arXiv:2304.08285v1 [cs.DB] 17 Apr 2023

DIALITE：发现、对齐与集成开放数据表

Aamod Khatiwada 东北大学 美国马萨诸塞州波士顿 khatiwada.a@northeastern.edu

Roee Shraga 东北大学 美国马萨诸塞州波士顿 r.shraga@northeastern.edu

Renée J. Miller 东北大学 美国马萨诸塞州波士顿 miller@northeastern.edu

摘要

我们展示了一种名为 DIALITE 的新型表发现流水线，用户可以借此发现、集成和分析开放数据表。DIALITE 包含三个主要阶段。首先，它利用最先进的表发现技术，帮助用户从开放数据平台中发现表。其次，DIALITE 将发现的表集成起来，生成一个集成表。最后，它允许用户在集成表上执行各种下游任务，以分析集成结果。我们的处理流程具有灵活性，用户可以轻松添加并比较其他发现和集成算法。

CCS 概念

• **信息系统** →**信息集成**。

关键词

数据湖、数据发现、数据集成、全析取

ACM 引用格式：

Aamod Khatiwada、Roee Shraga 和 Renée J. Miller。2023。DIALITE：发现、对齐与集成开放数据表。在 *2023*年国际数据管理会议（*SIGMOD-Companion '23*）论文集，*2023*年*6*月*18*日至*23*日，美国华盛顿州西雅图。 ACM，美国华盛顿州西雅图，4 页。 https://doi.org/10.1145/3555041.3589732

1 引言

数据发现已成为数据科学处理流程中的重要组成部分。发现过程会使用关键词搜索等技术 [13] 和表搜索 [4, 7, 9] 来发现一组表（数据集）。数据科学家利用这些表支持决策过程、训练机器学习模型、执行统计分析等。发现之后，自然的一步便是集成已发现的表。集成表提供了数据的统一视图，并允许用户运行超越单个表范围的查询和分析。

集成表时，我们希望以尽可能充分的方式合并来自不同表的元组，使集成后的元组包含尽可能多的信息。这不仅丰富了集成后的分析和决策过程，也提升了下游应用的质量。众所周知，外连接 [12] 算子不满足结合律，也不以最大化集成后元组之间的连接 [6]。因此，全析取 *(FD)* [6]被视为一种自然的方式，用于组合部分信息片段（事实），从而最大化这些事实之间的连接 [11]。FD 可以看作是外连接的一种结合律版本 [2]，已被用于集成关系表 [2, 10] 和网页表中的信息 [10]。在最近的一篇论文中 [8],，我们提出了 ALITE，这是一种基于全析取的新算法，用于集成从数据湖中发现的表。研究表明，在实际集成开放数据湖表时，ALITE 是正确的，且比现有的 FD 算法更快 [2, 10] 。 [8]有关其他集成算子及其与 FD 的比较的更多详情，请参阅 ALITE 论文全文 [8]，其中证明 FD 是更好的集成语义，因为它能生成结果，使实体解析等下游任务的执行更加准确。

在本次演示中，我们提出了 DIALITE，这是一种能够发现、对齐并集成开放数据表的新型系统。它扩展了前述的 ALITE [8],，后者不执行发现，而是将一组表作为输入，使用整体模式匹配对齐匹配列，并应用 FD 得到一个集成后的表。DIALITE 提供了用于不同表发现任务的先进系统 [7, 14]，然后再使用 ALITE 对这些表进行集成。此外，DIALITE 还提供了新的下游分析（此前尚未考虑过），用于评估集成质量。具体而言，DIALITE 允许用户

(1) 上传一个表，或使用 GPT-3 随机生成一个表作为查询表，并从给定的数据湖中发现与查询表可并集 [7] 或可连接 [14] 的相关表。除了现有的表发现算法外，DIALITE 还允许用户根据自己的偏好添加新的表搜索算法。

(2) 使用名为 ALITE [8]的新型表集成系统集成发现的表（或上传一组待集成的表）。除了 ALITE，我们还允许用户在可扩展架构中添加新的集成运算符。

(3) 分析并比较使用 ALITE 集成的表与通过其他集成技术处理的表，方法是执行下游应用。在我们的演示中，我们将外连接视为一种替代集成技术（或用户添加的其他查询/方法），并将数据分析（常见的聚合和统计）以及实体解析作为下游应用。在真实表（可能不完整）上应用这两种方法时，可以看出使用 FD 与使用外连接来最大限度地集成信息之间的显著差异。

