Hans Journal of Data Mining 数据挖掘, 2022, 12(3), 235-245
Published Online July 2022 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2022.123024

Hans汉斯

一种基于实例的数据转换方法

薄凤羽，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：2022年6月1日；录用日期：2022年7月5日；发布日期：2022年7月12日

摘　要

Web中包含大量有用的信息，但由于它们是半结构化的，非专家用户在进行数据转换和集成时不能很好地利用。为此本文提出了一种基于实例的数据转换方法，用户只需要提供适当的输入－输出示例就可以得到所需的转换。首先，利用基于序列比对的模式距离度量方法依据用户提供的示例生成代表性示例；其次，提出了一种基于信息熵的代码分析方法，利用该方法与代表性示例结合来筛选与转换任务相关的候选函数；最后，通过函数排名将相关函数先进行列转换，再行合成与所有示例一致的数据转换程序。本文利用房地产领域数据集进行了实验评估，结果表明，该方法可以处理目前许多现有系统不支持的常见转换，并且能够实现实验系统中近80%的数据转换，其准确率远高于其他同类型系统。

关键词

数据转换，示例转换，信息熵，PBE

An Instance-Based Data Transformation Method

Fengyu Bo, Gui Li, Zhengyu Li, Ziyang Han, Keyan Cao

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

Received: Jun. 1st, 2022; accepted: Jul. 5th, 2022; published: Jul. 12th, 2022

Abstract

The Web contains a lot of useful information, but because it is semi-structured, non-expert users are not able to make good use of it in data transformation and integration. Therefore, this paper proposes an instance-based data transformation method. Users only need to provide appropriate input-output examples to get the required transformation. First, a pattern distance measurement method based on sequence alignment is used to generate representative examples from user-provided examples. Secondly, a code analysis method based on information entropy is proposed, which is combined with representative examples to screen candidate functions related to

文章引用: 薄凤羽, 李贵, 李征宇, 韩子扬, 曹科研. 一种基于实例的数据转换方法[J]. 数据挖掘, 2022, 12(3): 235-245.
DOI: 10.12677/hjdm.2022.123024

transformation tasks. Finally, the related functions are converted into rows and columns through function rankings, and then a data conversion program is synthesized that is consistent with all the examples. In this paper, we use real estate data set to carry out experimental evaluation, and the results show that this method can deal with many common conversions that are not supported by existing systems, and can achieve nearly 80% of the data conversions in the experimental system, and its accuracy is much higher than other systems of the same type.

Keywords

Data Transformation, Example Transformation, Information Entropy, PBE

1. 引言

在 Web 数据集成过程中，数据通常是半结构化的状态，而数据转换(Data Transformation)是将数据从一种表现形式变为另一种表现形式的过程。用户在进行数据分析之前，通常需要对不同来源收集的原始数据进行预处理，其中包括数据清洗、数据归约、数据转换等。研究表明，数据分析师要花费近 80%的时间在数据准备上[1]，这对用户来说不仅困难而且耗时。为了解决这种现状，本文提出了一种基于实例的数据转换方法，它允许最终用户仅通过几个示例来实现数据转换，这种方法称为范例编程(PBE) [2]。PBE 的好处[3]在于用户不必学习一种编程语言就可以自动的完成一些重复性的任务，并且用户是通过一个他所熟悉的界面将任务演示给系统的。FlashFill [4]也使用这种范例进行数据转换，取得了很大的成功。但 FlashFill 是由领域特定语言(DSL)中预定义的少量字符串原语组成，表达性具有一定的限制。与现有的 PBE 系统相比，本文提出的方法可以从更大的搜索空间(成千上万个函数)中合成数据转换程序。

1.1. 主要实例

本文引用了实际项目中的城市房地产大数据，该数据是利用网络爬虫从各城市房地产交易备案信息网站所爬取，并经过数据集成处理的真实数据。图 1 给出了一个示例数据集，来自相同列的值是高度异构的，当数据来自不同的源，或者手动输入，这种情况经常发生。如果数据分析师想要找出某个月中哪一处房产销售最好，并不能通过执行一个 SQL 语句查询找到它。再比如说分析师可能想要根据城市或区域(从地址中提取)分析销售情况，这都需要非常复杂的数据转换。

	A	B	C	D	E	F	G
1	T_Date	City	Addr	Sqar	T_Price	Name	Tel
2	2021年5月3日	大连市	中山区碧桂园东港国际	150	19942	张*淼	(0411)3991188
3	2021.02.09	大连市	万科、甘井子区	115.14	11,986	白*爽	0411-8302327
4	2020/12/9	沈阳市	汇智山湖（浑南区）	200.86	12112	王*洲	86+13834690407
5	2020.7.23	沈阳市	皇姑区华润置地	125.7	2w	陈*	024-85491653
6	2021.1.5 12:04	大连市	沈河区 龙湖·天奕	156	3.8w	张*蜂	18345978251
7	2021.6.13 17:18	大连市	沙河口区万科	97	23,500	吴*霆	15703509635
8	2021年3月26日星期二	沈阳市	浑南区龙湖·云颂	125	16000	沈*军	38549214
9	二零二一年九月二十日 星期一	沈阳市	铁西，万象首府	65	9500	周*	18135437377

