于签名的代表性。对于 LSH，如上所述，绝大多数签名仅与参考表中的几行相关联。因此，连接大小仅由少数高频出现的签名决定。其次，我们的 LSH 方案每行重复签名生成过程(m = 6)次。这给我们在每个记录中丢弃签名的空间。例如，如果两个表 S 和 R 中的一对记录(S, R)具有大于 0.8 的 Jaccard 相似性，并且我们使用 k = 4 个最小哈希值令牌作为一个签名，那么使用 5 个而不是 6 个(默认 m)签名导致(S, R)至少共享一个签名的概率为 93%，使用 4 个签名的话至少共享一个签名的概率是 89%。换句话说，丢弃一个签名只会使他们共享至少一个签名的概率降低大约 2% (从 95%到 93%)，而丢弃两个签名会使这一概率降低 6% (从 95%到 89%)。这两个观察促使我们使用删除策略，删除与参考表中超过一定数量的记录相关联的签名，因为它们占据了运行的大部分时间，并且删除它们对召回率的影响最小。在实验中我们发现对召回率的影响可以忽略不计(远远小于理论分析)。进一步支持这种修剪的一个因素是对令牌使用 IDF 权重。高频出现签名通常对应于低权重的令牌(或令牌序列)。因此，删除这些签名对相似度的影响很小。

此外，上述修剪技术对前缀过滤无效。签名频率分布也并非偏斜分布，很多签名与数据库中的大量记录相关联；因此，试图通过剪枝获得显著的优化将导致召回率非常高。

5. 扩展模糊连接

在本节中，我们概述了如何实现模糊连接的扩展。类似于并行数据库中的传统连接，我们实现了模糊连接的两种情况。广播模糊连接和混洗模糊连接。当参考表 R 很小的情况下，并且 R 的签名的倒排索引适合于单个节点的内存时，可以使用广播模糊连接。对于参考表很大的情况下，可以采用混洗模糊连接。因为实际的连接步骤是在令牌 ID 空间中工作的，而不是在原始的字符串中，所以效率很高。广播和混洗模糊连接基本流程具有相同的初始步骤，将参考表和输入表转换为令牌 ID 空间。

5.1. 预处理

图 6 描述了预处理的基本流程。在预处理过程的最后，我们将参考表记录和输入表记录转换成令牌标识和令牌权重。

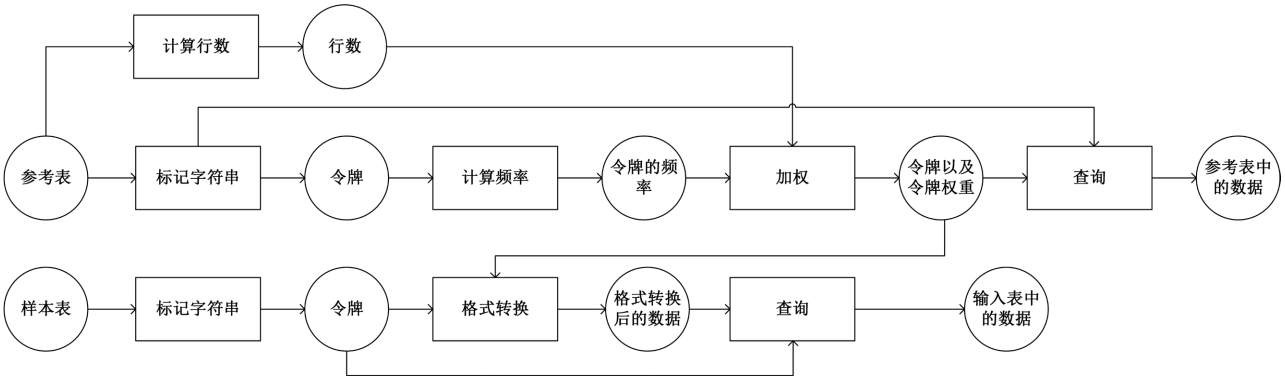

Figure 6. Preprocessing
图 6. 预处理

步骤 P1：计算参考表中的行数。计数存储在内存中，并在步骤 P2 中使用。通常情况下，该步骤可以与步骤 P2 (计算令牌频率)相结合，方法是每行插入一个空令牌，并将空令牌的计数用作行数。因为这一步通常并不需要花费多少时间成本，所以这种优化不会显著减少运行时间。

步骤 P2：计算 R 中的令牌频率。我们将右(参考)表的每一行中的字符串进行令牌化，并将它们分组。我们对令牌频率进行计数，并根据令牌值对它们进行排序，然后将它们变为<token, freq>数组。使用步骤 P1 中计算的行数，它计算每个令牌的 IDF 权重，并生成一个权重数组。因为我们使用的 LSH 签名生成方案要求令牌 ID 是可重复的，所以必须要进行排序。我们使用数组中的索引作为令牌 ID，因此每次模糊连接都为相同的令牌分配相同的令牌 ID。假设 R 中不同令牌的数量相对较小，并且它们都可以保存在内存中并广播给所有工作节点。

