使用 Explain-Da-V 解释语义数据版本管理中的数据集变更（技术报告）

Roee Shraga，Renée J. Miller 东
北大学 波士顿，马萨诸塞州，美国
{r.shraga,
miller}@northeastern.edu

arXiv:2301.13095v1 [cs.DB] 30 Jan 2023

摘要

在数据科学与分析采用协作方式进行的多用户环境中，同一数据集会生成多个版本。尽管数据版本的管理与存储已在研究文献中受到一定关注，但这类变更的语义属性仍未得到充分研究。在本研究中，我们介绍了 Explain-Da-V，这是一个旨在解释两个给定数据集版本之间变更的框架。Explain-Da-V 生成解释，利用数据转换来说明变更。我们还提出了一组度量指标，用于评估这些解释的有效性、可泛化性和可解释性。通过使用一个经过调整的现有基准和一个新创建的基准，我们以实证方式表明，Explain-Da-V 生成的解释优于现有的数据转换合成方法。

PVLDB 引用格式：
Roee Shraga、Renée J. Miller。使用 Explain-Da-V解释语义数据版本管理中的数据集变更（技术报告）。PVLDB，XX（XX）：XXX-XXX，2023年。doi:XX.XX/XXX.XX

PVLDB 工件可用性：
源代码、数据和/或其他工件已在以下地址提供：
https://github.com/shraga89/ExplainDaV。

1 引言

数据是任何决策过程中的最重要要素之一。数据的数量和规模不断增长，数据集也被重复用于多种分析。数据可能存储在不同的系统中（例如数据湖 [77]），格式各异，也可能包含或不包含元数据。数据项目通常涉及多个用户，他们共同或独立地处理数据集，从而创建出不同的数据版本。因此，数据版本管理成为数据管理中的重要组成部分 [21]。然而，即使版本得到了良好的管理 [22],，相关文档也可能十分表面化，例如嵌入文件名中，这往往非常不充分。此外，协作本身也可能缺乏结构或管理不当，每个用户都可能对数据执行不同的、通常是未记录的处理步骤 [55, 60–62, 106]。例如，如果数据中存在重复或缺失的信息，一些用户可能会通过删除行或列来清理数据。其他用户则提取特征，通过转换当前数据来创建新的列。

当前，工具对数据版本管理的支持有限 [60]。一般而言，与代码相比，数据的记录可能不够完善 [61, 106]，而且数据变更即使有记录，也很少附带有用的描述，因此难以理解这些变更 [55]。在紧密的协作团队中，可能会共享包含转换代码的笔记本，但组织之间很少这样做。例如，可以考虑开放数据门户上共享的重要数据集的众多版本 [42, 43, 56]，其中通常不会共享变换过程。版本描述文档不足，导致使用这些数据的用户的可复现性和信任度降低 [62, 106]。尽管文献已经关注数据版本的管理和存储 [21, 22, 57, 88, 105]，但此类变更的语义属性仍未得到充分探究。我们通过以下示例阐述开展这项工作的动机。

a0	a1	a2	a3	a4
m1	The Godfather (A)	175	9.2	Drama
m2	Hamilton (PG-13)	160	8.6	Drama
m3	The Avengers (UA)	143	8.0	Action
m4	Inception (UA)	NaN	8.8	Action
m5	Moana (U)	107	7.6	Animation

a0	a1	a2	a3	a4	a5	a6	a7	a8
m1	The Godfather (A)	175	9.2	Drama	A	2.91	4	17
m2	Hamilton (PG-13)	160	8.6	Drama	PG-13	2.67	3	16
m3	The Avengers (UA)	143	8.0	Action	UA	2.38	3	17
m5	Moana (U)	107	7.6	Animation	U	1.78	2	9

(b) 由 UserB 创建的数据集版本

图1：两位用户创建的电影示例数据集版本。属性名称见示例1。

示例 *1*。图*1*展示了两个关于电影的数据集版本。为呈现真实的数据湖场景，我们省略了图中的列名。为便于阅读，*a0* 是元组标识符，*a1* 表示电影名称，*a2* 表示以分钟为单位的电影时长，*a3* 为电影指定评分，*a4* 则提供电影类型。为方便说明，我们假设底部的表（图*1b*）由 *UserB* 创建，是对顶部由 *UserA* 创建的表（图*1a*）的派生。即使为数据表恰当地命名，例如将表 *1a* 命名为 *data1_v1.csv*、将表 *1b* 命名为 *data1_v2.csv*，或者知道表 *1b* 派生自表 *1a* *[21],*，也无法帮助 *UserA* 在语义上理解 *UserB* 对表做了哪些更改，或者更重要的是，*UserB* 执行了哪些数据处理步骤。

本作品根据知识共享署名-非商业性使用-禁止演绎 4.0 国际许可协议授权。请访问 https://creativecommons.org/licenses/by-nc-nd/4.0/ 查看该许可协议副本。如需进行超出该许可协议涵盖范围的任何使用，请发送电子邮件至 info@vldb.org以获取许可。版权归所有者/作者所有。出版权授权给 VLDB基金会。Proceedings of the VLDB Endowment，第 XX 卷，第 XX 期，ISSN 2150-8097。
doi:XX.XX/XXX.XX

示例 1 说明了对新数据集版本进行语义理解的必要性。为弥补这一不足，本文提供了相应的设置和新方法，用于 解释两个数据集版本之间的语义变化。具体而言，我们的目标是自动 解释（以简单易懂的方式）从一个数据集版本转变为另一个版本的步骤。例如， 图 1b 中的列 *a6* 是如何创建的？ 或者图 *1a* 中的第四行为什么被删除？请注意，变更者的意图具有主观性，因此无法真正进行逆向推断。我们的目标是向其他用户提供准确的解释，例如一组能够描述这些变化的函数。基于这一目标，我们回到最初的示例。

示例1（续）。图*2*展示了图*1b*的注释版本，用以解释这些变化。换言之，图*2*对*UserB* 所做的变更进行了逆向分析，使用户能够理解。具体而言，*UserB* 清理了包含NaN 值的行（本例中为m4），并提取了数值特征。电影认证信息（在a1中以括号标出）被提取出来，用于创建a5； 而列a6 则将a2中的电影运行时间单位从分钟转换为小时。由于电影评分（a3）的取值范围有限，*UserB*还将这些值离散化，在列a7中创建了四个评分类别。为了考察标题长度（在论文引用领域中发现的一种影响 *[44]*）在电影领域中的作用，*UserB* 新增了列a8， 用于提供a1中标题的长度。

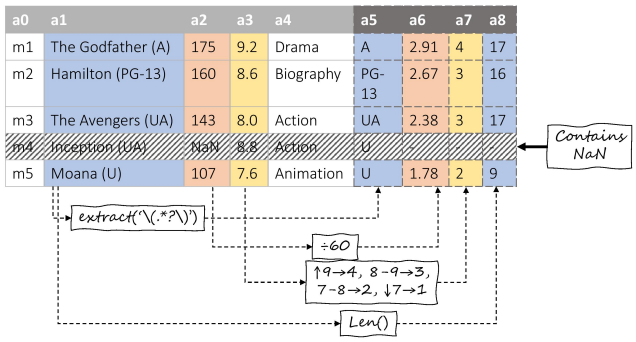


图2：对图1所示数据集版本之间变更的解读。列根据其来源着色（例如，a5 为蓝色，因为它源自蓝色的a1），列变换的注释列于底部。行变换的注释列于右侧；本例中删除了一行，图中也用覆盖该行的对角线条纹加以标示。

如示例 1所示，可能存在多种变换（例如，将数值型列的值乘以或除以某个常数，例如 a3），这可能会对数据集产生无限多种可能的变更。这些变更可以是垂直的（更改列）或水平的（更改行），可以添加信息（添加列/行）或删除信息（删除列/行），还可以涉及不同的数据类型，例如从文本型到数值型（a1 到 a8）或从数值型到类别型（a3 到 a7）。此外，用户还可以更改表中的单元格，例如，将第 NaN 行 m4 中的值替换为 146.25（该列其他值的均值），或者执行全表操作，例如转置表。变换发现方法用于多种数据管理任务，包括模糊连接 [109], 数据整理 实体整合 [25],数据整理 [45]以及更多 [18, 52, 54, 58]，主要关注文本型（文本到文本）的变换，并考虑转换后的值（而非变换本身）。我们的方法主要关注数据版本管理，因此，变换本身作为解释不同数据集版本之间变更的手段，是主要研究对象。我们所解析的变换还涵盖了涉及数值变换等多种类型的变换。“解释”一词最近变得相当常见，并且可能具有多种含义。例如，El Gebaly 等人 [47] 和 Kim 等人 [65] 都使用数据摘要作为解释。Explain3D [101], 与我们的研究具有相似背景，它通过基于语法溯源和数值的修改映射来解释数据集差异。在本文中，解释的主要组成部分是一个能够解释变更的变换。本文作出以下贡献。**(1) 语义数据版本控制定义：** 我们通过解释两个数据集版本之间的变更，定义并解决了语义数据版本控制这一新问题。

(2) 跨不同数据类型的垂直和水平数据变换解析：我们提出了一种语义数据版本控制问题的解决方案，考察涉及多种数据类型的垂直变换（添加/删除列）和水平变换（添加/删除行）。

(3) 语义数据版本控制指标：我们提供了一组评估指标，从有效性、泛化能力和可解释性方面考察解释的质量。

(4) **语义数据版本控制基准：**我们介绍了一个新的数据版本管理基准，其中包含 5 个版本集，共计 342 个不同的数据集版本，代表总计 1702 项变更。[1](5) **实证评估：**我们的实验表明，Explain-Da-V在我们提出的新版本基准和现有的数据科学流水线基准上，表现均优于多种基线方法。 [1]我们分析了解决方案中不同组件对性能的影响。

本文假设给定两个数据表，其中一个已知是从另一个数据表派生而来（即，是其一个版本），并且我们知道这两个数据表共有属性和元组之间的匹配关系。本研究聚焦于“内部”添加、删除或修改（将修改建模为先删除再添加）。外部添加，例如，查找可连接的数据表 [108] 并将其与一个数据表连接以创建新版本，则留待未来研究。

2 相关工作

据我们所知，我们是首个研究数据版本管理语义层面问题的团队。不过，已有相关研究，涵盖从合成数据转换到探索数据变更等方面。

2.1 数据版本管理

数据版本管理研究主要致力于开发版本管理器，以减少存储大型数据集的多个版本的需求 [77]。例如，DataHub 提供了类似 Git 的界面，利用有向版本图来管理、存储、重建和检索版本 [21]。后续研究以系统化的方式进一步分析了六种不同设置下重建与存储之间的权衡 [22]。最近，Schüle 等人提出了 TardisDB [88],

[1]代码和基准已公开发布 [15]。

一种用于支持版本管理的 SQL 扩展。TardisDB 在数据表上使用命名分支，以监控数据表版本并跟踪其修改历史。相比之下，我们关注数据版本管理的语义层面，重点解释数据集版本之间的语义差异。模式版本控制也得到了研究 [85]。尽管模式可能会随时间发生变化 [94],这为数据变更提供了语义提示，但我们假设元数据并不总是完整的，并且可能存在歧义 [77]。因此，我们仅关注数据本身的版本管理。

有多种方法可以查找或发现相关表 [33, 41] （例如可连接的 [108] 和可并集的 [63, 78]）。虽然表的不同版本可能存在关联，并且可以通过这些方法找到，但我们的工作假设相关表的发现工作已经完成，旨在为版本之间的差异提供语义解释。

2.2 数据变更、差异与集成

我们假设版本的属性和元组之间存在某种匹配关系。这一假设源于多年的数据集成研究，这些研究探索了属性匹配（模式匹配）[84, 90], 、元组匹配（实体解析） [48, 70], 以及其他方法 [20, 64, 73]。早期研究探讨了结构化数据中的变更和复制检测 [34, 35], , 之后又将其扩展到 XML 等半结构化文档 [38, 79, 102]。

Bleifuß 等人认识到数据变更， [23] 并设想了能够以交互方式探索此类变更的系统。他们提出了一个用于描述变更内容、位置、时间和方式的模型，并将其称为变更立方体，利用时间序列聚类等方法监控变更随时间演变的历史 [26]。DBChEx [24] 是一种利用一组探索原语来探索数据和模式变更的工具。虽然性质相似，但这项工作主要关注如何探索变更，旨在回答诸如“最近几分钟内发生了多少次变更？”以及“表 Y 中的实体有多旧？它们上次更新是什么时候？”等问题 [23, 24]。我们的工作关注版本之间的局部变更，以及这些变更是如何执行的（应用了哪些变换？），例如，UserB 如何根据图1b 中的图1a创建了图中的表。另一个相关研究领域是解释查询答案 [72, 87]。给定一个查询和数据库，研究人员利用给定数据库中的元组来解释查询答案，例如使用数据溯源 [36]。相比之下，我们使用数据转换来解释数据集变更。虽然可以将变更视为一组查询，但涉及非平凡转换的查询的解释（例如，在 Example 1中添加a5 ）不能仅使用元组来解释。

2.3 示例驱动的数据转换

我们介绍的最后一类相关研究旨在自动转换数据。总体而言，给定输入表和输出表（数据集）或它们的子集（示例），这类方法的目标是找到一种变换（程序），使其应用于输入后能够得到输出。值得注意的是，早期研究将这一问题称为查询逆向工程 [80, 98], , 其思路大致相同，即找到一个利用输入生成输出的查询。这类研究可分为两大类。