Figure 1. Urban real estate transaction data with heterogeneous values
图 1. 具有异构值的城市房地产交易数据

	A	B	C	D	E	F	G
1	T_Date	City	Addr	Sqar	T_Price	Name	Tel
2	2021年5月3日星期一	大连市	中山区碧桂园东港国际	150.00	19,942	张*淼	0411-3991188
3	2021年2月9日星期二	大连市	甘井子区万科	115.14	11,986	白*爽	0411-8302327
4	2020年12月9日星期三	沈阳市	浑南区汇智山湖	200.86	12,112	王*洲	13834690407
5	2020年7月23日星期四	沈阳市	皇姑区华润置地	125.70	20,000	陈*	024-85491653
6	2021年1月5日星期二	大连市	沈河区龙湖·天奕	156.00	38,000	张*蜂	18345978251
7	2021年6月13日星期日	大连市	沙河口区万科	97.00	23,500	吴*霆	15703509635
8	2021年3月26日星期二	沈阳市	浑南区龙湖·云颂	125.00	16,000	沈*军	024-38549214
9	2021年9月20日星期一	沈阳市	铁西区万象首府	65.00	9,500	周*	18135437377

Figure 2. Standardized urban real estate transaction data
图 2. 规范化的城市房地产交易数据

本文针对的是不同字段的规范化转换，其中输入是不同字段的字符串值，输出是对应字段规范化的数据类型值。用一个例子来说明这种转换，如图 2 所示。图 2 是图 1 中的数据经过转换后得到的规范化数据，也是用户想要的最终数据形式。

1.2. 本文主要工作

为了提高数据转换的效率，减少用户工作量，本文利用基于序列比对的模式级度量方法来生成代表性示例。在生成代表性示例之后，我们需要弄清楚哪些函数可能与给定的转换任务相关，由于盲目测试所有爬取到的函数会耗费大量的精力，所以本文提出了一种基于信息熵的代码分析方法，把代表性示例作为函数参数，用该方法来筛选候选函数。之后对候选函数进行排名，目的是找到要执行的相关函数，根据执行结果合成数据转换程序，最终生成目标输出。通过实验验证了没有编程能力的用户利用该方法也能够进行相关的数据转换工作，同时减少了专家用户的精力消耗。本文做的主要贡献如下：

1) 由于用户提供的示例有限，本文在引入模式距离、元素距离等概念的基础上，利用基于序列比对的模式级距离度量方法，生成实现数据转换的代表性示例。

2) 本文提出了一种基于信息熵的代码分析方法，用于筛选与转换任务相关的候选函数。并利用 L1-Ranking 和 L2-Ranking 两种排名方法，对候选函数进行排名，根据执行结果选择相关字段的转换函数(列转换)，综合生成数据转换程序(行合成)。

3) 在房地产大数据的真实数据集上进行了实验，实验结果表明本文提出的基于实例的数据转换方法与现有的数据转换方法相比具有良好的效果。

2. 相关研究

目前在数据转换方面的代表性研究方法大致可分为如下几类：

1) 基于菜单的转换

大多数现有的数据准备系统都有常用于转换的菜单，用户也希望在这些菜单上找到合适的转换，比如 Paxata 和 OpenRefine，但是这两个系统只支持有限数量的简单类型数据转换。

2) 基于语言的转换

有的现有系统定义了自己的转换语言，用户需要学习怎么编写数据转换程序。比如 Trifacta 定义了一种叫 Trifacta 的“牧马人”语言；OpenRefine 也有自己的表达式语言。但是对于这种领域特定语言来说，它们不仅表达能力有限，而且具有局限性。

3) 基于输入进行转换

Trifacta 采用了一种 PBE 变体形式，用户可以选择输入一部分数据，然后系统给出可能的转换建议(称

为预测性交互[5])。同样的还有 Informatica Rev 和 Talend，这两个系统也可以根据所选输入建议进行数据转换。但是仅通过输入，系统不确定要使用哪个输出，而且经常会产生许多不相关的建议。

4) 使用字符串原语的示例驱动搜索进行的转换

FlashFill [3]使用 PBE 范例用于表格数据转换，以及与 FlashFill 相类似的系统 Foofah [6]、Blinkfill [7]、Flashextract [8]等，它们通过用户提供输入－输出示例，之后系统将使用简单的字符串原语来查找与示例一致的程序。尽管 PBE 范例在很大程度上改善了易用性，但是现有系统的表达能力受到了限制，不支持需要特定领域知识的复杂转换。