步骤 P3：我们对输入表 s 中的字符串进行令牌化，并将他们的令牌进行分组。然后我们将步骤 P2 中收集到的<token, freq>数组广播到所有工作节点，并进行映射步骤，对于其中的每个令牌，我们计算它 k 次编辑内的所有令牌。我们计算 k 次编辑内所有令牌的方法采用了 tire 的方法。我们通过将数组中的所有令牌插入 tire 中来对 tire 进行初始化。对于输入表 S 中的每一个令牌，我们遍历 tire，如果前缀已经超过 k 次编辑，那么我们可以不再访问该 tire 中的所有子树。尽管有这种优化，这种计算成本也是昂贵的(一组数百万个令牌，每个令牌大约 1 毫秒)；我们在扫描和洗牌所有令牌中进行了权衡，决定每个令牌只计

算一次编辑转换规则。我们将转换规则收集成为一个数组<token, array of token-IDs of tokens within k-edits>。同样，由于转换规则的数量相对较少，该数组可以保存在内存中并进行广播。

步骤 P4：为连接参考表行。我们将参考表中的每一行转换成<rid, (<token-ID, weight>)*>。为此，我们广播了在步骤 P2 中生成的(token-ID, weight)数组，并构建了一个哈希表。

步骤 P5：为连接准备输入表行。我们将输入表中的每一行转换为<rid, (<token-ID, weight>)*，(<position, (token ID and weight of transformed token)*>)*>，我们广播在步骤 P2 中生成的令牌权重数组和步骤 P3 中具有相似令牌 ID 与权重的数组，并构建哈希表。然后我们运行一个映射步骤，为输入表中的每个令牌化记录查找每个令牌的 ID 和权重，如果符合编辑距离变换规则，则构造位置变量以及变换后与参考表 token 对应的 token-ID 和权重数组。我们在这里使用位置变量，因为相同的令牌可以在字符串中多次出现，并且每个转换都需要独立跟踪。

5.2. 广播模糊连接

图 7 描述了广播连接管道。它将预处理流程的步骤 P4 和 P5 中产生的“准备好的”参考表和输入表记录作为输入。过程的主要步骤有：

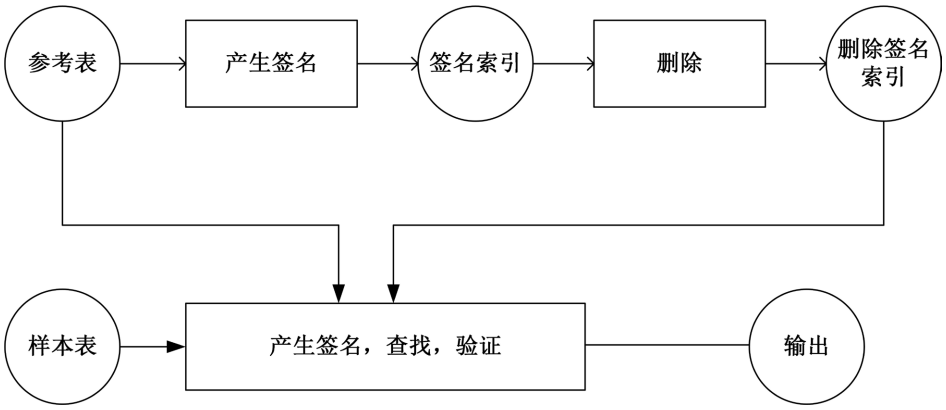

Figure 7. Broadcast fuzzy join
图 7. 广播模糊连接

步骤 B1：为参考表生成签名：我们运行一个映射步骤，对于预处理过程中步骤 P3 产生的 dataframe 中的每一记录，调用 LSH 签名生成器来生成签名，并将其展平为(rid, signature)的 dataframe。这是参考表的签名索引。

步骤 B2：删除签名：根据步骤 B1 中产生的签名索引进行删减的签名集。我们按签名对签名索引进行分组，并过滤掉那些计数高于指定截止值的索引。我们观察到，即使对于广播模糊连接，修剪也会导致加速，因为对签名索引的查找更少(尽管对于混洗模糊连接来说，增益要大得多)。

步骤 B3：生成并验证候选对。我们收集在步骤 B1 和 B2 中生成的 dataframe，并将它们广播给所有工作节点。然后，我们在预处理过程的步骤 P4 中产生的 dataframe 上运行映射步骤:我们调用签名生成器来获取签名，确保签名没有被删减，查找索引来找到候选行，最后验证相似性是否高于给定的阈值。