第一组根植于一种称为基于示例的编程（PBE）的范式 [25, 50, 52, 58, 59, 92, 93, 109], , 其目标是合成一个程序，处理给定输入以获得给定的输出。为此，各种方法设计不同的搜索空间（应用于输入的算子），并采用不同的搜索算法。例如，Foofah [58]使用诸如删除列（drop）和拆分列（split）（按某个分隔符分隔一列）等算子构建搜索空间，并使用 A*启发式搜索在该空间中进行搜索。CLX [59]还将字符串模式（如正则表达式）和分词引入搜索空间。Data Diff [95] 采用搜索方法对变换进行“修补”，总结分布变化，其中包括一种支持线性变换的数值补丁（算子），其参数为预定义（随机选取）的参数。Muller et al. 使用类似的搜索方法，通过他们称为“更新距离”的指标 [75] 描述关系数据库之间的差异。最后，Bogatu et al. 引入函数依赖来导航搜索空间 [25], , 我们在工作中也采用了这一方法。

第二类研究侧重于从 Web 表单、知识库、 [18, 81], GitHub 和 Stackoverflow [53, 54]等外部来源创建变换代码仓库。Transform Data by Example（TDE） [53], 并不在预先定义的可能操作空间中进行搜索，而是创建一个搜索引擎，从 GitHub 和 Stackoverflow 抓取变换函数。TDE 不采用 A* 等启发式搜索，而是对候选函数进行排序，以找到相关函数。后来，TDE 扩展为支持基于模式的变换搜索 [54]。DataX-Former [18] 和 Proteus [81] 则创建一个数据表代码仓库，可从中提取所需的输出值。

另一条相近的研究路线致力于解决数据准备和分析变换问题 [19, 103, 104]。AutoPandas [19]专注于 Pandas 库 [14] , 旨在利用 pandas 函数合成程序。Auto-pipeline [104] 将“按示例编程”范式扩展为“按目标编程”，这意味着用户提供的输出不一定与输入对齐，可能需要执行表重塑操作（例如分组）。Auto-pipeline 有两种变体：搜索（与上述方法等效，但有所扩展）和深度强化学习。前者可以作为我们方法的候选基线；后者需要训练数据，而我们假设在当前场景中不存在此类数据。

与基于示例的编程和查询逆向工程不同，我们不专注于匹配输入与输出。我们的搜索并非只关注成功率（即变换是否有效），而是遵循创建有效、可泛化且可解释的变换这一原则。在我们的方法中，这些变换可以是多维的（添加/移除属性/元组），并且适用于多种数据类型（例如数值型、类别型和文本变换）。我们的基于字符串的变换解析基于扩展后的 Foofah；此外，我们还支持使用可解释的机器学习算法进行数值变换，以拟合适当的变换，而非在庞大的可能变换空间中搜索。我们进一步扩展了 Foofah 的能力，以支持文本到数值的变换（例如测量字符串长度）和文本清理操作（例如停用词移除和词形还原）。

3 语义数据版本管理

数据集由表 T 表示，该表由一组属性 $T_A = \{A_1、\ldots\ldots、A_n\}$ 和元组 $T_r = \{r_1、\ldots\ldots、r_m\}$ 组成。每个元组定义为 $r_i = r_{i0}，r_{i1}、\ldots\ldots、r_{in}$，其中 r_{i0} 是元组标识符，$r_{ij}(j \neq 0)$ 是分配给元组 r_i 中属性 A_j 的值。

我们通常可能有两个数据集，并且知道其中一个是由另一个派生而来，但实际的转换代码或文档已经丢失 [55, 60–62, 106]。在下文中，我们将讨论解释两个数据集版本之间变更的问题。

给定两个数据集版本 T和 T'，不失一般性地，我们假设后者是一个派生表，即用户更改了表 T，从而得到表 T'，其包含 $T'_A = \{A'1, \ldots\ldots, A'n\}$以及元组 $T'r = \{r'1, \ldots\ldots, r'm\}$。我们假设已给定 T_A和 $T'A$之间的对齐关系（属性匹配，记为 Σ_A），并且假设 T 和T'中具有相同标识符（r_{0i}和 r'_{0i}）的元组表示同一个现实世界实体。如果属性 $A_i \in T$（或 $A'_j \in T'$）未出现在 Σ_A中，则认为该属性未匹配。如果记录 $r_i \in T$（或 $r'_j \in T'$）在 T'中（分别在 T中）不存在具有相同标识符的记录，则认为该记录未匹配。

给定一个属性匹配 Σ_A，我们使用三符号表示法来定义两个数据集版本之间需要解释的变化。第一个符号表示数据集是左侧数据集（L）还是右侧（修订）数据集（R），第二个符号表示它是匹配的（∇）还是未匹配的（未匹配也称为增量（Δ）），第三个符号表示属性（A）还是元组（r）。具体而言，$L\Delta_A$（左侧增量属性）和 $L\nabla_A$（左侧匹配属性）分别是 T 中未匹配（增量）属性和匹配（一致）属性的集合。同样，$R\Delta_A$ 和 $R\nabla_A$ 分别是 T'中的未匹配属性和匹配属性。利用这些集合，我们创建投影元组。设 r_j 是表 T 的一个元组，则投影元组表示为 $\pi_{L\nabla_A}[r_j]$，其中投影去除了不匹配的属性。给定这样的投影元组，我们可以为元组定义类似的集合，即 $L\Delta_r$（左侧增量元组）、$R\Delta_r$（右侧增量元组）、$L\nabla_r$（左侧一致元组）和 $R\nabla_r$（右侧一致元组）。我们在表 1 中总结了这种表示法。直观而言，我们关注的是解释数据集之间的增量，即 $L\Delta_A$、$R\Delta_A$、$L\Delta_r$ 和 $R\Delta_r$。

示例 2。给定图 *1a(T)* 和 *1b(T′)* 中的数据集版本，属性匹配只需通过对齐列标题即可得到（例如，a 2 ↔ 2 0 1 a）。元组标识符列于a下方（例如，mm）。变更集如下：$L\Delta_A = \emptyset$（无删除列）、$R\Delta_A = \{$a5、a6、a7、a8$\}$（四个新增属性）、$L\Delta_r = \{$m4$\}$（一个删除元组）以及 $R\Delta_r = \emptyset$（无新增元组）。

表 1：本文使用的符号说明。变更 $L\Delta_A$、$L\nabla_A$、$L\Delta_r$ 和 $L\nabla_r$均相对于左侧表 T定义。将下方的 T替换为 T'即可得到右侧符号。

	符号	含义
Basic	T	左侧数据集
	T'	右侧（修订）数据集
	T_A	数据集 T的属性集
	T_r	数据集 T的元组集
Changes	$L\Delta_A$	T 中的未匹配属性集合 $\{A_i\colon\ A_i \in T_A \wedge \nexists A'_j \in T'\colon (A_i,\ A'j) \in \Sigma_A\}$
	$L\nabla_A$	T 中的匹配属性集合 $T_A \backslash\ L\Delta_A$
	$L\Delta_r$	T 中的未匹配元组集合 $\{\pi_{L\nabla_A}[r_j]\colon\ r_j \in T_r \wedge \nexists r'_j \in T\colon\ r_{0i} = r'_{0i}\}$
	$L\Delta_r$	T 中匹配元组的集合 $\{\pi_{L\nabla_A}[r_j]\colon\ r_j \in T_r\} \backslash L\Delta_r$

3.1 变更解释

我们使用解释一词来指代一种以用户友好的方式解读两个关系之间变化的方法。直观地说，解释是一个将源 O 变换为目标 G 的变换 P。形式上，解释 E 是相对于一个目标G 定义的，其中目标的名称为 G*name*，并由关联关系 $G_{relation}$ 表示。与目标一样，源也关联有一个名称（O*name*）和一个关系（$O_{relation}$）。变换 P 是一个将源关系 $O_{relation}$ 变换为目标关系 $G_{relation}$的表达式。源关系也可能为空。形式化定义如下

定义 3（解释（$\mathcal{E}$））。设$\mathcal{G}$ 为一个目标。目标$\mathcal{G}$ 的解释 $\mathcal{E}_{\mathcal{G}} = (\mathcal{O}, \mathcal{P})$ 由来源$\mathcal{O}$ 和变换$\mathcal{P}$组成，满足$\mathcal{G}_{relation} = \mathcal{P}(\mathcal{O}_{relation})$。

3.2 解释数据集变更

我们重点关注解释的两种方向，即垂直解释和水平解释。我们进一步区分移除解释和加法解释。修改可以建模为先移除后加法。解释类型的差异在于源和目标所表示的关系类型不同。

目标（$G_{relation}$）和来源（$O_{relation}$）关系是针对版本 T或 T' 定义的。具体而言，关系 $O_{relation}$ 和 $G_{relation}$ 可以是对任一 T或 T'进行的投影（属性子集）或选择（元组子集）。在垂直解释（添加或移除属性）中，相关的目标名称和来源名称（G*name* 和 O*name*）就是投影后的属性。在水平解释（添加或移除元组）中，G*name* 和 O*name* 对应于子集中的元组标识符集合。根据上下文可以明确时，我们直接用名称指代目标和来源。

示例 4。回顾图 *1a(T)* 和 *1b(T′)* 中的版本。目标 $\mathcal{G} = (a6, \pi_{a6}[T'])$ 的一个示例解释由来源 $\mathcal{O} = (a2, \pi_{a2}[T])$ 和变换 $\mathcal{P} = \pi_{a2}[T] \div 60$组成，该变换基于元组，即将 $\pi_{a2}[T]$ 中的每个元组除以 *60*。当上下文明确时，我们将此解释记为 $\mathcal{E}_{a6} = (a2, a2 \div 60)$。

回顾这些变更集，我们旨在为 $L\Delta_A$寻找纵向添加解释，为 $R\Delta_A$寻找纵向删除解释，为 $L\Delta_r$寻找横向添加解释，并为 $R\Delta_r$寻找横向删除解释。第 7.2 节给出了一个明确的问题定义，该定义依赖于解释的质量。第 7.2 节。

示例 5。回顾图*1*中的版本、图*2*中标注的变更，以及 $L\Delta_A$、$R\Delta_A$、$L\Delta_r$和 $R\Delta_r$，它们在示例*2*中定义。以下是一组可能用于解释这些变更的解释：
$E_{a5} = (a1,\ \text{extract(a1, '(.*?)'))}$ $E_{a6} = (a2,\ a2 \div 60)$

$$\mathcal{E}_{\mathtt{a7}} = \left(a3, \begin{cases} 4, & if\, 9 \le a3 \\ 3, & if\, 8 \le a3 < 9 \\ 2, & if\, 7 \le a3 < 8 \\ 1, & otherwise \end{cases} \right)$$

$$\mathcal{E}_{\mathtt{a8}} = (a1, \mathtt{len(a1)})$$

$$\mathcal{E}_{\mathtt{m4}} = (\emptyset, \mathtt{has_NaN})$$

大多数解释都不言自明（本就应该如此）。有趣的情况是$\mathcal{E}_{a5}$，它会提取括号中的值。另一个例子是水平解释$\mathcal{E}_{m4}$，其来源是空集，而该元组因NaN（相当于 *null*）的值而被移除。虽然目标是 $m4$，但我们发现的变换会移除 $m4$，而不会移除其他元组。

Explain-Da-V是一种数据驱动的[2]方法，由四个部分组成，分别对应于属性/元组的添加或删除。我们首先介绍核心解释方法（第4节），随后利用这些方法解释垂直变化（第5节）和水平变化（第6节）。

4 核心语义解释方法

我们的核心解释方法依赖于将适当的解释方法论应用于来源 O 中的数据类型和目标 $\mathcal{G}$。不同于传统的数据库属性类型（字符串、整数、浮点数等），鉴于我们分析的性质，我们考察机器学习特征类型 [89]。我们重点关注三种主要类型，即数值型、类别型和文本型（混合类型被视为文本型），这些类型刻画了Explain-Da-V所涵盖的核心变化。[3] 为了解决高度多样化的变化，我们基于不同类型的源集合，采用多种利用变化类型的方法来构建方法。相应地，Explain-Da-V可以应用于任意一对版本，无论版本之间相隔多远（即应用了多少次转换）。如果未能针对某个特定变化找到解释，则将其声明为特发性的（无法解释）。

请注意，在不同类型的变更中，垂直添加最为常见且复杂，因此本文所介绍的方法主要针对这种场景。具体而言，在介绍方法时，我们假设目标是一个包含其数据值的单一属性（需要解释的右侧属性）。此外，给定一个目标，找到其来源并非易事。暂且假设来源是原始的左侧表 T。在第4.6节中，我们将讨论一种给定目标以“找到”来源的方法。

4.1 数值变化解释

每当我们需要使用包含数值数据的来源来解释数值型目标时，就会将问题定位为<style id='1'>回归，其中源关系元组被视为相互独立的变-

表格和目标关系元组作为因变量。为了实现可解释变换，我们在线性回归的基础上构建 [30]。为了降低模型复杂度并防止过拟合 [96]，我们尝试了 Lasso 和岭正则化。

[2]数据驱动意味着我们只使用数据值（不使用元数据）。[3]Explain-Da-V 可以轻松扩展，以支持日期等其他类型。

示例 6。 图 *1b*中给出了一个数值变换，其中对a6 的解释可以通过拟合回归器$160 \cdot a2$来求解。

并非所有数值变换都可以由线性函数表示。因此，为了实现更丰富、更灵活的数值变换，我们扩展特征空间（即来源），生成额外的特征。请注意，尽管这些扩展的设计受常用数据科学和工程操作的启发 [67]，，但它们并不能（也不可能）涵盖所有可能的变换。