5) 使用搜索引擎的示例驱动进行的转换

DataXFormer [9]使用搜索引擎查找相关的 Web 表，可以从中提取转换任务所需的输出值。值得注意的是，与只使用示例的 PBE 不同，DataXFormer 需要列标题作为构建关键字查询的输入。虽然 Web 表是重要的数据源，但是 DataXFormer 缺乏综合处理复杂转换的结果的能力。

表 1 总结了相关系统之间的比较情况。

Table 1. Comparison of related systems
表 1. 相关系统的比较

系统	PBE 范例	转换方法	转换对象	局限性
Paxata	×	基于菜单的转换	各种数据源	支持有限数量的简单转换
OpenRefine	×	基于菜单的转换	列、字段	以列和字段的方式工作，对于增加新行内容表现不佳
Trifacta	√	基于语言的转换	部分输入数据	表达能力有限
Informatica Rev	√	基于输入进行转换	各种数据源	不确定使用哪个输出，经常会产生一长串不相关的建议
Talend	√	基于输入进行转换	数据库、文件	需要手工调整，编写 Java
FlashFill	√	使用字符串原语的示例驱动搜索	表格	不支持需要特定领域知识的复杂转换
DataXFormer	√	采用搜索引擎示例驱动搜索	Web 表	缺乏综合处理复杂转换结果的能力

3. 系统概述

在本节中主要展示了系统的工作框架，该系统有两个主要阶段：离线阶段和在线阶段，下面将叙述这两个阶段的主要工作内容。

离线阶段：首先从现有的资源 GitHub 上爬取了大量代码片段，索引了.Net 系统库中的所有函数，以及来自[10]变体中的映射关系。鉴于爬取到的代码片段数量庞大，最关键的一步是在需要运行时快速地找到与转换任务相关的函数。所以对函数进行离线分析，使用代码分布分析技术，筛选出候选函数(第 4 节)。映射关系与字典查找类似，是补充代码的一部分(本文不做叙述)。对于一些需要大量外部资源的复杂转换，该系统还会用到 Web 服务，因为只有 Web 服务具有复杂的特定于域的逻辑，可以执行较为复杂的转换。

在线阶段：在这个阶段，给定一个转换任务，系统使用离线阶段构建的索引快速查找相关的转换函数，然后将其用于合成与所有输入－输出示例匹配的数据转换程序。本文设计了两种排序方法，以及程序合成算法，可以有效地将相关函数组合成复杂的数据转换程序(第 5 节)，从而完成用户提交的转换任务。

系统架构图如图 3 所示。

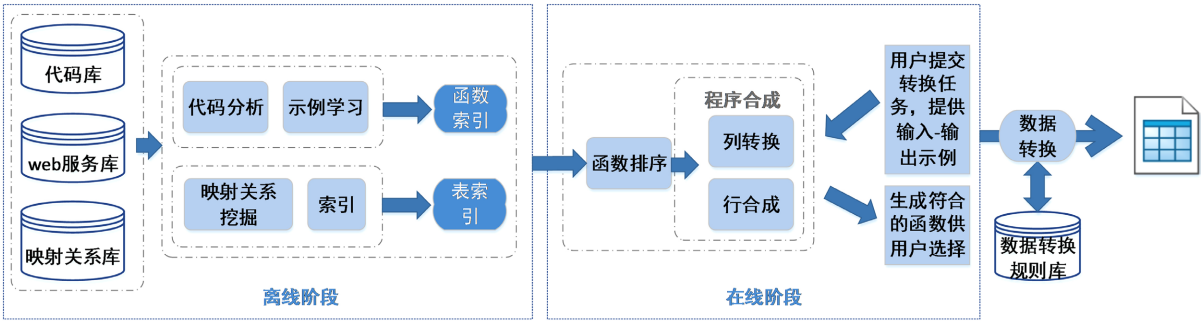

Figure 3. System architecture
图 3. 系统架构图

4. 离线函数索引

在给定大量代码的情况下，最关键的是我们要弄清楚哪些函数与转换任务相关。盲目测试所有爬取到的函数会大大降低数据转换的效率，所以我们对黑盒函数进行离线分析。先前的代码分析技术(静态编码和动态生成)利用的是代码中的现有数据，尽管有用，但是这几种技术可能会受代码编写方式的限制。因此，本文提出的代码分析技术，利用“信息论”来测试黑盒函数。首先通过序列比对的方式生成一组代表性示例 E (包括很多数据类型)，然后将 E 中的每个值作为参数传递给每个函数 f，通过观察 f 在其输出分布方面的行为，来判断适合 f 的合适输入。下面将介绍该方法的具体理论研究。