5.3. 混洗模糊连接

图 8 描述了混洗连接流程。与广播版本一样，它也将预处理流程的步骤 P4 和 P5 中产生的“准备好的”参考和输入表行作为输入，然后如下进行：

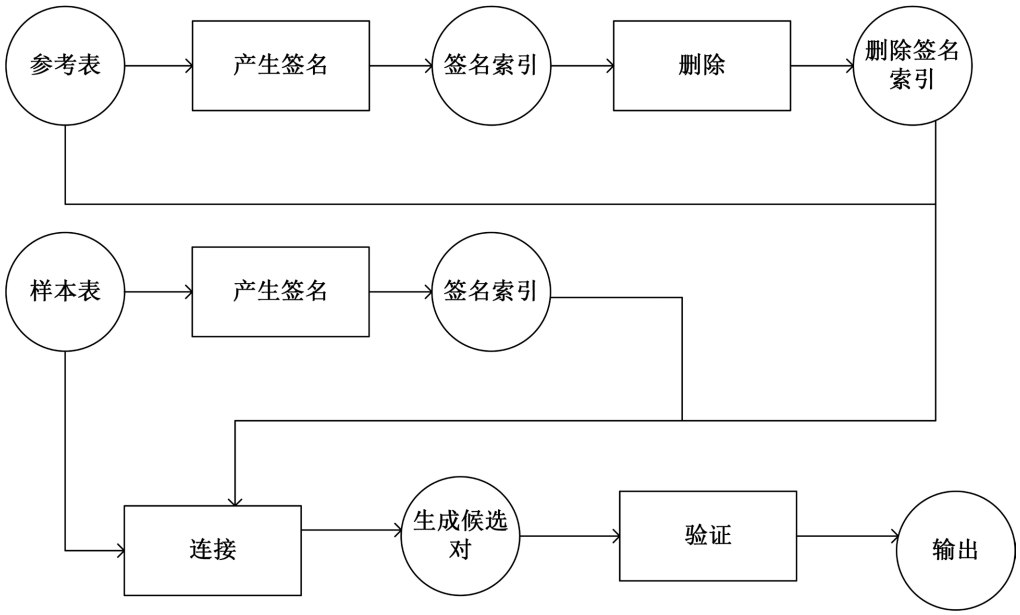

Figure 8. Shuffle fuzzy join
图 8. 混洗模糊连接

步骤 S1：为参考表生成签名。我们运行一个映射步骤，对于预处理步骤中步骤 P3 产生的 dataframe 中的每一行，调用 LSH 签名生成器来生成签名，并将其展平为(rid, signature)的 dataframe。这是参考表的签名索引。

步骤 S2：删除签名。我们通过签名对签名索引进行分组，并提取计数小于截止阈值的索引。它们是修剪后剩下的签名。

步骤 S3：为输入表生成签名。我们为预处理步骤 P4 中产生的数据帧中的每一行运行一个映射步骤，调用 LSH 签名生成器并将其展平为(rid, signature)的数据帧。

步骤 S4：生成候选对。该步骤将步骤 S2 中产生的 R 上的修剪签名索引与步骤 S3 中的签名索进行连接。然后我们消除重复的(s, r)对。输出是(ridleft, ridright)的 dataframe。最后，我们将这个 dataframe 与左右表连接起来，每一记录都是要验证的候选对。

步骤 S5：验证候选对。该映射步骤通过调用相似性函数来验证候选对，并输出满足给定阈值 θ 的对。

5.4. 成本分析

广播连接的总成本与输入 $O(|R|)+O(|S|)$ 的大小成正比，其中 R 为参考表，S 为输入表，加上模糊连接的输出大小；而且它完全避免了洗牌。在混洗模糊连接的情况下，S4 步骤的时间成本是最昂贵的，为了控制流程的时间的成本，因为：1) 有三个连接，连接输出的大小可能比输入的大得多。2) 对中间连接的大结果进行洗牌包括大量 I/O。根据签名方案的选择性(通过右表中与左表中的一行连接的平均行数来量化，此后表示为 α)，成本可以建模为 $O(\alpha\times(|R|+|S|))$。

例如，在输入表的大小远大于参考表的大小的设置中，如果 α 为 200，连接大小大约是输入大小的 200 倍。这意味着，即使对于一个大小为 50 GB 输入表，也需要洗牌 10 TB 的数据，这不仅会占据大量的运行时间，还会导致一些节点耗尽内存。事实上，我们确实在一些数据集中看到前缀过滤签名方案的这种内存不足行为，因为它的签名代表性可能很差；而在 LSH 签名方案中没有观察到这种行为，在该方案中可以应用有效的签名修剪。