多项式回归和特征间交互特征：为了解释多项式变换，我们在 O 上生成额外的多项式特征 [46]。给定预定义的阶数 d，O 的多项式扩展表示为 $poly(\mathrm{O})$，其属性对应于 $\{A_i^2$、…、A_i^d、$\forall A_i \in \mathrm{O}_{name}\}$。通过应用相关操作，在元组级别创建扩展后的源关系。例如，扩展关系中元组 r_j 的属性 A_i^2 将获得属性 A_i 的平方值，即 $\pi_{A_i}[r_j]^2$。我们还引入特征间交互，即 O 中不同属性值之间的乘法和除法。请注意，使用线性回归时已经支持加法和减法。O 的特征间交互扩展 $inter(\mathrm{O})$ 对应于 $\{A_i \cdot A_j$、$A_i \div A_j \ldots$、$\forall A_i$、$A_j \in \mathrm{O} \wedge A_i \neq A_j\}$。这里的变换同样在元组级别进行，例如，元组 r_j 的属性 $A_i \cdot A_j$ 将获得值 $\pi_{A_i}[r_j] \cdot \pi_{A_j}[r_j]$。这些扩展还可以连续应用，例如 $inter(poly(\mathrm{O}))$，以创建诸如 $A_i \div A_j^2$ 的属性，从而求解例如身体质量指数公式（$kg \div m^2$）。例如，回顾图1，令 $\mathrm{O}_{name} = \{a_2$、$a_3\}$，使用 $poly(inter(\mathrm{O}))$ 且 $d = 2$ 时得到的扩展特征为：

a3	a4	$a3^2$	$a4^2$	$a3 \cdot a4$	$\frac{a3}{a4}$	$\frac{a4}{a3}$	$a3^2 \cdot a4$	…
175	9.2	30625	84.64	1610	19.02	0.05	281750	…
160	8.6	25600	73.96	1376	18.6	0.05	220160	…
143	8.0	20449	64	1144	17.88	0.06	163592	…
NaN	8.8	NaN	77.44	NaN	NaN	NaN	NaN	…
107	7.6	11449	57.76	813.2	14.08	0.07	87012	…

图 3：关于 a3和a4 图 1a的多项式和特征间交互扩展。

数学变换：在数值数据上生成新特征时，使用数学运算 [27] 也很常见。具体而言，为支持此类解释，我们为 O 生成数学扩展 $math(\mathrm{O})$，其属性为 $\{log(A_i)$ $sqrt(A_i)$ $reciprocal(A_i)$ $exp(A_i) \ldots, \forall A_i \in \mathrm{O}\}$，其中 $sqrt(A_i) = \sqrt{A_i}$、$reciprocal(A_i) = A_i^{-1}$，以及$\overline{\ }exp(A_i) = e^{A_i}$。这些变换基于元组，例如，扩展关系中元组 r_j 的特征 $log(A_i)$ 将获得值 $log(\pi_{A_i}[r_j])$。作为示例，回顾我们的运行示例，使用 $math(\{a_2, a_3\})$ 的扩展特征如下：

全局聚合： 我们还会生成聚合特征。在考察机器学习中最常见的变换之一，即（值）归一化 [107]时，这一扩展集合尤其重要。我们引入 O 的扩展 $agg(\mathrm{O})$，其中包含属性 $\{sum(A_i)$ $mean(A_i)$ $max(A_i)$ $min(A_i) \ldots, \forall A_i \in \mathrm{O}\}$。在这里，尽管这些扩展值是在元组级别上赋予的，

a3	a4	$log(a3)$	$sqrt(a3)$	$reciprocal(a3)$	...	$exp(a4)$
175	9.2	2.24	13.23	0.006	...	9897
160	8.6	2.20	12.65	0.006	...	5432
143	8.0	2.155	11.96	0.007	...	2981
NaN	8.8	NaN	NaN	NaN	...	6634
107	7.6	2.029	10.34	0.009	...	1998

图 4：在a3 和a4 图1a上的数学扩展。

在该属性的所有值上进行计算。例如，元组 r_j 的特征 $sum(A_i)$ 将获得值 $\sum_{r_{ik} \in \pi_{A_i}(T)} r_{ik}$。然后，如果用户对 A_i 应用总和归一化，则包含特征 $A_i \div sum(A_i)$ 的特征集 $agg(inter(\mathrm{O}))$ 将能够解析并解释这一新增属性。类似地，在最小-最大归一化的情况下，[107] $min(A_i)$ 和 $max(A_i)$ 的附加特征可用于生成准确的解释。另一个例子是常见的协同过滤变换，即减去均值 ($A_i - mean(A_i)$) [71]。回顾我们的运行示例，令 $\mathrm{O} = \{a_2, a_3\}$，使用 $agg(\mathrm{O})$ 的扩展特征如下：

a3	a4	$sum(a3)$	$mean(a3)$	$std(a3)$	$min(a3)$	...	$max(a4)$
175	9.2	585	146.25	25.33	107	...	9.2
160	8.6	585	146.25	25.33	107	...	9.2
143	8.0	585	146.25	25.33	107	...	9.2
NaN	8.8	585	146.25	25.33	107	...	9.2
107	7.6	585	146.25	25.33	107	...	9.2

图 5：在a3 和a4 图1a上的聚合扩展。

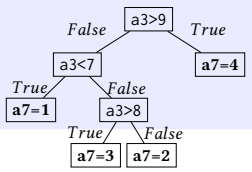
使用扩展特征集来拟合一个回归器，该回归器为扩展特征集分配系数。理想的回归器应能够处理一个包含所有特征的集合（即 $poly\ inter \cup ((\ldots inter(\dot{O}))(poly((\dot{O})))$；然而，既然不能现实地期望一个可解释的回归器做到这一点，我们便独立处理每个集合（扩展项的幂集），并生成多个“可能的”解释。第 7.4节讨论如何从中进行选择。

请注意，变换仅基于两个数据集版本 T和 T' 进行“学习”（拟合），**无需额外的训练数据**。第5.1节讨论数值变化解释的主要应用，并通过多个示例加以说明。

4.2 类别型变更解释

为了使用包含数值型数据的源关系来解释类别型目标，我们将问题表述为分类问题，其中源关系的元组被视为解释变量，而目标关系元组则作为输出类别标签。输出类别可以是二元的（例如，电影是否超过两小时），也可以是多类别的（例如，a7，示例1）。为了实现可解释性，我们聚焦于决策树 [30]。请注意，决策树仅涵盖可表示为合取项析取的解释 [74]。从根节点到叶节点的每条路径都对应一个合取项，而整棵树则是这些合取项的析取。

示例7。图 *1b* 提供了一个.....的示例 类别转换，即 a7，可以通过



以下决策树加以解决。

4.3 文本型变更解释

当源关系和/或目标关系为文本型时，我们采用基于示例的编程方法（参见第2.3节），使用基于搜索的解决方案。具体而言，我们采用一个名为 Foofah 的现有框架 [58]。基于示例的编程解决方案由可能算子空间的设计和一种搜索算法构成。搜索算法（遵循 Foofah 的 A* 算法）利用启发式函数（基于表格差异度）在算子空间中进行搜索，以估算任何候选部分解决方案的代价。为提高搜索速度，搜索空间会被剪枝 [58]。

4.3.1 文本到文本。针对数据版本管理，我们扩展了传统的基于示例的编程算子，加入了涵盖常见文本处理步骤的算子 [100],，包括文本小写化、词形还原、移除特殊字符（例如标点符号和数值）以及词元（例如停用词和 HTML 标签）。Foofah 中实现的所有附加算子均已提供于我们的代码仓库 [6]。

不同于 Foofah，我们还考虑文本到数值和文本到类别型。需要注意的是，尽管近年来基于 Transformer 的模型已成为从文本中提取（潜在）特征的标准方法，但针对文本的传统特征工程，即从文本值中提取手工设计的数值型和类别型特征，在自然语言处理 [37, 40, 66] 以及其他研究领域（如人机交互 [31] 和信息管理 [51]）中仍然是重要组成部分。

4.3.2 文本到数值。用于解决这类变换的搜索空间包含一种元操作，用于统计某个模式 pat在一个值中（$\mathrm{count}^{pat}(r_{ij})$），其中$r_{ij}$是表中的一个值，参见第3)。使用此操作，我们可以定义诸如单词_数量_count‘.’(r_{ij})和问号_数量_count‘?’(r_{ij})这样的操作。我们还涵盖了对预定义的一组停用词和标点符号进行计数。

4.3.3 文本到类别。我们为模式存在性定义了一个类似的元操作（$\mathrm{contains}^{pat}(r_{ij})$），用于生成诸如包含_百分比 $=\mathrm{contains}^{\text{‘\%’}}(r_{ij})$这样的操作。

示例 8。图*1b* 展示了一个文本到文本转换的示例，即a5，该转换可以借助 *Foofah* 环境的原始实现来解决。具体而言，如果我们将a1 视为来源，*Foofah* 会将 $\pi_{a1}[T]$ 视为输入示例，并将 $\pi_{a5}[T']$ 视为相应的输出示例（用我们的术语来说即目标），并合成一个基于元组的数据转换程序(1) t = split(t,0, ‘(’), (2) t = split(t,1, ‘)’), (3) t = drop(t,0), (4) t = drop(t, 2)例如，元组值Moana (U)将按如下方式转换：[Moana , U)] →[Moana , U,] → [U,] → U。

解决a8 需要Explain-Da-V的扩展，其中包括文本到数值转换 (len()) 。

请注意，Foofah旨在使用给定的一组示例元组来合成变换，并且通常只需少量示例即可完成这一任务。相比之下，我们的目标是生成一个能够正确解释整个表转换（即数据集版本）的解释。因此，将搜索空间从文本到文本的变换大幅扩展至更广泛的范围，在其中发挥了关键作用。此外，第4.6节介绍了一种在此情境下剪枝搜索空间的技术。

4.4 类别型编码变更解释

在处理混合类型时，文本型值和类别型值可能需要进行编码。我们在此采用的一种常见编码方法是独热编码 [86]。令 $A_i \in O$ 表示来源中的一个文本型/类别型属性。对该属性进行独热编码后，会为 A_i 中的每个唯一值（或类别）生成一个附加属性，并为对应于该值（类别）的每个元组赋值 1。这种加法不仅可以实现这一常见编码方案，还能提供更丰富的表示形式，可用于实现其他类型的编码（例如序数编码）和附加变换。这类编码也常用于机器学习的数据准备 [28]。回顾示例 1，如果用户希望预测电影评分，提取是否_剧情片或是否_动作等特征会有助于模型学习。

4.5 重塑变更解释

通常，分组依据是一种表重塑操作 [104],，也就是说，自然属性匹配和元组匹配并不存在。然而，在特征工程中，分组依据也可用于基于其他属性生成聚合特征。后者的处理方式与数值变化解释类似。对于每个数值型属性，都会创建一个扩展来源，类似于第 4.1 节 $A_j \in T_A$，并针对每个文本型/类别型属性 $A_i \in O$ 分别或联合进行（依据多个属性分组）。为清晰起见，我们使用 SQL 语法。

可以使用辅助查询（图6）生成数值型属性的扩展分组依据特征 $A_j \in T_A$，针对文本型/类别型属性 $A_i \in O$。

SELECT A_i, 均值(A_j), 最大值(A_j)... FROM T 分组 A_i

图 6：分组查询

如果存在多个数值型属性，它们将被添加到 GROUP BY 和 SELECT 子句中。使用此辅助查询并将其与来源连接后，我们可以获得其他可能的属性。例如，考虑图1a中的属性 a4。其他聚合特征将在数值型属性 a2 和 a3 上生成，并连接到表中，如图 7所示。其中包括每个 a4 （运行时间）对应的 a3 （评分）和 a4 （类型）的均值，从而支持此类解释。

a3	a4	a5	*sum(a3) by a5*	*mean(a3) by a5*	...	*min(a4) by a5*	*max(a4) by a5*
175	9.2	剧情片	167.5	335	...	8.6	9.2
160	8.6	剧情片	167.5	335	...	8.6	9.2
143	8.0	动作	143	143	...	8	8.8
NaN	8.8	动作	143	143	...	8	8.8
107	7.6	动画	107	107	...	7.6	7.6

图7：在 a3 和 a4 上进行分组特征扩展，图 1a。

我们还考虑了先前工作中介绍的重塑情形，例如， [104]。重塑通常被视为搜索过程中可应用于表的一种可能操作（参见第2.3节）。我们提出了一种替代性的数据驱动方法。具体而言，仅当不存在元组匹配时，才将重塑与属性匹配相关联。我们使用图6 中的查询显式重塑该表，并在其上拟合一个回归器。目前，Explain-Da-V 尚不支持其他重塑变换，如转置和透视，我们计划在未来工作中对此进行探索。

4.6 查找来源

在此前的讨论中，我们假设变换 $\mathcal{P}$ 的来源是原始关系 T。然而，如果我们能够确定，对于特定目标，来源仅是 T的一部分，就可以提高搜索效率。因此，我们的目标是“查找来源”。不同于一些相关文献中输入—输出范围明确的情况（参见第 2.3节），在数据版本管理中，我们不知道使用了哪些数据来推导特定目标。因此，我们的方法首先搜索该变换的来源。

一种朴素的解决方案是使用所有可用的数据值。例如，在添加属性的场景中，将 T的所有属性作为来源，即 $O = (T_A, T)$。这种方法有两个问题。首先，在为目标寻找适当变换时，不相关的属性可能会成为噪声。例如，回到示例1，如果使用所有属性（a5 使用a1-a4）来解决问题，其搜索空间会比使用 a2大得多。第二个问题与数据类型有关。使用数值变化解释（第4.1节）可能比使用文本变化解释（第4.3节）更有益。例如，使用 a2 作为来源来解释 a5，而不是使用 a1 到 a4。

在考察如何从现有属性创建新属性时，我们会发现，创建函数依赖是其副作用之一：该依赖连接来源与新属性（目标）。例如，如果两部电影的分钟运行时间相同（a2），那么它们的小时运行时间也相同（a5）。根据定义，这构成了a2 与a5之间的函数依赖。因此，我们使用函数依赖发现算法 [82]来寻找来源。[4]我们寻找以目标为被决定属性集的依赖，并将发现的决定项用作来源。请注意，可能有多个属性集能够决定目标，因此也可能产生多个来源。

我们将每个决定属性集都视为候选来源，对所有决定属性集进行分析。因此，一个目标可能产生多个解释。第7.4 节将介绍如何从中进行选择。具体而言，我们根据决定项的大小和基数对其排序；如有需要，还可以根据所发现变换的大小或质量设置提前停止条件。

示例 9。 回顾图 *1*，并将属性 a7 视为目标。由于示例数据表较小，*{*a1、a2、a3、a4*}* 中的任意属性组合都可以视为来源。如果未能针对单一属性找到高质量解释，算法可以考虑属性组合。在较大的实际示例中，可能只有少数属性或其组合是来源。

5 解释垂直变化

借助我们丰富的解释方法，我们现在来探讨如何利用这些方法解释变更。我们首先讨论属性添加，随后介绍我们处理属性删除的方法。

[4]在我们的实验中，我们使用一种称为 FDEP 的发现算法 [49].