相关工作。 据我们所知，DIALITE 是首个能够实现完整表搜索流水线的系统，涵盖从

特此授权免费制作本作品全部或部分内容的数字副本或纸质副本，供个人或课堂使用，但不得为盈利或商业利益而制作或分发副本，且副本必须包含本声明及首页上的完整引文。对于本作品中由作者以外的其他人拥有版权的部分，必须尊重其版权。注明出处后，允许摘录。若要以其他方式复制、重新发布、上传至服务器或分发至列表，须事先取得具体许可和/或支付费用。请通过 permissions@acm.org 申请许可。*SIGMOD-Companion '23*，*2023*年*6*月*18*日至*23*日，美国华盛顿州西雅图
©2023 版权所有归所有者/作者所有。出版权已授权给 ACM。ACM ISBN 978-1-4503-9507-6/23/06...$15.00https://doi.org/10.1145/3555041.3589732

表发现，再到表集成，以及基于集成表的下游应用。早期有一个名为 CO- COA [3] 的系统，专注于查找具有相关属性、可扩展查询表的可连接表。找到这些表后，COCOA 会在查询表与发现的表之间应用左连接，作为集成算子。另一方面，DIALITE 会查找一组可与查询表集成的相关表（包括可并集表和可连接表）。Auctus [1]也会发现相关表，并通过应用内连接或并集来集成表对。与 Auctus 不同，DIALITE 不限制待集成表的数量。此外，DIALITE 使用函数依赖来集成表；理论上已证明，这种方法能够最大限度地增加表中元组之间的连接 [11]。

现有的表发现技术会搜索可连接、可并集或相关的表。例如，**JOSIE** [14] 和**LSH Ensemble** [15], 以查询表作为输入，由用户标记查询列，并返回一组可与查询表连接的前 k 个表。**SANTOS** [7] 以查询表作为输入，发现一组语义上可并集的表作为输出。其他表搜索技术 [4, 9] 也关注从大型存储库中进行表发现。但如何集成所发现的表这一重要问题尚未得到解决。**ALITE** [8]为这一问题提供了解决方案。由于数据湖表可能缺少一致且有意义的列标题，ALITE 会对所搜索表的集合执行整体模式匹配，并为匹配列集合分配一个名为"集成 ID"的虚拟列标题。随后，它基于集成 ID 应用自然函数依赖，以集成这些表。集成过程输出一个具有最大化集成元组的集成表 [6, 11]。

2 系统描述

DIALITE（如图 1所示）包含三个阶段（发现、对齐与集成、分析），这些阶段均纳入了我们的演示计划（第 3 节）。

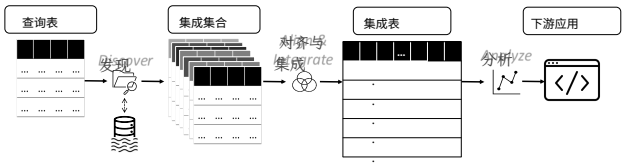

图 1：DIALITE 概览。

DIALITE 为发现、对齐、集成和分析提供了多种选项，包括允许用户自行实现方法。处理流程从用户提供的查询表开始 Q。不过，我们也允许用户使用 GPT-3 [5]随机生成表格。下面将详细介绍各个组成部分。

2.1 发现

给定查询表 Q，DIALITE 使用数据发现方法，在数据湖（表存储库）D 中查找可并集 [7], 可连接 [14], 或仅在语义上相似的表。DIALITE 支持用户从现有发现算法中进行选择，包括 SAN- T OS [7] 和 LSH Ensemble [15]。或者，用户也可以通过实现两个表之间的相似度方法来使用自己的发现算法。由于我们使用 ALITE [8] 进行对齐和集成，因此发现阶段与搜索类型无关。ALITE，旨在最大限度地增加表之间的关联，并决定应如何集成这些表。发现的输出是一个集成集 $D \subseteq$ D，即一组待集成的表，其中包括查询表。我们的发现技术允许用户控制返回的表的数量，系统也允许用户选择发现出的表的一个子集进行集成。

2.2 对齐与集成

给定一个集成集合 D，DIALITE 使用 ALITE [8] 来集成这些表。集成集合可以从发现部分（第 2.1 节）得出，也可以由用户作为输入提供。后一种情况代表传统的数据集成场景，即集成集合已给定。如图 1所示，此阶段的输出是一个集成表。