6. 实验

在本节的实验结果中主要实现了以下的目标：

1)：比较了前缀过滤与 LSH 签名方案之间的性能；

2)：研究了签名修剪技术对于前缀过滤与 LSH 的有效性；

3)：比较基于混洗和广播模糊连接方法的性能。

6.1. 数据集

本文使用了以下两个数据集，分别是城市房地产数据集 REAL1 以及另一个不同源的城市房地产数据集 REAL2，均包含大约四十余万条的房产信息，包括楼栋表、楼盘表、户表、许可表、开发商楼盘表、许可表以及栋许可表等各种类型的数据。

6.2. 混洗模糊连接性能

在这一节中，使用 LSH 和前缀过滤作为签名方案进行混洗流程，以是否运行转换规则作为变量，并且在这一节中不对签名进行修剪。在图 9、图 10 和图 11、图 12 中分别展示了以分钟为单位的运行时间以及在步骤 S4 中生成的签名对的数量。生成的签名对与运行时间都是对一个签名方案的较好的评价指标。通过实验我们发现两种现象：① 模糊连接是一种运算成本非常昂贵的运算，尤其是在运用转换规则的时候，运算成本就更加昂贵，其中有一个运行了转换规则的案例超过了 10 小时。② LSH 签名生成的签名对显著减少，并且在运行转换规则的时候花费的时间成本减少尤为显著。

6.3. 签名修剪得影响

在本节中，对所有频率超过指定截止值的签名进行修剪。并且将 LSH 的截止值设置为 50、100、150，并将修剪后的召回率与未进行修剪之前进行对比。召回率如图 13、图 14 所示。由该表可以得出，所有截止值的召回率都非常接近 1，其中因签名修剪而产生的误差可以忽略不计。签名对的运行时间和数量分别如图 15、图 16 与图 17、图 18 所示。与没有修剪的情况相比，修剪后运行时间和签名对的数量的减少十分显著。相反，当将签名的修剪运用到前缀过滤技术当中，对召回率的影响是十分巨大的，如图 19、图 20 所示。该部分实验证实了上文的推测，LSH 签名方案中固有的随机性以及签名冗余可以用来进一步提高 LSH 方案的性能。由此我们得出结论：签名剪枝方案对于 LSH 非常有效，但是并不适用于前缀过滤技术。

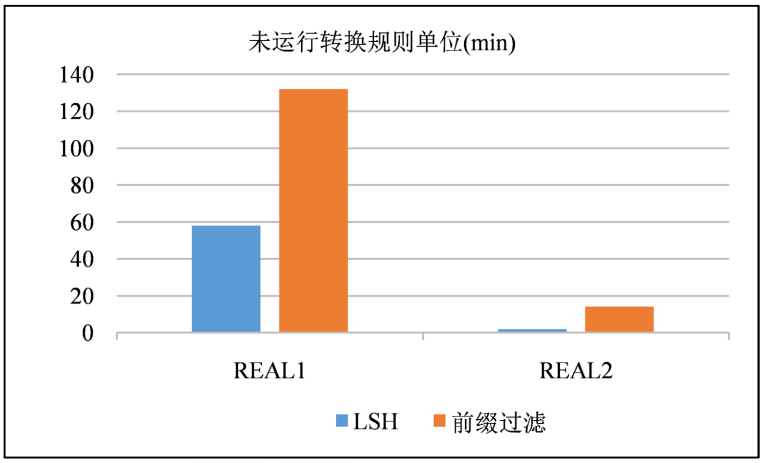

Figure 9. Comparison of time spent without transformation rules running
图 9. 未运行转换规则花费的时间成本比较

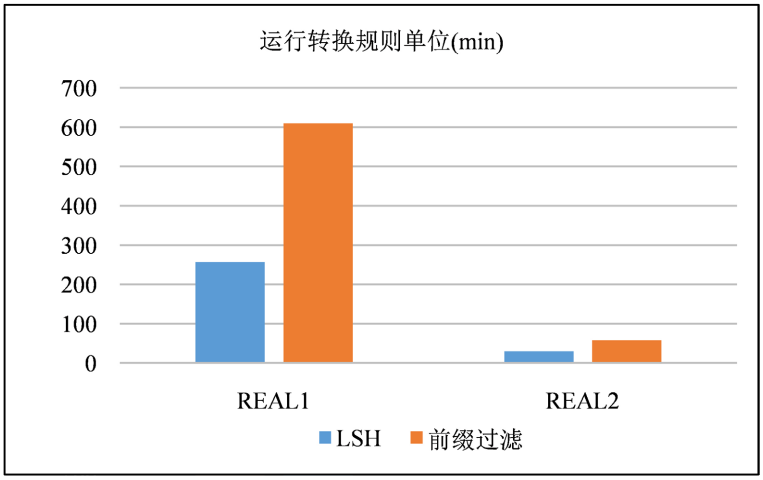

Figure 10. Comparison of time spent running transformation rules
图 10. 运行转换规则花费的时间成本比较