5.1 针对 $R\Delta_A$的加法解释

添加属性是数据科学中的一项非常常见的操作，主要涉及机器学习（ML）的数据准备和特征工程 [107]。新增属性通常是对现有数据应用某种变换的结果。我们首先找到来源（第4.6节），然后利用核心解释方法（第4节）来寻找应用于源关系后能够生成所需目标关系的变换。图8 展示了该方法的示意图（左侧和中间部分），并在属性添加的背景下突出显示了其主要创新点（右侧部分）。

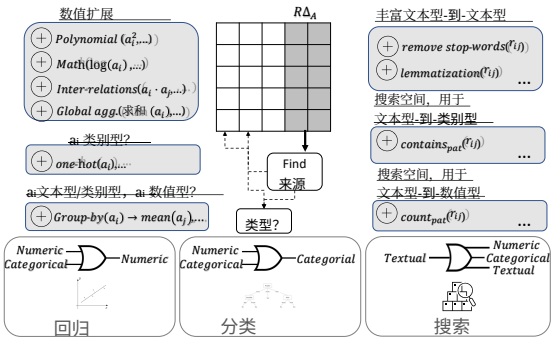


图8：解释属性添加

算法1： 属性添加解释

输入：待解释的属性集合 $R\Delta_A$
输出：每个属性的解释集合 $\mathrm{E}^{R\Delta_A}$
$\mathrm{E}^{R\Delta_A} := \emptyset$
对于 $A_i \in R\Delta_A$ **执行**
  $\mathcal{E}^{A_i} := \emptyset$; O^{A_i} :=查找来源 (第4.6节)
  对于 $O \in O^{A_i}$ **执行**
    如果 A_i 是数值型**则**
      如果O 是数值型**则**
        $\mathcal{E}^O_{A_i} \leftarrow$ 数值解释（第4.1节）
      else
        $\mathcal{E}^O_{text} \leftarrow$ 文本解释（第4.3节）
        $\mathcal{E}^O_{encode} \leftarrow$ 编码解释（第4.4节）
        $\mathcal{E}^O_{reshape} \leftarrow$ 重塑解释（第4.5节）
        $\mathcal{E}^O_{A_i} \leftarrow \mathrm{E}^O_{encode} \cup \mathrm{E}^O_{reshape}$
    否则如果 A_i 是类别型**则**
      如果O 是数值型**则**
        $\mathcal{E}^O_{A_i} \leftarrow$ 类别解释（第4.2节）
      else
        $\mathcal{E}^O_{text} \leftarrow$ 文本解释（第4.3节）
        $\mathcal{E}^O_{encode} \leftarrow$ 编码解释（第4.4节）
        $\mathcal{E}^O_{reshape} \leftarrow$ 重塑解释（第4.5节）
        $\mathcal{E}^O_{A_i} \leftarrow \mathrm{E}^O_{encode} \cup \mathrm{E}^O_{reshape}$
    否则如果 A_i 是文本型 **则**
      $\mathcal{E}^O_{A_i} \leftarrow$ 文本解释（第4.3节）
  $\mathrm{E}^{R\Delta_A} \leftarrow \mathrm{E}_{R\Delta_A} \cup \mathcal{E}^{A_i}$
返回：$\mathcal{E}^{R\Delta_A}$

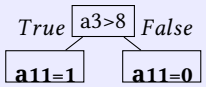
Explain-Da-V 属性添加解释算法如算法1所示。Explain-Da-V 以迭代方式运行，旨在逐一解决每个新增属性（即目标 $\mathrm{G} = (A_i, \pi_{A_i}[T_\prime])$ ），时间（第 4-25 行）。找到一组可能的来源 O^{A_i}（第 5 行）后，我们利用核心解释方法，为每个来源生成一组解释 $\mathcal{E}O^{A_i}$（第 6-24 行）。例如，如果为数值目标找到数值来源（第 9 行），Explain-Da-V 会使用数值解释方法（第4.1节）。请注意，对于每个目标都会生成多个解释；为此，我们在第 7.4 节中介绍了一种搜索策略。我们通过图 9对其进行说明，该图给出了图 1a中表的一个附加版本。

a0	a1	a2	a3	a4	a9	a10	a11	a12	a13	a14
m1	教父 (A) 175 9.2 剧情片				0.28	3.15	1	教父 a	8.9	1
m2	Hamilton（PG-13）	160	8.6	剧情片	0.28	3.23	1	Hamilton PG	8.9	1
m3	复仇者联盟（UA）	143	8.0	动作	0.24	3.36	0	复仇者联盟 UA	8.0	2
m5	Moana（U）	107	7.6	动画	0.23	4.26	0	moana u	7.6	3

图9：由UserC 基于图1a创建的数据集版本。

示例 *10*。在新增属性中，*a9* 和 *a10* 是数值型的（第 *8* 行）。属性 *a9* 是对 *a3*（归一化评分）中的值进行（求和）归一化得到的结果。*Explain-Da-V* 首先会找出其起源（第 *5* 行）。与 *a6* 的情况相同（见示例 *9*），*a9* 也可以由多个属性集确定。在可能的起源中，考虑 $O = a3$。由于 *a3* 和 *a9* 都是数值型的，*Explain-Da-V* 使用数值变化解释（第 *9* 行）。深入考察该方法（第*4.1*节），首先会通过在 *O* 上拟合回归器来生成基线解释。然后，将在该起源上应用不同的扩展，并通过拟合回归器为每个扩展关联一个解释。在生成的扩展中，我们会找到 *agg(inter(O))*，其中包含特征 $a3 \div sum(a3)$，并将对其拟合变换 $a9 = a3 \div sum(a3)$，以生成解释 $E_{a9} = (a3, a3 \div sum(a3))$，该解释将被添加到 $E\,O_{a9}$ 中。同样，*a10* 也由多个属性集确定。在这些属性集中，考虑 $O = \{a2, a3\}$，将在其上生成以下解释 $E_{a10} = (\{a2, a3\}, 60 \cdot a3 \div a2)$（每小时评分），并将其添加到 $E\,O_{a10}$ 中。

示例 10（续）。接下来我们查看 a11，它是类别型的，例如，考虑来源 O = a3。因此，Explain-Da-V 使用类别-



类别型变更解释（第*17*行），并可能会采用以下决策树来解释 a11。请注意，a11 可能是通过应用a3>*8.5* 创建的，这与 Explain-Da-V的数据驱动解释不同。不过，我们的目标是提供一个准确的解释，而这两者都是如此。有趣的是，如果我们将 a2 视为来源，就可以推导出一个类似的决策树，其根节点为 $a2 > 150$。这些问题涉及解释的可泛化性，将在第*7*节讨论。

新增属性a12 是文本型的，重点在于文本清理。考虑一个来源 O =a1，Explain-Da-V 将执行第*24*行，具体执行第*4*节*4.3.1*，其中描述了文本到文本转换。已解析的变换包括移除标点符号（例如“(”）和数值（“*13*”），并将文本转换为小写。请注意，这种变换需要对 *Foofah* 进行扩展，使用 *Foofah* 的原始实现无法准确解析 *[58]*。文本到类别（第*4*节*4.3.3*）的示例可以使用“*contains_multiple_words?*”；在此情况下，元组m1 和m3的值将为 *1*，而m2和m5的值为 *0*。

接下来，考虑a13，它是数值型的；在可能的来源中，考虑 O ={a3,a4}，它是混合型的。Explain-Da-V 将进行编码（第?? 行）和重塑（第??行）。考虑后者，并注意图7中展示了新的生成特征。Explain-Da-V 将使用变换1·（*mean*（*a3*） *by a4*）生成解释，其中表示按a4 分组，并计算a3 的均值（按类型分组的平均评分）。

最后，属性 *a14* 是类别型的，并将 *a4* 视为其来源。除了尝试寻找文本解释（第 *19* 行）之外，*Explain-Da-V* 还会转而采用编码（第 *20* 行）和重塑（第 *21* 行）解释。考虑前者，并注意到 *a14* 是属性 *a4* 的序数编码（剧情片→*1*、动作→*2*、动画→*3*）。这三个编码后的属性，即 *is_Drama*、*is_Action?* 和 *is_Animation?*，被用于解析 *a14*。请注意，在现实场景中，像 *a14* 这样的属性不一定会被识别为类别型属性（例如，它可能具有高基数，或被错误分类为数值型值）。在这种情况下，*Explain-Da-V* 会转到第 *12* 行，从而得到变换 $1 \cdot is_Drama? + 2 \cdot is_Action? + 3 \cdot is_Animation?$。

5.2 $L\Delta_A$的删除解释

移除属性并不常见，通常包括表层变换。我们将 $L\Delta_A$中的每个属性分别视为一个目标。我们介绍两种主要的移除解释类型，分别对应数据清洗（移除重复属性和噪声属性）。

首先，我们考察一种 与表无关的 属性删除，按照我们的术语，这反映了一个空来源（$O = \emptyset$）。具体而言，我们使用一个阈值来判断某个属性是否因缺失值（NaN）过多（超过阈值）而被删除。[5]在这种情况下，对已删除属性的解释 $A_i \in L\Delta_A$ 形式为 $\mathcal{E}_{A_i} = (\emptyset,$ ‘包含缺失信息’)。形式上，“包含缺失信息”可定义为 $A_i = \begin{cases} \emptyset, & \text{如果 NaN 值的比例 } A_i \\ & \text{否则，} \end{cases}$，其中 $\alpha \in [0,1]$ 是某个阈值。

作为第二个案例，我们研究重复信息。对于 T'中的相同属性，可以给出一个简单的解释。给定目标 A_i，来源是某个属性 $A'_j \in T'$，使得 $A'_j \notin T$，并且 $\pi_{A_i}[T] = \pi_{A'_j}[T']$（值完全重叠）。寻找重复信息的自然扩展是研究属性之间的相似性。我们研究两种相似性：衡量属性之间的重叠度，以及它们之间是否存在一对一依赖。我们会测量重叠度；如果达到某个阈值，就会使用重叠属性 $A'j$ 作为来源，并通过“与 $A'j$重叠”变换生成解释，该变换的定义方式与上文类似。我们还会检查 T'是否决定了4.6节中所述的类似方法） $A_i \in L\Delta_A$。显然，还存在许多其他相似性度量，我们计划在未来的工作中进行探索。最后需要注意的是，有时属性删除可能是特定性的，即用户只是因为对数据的某些部分不感兴趣而移除了某个属性。

[5]该阈值可以视为一个超参数，也可以由用户提供。

6 水平变更解释

我们首先讨论常见的数据清洗操作——移除元组，随后讨论添加元组。

6.1 针对 $L\Delta_r$的删除解释

元组删除是数据准备中非常常见的操作，主要涉及数据清理。我们的考察从迭代检查每个被删除的元组开始， $L\Delta_r$ 并分别进行分析。这可能会得到水平解释 $\mathcal{E}_{m4}$，来自 示例 5。最后，我们探究是否应用了某个谓词 来共同删除其余所有（尚未解释的）元组。

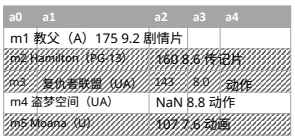
如第 5.2 节所述，我们旨在以独立于表的方式，找出因缺失值（NaN值）而被集体移除的元组，参见 m4 示例 5中的解释。对于依赖于表的解释，我们会寻找重复元组，这是使用实体解析进行数据清洗的常见结果 [48, 70]。我们仅关注相同元组的删除。请注意，这一严格要求可以放宽，并且可以使用任何实体解析技术（例如使用声明式规则） [29], 来寻找重复元组删除解释。具体而言，如果为目标元组 $r_i \in L\Delta_r$ 找到重复元组 $r'_j \in T'$，我们将创建如下形式的水平解释：Er_i =(∅，重复于 $r'j$）。这一变换可以表示为

as follows $r_i = \begin{cases} \emptyset, & \exists r'_j \in T' s.t. r_i = r'_j \\ r_i, & \text{otherwise} \end{cases}$

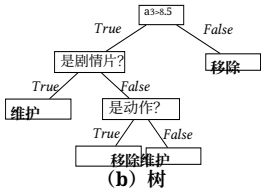
Finally,outlier detectors Z(method and四分位距方法 [32, 97]）也可作为删除元组的解释。

并非所有元组都能独立地得到解释，因此，对于所有未解释的元组，$L\Delta_{r\,unexplained}$，我们旨在以谓词的形式寻找联合解释。给定一组未解释的元组，我们使用分类解释方法（第4.2节）来寻找联合解释。与第5.1节类似，如果来源包含混合

类型，还会在表中对经过编码的属性应用决策树（使用类别型编码变更解释，第4.4节）。



（a）已创建的数据集版本由UserD基于图1a。



（b）树

图10： UserD 版本的图1a及其解释。

示例11。 在图 *1b* 的示例中，我们展示了一个由于值为 *NaN* 而进行元组删除的简单示例。图 *10a* 展示了在表上应用谓词的示例。为了解决该谓词，Explain-Da-V 首先会添加与 a4 *(* 是 _ 剧情片? 、是 _ 动作? 和是 _ 动画? ） 相对应的独热编码特征，然后使用决策树尝试解决该谓词。图 *10b* 中的决策树将被生成。

6.2 $R\Delta_r$的添加解释

元组的非自发性加法可能是过采样（自助法）的结果。为了检测这种变换，我们使用

与第6.1节中类似的方法论。给定一个新增元组 $r_{ni} \in R\Delta_r$，我们的目标是找到一个重复的（相同或相似的）元组 $r_j \in T$，以创建一条解释，说明该元组是通过自助法生成的。