ALITE 由两个主要部分组成，即对齐和集成。请注意，我们不要求集成集合中的列标题可靠。ALITE 的第一部分（对齐）应用整体模式匹配，以识别集成集合中的公共列。匹配的列会被赋予相同的集成 ID。集成部分将这些 ID 用作属性名称，并应用（自然）函数依赖（FD）。具体而言，ALITE 使用一种新的整体模式匹配算法，该算法已被证明优于最先进的匹配器。ALITE 使用一种新颖的算法来计算函数依赖（FD），并已证明该算法在真实表（包括含有空值的表）上正确且高效 [8]。DIALITE 还允许用户添加其他集成算子，例如外连接，该算子用于演示。

2.3 分析

给定一个集成表（用户也可以将其作为文件上传），DIALITE 允许用户探索集成的益处。具体而言，用户可以选择一个下游应用，并将其应用于集成表。一个简单的应用是聚合查询，可以应用于集成表，正如我们在第3.1节中所示。在第3.2节中，我们还介绍了一个更复杂的下游应用：实体解析（ER）。

2.4 实现

DIALITE 使用 Python 3.8 实现，演示采用 Web 应用。[1] 在我们的处理流程中，我们使用 SANTOS [2]、LSH Ensemble [3] 和 ALITE [4]，并采用它们公开提供的代码。此外，我们使用 *py_entitymatching* 软件包，将 ER 展示为下游应用。[5] 我们允许用户在每个步骤完成后与系统交互，以便验证中间结果。

3 演示计划

接下来，我们将介绍演示。演示视频链接可在 GitHub 仓库中找到。[1] 我们的演示包含两部分。首先，我们展示第2节所述处理流程的应用案例。这部分实际上由三个可独立演示的演示项目组成。然后，我们演示 DIALITE 在发现、集成和分析新算法方面的可扩展性。

[1] https://github.com/northeastern-datalab/dialite [2]
https://github.com/northeastern-datalab/santos [3]
https://github.com/ekzhu/datasketch [4]
https://github.com/northeastern-datalab/alite [5]
https://github.com/anhaidgroup/py_entitymatching

3.1 DIALITE 用例

发现。 DIALITE 的用户可以从现有的表集合中上传 CSV格式的查询表。[6] 请注意，我们将在演示中提供一个供用户使用的数据湖。该数据湖中的表来自开放数据，目前已针对我们的发现算法进行了预处理。 [7, 15]具体而言，SANTOS 中使用的索引 [7] 和 LSH Ensemble 中使用的索引 [15] 均已离线构建，也就是说，用户可以直接使用这些索引。用户也可以轻松地对自己的数据湖进行预处理和链接。

所有表发现系统发现的表集合都会保存下来，以供下一步使用。由于可并集和可连接搜索结果可能存在重叠，我们会保留所有技术找到的表集合，以形成一个集成集合。

T1 (查询表)

TID	国家	City	疫苗接种率 (1+ 剂)
t_1	德国	柏林	63%
t_2	英格兰	曼彻斯特	78%
t_3	西班牙	巴塞罗那	82%

T2 (检索到的可并集表)

TID	国家	City	疫苗接种率 (1+ 剂)
t_4	加拿大	多伦多	83%
t_5	墨西哥	墨西哥城	±
t_6	USA	波士顿	62%

T3 (检索到的可连接表)

TID	City	病例总数	死亡率 (每10万居民)
t_7	柏林	1.4M	147
t_8	巴塞罗那	268万	275
t_9	波士顿	263k	335
t_{10}	新德里	2M	158

图 2：展示不同地区 COVID-19 病例的表格。符号 ± 表示输入表中的空值（“缺失值”）。

示例 1. 考虑表 T_1、 T_2 和 T_3，它们包含不同地点的 *COVID-19* 病例数据，如图 *2*所示。此处，*TID*（元组 *ID*）并非真实的数据列，仅用于指代元组。设 T_1 为查询表，表 T_2 和 T_3 位于数据湖存储库中。在我们的演示中，用户选择城市作为意图列和查询列，分别使用 *SANTOS* 搜索可并集的表 *[7]*，并使用 *LSH Ensemble* 搜索可连接的表 *[15]* 。有关搜索算法的更多细节，请参阅相应论文。设 T_2 和T_3 分别为并集搜索和连接搜索的结果。因此，发现步骤的结果是一个由表 T_1、 T_2和 T_3组成的集成集合。请注意，为简明起见，此处列出了列名；发现技术并不使用这些列名。这些技术旨在应对数据湖中数据的模糊性，即元数据不可靠或缺失的问题。