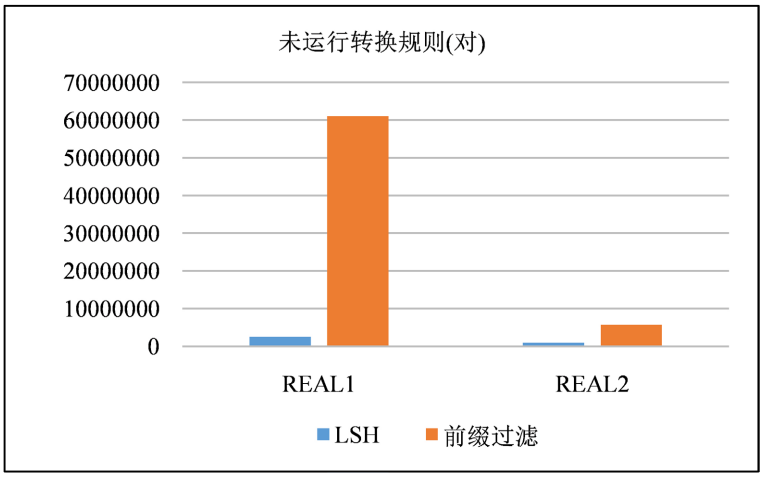

Figure 11. Signature pair count comparison without transformation rules running
图 11. 未运行转换规则生成签名对数量比较

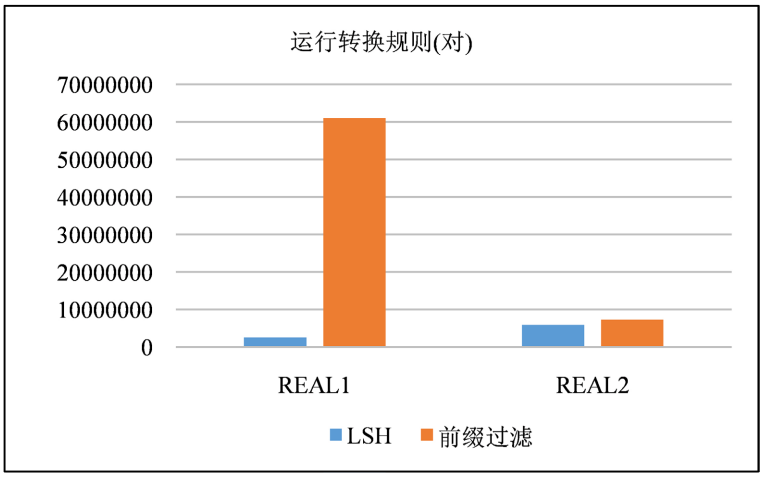

Figure 12. Signature pair count comparison with transformation rules running
图 12. 运行转换规则生成签名对数量比较

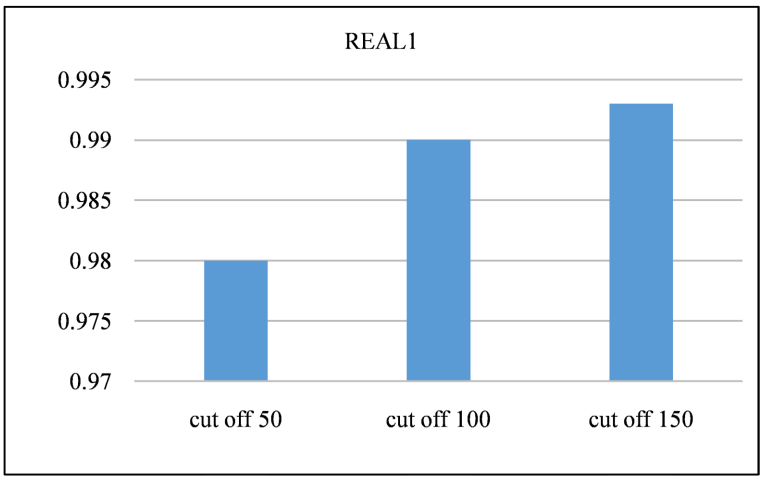

Figure 13. Recall of REAL1 dataset at different cutoff points
图 13. REAL1 数据集在不同截止点的召回率