7 评估解释

我们的目标是生成能够捕捉变化语义、便于用户理解的解释。具体而言，该解释（变换）能够重现这一变化，并将其泛化到特定设置之外。为了评估此类语义，下面将介绍我们如何评估解释。有时，针对某一变化可以生成多个解释。回顾示例10，其中我们给出了关于a11的两个可能的有效解释。第一个决策树解释以

$$a3 > 8 \qquad a2 > 150$$

为根，第二个则以为根。此外，以为根的决策树也可能是一个（无效的）解释。在下文中，需要提出的一个重要问题是：如何比较（并在多个可能的解释中进行选择）？

关于数据转换的相关工作（见第2.3节）采用成功率，以衡量通过对输入应用变换是否成功生成了输出。Yang 等人还针对可能的变换（在其术语中为流水线）引入了一个排序指标（平均倒数排名，MRR），但该指标仍将变换视为一个整体[104]。我们认为，仅使用此类指标无法捕捉变换的真正性质，尤其是在评估属性到属性的变换（第5.1节）时。为了提供更细粒度的评估，我们评估基于转换后的值计算的有效性和泛化能力，以评估变换的覆盖率。由于我们关注于提供可解释的解决方案，因此还采用可解释性的两个维度：简洁性和集中度。

7.1 解释有效性和泛化能力

属性（垂直）添加解释比移除或元组转换更丰富，因此相应地进行评估。**垂直添加：**我们将此评估分为有效性（生成的变换能否利用来源重现目标？）和可泛化性（生成的变换能否利用相似的来源重现相似的目标？回顾定义3 以及来源（O）、目标（$\mathcal{G}$）和变换（$\mathcal{P}$）的记法。为简便起见，我们将变换应用于˙源关系后的输出记为$\mathcal{G} = \mathcal{P}(O_{relation})$

。有效性以˙基于元组的方式在值对 $(\hat{r}_{ij}, r_{ij})$ 上计算，其中˙$r_{ij} \in \mathcal{G}$ 是与目标值 $r_{ij} \in \mathcal{G}_{relation}$ 对应的转换值，即˙$r_{i0} = r_{i0}$ (r_{i0} 是元组标识符，因此这意味着元组相匹配，参见第3节）。解释有效性的衡量方式如下：

$$Val(\mathcal{E}_{\mathcal{G}}) = \frac{1}{|\mathcal{G}_{relation}|} \sum_{\substack{\hat{r}_{ij} \in \hat{\mathcal{G}}, r_{ij} \in \mathcal{G}_{relation}: \\ s.t. \hat{r}_{i0} = r_{i0}}} \mathbb{I}(\hat{r}_{ij} = r_{ij}) \qquad (1)$$

其中，$\mathbb{I}(\hat{r}_{ij} = r_{ij})$ 是一个指示函数：如果转换值等于相应的目标值，则返回值 1，否则返回 0。有效性可以视为基于元组的成功率，即成功完成变换的元组所占的比例。

示例 12。 回顾为图 $\mathcal{E}_{a9} = (a3,\ a3 \div sum(a3))$ 中属性 a9 创建的垂直解释，该图见 9 （参见示例6）。此外，考虑另一种垂直解释 $\mathcal{E}'_{a9}$ $=(a3,\ a3 \div 33.4)$。两个解释的有效性得分均为1，因为应用相应的变换可以完美地重新生成a9。

如示例所示，有效性只考察给定的数据集版本 T和 T'，这可能导致过拟合（例如，在$a3$的情况下选择 $a3 \div 33.4$ 而非 $a3 \div sum$）。为了衡量此类情形，我们引入可泛化性，用于衡量一个解决方案能够解释一组等价版本的程度。具体˜而言，如果存在一对版本 T和

˜ ˜ ˙T'，使得 T' 是使用生成 T' 时所采用的相同变换生成的 T 的一个版本，则可以衡量可泛化性。设 ˜O 为关于 $\tilde{T}$ 的来源， ˜G 为关于 $\tilde{T}'$ 的目标。解释 Gen（EG）的可泛化性˜通过在 O 上应用 P 生成 ˆG 来衡量，并按照公式 1 在 ˆG 和 ˜G 上计算。我们通过以下示例说明可泛化性的重要性。

a0	a1	a2	a3	a4
m6	《低俗小说》 (A)	154	8.9	剧情片
m7	《电锯惊魂》 (UA)	103	7.6	恐怖片
m8	《两杆大烟枪》 (UA)	104	NaN	犯罪片

a0	a1	a2	a3	a4	a9	a10	a11	a12	a13	a14
m6	《低俗小说》 (A)	154	8.9	剧情片	0.36	3.47	1	《低俗小说》 a	8.9	1
m7	《电锯惊魂》 (UA)	103	7.6	恐怖片	0.31	4.43	0	《电锯惊魂》 UA	7.6	4
m9	《金刚》 (已通过)	100	7.9	冒险片	0.32	4.74		0	《金刚》 已通过	7.9	5

图11：可泛化性的示例版本

示例*13*。图*11*提供了两个数据集版本。顶部的表与图*1a*相似，底部的表对应于图*9*，因此在图*1a*上应用相同的变换即可生成图*9*。回顾解释 $E_{a9} = (a3, a3 \div sum(a3))$ 和 $E'_{a9} = (a3, a3 \div 33.4)$。虽然这两个解释都是有效的，但利用图*11*，我们观察到 E_{a9} 也具有可泛化性，而 E'_{a9} 则没有。具体而言，如果我们在图*11*中的 *a3* 上应用 E_{a9}，就会得到底部表中 *a9* 的值。然而，如果应用 E'_{a9}，则会分别为记录 *m6*、*m7* 和 *m9* 得到值 *0.27*、*0.23* 和 *0.24*，从而导致可泛化性为 *0*。

在实际场景中，当相同的更改应用于多个数据集时，可以计算可泛化性，例如在 ETL 等数据管道中 [99]。对于我们的新基准（第8.1.1 节），我们生成一个带注释的 留出集，用于计算可泛化性。

其他解释：对于元组删除，我们采用一种重构方法，在全局范围内评估验证和可泛化性。我们收集所有生成的解释，并将其应用于 T，尝试重新生成 T′。然后，我们检查已删除元组与未包含在 T′ 中的元组之间的重叠度。例如，如果使用某个 a3>8 谓词来解释已删除的元组，则会将相同的谓词应用于 T，并与 T′ 进行比较。该重叠度，即使用 Explain-Da-V 的解释正确删除的元组所占的比例，被用作元组删除的总体有效性。可泛化性以与上述类似的方式进行度量，使用额外的数据集版本对 $\tilde{T}$ 和 $\tilde{T}$′。

我们还计算其他解释的有效性和可泛化性。被移除属性或被添加元组的验证和可泛化性分别计算；也就是说，如果某个属性被正确移除，或某个元组被正确添加，则相应得分为 1。

7.2 问题定义

现在，我们正式阐述解释数据版本的问题。回顾一下定义在 T 和 T'（参见第 3.2 节）。

定义 14。 给定 T'，一个 T 的版本，其目标为 $L\Delta_A$（左侧增量属性）、 $R\Delta_A$（右侧增量属性）、$L\Delta_r$（左侧增量元组）和 $R\Delta_r$（右侧增量元组）。在可能解释构成的搜索空间中，版本解释问题是针对每个目标，找出一组有效性最高的解释。

版本解释问题的一个解，是一组组合后最接近生成 T' 自 T 的解释。请注意，仅凭有效性可能会出现平局（正如我们的示例中存在多个有效性为 1 的解释）。如果我们有多个数据集（或一个数据集留出集），则可以使用可泛化性在多个解中进行选择。我们还可以放宽 定义 14，寻找有效性处于最佳解一定范围内的解，然后使用可泛化性在这些候选解中进行选择。此外，我们还可以在此次选择中使用可解释性（下一节将对此进行介绍），选择用户更容易理解的解释。

7.3 解释的可解释性

如上文所述，我们关注生成结果的可解释性。我们再次主要关注属性添加变换。由于我们使用不同的模型生成解释（回归器、决策树和程序），因此需要找到一种通用的可解释性衡量方法。受 Narayanan 等人 [76] 和 Lakkaraju 等人 [68], 关注决策集的研究启发，我们引入两个可解释性维度，即 简洁性和 集中度，用于衡量不同类型解释的可解释性。

可解释性简洁性： 研究表明，模型中的组件越少、长度越短，用户就越容易理解它 [39, 83]。接下来，我们用变换所包含的组件数量（N_c）来衡量其简洁性。对于回归模型，我们使用系数数量；对于决策树，我们使用节点数量；对于程序，我们使用实现代码行数。

示例15。 考虑一个 $exp(A_i)$ 变换。显然，理想的解释应使用 $math(\cdot)$ 扩展（第4.1节）来生成一个有效且可泛化的解释 $(A_i, exp(A_i))$ 其解释可解释性简洁性为1（仅含一个系数）。

另一种解释会在 $poly(\cdot)$ 扩展上使用泰勒级数，以生成有效且可泛化的解释 $(A_i, 1 + A_i + \frac{A_i^2}{2} + \frac{A_i^3}{6} + \dots)$，其解释可解释性简洁性为 $\frac{1}{d+1}$，其中 d 是多项式次数。请注意，这个例子也凸显了有效性（或可泛化性）与可解释性之间的权衡。所选次数越高，有效性（以及可泛化性）就越高，而可解释性则越低。

可解释性集中度： 虽然更简洁的解释更受欢迎，但其中也应包含尽可能少的组件 [69]（即应尽可能集中）。具体而言，由于人类的工作记忆有限，将信息归纳为更少信息块的解决方案更受欢迎 [76]。例如，线性回归函数比带有倒数和对数变换的多项式回归更容易理解，即使前者更长。对于回归器，我们统计其所使用的扩展（例如多项式和数学运算）。对于决策树，我们统计表示条件的内部节点数量，对于程序，我们使用中间变换的数量。设 N_g 为信息块的数量，则可解释性集中度可表示为 $1 \div N_g$，因此，越集中的变换得分越高。

示例16。 $(A_i, exp(A_i))$ 和 $(A_i, 1 + A_i + \frac{A_i^2}{2} + \dots)$ 的集中度分别为0.5（1个扩展，1个次数）和 $\frac{1}{d}$。

为突出简洁性与集中度之间的差异，请考虑例如 $\mathcal{E}_1 = A_1 + A_2 + 5$ 和 $\mathcal{E}_2 = log(A_1) + A_2^2 \cdot A_1$。虽然 $\mathcal{E}_1$ 的简洁性较低（13 对 12 的 $\mathcal{E}_2$），但其集中度（1）高于 $\mathcal{E}_2$（13），后者涉及两个额外的扩展。

总体可解释性是简洁性与集中度的线性组合，可由用户或系统定义。[6]

7.4 关于选择解释

Explain-Da-V 以迭代方式运行，旨在根据第5节和第6节，为检测到的每项变更找到有效解释。如上所述，对于每个目标，都可以生成多个解释；例如，当存在多个来源（第4.6节）时，或者我们有多种方法论来解释某个变换时（例如，第4.1节中的不同扩展）。Explain-Da-V 会为每个目标选择最具可解释性的有效解释。

每个独立解释的推导方式，都是在相应上下文中优化某种误差度量；而这种度量并不总是与我们对有效性的定义相同。回归器（第4.1节）将均方误差降至最低，基于示例的编程解（第4.3节）通过搜索直接优化准确率，而决策树则（以贪心方式）优化节点的划分函数。给定一组解释，我们按如下方式选择其中一个：(1) 找出该集合中的最高有效性。(2) 如果多个解释共享这一值，则根据总可解释性返回最具可解释性的解释（见第 7.3 节）。

请注意，除非我们能够访问 $\tilde{T}$ 和 $\tilde{T}'$（见第7.1节），否则不能使用可泛化性来选择解释。

变换的数量可能非常庞大。为处理这一规模的问题，集合中的解释会根据其来源的大小和基数按排序顺序生成（见第4.6节）。类似地，在回归模型中，解释会按照所应用扩展的数量排序（即首先考虑不含扩展的模型）。因此，我们引入一个提前停止条件：如果某个解释达到预定义的有效性和可解释性阈值，则返回该解释并停止搜索。[7] 根据经验，近70%的案例会提前终止。

8 实证评估

现在，我们将性能与基线方法进行比较（第 8.2 节），并通过消融研究分析各个组件（第 8.3 节）。

8.1 实验设置

现在详细介绍我们的基准测试、实现和基线方法。基准测试和代码可在我们的代码仓库中获取 [15]。

[6] 在实验中，我们采用统一组合。[7] 在实验中，阈值设为 .95。

表 2：语义数据版本控制基准详情。

主题（名称）	原始元组数	原始元组 属性	# of 版本	# of 版本对
电影和电视节目 [8] (IMDB)	1,000 6		72	29
NBA球员 [10] (NBA)	11,700	9	68	27
葡萄酒评论 [12] (WINE)	129,971	6	72	29
鸢尾花 [9] (IRIS)	150	5	58	22
泰坦尼克号乘客 [11] (TITANIC)	891	6	72	29

8.1.1 基准测试。我们针对语义数据版本控制这一新颖任务设计了一个新的基准测试，称为语义数据版本控制基准（SDVB），由五个版本集组成。我们还采用了由 Yang 等人设计的一个公开数据集 [104]，用于合成数据管道这一类似任务。