4.1. 生成代表性示例

标点符号通常充当丰富数据类型的结构定界符，因此本文通过使用标点符号对字符串进行“分段”来生成标点符号诱导序列。

定义 1 给定字符串 v，定义 s_v 是其标点符号诱导序列，其中每个元素要么是单个标点符号，要么是不带标点符号的最大子字符串。

例如，“2020/2/9”的标点符号诱导序列为{“2020”，“/”，“2”“/”，“9”}。

给定两个序列 s_u 和 s_v，根据序列比对定义它们的模式水平距离。在一般的序列比对中[11]，两个序列 s_u 和 s_v 之间的距离是通过用特殊间隙符号“-”来填充两个序列并计算的，从而产生具有相同数量元素的序列 s_u 和 s_v，以便它们可以“对齐”，并且在所有这样的 s_u 和 s_v 上最小化它们的元素距离之和，计算公式如下：

$$\sum_{i\in[|\bar{s}_u|]} d\left(\bar{s}_u[i],\bar{s}_v[i]\right) \tag{1}$$

其中 $s[i]$ 表示序列 s 的第 i 个元素。

本文采用此序列比对框架来定义模式距离，然后基于字符类定义不同的元素距离。具体来说，我们考虑三类字符：字符，数字和标点符号 P。一对元素之间的距离定义如下：

定义 2 两个元素 e，e'之间的距离为 $d(e,e')$，则

$$d(e,e')=\begin{cases}1, & \text{if } e=\text{“-”} \text{ or } e'=\text{“-”}\\ 0, & \text{if } e,\ e'\in P,\ e=e'\\ 1, & \text{if } e,\ e'\in P,\ e\neq e'\\ 1, & \text{if } e\in P,\ e'\notin P\\ 1-\dfrac{\min\left(n_l(e),n_l(e')\right)+\min\left(n_d(e),n_d(e')\right)}{\max\left(n(e),n(e')\right)}, & \text{if } e,\ e'\notin P\end{cases} \tag{2}$$

其中“-”表示插入的特殊间隙字符；$n(e)$、$n_l(e)$、$n_d(e)$ 分别表示 e 中的字符、字母、数字总数，当 $e \notin P$ 时，$n(e)=n_l(e)+n_d(e)$。

上述定义确保了元素之间的距离 $d(e,e') \in [0,1]$。然后将两个序列的模式距离定义为它们的平均元素距离，类似于其他序列比对距离，定义如下：

定义 3 序列 s_u 和 s_v 之间的模式距离 $d(s_u,s_v)$ 定义为

$$d(s_u,s_v)=\min_{\bar{s}_u,\bar{s}_v} avg_{i\in[|\bar{s}_u|]} d(\bar{s}_u[i],\bar{s}_v[i]) \tag{3}$$

使用这种模式级别的距离，我们可以计算出相同列中的字符串之间的距离都很短，所以是“相似的”，因此我们可以选择其中一个字符串来代表每一列，该字符串即为代表性示例。

4.2. 基于熵的分布分析算法的相关理论

在获得以 E 表示的代表性示例之后，将 E 作为参数来执行所有的候选函数 F。其思想是，通过观察在 $f \in F$ 上执行 E 之后的结果分布，我们可以推断出 $E_f \subset E$ 是 f 的合适输入。具体来说，有两种方案：

方案一：函数 f 需要严格检查输入参数的有效性，如果输入值不是预期的格式，则 f 会出错(引发异常，返回空对象等)。例如，如果输入不是函数可以处理的预期日期时间字符串，比如 DateTime.Parse() 函数将引发格式错误。对于此类函数，由于 E 中只有一小部分合适的值会被执行，因此观察所有 E 组的结果会使 E_f 变得简单。

方案二：并非所有函数都会严格检查输入是否存在错误，许多函数可能不会因为输入不当而出错，但仍会返回一些结果，基于分布的分析仍将显示适合 f 的 $E_f \subset E$。具体来说，f 通常对特定类型的输入数据执行某种类型的转换，然后用适当的结果填充返回对象。例如，一个解析日期值的 GitHub 函数返回一个日期对象，该对象具有诸如 int year，int month，int day，string day-of-the-week 等等的属性，当日期字符串有效并且作为参数传递时，返回对象中的这些属性值将被正确填充。但是，如果输入值不是日期时间而是其他字符串，则该函数可能会以不同的方式终止，返回对象中的属性值将具有默认初始值，在这种情况下，年，月，日为 0，一周中的某天为空字符串。

在 f 上执行 E 之后，我们可能会看到返回了一大堆(可能是“默认”)值，然后是一小群不同的值(可能是 f 有意义的输出)。一大组相同结果(可能毫无意义)和一小撮不同结果(可能有意义)之间的划分实际上始终适用于方案一和方案二。在方案一中，我们也可以将异常视为输出。

由于这些期望的分布对应于信息论中的低熵分布[12]，因此我们使用熵来识别此类函数 f 及其对应的 E_f。分布 X 的熵定义为：

$$H(X)=-\sum_i^n P(x_i)\log P(x_i) \tag{4}$$

其中 $P(x_i)$ 是第 i 个结果 x_i 在 X 的概率。因为熵对于具有更大域的变量(例如，字符串/布尔型)会自然变大，因此我们使用归一化使之对域不敏感，用归一化熵定义为：

$$N(X)=-\frac{\sum_i^n P(x_i)\log P(x_i)}{\log n} \tag{5}$$

$N(X)$ 小表明分布高度偏斜，可以从中相应地识别出 E_f。

这种基于熵的方法适用于多种函数，例如处理特定数据类型(IP 地址，电话号码等)的特定于域的函数，以及诸如阶乘函数(只接受非负整数)等一般数值函数。当我们推断出给定函数 f 的 E_f 之后，这些 E_f 就可以用于填充表 2 中的索引表，该索引表又将用于函数排名，接下来将进行讨论。