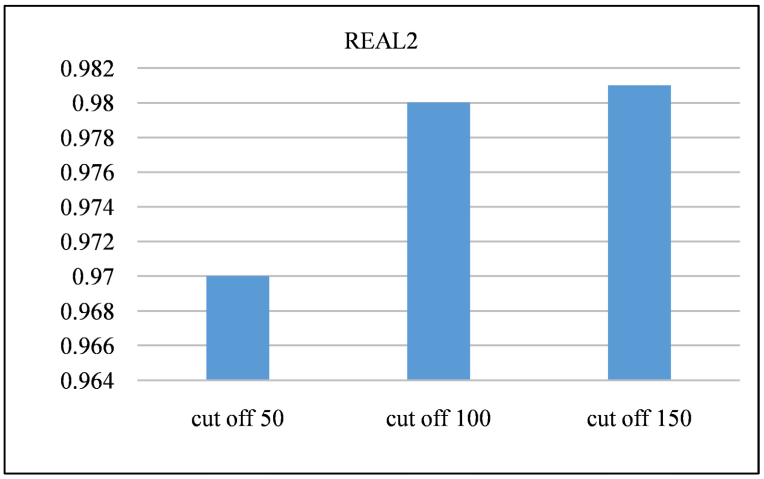

Figure 14. Recall of REAL2 dataset at different cutoff points
图 14. REAL2 数据集在不同截止点的召回率

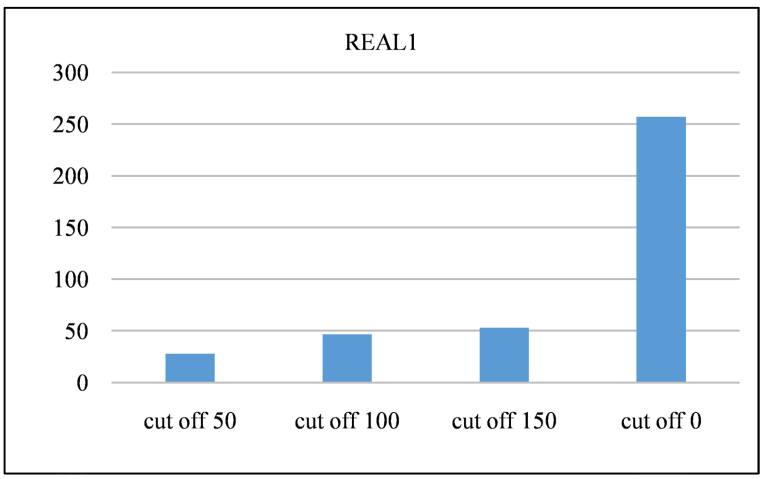

Figure 15. Time spent on REAL1 dataset at different cutoff points
图 15. REAL1 数据集在不同截止点消耗的时间

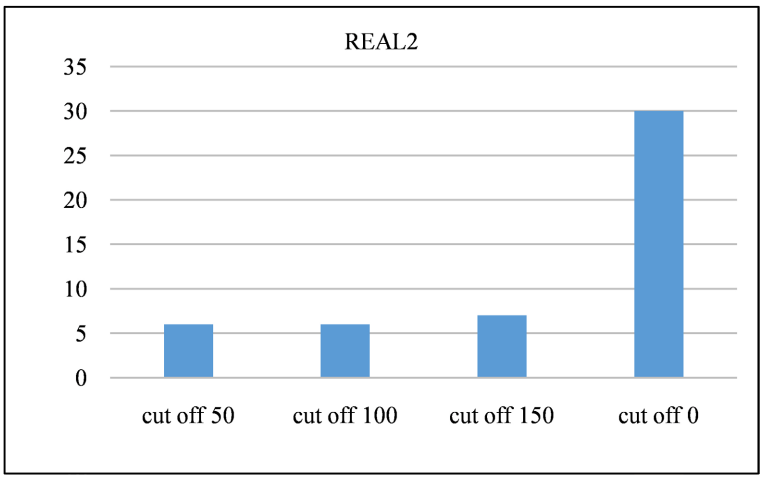

Figure 16. Time spent on REAL2 dataset at different cutoff points
图 16. REAL2 数据集在不同截止点消耗的时间

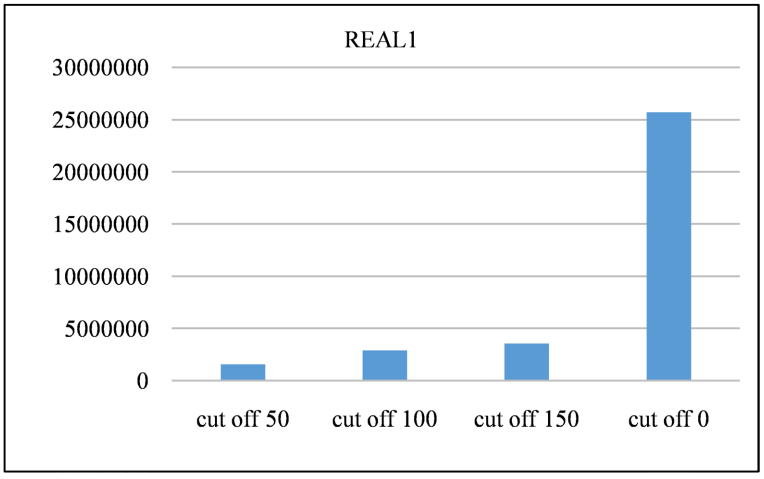

Figure 17. Signature pairs generated by REAL1 dataset at different cutoff points
图 17. REAL1 数据集在不同截止点产生的签名对