语义数据版本控制基准（SDVB）： SDVB 包含来自五个不同主题的共 342 个数据集版本（136 个版本对），这些数据集在长度（元组数量）和宽度（属性数量）方面各不相同。[8] 每个主题代表一个版本集，该版本集源自表2中详细介绍的知名种子数据集，其中既包括较小的数据集（例如 IRIS），也包括较大的数据集（例如 WINE）。**版本生成：** 给定一个种子数据集，我们首先选择一组变更维度（例如，$R\Delta_A$和 $L\Delta_r$），然后对其进行修订以生成数据集版本。接着，根据所选维度执行一组变换（其中一些通过采样获得，另一些由人工创建）。我们确保五个版本集中的每一个都涵盖所有变更维度。˜在生成版本之前，每个数据集都会被拆分为 T和 T（80%–20%），后者作为留出集，用于计算可泛化性。根据第7.1节，将应用于 T以生成T，的相同变更，应用于 $\tilde{T}$以生成 $\tilde{T}$，。[9]

最后需要注意的是，一个版本可能通过不止一项更改创建；实际上，上述版本数量由**1,702 项更改组成**。例如，为创建 72 个 WINE 数据集版本，总共对原始数据集及其各个版本应用了 681 项更改。

自动管道基准 [1]：该基准包含从 GitHub notebooks 中提取的真实数据管道。由于我们关注数据集版本，因此会过滤掉包含多个表的数据管道（例如使用连接操作的管道）。遵循 Yang 等人 [104], 的做法，我们将“测试”表视为 T，将“目标”表视为 T，。为确保公平比较，我们在全部数据上运行 Explain-Da-V 和所有基线方法，并且不考虑该基准的可泛化性。

8.1.2 实现。 Explain-Da-V使用 Python 实现，遵循第 5 节和第 6节。代码的主要部分已提供在我们的代码仓库中 [15]。使用 Lasso [13] 和 Ridge [16],[10] 正则化的线性回归以及决策树 [3] 均通过 Scikit-learn 实现。我们扩展了 Foofah 的 Python 公共实现 [7]。我们使用 Featuretools [5] 框架生成聚合特征和分组特征（见第 5.1节）。

8.1.3 基线方法。 **Foofah** [58] 被用作一种基于示例的编程基线方法（见第 2.3节）。 **Foofah+** 表示采用我们新扩展的 Foofah（例如，文本到数值，见第 4.3节）。由于 Auto-pipeline 的实现并未公开，我们通过使用 Foofah 框架实现 Yang 等人提供的算子，复现了其搜索方法[11] （**Auto-pipeline***） [104]。搜索的最坏情况时间复杂度为指数级，因此我们按照 Foofah 中的默认设置，为所有方法设置 60 秒的超时限制 [58]。[12]

我们还使用其可用实现运行了 AutoPandas [19] [2]。AutoPandas 根据 Pandas [14] 操作构建搜索空间，并使用深度学习剪枝程序搜索空间。与 Auto-pipeline 报告的性能类似，[104], AutoPandas 的性能较差，因此未予报告。我们还试验了 SQUARES [80], 这一近期的查询反向工程框架，同样由于其结果较差而未予报告。由于 SQUARES 的设计目的是合成传统 SQL 查询，因此有时能够处理选择谓词；然而，它无法应对其他变更，例如通过变换添加的属性。

最后，基线方法的朴素实现会将所有 T和 T，用作输入输出示例。不过，为了进行公平比较，我们为每种基线方法“查找来源”（见第 4.6 节），并迭代求解垂直解释（每次处理一个属性）。对于横向添加解释，使用 T中的元组作为输入，并使用 T，中的元组作为输出（横向删除的处理类似，以 T，为输入，以 T为输出）。

8.1.4 评估指标。 我们的基线方法提供的解释只有一种类型（程序，而不是回归器或决策树），因此，在 第8.2节中，我们将 Explain-Da-V 与基线方法的有效性（Val.）和泛化能力（Gen.）进行比较。由于 Explain-Da-V可以返回有效性/泛化能力得分不为 1.0 的解释，因此我们还报告这类解释占所有输出解释的比例。我们进一步报告方法用于选择最具可解释性的有效解释的平均解释数量（$\#\mathcal{E}$）（见 第 7.4 节）。我们还对运行时间进行比较并予以报告。 第 8.3 节 还使用可解释性（简洁性和集中度）。

8.2 Explain-Da-V 与基线方法的比较

在基准测试（第 8.1.3 节）上，Explain-Da-V 与基线方法之间的比较结果（第 8.1.1 节）见表6。

Explain-Da-V 之所以明显优于基线方法，主要是因为它能够处理不同的数据类型（除文本型之外，还包括数值型和类别型）。经过调整的自动管道基准是一个例外，在该基准中，Explain-Da-V 仅略优于 Auto-pipeline*。此外，即使我们只聚焦于文本转换（技术报告中提供了相关内容 [91]），Explain-Da-V 仍然优于所有基线方法。即使我们只评估由 Explain-Da-V 返回的100%有效/可泛化的解释（在表6的括号中标出），我们仍观察到显著改进。在各基线方法中，我们观察到扩展 Foofah（Foofah+）使有效性和可泛化性平均提升了9.5%，这表明扩展搜索空间的益处。所有方法都会基于多个来源（参见$\mathcal{E}$，见第8.1.4节）从多个解释中进行选择（参见第8.1.3节）。Explain-Da-V 考虑的解释几乎是其他方法的两倍，因为它为数值解释生成了扩展来源（参见第4.1节）。

[8]并非所有版本都会使用全部原始属性。[9]表2 中报告的数字包含留出集。[10]我们首先尝试使用 Lasso；如果失败，则使用 Ridge。

[11]强化学习需要训练数据，而我们假设这些数据不可用。[12]我们注意到，Auto-pipeline 的默认超时限制为一小时

表3：方法在有效性（Val）、可泛化性（Gen）以及所选解释的平均数量（# $\mathcal{E}$）方面的表现。对于 Explain-Da-V，我们还报告了有效性/可泛化性得分为 1 的解释所占比例（括号内）。

数据集→ / ↓方法	IMDB			NBA			WINE			IRIS			TITANIC			Auto-pipeline		
	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	Gen	# $\mathcal{E}$	Val	可泛化性	# $\mathcal{E}$
Foofah	.42	.42	3.7	.28	.28	4.2	.29	.29	3.9	.23	.23	3.1	.29	.29	4.1	.55	-	3.3
Foofah+ Auto-pipeline*	.44	.44	3.7	.29	.29	4.2	.34	.34	3.9	.25	.25	3.1	.37	.37	4.1	.55	-	3.3
	.44	.44	3.7	.30	.30	4.2	.33	.33	3.9	.26	.26	3.1	.37	.37	4.1	.78	-	3.3
Explain-Da-V	.73 (.64)	.60 (.56)	6.4	.90 (.89)	.79 (.69)	7.3	.87 (.76)	.81 (.59)	6.8	.93 (.88)	.83 (.76)	8.9	.88 (.79)	.77 (.68)	7.2	.82 (.78)	-	5.7
+高于基线	+65%	+36%		+202%	+167%		+156%	+138%		+254%	+217%		+140%	+109%		+5%	-	

表 4：Foofah、Foofah+、Auto-pipeline* 和 Explain-Da-V 在数值目标的有效性（验证集）和可泛化性（泛化）方面的表现

数据集→ / ↓方法	IMDB		NBA		WINE		IRIS		TITANIC	
	Val.	Gen.	Val.	Gen.	Val.	Gen.	Val.	Gen.	Val.	Gen.
Foofah	.20	.20	.11	.11	.14	.14	.18	.18	.12	.12
Foofah+ Auto-pipeline*	.20	.20	.15	.15	.16	.16	.21	.21	.24	.24
	.22	.22	.16	.16	.18	.18	.22	.22	.26	.26
Explain-Da-V	**.97** (+340%)	**.85**(+286%)	**.72** (+350%)	**.66** (+313%)	**.87** (+383%)	**.75** (+316%)	**.99** (+350%)	**.85** (+286%)	**.92** (+253%)	**.84** (+223%)

表 5：Foofah、Foofah+、Auto-pipeline* 和 Explain-Da-V 在类别目标的有效性（验证集）和可泛化性（泛化）方面的表现

数据集→ / ↓方法	IMDB		NBA		WINE		IRIS		TITANIC	
	Val.	Gen.	Val.	Gen.	Val.	Gen.	Val.	Gen.	Val.	Gen.
Foofah	.15	.15	.09	.09	.16	.16	.23	.23	.22	.22
Foofah+ Auto-pipeline*	.25	.25	.12	.12	.18	.18	.25	.25	.22	.22
	.27	.27	.12	.12	.17	.17	.26	.26	.22	.22
Explain-Da-V	**.86** (+218%)	**.86**(+218%)	**.99** (+725%)	**.87** (+625%)	**.89** (+423%)	**.78** (+358%)	**.98** (+277%)	**.91** (+250%)	**.88** (+300%)	**.83** (+277%)

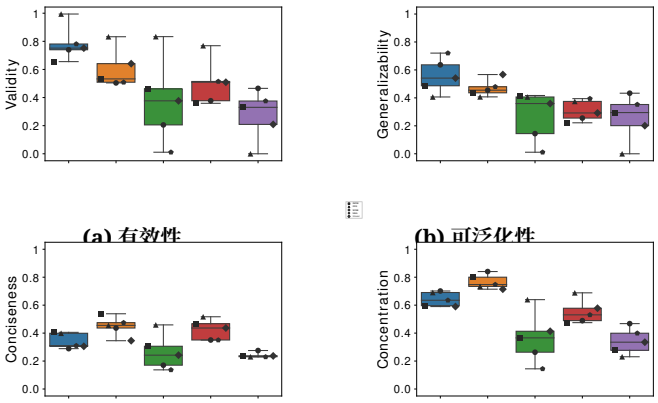
表6：Foofah、Foofah+、Auto-pipeline* 和 Explain-Da-V 在文本目标方面的有效性（验证集）和可泛化性（泛化）表现

数据集→ / ↓方法	IMDB		NBA		WINE		IRIS		TITANIC	
	Val.	Gen.	Val.	Gen.	Val.	Gen.	Val.	Gen.	Val.	Gen.
Foofah	.50	.50	.48	.48	.44	.44	.33	.33	.47	.47
Foofah+ Auto-pipeline*	.52	.52	.50	.50	.53	.53	.42	.42	.48	.48
	.52	.52	.56	.56	.53	.53	.38	.38	.50	.50
Explain-Da-V	**.62** (+19%)	**.57**(+10%)	**.88** (+57%)	**.67** (+20%)	**.58** (+9%)	**.56** (+6%)	**.82** (+115%)	**.72** (+89%)	**.79** (+58%)	**.69** (+38%)

在比较不同版本集时，我们观察到，在 IRIS 数据集上，`Explain-Da-V` 取得了最佳性能（.927 验证集和 .831 泛化）以及最高的提升幅度。对于 IMDB 版本集，`Explain-Da-V` 取得了最差的性能（.732 验证集和 .602 泛化）以及最低的提升幅度（在新提出的基准版本集中）。IRIS 主要由数值型属性组成（5个属性中有4个），这些属性通过我们的数值变化解释得到解决，而基线方法无法处理这些属性。需要注意的是，相应地，`Explain-Da-V` 考虑的解释数量几乎是其他方法的三倍。然而，第4.1节中引入的数值扩展，以及我们能够找到来源这一事实，有助于快速锁定有效解决方案（见下文的运行时间）。另一方面，IMDB 版本集包含更多文本属性（6个属性中有5个），并涉及一些`Explain-Da-V` 无法解决的变更。例如，其中一个 IMDB 版本集涉及一种变换添加一个属性，其中包含某个类型属性中类型数量的计数。在类型属性中，各类型以逗号分隔（例如，剧情片、爱情）。例如，一种正确的变换方式是统计逗号的数量并加 1。虽然寻找一种统计逗号数量的变换方式，对`Explain-Da-V` 来说是一项实际任务（它包含文本型变换和聚合），但目前尚无法实现这样的组合。相反，解释`Explain-Da-V` 选择的方案（有效性最高，见第 7.4 节）使用IMDB 评分属性，通过决策树来确定类型数量，其有效性为 0.65。[13]

最后，我们注意到，性能会随不同的变更维度而变化。有趣的是，如果我们只考察垂直移除（第 5.2 节），三个基线方法的有效性和泛化能力得分均为 1。原因在于它们能够

[13]解释可在代码仓库中查看 [4]。



(a) 有效性 (b) 可泛化性

(c) 简洁性 (d) 集中度

图12：对 SDVB 数据集进行的消融研究。

发现投影（用它们的术语来说，即对某个属性执行 删除列 操作）。尽管它成功找到了这些变换，却无法解释属性删除的语义。 Explain-Da-V虽然有效性和可泛化性并非完美（0.95），但在解释移除方面表达能力更强。例如，它可以解释某个属性为何被移除：因为该属性包含重复信息（见第 5.2 节）。考察元组删除时，Explain-Da-V 的表现远优于基线方法。由于Auto-pipeline“未考虑行级过滤”，[104], 我们再次回顾与SQUARES 的比较（见第 8.1.3 节）。尽管 SQUARES 专注于学习选择谓词，但它能够处理谓词以 0.6 的有效性应用的情况（Explain-Da-V 在这些变更上的得分为 0.73）。这是因为SQUARES 无法处理删除包含 NaN 值的元组和重复元组的情况（基准中的两个案例）。

运行时间：在这些实验中，排除超时情况（见第 8.1.3节），使用 Foofah 查找解释平均耗时 4.9 秒，Foofah+ 12.4 秒，Auto-pipeline* 8.1 秒，而 Explain-Da-V 耗时 2.4 秒。造成这一差异的原因之一是，拟合回归器（线性时间复杂度）和学习决策树（二次复杂度）比搜索（指数级）更高效。

8.3 Explain-Da-V消融研究

图 12 展示了 Explain-Da-V 的消融研究。我们重点研究垂直加法解释，并分析 Explain-Da-V 在不查找来源（不查找来源）时的性能，即使用 T整体来解释给定目标，以及不使用数值到数值变换的扩展（不使用扩展）时的性能。我们还通过应用 Explain-Da-V，假设所有类型均为数值型（全部数值型）或均为文本型（全部文本型），来分析解析后的数据类型。