对齐与集成。 形成集成集合后，DI-ALITE 允许用户应用 ALITE 的整体架构匹配来生成集成 ID。我们基于这些 ID 应用 ALITE 的函数依赖。

示例 2. 考虑表的集成集合 T_1， T_2 和 T_3 所示图 *2* 是在表发现步骤之后形成的，如例 *1*所示。*DIALITE* 对这些表应用 *ALITE* 的集成算法，返回一个集成表，如图 *3*所示。集成语义在 *ALITE* 论文中有所解释 *[8]*。

[6]在演示中，我们还允许用户随机生成查询表（参见第3.2节）。此外，用户也可以选择上传查询表，但要注意该表可能与数据湖无关，因此可能无法获得任何结果。

FD(T_1, T_2, T_3)（ALITE 集成的表）

OID	TIDs	国家	City	疫苗接种率 (1+ 剂)	病例总数	死亡率 (每10万居民)
f_1	{t_1, t_7}	德国	柏林	63%	1.4M	147
f_2	{t_2}	英格兰	曼彻斯特	78%	⊥	⊥
f_3	{t_3, t_8}	西班牙	巴塞罗那	82%	268万	275
f_4	{t_4}	加拿大	多伦多	83%	⊥	⊥
f_5	{t_5}	墨西哥	墨西哥城	±	⊥	⊥
f_6	{t_6, t_9}	USA	波士顿	62%	263k	335
f_7	{t_{10}}	⊥	新德里	⊥	2M	158

图3：将 ALITE 应用于图2中的表所得的结果。符号 ⊥ 表示由于信息缺失，在集成过程中产生的空值（“产生的空值”）。

分析。 集成后，ALITE 提供的下一个功能是分析集成表。我们展示了如何对集成表执行聚合查询。

示例 3. 借助集成表，我们现在允许用户使用超越单个表范围的查询。例如，在集成表（图 *3*）中，用户可以发现波士顿的疫苗接种率最低，而多伦多最高。为了了解其中的原因，用户可以探索疫苗接种率（见 T_1 和 T_2）、病例数和死亡率（见 T_3）之间的关系。例如，用户可以计算疫苗接种率与死亡率之间的相关性，结果显示二者的（皮尔逊）相关系数为 0。16，而病例数与疫苗接种率之间的相关系数则为（有些出乎意料的）0。9。尽管这一分析结果有些违背直觉，却揭示了有关疫苗接种本质的一项有趣见解：在死亡率较高、病例更多的城市，政府会重点推行疫苗接种计划，而民众也更愿意接种疫苗。

3.2 DIALITE 的可扩展性

如第 2 节所述，用户可以通过在处理流程中实现自己的替代组件来扩展 DIALITE。具体而言，我们旨在展示用户实现新发现算法、集成方法并进行所需分析的能力（使用 Python 代码）。

示例 4. 为便于说明，下面提供了一段示例代码，用于添加一个新的可连接表发现算法，见 图 *4*。用户只需实现两个数据集（*df1* 和 *df2*）之间的相似度函数，*DIALITE* 就会用它来进行表发现。

```
def new_joinability_discovery_algorithm(df1, df2):
    join_df = pd.merge(df1, df2, how ='inner')
    return len(join_df)/max(len(df1), len(df2))
```

图 4：基于内连接实现用户自定义发现算法。

我们还考虑了这样一种场景：用户可能没有查询表来启动分析。因此，DIALITE 允许用户通过简单的提示生成用于分析的查询表。我们使用基于 GPT-3 的实现来展示此功能 [5]，并允许用户根据提示生成查询表，例如

如 图5所示。这里，我们生成了一个关于 COVID-19 病例的查询表，其中包含 5 列和 5 行。

```
query_table = dialite.randomly_generate_query_table(
    'a table about covid with 5 columns and 5 rows'
    )
query_table.head(5)
```
✓ 10.1s

	Country	Cases	Deaths	Recovered	Active
0	USA	5742812	178701	2633567	2930544
1	Brazil	3713876	116476	2788841	808559
2	India	3444061	61529	2643788	738744
3	Russia	982822	16841	745930	219051
4	Mexico	704016	73814	442309	187893

图5：使用 GPT-3 生成查询表的代码片段。

此外，用户可以在默认集成系统（ALITE）上添加其他集成算子。例如，图6展示了一个用户定义的代码片段，它实现了常用于集成的外连接算子。在演示中，我们通过以下示例说明使用 ALITE 相较于标准外连接的优势。

```
def new_outer_join_integration_algorithm(integration_set):
        table1_loc = integration_set.pop()
        table1 = pd.read_csv(table1_loc)
        for table2_loc in integration_set:
                table2 = pd.read_csv(table2_loc)
                table1 = table1.merge(table2, how = "outer")
        return table1
```