Table 2. Sample functional index table
表 2. 示例函数索引表

Function	Input-1	Input2	...	Output
System.DateTime.Parse()	2021 年 5 月 3 日			obj{...}
System.DateTime.Parse()	2021.02.09			obj{...}
System.DateTime.Parse()	2020/12/9			obj{...}
System.DateTime.Parse()	2021.6.13 17:18			obj{...}
...	...			...
processDateTime_1()	2020/12/9			false
processDateTime_2()	2020/12/9			2020
...	...			...

5. 在线程序合成

使用第 4 节中的技术，我们可以离线地将每个函数 f 与适合 f 的示例参数 E_f 关联。在本节中，我们讨论在线阶段的以下几个步骤：

1) 给定一个带有输入/输出示例的用户任务 T，对所有候选函数 F 进行排名以找到要执行的相关函数；
2) 利用 L1-Ranking 和 L2-Ranking 两种排名方法，对候选函数进行排名；
3) 根据执行结果选择相关字段的转换函数(列转换)，综合生成数据转换程序(行合成)。

5.1. L1-Ranking

按输入模式进行 L1 排名。该方法是基于比较函数 f 接受的输入参数的模式和任务 T 的输入值的模式。令 $T.I$ 为任务 T 的输入示例，函数 f 的输入排序为

$$R(f,T)=\min_{v\in E_f,u\in T.I} d(v,u) \tag{6}$$

因为我们使用距离，所以得分越低表示匹配越好。这种排名方法适用于各种语义数据值，例如 IP 地址、电话号码、电子邮件、日期时间、邮政地址等，其中大多数都有固定标点序列的结构化模式，基于模式的排名可以很好地识别。

按输入－输出关系对进行 L1 排名。该方法适用于语法转换的一类函数，例如用于驼峰/标题换行的函数，或用于修剪空白或删除连续空白的函数。对于此类函数，输入可以是没有独特模式的任何字符串。具体来说，使用英文字母中的标准字符分组(例如，小写/大写字母，字母，数字，字母数字等)，我们“描述”函数 f 的输入/输出之间的语法差异。例如，对于修剪空白的函数，一种描述其输入/输出差异的简洁方法是从输入中删除空白，而其他类别的字符不受影响(描述为{删除：{“ ”}，插入：∅，更新：∅})。

然后给定用户任务 T(例如标题框)，我们可以类似地将其输入/输出差异简要描述为{删除：∅，插入：∅，更新：{小写→大写}}。可以通过计算删除/插入/更新类别之间的平均相似性得分，并相应地对函数 f 进行排名，从而将 T 的描述与每个 f 的描述进行比较。对于这种 T 函数(例如上壳体、骆驼壳体)将排在较高的位置，而空间修剪函数将排在较低的位置。

两种 L1 排序方法是相当互补的(一种侧重于处理语义数据的函数，而另一种则更适合于语法函数)，在转换任务中常常从两个排名中获取 top-K 函数的并集。

5.2. 列转换

R_K 表示 L1 排序器返回的前 K 个函数，T 表示转换任务，T 中总共提供了 m 个(通常为 3 个)输入/输出示例。令 $T.I[i]$ 为第 i 个输入单元，$T.O[i]$ 为第 i 个输出单元。目标是使用 $T.I[i]$ 作为输入执行 R_K 中的函数，为所有 $0 \le i \le m$ 生成目标输出 $T.O[i]$。

我们首先考虑使用 $T.I[i]$ 作为输入执行 $f \in R_K$，以产生目标 $T.O[i]$，$\forall i \in [m]$。其中，目标 $T.I[i]$ 是字符串，而函数 f 的输出可以是字符串，但在一般情况下，函数返回面向对象语言的对象。但是即使我们对 T 具有正确的函数 f，f 的输出也不可能与目标输出完全匹配。这里的关键是自动合成程序把结果对象转换为目标输出字符串。

拿转换图 1 中的销售时间任务 T_{date} 来说，由于函数 System.DateTime.Parse( )对于 T_{date} 排名较高，因此我们将使用 $T_{\text{date}}.I[i]$ 中的输入作为参数来执行它，这将导致 DateTime 对象。我们使用一种称为反射[13]的编程语言机制，该机制允许我们以编程方式访问每个活动对象并在运行时遍历其成员属性和方法。具体来说，如图 4 所示，我们能够使用此反射来“转储”每个返回的 DateTime 对象的所有成员属性的值，其中包括几十个属性，例如 Year，Month 等。此外，我们可以使用反射来枚举 DateTime 对象中的成员方法，例如 ToLongDateString( )和 ToUTC( )。由于这些都是无参数方法，因此我们可以再次从每个返回的 DateTime 对象中调用以产生更加丰富的中间结果，如图 4 所示。