如图 12a 和 12b所示，完整的 Explain-Da-V 表现出最具有效性和可泛化性的性能。添加扩展和查找来源分别使有效性方面的性能平均提升了30%和107%，而将所有属性类型都作为数值型和文本型处理，则分别使有效性降低了35%和64%。NBA版本集展示了一个有趣的案例。由于其中包含类型各异的多种属性，在不查找来源的情况下，Explain-Da-V 获得了极低的有效性和可泛化性。同样，如上所述，由于IRIS版本集主要由数值属性组成，将所有属性都作为文本型处理会导致性能极低。

考察可解释性（图 12c-12d）后，我们观察到，尽管没有扩展的解释有效性和泛化能力较低，但其更加简洁且更为集中。如果未找到来源，解释通常不够简洁，集中程度也低得多。最后，数值解释比文本解释更具可解释性，尤其是在简洁性方面。**通过有趣案例探讨局限性：** 考虑 NBA 版本集。在该数据集中进行一次纵向加法，会创建一个二元属性，用于表示“两双”表现的指示函数。[14] 由 Explain-Da-V 生成的有效且可泛化的变换，在1超过 $A_{ast} > 10$ （助攻超过10次）时返回$r_i \in T$，因为对于每个元组 $\pi_{A_{ast}}[r_i] > 10$，也会出现 $\pi_{A_{pts}}[r_i] > 10$（得分超过10分）的情况。尽管在特定场景下是正确的，但它没有捕捉到两双的语义。

另一个有趣案例是在 IMDB 版本集中的电影评分（A_{rating}）上进行平方根变换。 Explain-Da-V生成了两个有效且可泛化的解释：一个使用变换 $sqrt$（A_{rating}）（使用 $math$扩展），另一个使用多项式变换 $1.074 + 0.262 \cdot A_{rating} - 0.005 \cdot A^2_{rating}$。显然，这两种变换并不相同；然而，如果我们聚焦于范围 $x \in [6, 10]$ （IMDB 中的典型评分），就会发现这两种变换的表现几乎相同。例如，评分为9时，经过变换后分别得到3和3.02。我们的代码仓库中提供了示例 [17]。

9 结论

本研究为解释数据版本管理中的语义变化奠定了基础。Explain-Da-V 使用不同类型的技术来解决并解释一对数据集版本之间的变化。我们引入了用于评估解释的度量指标，并表明，与多个基线方法相比，Explain-Da-V 在一个现有的改编基准和一个新提出的数据版本管理基准上表现更好。在未来的工作中，我们计划扩展 Explain-Da-V，以处理更多数据类型（例如日期），并应对由外部数据触发的变化，例如执行连接和并集。未来的另一项挑战是将版本解释问题表述为一个多目标优化问题，以共同优化有效性、可泛化性和可解释性。

参考文献

[1] 2022。自动管道基准。https://gitlab.com/jwjwyoung/autopipeline-benchmarks。[2] 2022。AutoPandas 实现。https://github.com/rbavishi/autopandas。[3] 2022。决策树。https://scikit-learn.org/stable/modules/tree.html。[4] 2022。解释示例。https://github.com/shraga89/ExplainDaV/blob/ main/Explanation_Example.md。[5] 2022。Featuretools。https://www.featuretools.com/。[6] 2022。FoofahExtensions（代码仓库）。https://github.com/shraga89/ExplainDaV/ blob/main/Code/Ops_added_to_Foofah.py。[7] 2022。Foofah 实现。https://github.com/umich-dbgroup/foofah。[8] 2022。初始 IMDB数据集。https://www.kaggle.com/datasets/harshitshankhdhar/imdb-dataset-of-top-1000-movies-and-tv-shows。[9] 2022。初始 IRIS 数据集。https://www.kaggle.com/uciml/iris。[10] 2022。初始 NBA数据集。https://www.kaggle.com/justinas/nba-players-data。[11] 2022。初始 TITANIC数据集。https://www.kaggle.com/competitions/titanic。

[14]在篮球中，两双是指一名球员在两个统计类别中的数据均达到或超过十项 https://en.wikipedia.org/wiki/Double-double。

[12] 2022。初始 WINE 数据集。https://www.kaggle.com/christopheiv/winemagdata130k。[13] 2022。套索正则化。https://scikit-learn.org/stable/modules/generated/ sklearn.linear_model.Lasso.html。[14] 2022 。Pandas。https://pandas.pydata.org/。[15] 2022。代码仓库。https://github.com/shraga89/ExplainDaV。[16] 2022。岭正则化。https://scikit-learn.org/stable/modules/generated/ sklearn.linear_model.Rigde.html。[17] 2022。平方根示例（代码仓库）。https://github.com/shraga89/ExplainDaV/ blob/main/sqrt_example.pdf。[18] Ziawasch Abedjan、John Morcos、Ihab F. Ilyas、Mourad Ouzzani、Paolo Papotti 和 Michael Stonebraker。2016年。DataXFormer：一种稳健的变换发现系统。载于第 32 届 IEEE 国际数据工程会议，ICDE 2016，芬兰赫尔辛基，2016年5月16日至20日。IEEE Computer Society，1134–1145。https://doi.org/10 1109 2016 7498319 。/ICDE。[19] 3, 2019 168 1 168 27 10 1145 3360594 Rohan [20] 2013 Bavishi、Caroline Lemieux、Roy Fox、Koushik Sen 和 Ion Stoica。2019年。AutoPandas：用于程序合成的神经网络支持的生成器。ACM 程序设计语言会议论文集。OOPSLA（），：–：。https://doi.org/。/ Ladjel Bellatreche 和 Robert Wrembel。[21] 。关于语义数据集成系统演化与版本管理的专刊。57–59 页。15_ Anant P. Bhardwaj、Souvik Bhattacherjee、Amit Chavan、Amol Deshpande、Aaron J. Elmore、Samuel Madden 和 Aditya G. Parameswaran。2015年。DataHub：大规模协作式数据科学与数据集版本管理。载于第七届双年度创新数据系统研究会议，CIDR 2015，美国加利福尼亚州阿西洛马，2015年1月4日至7日，在线论文集。www.cidrdb.org。http://cidrdb. org/cidr2015/Papers/CIDR15_ Paper18.pdf[22] Souvik Bhattacherjee、Amit Chavan、Silu Huang、Amol Deshpande 和 Aditya Parameswaran。2015年。数据集版本控制原则：探索重建与存储之间的权衡。载于 Proceedings of the VLDB Endowment。国际超大型数据库会议，第 8 卷。NIH Public Access，1346。[23] Tobias Bleifuß、Leon Bornemann、Theodore Johnson、Dmitri V Kalashnikov、Felix Naumann 和 Divesh Srivastava。2018年。探索变更：数据分析的新维度。Proceedings of the VLDB Endowment 12，2（2018年），85–98。[24] Tobias Bleifuß、Leon Bornemann、Dmitri V Kalashnikov、Felix Naumann 和 Divesh Srivastava。2019年。DBChEx：数据与模式变更的交互式探索。载于 CIDR。[25] Alex Bogatu、Norman W. Paton、Alvaro A. A. Fernandes 和 Martin Koehler。2019年。迈向自动数据格式转换：大规模数据整理。Comput. J. 62，7（2019年），1044–1060。https://doi.org/10.1093/comjnl/bxy118[26] Leon Bornemann、Tobias Bleifuß、Dmitri Kalashnikov、Felix Naumann 和 Divesh Srivastava。2018年。使用时间序列聚类探索数据变更。Datenbank-Spektrum 18，2（2018年），79–87。[27] Richard J Brook 和 Gregory C Arnold。2018年。应用回归分析与实验设计。CRC出版社。[28] Jason Brownlee。2022年。机器学习的数据准备。[29] Douglas Burdick、Ronald Fagin、Phokion G. Kolaitis、Lucian Popa 和 Wang-Chiew Tan。2016年。实体链接的声明式框架。ACM Trans. Database Syst. 41，3（2016年），17:1–17:38。[30] Nadia Burkart 和 Marco F Huber。2021年。监督式机器学习可解释性综述。Journal of Artificial Intelligence Research 70（2021年），245–317。[31] Hancheng Cao、Vivian Yang、Victor Chen、Yu Jin Lee、Lydia Stone、N' godjigui Junior Diarrassouba、Mark E Whiting 和 Michael S Bernstein。2021年。我的团队将继续前进：通过团队互动区分高可行性团队与低可行性团队。Proceedings of the ACM on Human-Computer Interaction 4，CSCW3（2021年），1–27。[32] Varun Chandola、Arindam Banerjee 和 Vipin Kumar。2009年。异常检测：综述。ACM computing surveys（CSUR）41，3（2009年），1–58。[33] Adriane Chapman、Elena Simperl、Laura Koesten、George Konstantinidis、Luis-Daniel Ibáñez、Emilia Kacprzak 和 Paul Groth。2020年。数据集搜索：综述。VLDB J. 29，1（2020年），251–272。[34] Sudarshan S Chawathe 和 Hector Garcia-Molina。1997年。结构化数据中的有意义变更检测。ACM SIGMOD Record 26，2（1997年），26–37。[35] Sudarshan S Chawathe、Anand Rajaraman、Hector Garcia-Molina 和 Jennifer Widom。1996年。层次结构化信息中的变更检测。Acm Sigmod Record 25，2（1996年），493–504。[36] James Cheney、Laura Chiticariu、Wang-Chiew Tan 等。2009年。数据库中的数据溯源：原因、方法与位置。Foundations and Trends® in Databases 1，4（2009年），379–474。[37] Anton Chernyavskiy、Dmitry Ilvovsky 和 Preslav Nakov。2021年。变换器："自然语言处理的历史终结"？载于欧洲机器学习与数据库知识发现联合会议。Springer，677–693。[38] Gregory Cobena、Serge Abiteboul 和 Amelie Marian。2002年。检测 XML 文档中的变更。载于第 18 届国际数据工程会议论文集。IEEE，41–52。[39] Nicole Cruz、Jean Baratgin、Mike Oaksford 和 David E Over。2015年。使用 if、and 和 or 进行贝叶斯推理。Frontiers in psychology 6（2015年），192。[40] Giovanni Da San Martino、Seunghak Yu、Alberto Barrón-Cedeno、Rostislav Petrov 和 Preslav Nakov。2019年。新闻文章中的宣传细粒度分析。载于《自然语言处理实证方法会议》和第9届《国际自然语言处理联合会议》（EMNLP-IJCNLP）2019年会议论文集。5636–5646。[41] Anish Das Sarma、Lujun Fang、Nitin Gupta、Alon Halevy、Hongrae Lee、Fei Wu、Reynold Xin 和 Cong Yu。2012年。查找相关表。载于2012年ACM SIGMOD数据管理国际会议论文集（美国亚利桑那州斯科茨代尔）（SIGMOD '12）。计算机协会，纽约州纽约，美国，817 828 10 1145 2213836 2213962 – 。https://doi.org/ . / 。[42] 2020 加拿大开放数据。https://open.canada.ca/en/open-data [43] 英国开放数据。2020年。https://data.gov.uk/ [44] Boer Deng。2015年。标题较短的论文获得更多引用。《自然》新闻26（2015）。[45] Dong Deng、Wenbo Tao、Ziawasch Abedjan、Ahmed K. Elmagarmid、Ihab F. Ilyas、Guoliang Li、Samuel Madden、Mourad Ouzzani、Michael Stonebraker 和 Nan Tang。2019年。用于实体整合的无监督字符串转换学习。载于第35届IEEE国际数据工程会议（ICDE 2019），中国澳门，2019年4月8日至11日。IEEE，196–207。https://doi.org/10.1109/ICDE.2019.00026[46] Jeffrey R Edwards。2002年。差值分数的替代方法：多项式回归和响应面方法。测量与数据分析进展（2002年），350–400。[47] Kareem El Gebaly、Parag Agrawal、Lukasz Golab、Flip Korn 和 Divesh Srivas- tava。2014年。可解释且信息丰富的结果解释。《VLDB基金会学报》8，1（2014），61–72。[48] Ahmed K Elmagarmid、Panagiotis G Ipeirotis 和 Vassilios S Verykios。2006年。重复记录检测：综述。IEEE《知识与数据工程汇刊》19，1（2006），1–16。[49] Peter A Flach 和 Iztok Savnik。1999年。数据库依赖发现：一种机器学习方法。《人工智能通讯》12（3）（1999），139–160。http://content.iospress.com/articles/ai-communications/aic182 出版商：IOS出版社。[50] Yihan Gao、Silu Huang 和 Aditya G. Parameswaran。2018年。使用DATAMARAN驾驭数据湖：自动从日志数据集中提取结构。载于2018年数据管理国际会议论文集，SIGMOD 2018会议，美国得克萨斯州休斯敦，2018年6月10日至15日，Gautam Das、Christopher M. Jermaine 和 Philip A. Bernstein（编）。ACM，943–958。https://doi.org/10.1145/3183713.3183746[51] Dimitris C Gkikas、Katerina Tzafilkou、Prokopis K Theodoridis、Aristogiannis Garmpis 和 Marios C Gkikas。2022年。文本特征如何影响社交媒体帖子中的用户参与度：对脸书中文本可读性、长度和话题标签数量进行建模。《国际信息管理杂志：数据洞察》2，1（2022），100067。[52] William R. Harris 和 Sumit Gulwani。2011年。通过示例进行电子表格数据表转换。载于第32届ACM SIGPLAN编程语言设计与实现会议论文集，PLDI 2011，美国加利福尼亚州圣何塞，2011年6月4日至8日，Mary W. Hall 和 David A. Padua（编）。ACM，317–328。https://doi.org/10 1145 1993498 1993536 . / 。Yeye He、Xu Chu、Kris Ganjam、[53] Yudian Zheng、Vivek R. Narasayya 和 Surajit Chaudhuri。2018年。Transform-Data-by-Example（TDE）：可扩展的数据转换搜索引擎。《VLDB基金会学报》11，10（2018），1165–1177。https://doi.org/10 14778 3231751 3231766 . / 。Yeye He、Zhongjun Jin 和 Surajit [54] Chaudhuri。2020年。Auto-Transform：通过模式学习转换。《VLDB基金会学报》13，11（2020），2368–2381。http: //www.vldb.org/pvldb/vol13/p2368-he.pdf [55] Fred Hohman、Kanit Wongsuphasawat、Mary Beth Kery 和 Kayur Patel。2020年。理解并可视化机器学习中的数据迭代。载于2020年CHI会议计算机系统中的人为因素会议论文集。1–13。[56] 美国政府开放数据主页。2020年。https://data.gov/ [57] Silu Huang、Liqi Xu、Jialin Liu、Aaron J Elmore 和 Aditya Parameswaran。2017年。ORPHEUSDB：关系数据库的附加式版本管理。《VLDB基金会学报》10，10（2017）。[58] Zhongjun Jin、Michael R. Anderson、Michael J. Cafarella 和 H. V. Jagadish。2017年。Foofah：通过示例转换数据。载于2017年ACM数据管理国际会议论文集，SIGMOD 2017会议，美国伊利诺伊州芝加哥，2017年5月14日至19日，Semih Salihoglu、Wenchao Zhou、Rada Chirkova、J un Yang 和 Dan Suciu（编）。ACM，683 698 10 1145 3035918 3064034 – 。[59] https://doi.org/ . / 。Zhongjun Jin、Michael J. Cafarella、H. V. Jagadish、Sean Kandel、Michael Minar 和 Joseph M. Hellerstein。2019年。CLX：迈向可验证的基于示例的编程数据转换。载于《数据库技术进展——第22届扩展数据库技术国际会议》，EDBT 2019，29，2019年3月26日，葡萄牙里斯本，Melanie Herschel、Helena Galhardas、Berthold Reinwald、Irini Fundu- laki、Carsten Binnig 和 Zoi Kaoudi（编）。OpenProceedings.org，265–276。https://doi.org/10 5441 002 2019 24 . / /edbt. 。Mary Beth Kery、Amber [60] Horvath 和 Brad Myers。2017年。Variolite：支持数据科学家的探索式编程。载于2017年CHI会议计算机系统中的人为因素会议论文集。1265–1276。