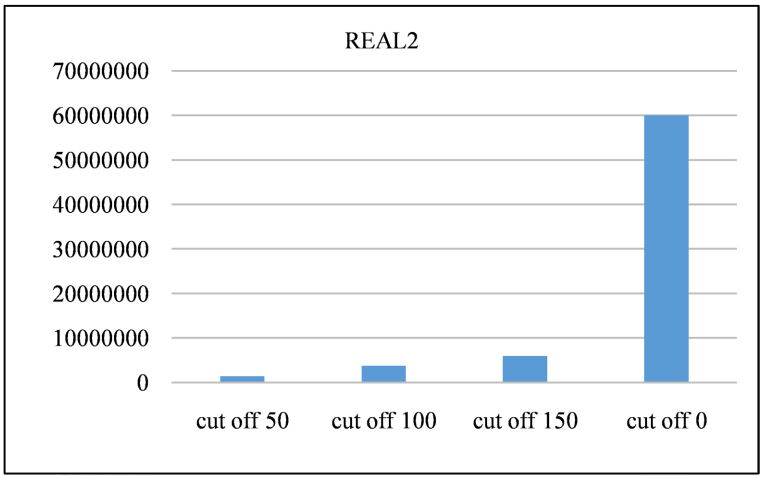

Figure 18. Signature pairs generated by REAL2 dataset at different cutoff points
图 18. REAL2 数据集在不同截止点产生的签名对

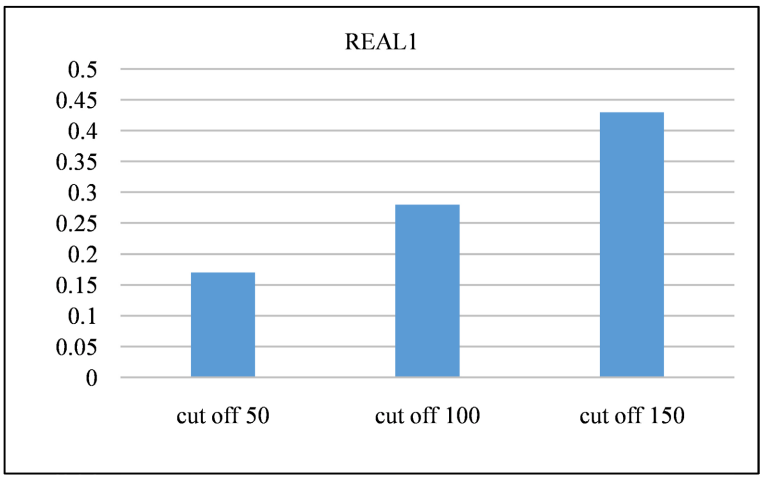

Figure 19. REAL1 dataset using prefix filtering to generate recall at different cutoff points
图 19. REAL1 数据集采用前缀过滤在不同截止点产生的召回率

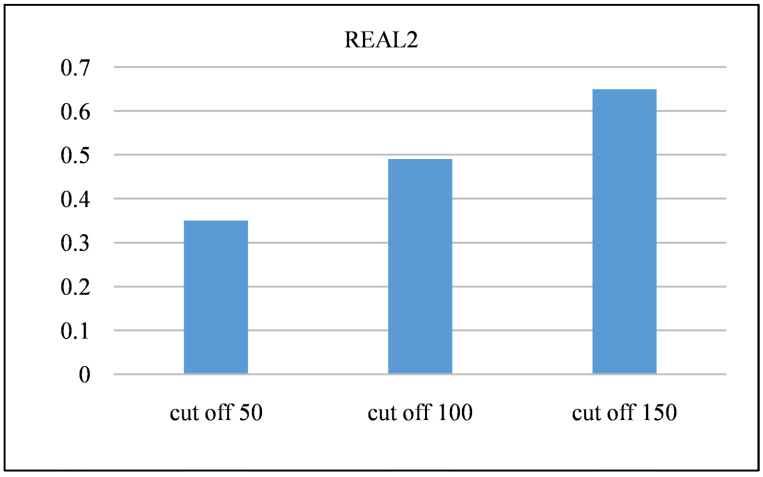

Figure 20. REAL2 dataset using prefix filtering to generate recall at different cutoff points
图 20. REAL2 数据集采用前缀过滤在不同截止点产生的召回率