给定中间表具有从输入值派生的丰富语义信息，从而“组合”其中的位和片段以产生目标 $T.O[i]$。

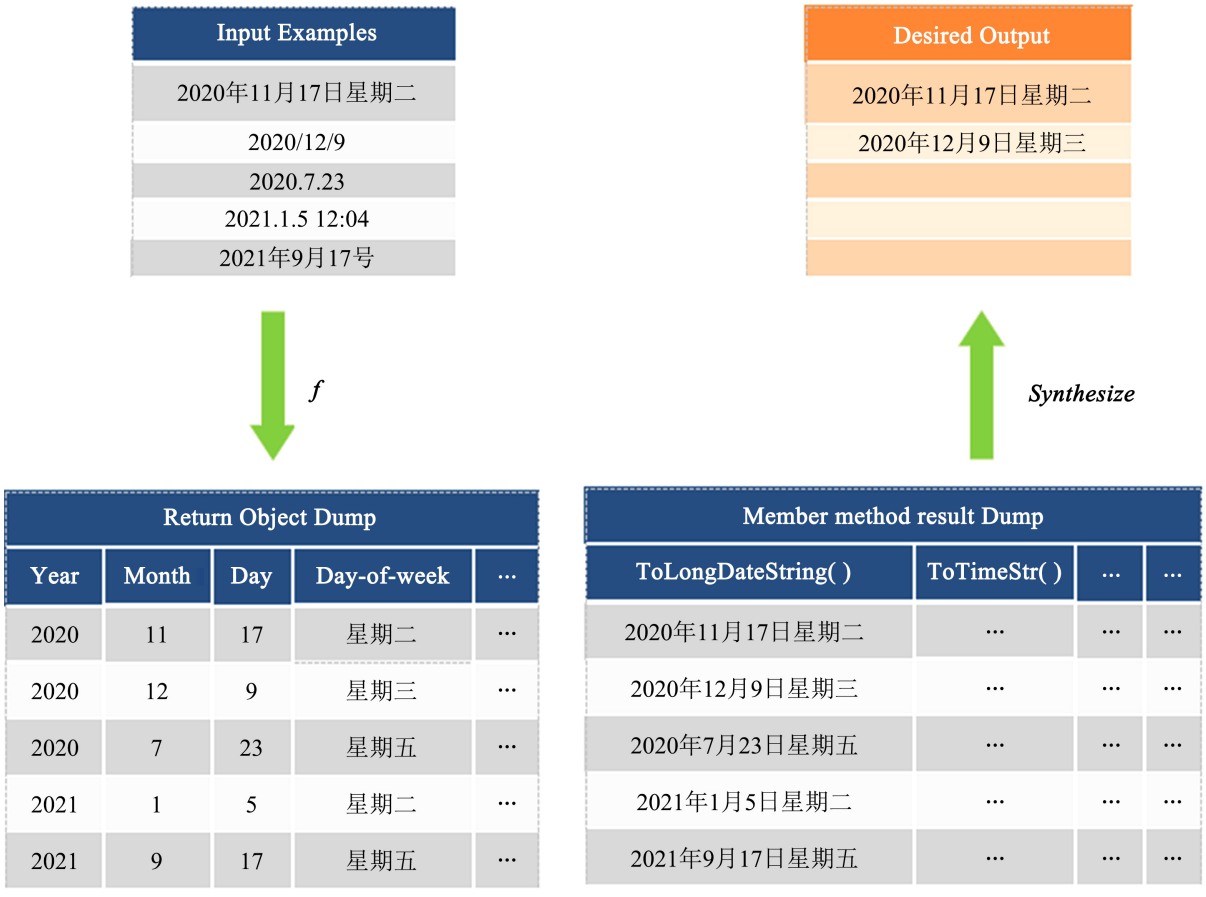

Figure 4. Reflection dump process of T_{date}
图 4. T_{date} 反射转储过程

5.3. L2-Ranking

给定任务 T，由于程序合成过程利用了 R_K 中的许多函数，因此它可能会产生与所有给定示例一致的多个程序。为了适当地排序合成程序以供用户检查，我们引入了另一级别的排名，称为 L2 排名。在 L2 排名中，我们使用 L1 排名中不可用的执行信息来对合成程序进行重新排名。

我们的观察结果是，一个数据转换程序的复杂性通常是一个很好的指标，它可以很好地概括 T 中的几个示例(通常只有 3 个)。例如，对于图 4 中的输入，假设目标输出是星期几，一个正确的数据转换程序利用函数 DateTime.Parse()产生目标输出。该程序总体上将(1)调用一个外部函数，(2)从结果中合成一个子字符串操作，总复杂度为 2。