[61] Mary Beth Kery、Bonnie E John、Patrick O'Flaherty、Amber Horvath 和 Brad A Myers。2019年。通过帮助数据科学家寻找过去的分析选择，实现有效的信息搜寻。在《2019年计算机系统中的人为因素CHI会议论文集》中。1–13。[62] Mary Beth Kery、Marissa Radensky、Mahima Arya、Bonnie E John 和 Brad A Myers。2018年。笔记本中的故事：使用文读编程工具进行探索性数据科学。在《2018年计算机系统中的人为因素CHI会议论文集》中。1–11。[63] Aamod Khatiwada、Grace Fan、Roee Shraga、Zixuan Chen、Wolfgang Gatter- bauer、Renée J Miller 和 Mirek Riedewald。2023年。SANTOS：基于关系的语义表联合搜索。在SIGMOD会议2023中。ACM。https: //arxiv.org/pdf/2209.13589.pdf [64] Aamod Khatiwada、Roee Shraga、Wolfgang Gatterbauer 和 Renée J. Miller。2022年。集成数据湖表。Proc. VLDB Endow. 16, 4（2022年），932–945。https://www.vldb.org/pvldb/vol16/p932-khatiwada.pdf [65] Alexandra Kim、Laks VS Lakshmanan 和 Divesh Srivastava。2020年。总结分层多维数据。在《2020年IEEE第36届国际数据工程会议（ICDE）》中。IEEE，877–888。[66] Olga Kovaleva、Alexey Romanov、Anna Rogers 和 Anna Rumshisky。2019年。揭示BERT的黑暗秘密。在《2019年自然语言处理实证方法会议暨第9届国际自然语言处理联合会议（EMNLP-IJCNLP）论文集》中。4365–4374。[67] Max Kuhn 和 Kjell Johnson。2019年。特征工程与选择：预测模型的实用方法。CRC出版社。[68] Himabindu Lakkaraju、Stephen H Bach 和 Jure Leskovec。2016年。可解释决策集：用于描述和预测的联合框架。在《第22届ACM SIGKDD知识发现与数据挖掘国际会议论文集》中。1675–1684。[69] Benjamin Letham、Cynthia Rudin、Tyler H McCormick 和 David Madigan。2015年。使用规则和贝叶斯分析的可解释分类器：构建更优的中风预测模型。《应用统计年刊》9, 3（2015年），1350–1371。[70] Yuliang Li、Jinfeng Li、Yoshihiko Suhara、AnHai Doan 和 Wang-Chiew Tan。2020年。使用预训练语言模型进行深度实体匹配。《Proceedings of the VLDB Endowment》14, 1（2020年），50–60。[71] Benjamin Marlin。2004年。协同过滤：机器学习视角。多伦多大学多伦多分校。[72] Zhengjie Miao、Qitian Zeng、Boris Glavic 和 Sudeepa Roy。2019年。超越数据溯源：使用基于模式的反事实解释查询答案。在《2019年数据管理国际会议论文集》中。485–502。[73] Renée J Miller。2018年。开放数据集成。《Proceedings of the VLDB Endowment》11, 12（2018年），2130–2139。[74] Tom Mitchell。1997年。决策树学习。《机器学习》414（1997年），52–78。[75] Heiko Müller、Johann-Christoph Freytag 和 Ulf Leser。2006年。描述数据库之间的差异。在《第15届ACM信息与知识管理国际会议论文集》中。612–621。[76] Menaka Narayanan、Emily Chen、Jeffrey He、Been Kim、Sam Gershman 和 Fi- nale Doshi-Velez。2018年。人类如何理解来自机器学习系统的解释？对解释的人类可解释性进行评估。arXiv预印本 arXiv:1802.00682（2018年）。[77] Fatemeh Nargesian、Erkang Zhu、Renée J. Miller、Ken Q. Pu 和 Patricia C. Arocena。2019年。数据湖管理：挑战与机遇。Proc. VLDB Endow. 12, 12（2019年8月），1986–1989。https://doi.org/10.14778/3352063. 3352116[78] Fatemeh Nargesian、Erkang Zhu、Ken Q. Pu 和 Renée J. Miller。2018年。开放数据上的表联合搜索。Proc. VLDB Endow. 11, 7（2018年3月），813–825。https://doi.org/10.14778 3192965 3192973 。/ 。[79] Andrew Nierman 和 HV Jagadish。。评估XML 文档中的结构相似性。。在webdb，第2卷。Citeseer，61–66。2002 [80] Pedro Orvalho、Miguel Terra-Neves、Miguel Ventura、Ruben Martins 和 Vasco Manquinho。2020年。SQUARES：使用查询逆向工程的SQL合成器。《Proceedings of the VLDB Endowment》13, 12（2020年），2853–2856。[81] Aslihan Özmen、Mahdi Esmailoghli 和 Ziawasch Abedjan。2021年。将示例驱动编程与大型数据库中的转换发现相结合。在《商业、技术与网络数据库系统（BTW 2021），德国信息学会数据库与信息系统专业委员会第19届会议，2021年9月13日至17日，德国德累斯顿，论文集（LNI）》中，Kai-Uwe Sattler、Melanie Herschel 和 Wolfgang Lehner （编），第 P-311 卷。德国信息学会，波恩，313 324 10 18420 2021 16 – 。[82] https://doi.org/ 。/btw - Thorsten Papenbrock、Jens Ehrlich、Jannik Marten、Tommy Neubert、Jan-Peer Rudolph、Martin Schönberg、Jakob Zwiener 和 Felix Naumann。2015年。函数依赖发现：七种算法的实验评估。Proc. VLDB Endow. 8, 10（2015年），1082–1093。https://doi.org/10.14778/2794367. 2794377[83] Forough Poursabzi-Sangdeh、Daniel G Goldstein、Jake M Hof man、Jennifer Wort- man Wortman Vaughan 和 Hanna Wallach。2021年。操纵和测量模型可解释性。在《2021年CHI会议：人为因素》论文集中的部分在计算系统中。1–52。[84] Erhard Rahm 和 Philip A Bernstein。2001年。自动模式匹配方法综述。the VLDB Journal 10, 4 (2001年), 334–350。[85] John F Roddick。1995年。数据库系统模式版本控制问题综述。Information and Software Technology 37, 7 (1995年), 383–393。[86] Pau Rodríguez, Miguel A Bautista, Jordi Gonzalez, 和 Sergio Escalera。2018年。超越独热编码：低维目标嵌入。Image and Vision Computing 75 (2018年), 21–31。[87] Sudeepa Roy, Laurel Orr, 和 Dan Suciu。2015年。使用可解释数据库解释查询答案。Proceedings of the VLDB Endowment 9, 4 (2015年), 348–359。[88] Maximilian E Schüle, Josef Schmeißer, Thomas Blum, Alfons Kemper, 和 Thomas Neumann。2021年。TardisDB：扩展 SQL 以支持版本管理。载于《2021年数据管理国际会议论文集》。2775–2778。[89] Vraj Shah, Jonathan Lacanlale, Premanand Kumar, Kevin Yang, 和 Arun Kumar。2021年。迈向 AutoML 平台的特征类型推断基准测试。载于《2021年数据管理国际会议论文集》。1584–1596。[90] Roee Shraga, Avigdor Gal, 和 Haggai Roitman。2020年。Adnev：使用深度相似度矩阵调整与评估进行跨域模式匹配。Proceedings of the VLDB Endowment 13, 9 (2020年), 1401–1415。[91] Roee Shraga 和 Renée J. Miller。2022年。使用 Explain-Da-V 解释语义数据版本控制中的数据集变更（技术报告）。https://github.com/shraga89/ExplainDaV/blob/main/Explain_Da_V_TR.pdf [92] Rishabh Singh。2016年。BlinkFill：用于句法字符串转换的半监督示例编程。Proc. VLDB Endow. 9, 10 (2016年), 816–827。https://doi.org/10.14778 2977797 2977807 。/。[93] Rishabh Singh 和 Sumit Gulwani。2012 学习从示例中提取语义字符串转换。Proc. VLDB Endow. 5, 8 (2012年), 740–751。https://doi.org/10.14778/2212351.2212356[94] Richard T S nodgrass, Curtis Dyreson, Faiz Currim, Sabah Currim, 和 Shailesh Joshi。2008年。验证quicksand：XSchema中的时态模式版本控制。Data & Knowledge Engineering 65, 2 (2008年), 223–242。[95] Charles Sutton, Timothy Hobson, James Geddes, 和 Rich Caruana。2018年。Data Diff：用于数据整理的、可解释且可执行的分布变更摘要。载于《第24届 ACM SIGKDD 知识发现与数据挖掘国际会议论文集》。2279–2288。[96] Christos Thrampoulidis, Samet Oymak, 和 Babak Hassibi。2015年。正则化线性回归：对估计误差的精确分析。载于《学习理论会议》。PMLR, 1683–1709。[97] Kai Ming Ting, Sunil Aryal, 和 Takashi Washio。2018年。我应该使用哪种异常值检测器？载于《2018年 IEEE 国际数据挖掘会议（ICDM）》。IEEE, 8–8。[98] Quoc Trung Tran, Chee-Yong Chan, 和 Srinivasan Parthasarathy。2014年。查询逆向工程。The VLDB Journal 23, 5 (2014年), 721–746。[99] Panos Vassiliadis。2009年。抽取-转换-加载技术综述。International Journal of Data Warehousing and Mining (IJDWM) 5, 3 (2009年), 1–27。[100] S Vijayarani, Ms J Ilamathi, Ms Nithya, 等。2015年。文本挖掘预处理技术概述。International Journal of Computer Science & Communication Networks 5, 1 (2015年), 7–16。[101] Xiaolan Wang 和 Alexandra Meliou。2019年。Explain 3D：解释不相交数据集中的分歧。Proceedings of the VLDB Endowment 12, 7 (2019年)。[102] Yuan Wang, David J DeWitt, 和 J-Y Cai。2003年。X-Diff：一种有效的 XML 文档变更检测算法。载于《第19届国际数据工程会议论文集（目录号 03CH37405）》。IEEE, 519–530。[103] Cong Yan 和 Yeye He。2020年。Auto-suggest：利用数据科学笔记本学习推荐数据准备步骤。载于《2020年 ACM SIGMOD 数据管理国际会议论文集》。1539–1554。[104] Junwen Yang, Yeye He, 和 Surajit Chaudhuri。2021年。Auto-pipeline：通过强化学习和搜索按目标合成复杂数据管道。Proceedings of the VLDB Endowment 14, 11 (2021年), 2563–2575。[105] Gunce Su Yilmaz, Tana Wattanawaroon, Liqi Xu, Abhishek Nigam, Aaron J Elmore, 和 Aditya Parameswaran。2018年。Datadiff：用于协作式数据分析的用户可解释的数据转换摘要。载于《2018年数据管理国际会议论文集》。1769–1772。[106] Amy X Zhang, Michael Muller, 和 Dakuo Wang。2020年。数据科学工作者如何协作？角色、工作流程和工具。Proceedings of the ACM on Human-Computer Interaction 4, CSCW1 (2020年), 1–23。[107] Alice Zheng 和 Amanda Casari。2018年。机器学习特征工程：面向数据科学家的原理与技术。" O'Reilly Media, Inc."。[108] Erkang Zhu, Dong Deng, Fatemeh Nargesian, 和 Renée J Miller。2019年。Josie：用于在数据湖中查找可连接表的重叠集合相似性搜索。载于《2019年数据管理国际会议论文集》。847–864。[109] Erkang Zhu, Yeye He, 和 Surajit Chaudhuri。2017年。Auto-Join：利用变换进行表连接。Proc. VLDB Endow. 10, 10 (2017年), 1034–1045。https://doi.org/10.14778/3115404.3115409