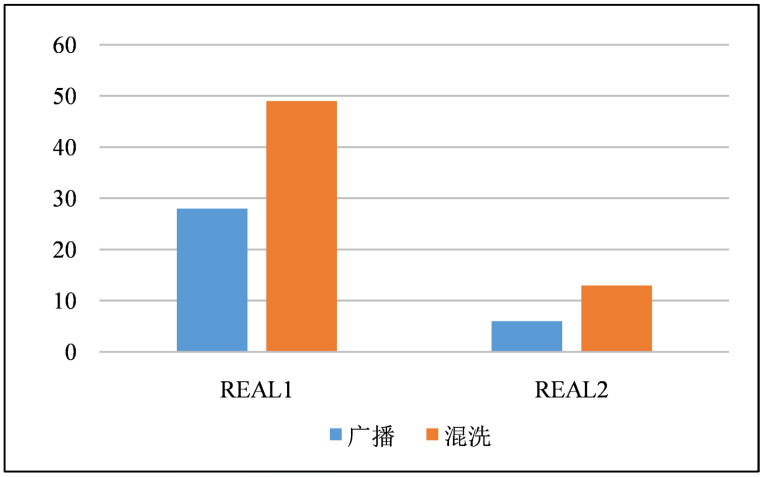

Figure 21. Comparison of time spent between broadcasting and shuffling fuzzy join
图 21. 采用签名修剪方案的 LSH 在广播与混洗模糊连接之间消耗的时间比较

6.4. 广播模糊连接与混洗模糊连接

在本节中，对模糊连接的广播和混洗版本进行了比较。两种版本均使用的是采用了签名修剪的 LSH 签名方案。图 21 所示，对于每个数据集，广播模糊连接都要比混洗模糊连接来的更快，因为我们的 VM 有足够的内存在每个工作节点的内存中保存 R 的索引。因此，在实践中，当参考表相对较小时，广播模糊连接的速度会更快。

7. 总结

本文提出了一种基于局部敏感哈希以及模糊连接的实体解析算法。该方法可以通过对转换规则的定制来展现较好的可扩展性。该方案与目前较为先进的技术前缀过滤相比展现了较好的性能，并且该方法通过签名剪枝方案进一步提升了性能。并且针对使用者内存大小的不同以及参考表 R 大小的不同，提出了两种解决方案分别是广播模糊连接与混洗模糊连接，当参考表 R 较小时，使用广播模糊连接方案，因为其性能比混洗模糊连接更为优越，当参考表 R 过大时，无法使用广播模糊连接，这时候混洗模糊连接成为了唯一的选择。

参考文献

[1] Koudas, N., Sarawagi, S. and Srivastava, D. (2006) Record Linkage: Similarity Measures and Algorithms. *Proceedings of SIGMOD*, Philadelphia, June 2006, 802-803. https://doi.org/10.1145/1142473.1142599

[2] Gravano, L., Jagadish, H., Ipeirotis, P.G., Srivastava, D., Koudas, N. and Muthukrishnan, S. (2001) Approximate String Joins in a Database (Almost) for Free. *Proceedings of VLDB*, Roma, 491-500.

[3] Arasu, A., Ganti, V. and Kaushik, R. (2006) Efficient Exact Set-Similarity Joins. *Proceedings of VLDB*, Seoul, 12-15 September 2006, 918-929.

[4] Sarawagi, S. and Kirpal, A. (2004) Efficient Set Joins on Similarity Predicates. *Proceedings of SIGMOD*, Seattle, June 2004, 743-754. https://doi.org/10.1145/1007568.1007652

[5] Arasu, A., Chaudhuri, S. and Kaushik, R. (2008) Transformation-Based Framework for Record Matching. *Proceedings of ICDE*, Cancun, 7-12 April 2008, 40-49. https://doi.org/10.1109/ICDE.2008.4497412

[6] Gionis, A., Indyk, P. and Motwani, R. (1999) Similarity Search in High Dimensions via Hashing. *Proceedings of VLDB*, Edinburgh, 518-529.

[7] Fier, F., Augsten, N., Bouros, P., Leser, U. and Freytag, J.-C. (2018) Set Similarity Joins on Mapreduce: An Experimental Survey. *Proceedings of VLDB*, **11**, 1110-1122. https://doi.org/10.14778/3231751.3231760

[8] Vernica, R., Carey, M.J. and Li, C. (2010) Efficient Parallel Set-Similarity Joins Using Mapreduce. *Proceedings of SIGMOD*, Indianapolis, June 2010, 495-506. https://doi.org/10.1145/1807167.1807222

[9] Chaudhuri, S., Ganti, V. and Kaushik, R. (2006) A Primitive Operator for Similarity Joins in Data Cleaning. *Proceedings of ICDE*, Atlanta, 3-7 April 2006, 5. https://doi.org/10.1109/ICDE.2006.9

[10] Baeza-Yates, R. and Ribeiro-Neto, B. (2011) Modern Information Retrieval. Pearson Addison Wesley, Boston.