5.4. 行合成

把图 1 中异构的数据转换成图 2 规范化的数据，首先需要对表中的每个字段进行数据转换，由于表中的每行记录都是由不同类型的字段组成的，因此需要用到不同的转换函数。为了转换成用户最终期望得到的规范化数据表，我们需要对表中多个字段进行列转换(第 5.2 节)，之后再进行行合成。行合成的定义如下：

定义 4 定义 $\Gamma=(T_1,T_2,\cdots,T_n)$，其中 T_n 表示第 n 个转换任务，每个转换任务都是针对列进行的，所以每个 T_n 都有其各自对应的 R_K。对 T_n 做列转换，把与 T_n 所有相关的 $f\in R_K$ 进行程序合成，合成一组与 Γ 对应的组函数 $\Lambda=(f_1,f_2,\cdots,f_n)$。

6. 实验与分析

6.1. 数据集

本文采用的数据集是从中国土地网、安居客等房地产网站爬取的关于房产销售方面真实数据，经过数据清洗、无关信息删除两个预处理过程，数据信息属性见表 3 所示。为了验证本文提出的方法的可行性和有效性，我们还构建了关于房地产销售方面的数据转换任务实例，以供我们进行实验研究。

Table 3. Property description of real estate information data sheet
表 3. 房地产信息数据表属性说明

序号	列名	数据类型	说明
1	T_Date	varchar	交易日期
2	City	varchar	城市
3	Addr	varchar	楼盘地址
4	Sqar	double	房子面积
5	T_Price	varchar	交易价格
6	Name	varchar	客户姓名
7	Tel	varchar	联系方式

6.2. 实验设置

大多数任务有 5~6 对输入/输出示例，对于所有被测试的系统，我们使用 3 对输入/输出作为训练示例，并将其余的作为测试用例进行测试，以验证数据转换合成程序是否正确。只有当一个程序的所有输出都

符合基本事实时，它才被认为是正确的。在本文中，我们测试每个系统的平均精度，定义为解决的任务数量除以总任务数量，如下列公式所示：

$$\text{Precision} = \frac{\text{解决的任务数量}}{\text{总任务数量}} \times 100\%$$

本文提出的 L1-Ranking 排名方法是从索引到的所有函数中选择一组有希望的函数，它是判断系统与用户交互性能的关键组成部分：排名越好，程序合成的速度就越快。

6.3. 实验结果及分析

图 5 显示了在同一数据集下，本文原型系统 TDE 与其他系统的比较结果。总的来说，TDE 为近 80% 的转换任务生成了预期的合成程序，大大优于其他系统，因为 TDE 利用了其他系统所没有的各种特定于领域的强大功能。

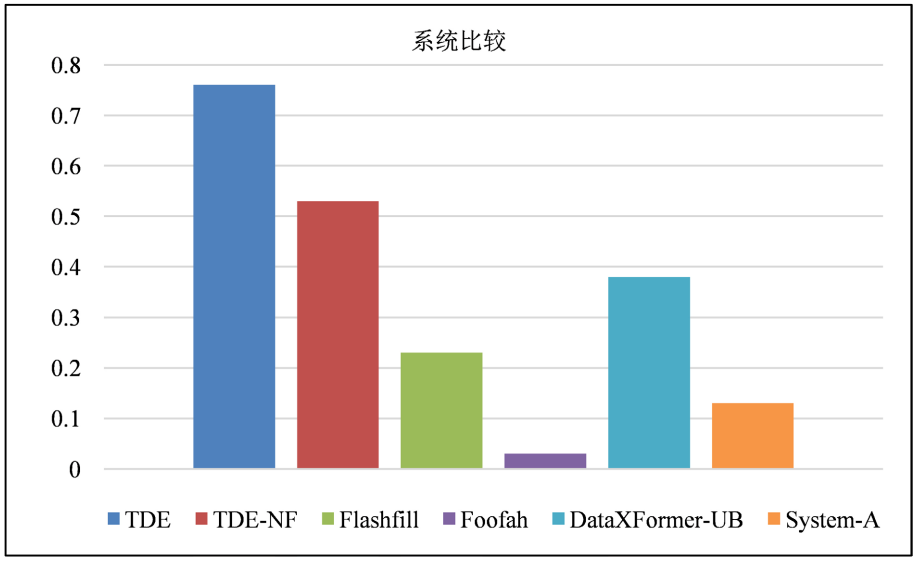

Figure 5. System comparison chart
图 5. 系统比较情况图

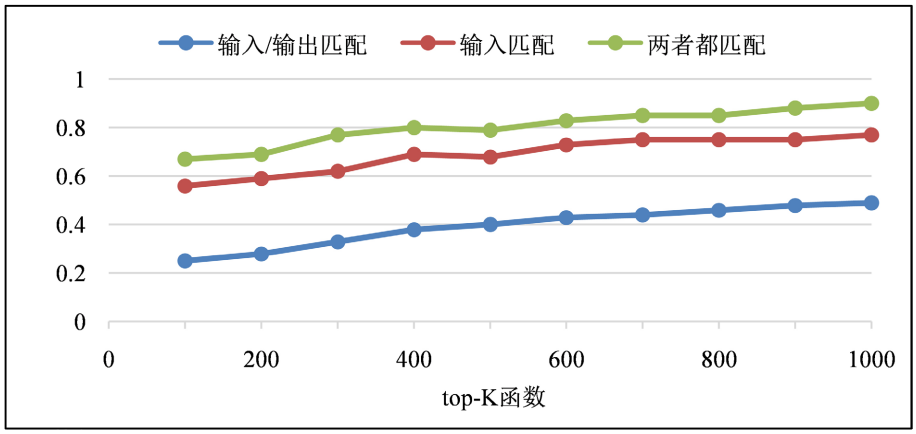

Figure 6. Validity test of L1-Ranking
图 6. L1-Ranking 的有效性测试

图 6 评估了两种 L1-Ranking 方法的有效性，其中 y 轴显示仅使用 L1-Ranking 的 top-K 函数可以解决的任务百分比，x 轴显示数量 K，K 会影响响应时间。从图中我们可以看到，两种 L1-Ranking 方法是互补的，并且它们结合起来要好得多。总体而言，使用 top-200 的函数可以解决大约 70%的情况，而使用 top-1000 的函数可以解决 90％的情况(对应于我们计算机上约 5 秒的响应时间)。

7. 结论

本文首先通过用户提供的原始数据，利用序列比对的方式生成代表性示例，并通过基于信息熵的代码分析方法筛选候选函数，之后通过 L1-Ranking 方法对候选函数排名，并针对给定的数据转换实例综合生成 Web 表的字段(列)和记录(行)数据转换程序。实现验证了本文提出的方法在很大程度上为缺乏编程技能的用户提供了便利，能完成近 80%的转换任务，其中通过实验评估了两种 L1-Ranking 方法的有效性。

参考文献

[1] Dasu, T. and Johnson, T. (2003) Exploratory Data Mining and Data Cleaning. John Wiley & Sons, Inc., New York. https://doi.org/10.1002/0471448354

[2] Lieberman, H., Ed. (2001) Your Wish Is My Command: Programming by Example. Morgan Kaufmann.

[3] 王飞龙. PBE 技术在文本搜索中的应用[D]: [硕士学位论文]. 哈尔滨: 哈尔滨理工大学, 2007.

[4] Harris, W.R. and Gulwani, S. (2011) Spreadsheet Table Transformations from Examples. *ACM SIGPLAN Notices*, **46**, 317-328. https://doi.org/10.1145/1993498.1993536

[5] Heer, J., Hellerstein, J.M. and Kandel, S. (2015) Predictive Interaction for Data Transformation. 7*th Biennial Conference on Innovative Data Systems Research* (*CIDR*'15), 4-7 January 2015, Asilomar, California.

[6] Jin, Z., Anderson, M.R., Cafarella, M. and Jagadish, H.V. (2017) Foofah: Transforming Data by Example. *SIGMOD*'17: *Proceedings of the* 2017 *ACM International Conference on Management of Data*, Chicago, May 2017, 683-698. https://doi.org/10.1145/3035918.3064034

[7] Singh, R. (2016) Blinkfill: Semi-Supervised Programming by Example for Syntactic String Transformations. *Proceedings of the VLDB Endowment*, **9**, 816-827. https://doi.org/10.14778/2977797.2977807

[8] Le, V. and Gulwani, S. (2014) FlasheXtract: A Framework for Data Extraction by Examples. *PLDI*'14: *Proceedings of the* 35*th ACM SIGPLAN Conference on Programming Language Design and Implementation*, Edinburgh, June 2014, 542-553. https://doi.org/10.1145/2594291.2594333

[9] Abedjan, Z., Morcos, J., Ilyas, I.F., Ouzzani, M., Papotti, P. and Stonebraker, M. (2016) DataXformer: A Robust Transformation Discovery System. 2016 *IEEE* 32*nd International Conference on Data Engineering* (*ICDE*), Helsinki, 16-20 May 2016, 1134-1145. https://doi.org/10.1109/ICDE.2016.7498319

[10] Wang, Y. and He, Y. (2017) Synthesizing Mapping Relationships Using Table Corpus. *SIGMOD*'17: *Proceedings of the* 2017 *ACM International Conference on Management of Data*, Redmond, May 2017, 1117-1132. https://doi.org/10.1145/3035918.3064010

[11] Gollery, M. (2005) Bioinformatics: Sequence and Genome Analysis. *Briefings in Bioinformatics*, **5**, 393-396. https://doi.org/10.1093/bib/5.4.393-a

[12] 夏源, 赵蕴龙, 范其林. 基于信息熵更新权重的数据流集成分类算法[J]. 计算机科学, 2022, 49(3): 92-98.

[13] Smith, B.C. (1982) Procedural Reflection in Programming Languages. PhD Thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.