图6：将外连接实现为集成算子。

示例5。考虑表 T_4、 T_5 和 T_6，如图7*(a)* 所示；在表发现后，它们构成一个集成集合。这些表描述了 *COVID-19* 疫苗、其原产国以及批准这些疫苗的监管机构。为便于说明，假设用户使用外连接作为替代集成算法（见图*6*）。对这些表应用外连接和 *ALITE*（函数依赖）得到的结果分别如图*8(a)* 和图*8(b)* 所示。现在假设用户希望通过应用 *py_entity matching*。[5] 将实体消解（*ER*）作为下游应用。对外连接和函数依赖结果进行此分析的结果分别如图*8(c)* 和图*8(d)* 所示。外连接产生的输出元组比函数依赖更多；然而，它没有生成任何包含批准强生（*J&J*）疫苗的机构的元组。另一方面，函数依赖会生成一个输出元组 f_{13}，提供了这一信息（可通过 t_{13} 和 t_{15}生成）。此外，由于外连接会产生不完整的元组，实体消解无法解析 f_9 和 f_{10}。这表明，使用默认的 *ALITE* 算子而非外连接来集成这些表具有优势。

致谢

本研究部分由美国国家科学基金会通过资助编号 IIS-1956096 和 IIS-2107248 提供支持。

T_4

TID	疫苗	审批方
t_{11}	辉瑞	FDA
t_{12}	JnJ	±

T_5

TID	国家	审批方
t_{13}	美国	FDA
t_{14}	美国	±

T_6

TID	疫苗	国家
t_{15}	J&J	美国
t_{16}	JnJ	USA

图7：关于 COVID-19 疫苗、其原产国及审批机构的表集成集合。

$T_4 \bowtie T_5 \bowtie T_6$

OID	TIDs	疫苗	批准机构	国家
f_8	$\{t_{11}, t_{13}\}$	辉瑞	FDA	美国
f_9	$\{t_{12}\}$	JnJ	±	⊥
f_{10}	$\{t_{14}\}$	⊥	±	USA
f_{11}	$\{t_{15}\}$	J&J	⊥	美国
f_{12}	$\{t_{16}\}$	JnJ	⊥	USA

(a) 使用外连接生成的输出表

疫苗	批准机构	国家
辉瑞	FDA	美国
JnJ	±	⊥
⊥	±	USA
J&J	⊥	美国

(c) 对以下内容进行实体消解外连接结果

函数依赖(T_4, T_5, T_6)

OID	TIDs	疫苗	审批机构	国家
f_8	$\{t_{11}, t_{13}\}$	辉瑞	美国食品药品监督管理局	美国
f_{12}	$\{t_{16}\}$	JnJ	⊥	USA
f_{13}	$\{t_{13}, t_{15}\}$	J&J	FDA	美国

(b) 使用函数依赖生成的输出表（ALITE）

疫苗	批准机构	国家
辉瑞	美国食品药品监督管理局	美国
J&J	FDA	美国

(d) 对函数依赖进行实体消解（ALITE）结果

图8：使用 图7 中的外连接和函数依赖来集成表。

参考文献

[1] Sonia Castelo、Rémi Rampin、Aécio S. R. Santos、Aline Bessa、Fernando Chirigati 和 Juliana Freire。2021。Auctus：用于数据发现和数据增强的数据集搜索引擎。《VLDB 汇刊》14，12（2021），2791–2794。[2] Sara Cohen、Itzhak Fadida、Yaron Kanza、Benny Kimelfeld 和 Yehoshua Sagiv。2006。全析取：多项式延迟迭代器的实际应用。载于 VLDB 2006。ACM。http://dl.acm.org/citation.cfm?id=1164191[3] Mahdi Esmailoghli、Jorge-Arnulfo Quiané-Ruiz 和 Ziawasch Abedjan。2021。COCOA：相关系数感知数据增强。载于 EDBT 2021。331– 336 10 5441 002 2021 30 。https://doi.org/ 。 / / edbt. 。Grace Fan、[4] Jin Wang、Yuliang Li、Dan Zhang 和 Renée J. Miller。2023。利用上下文化列表示学习，从数据湖中发现语义相关数据集。《VLDB 汇刊》16，7（2023），1726–1739。https: //doi.org/ 10 14778 3574245 3574274 。 / 。Luciano Floridi 和 Massimo Chiriatti。2020。[5] GPT-3：其性质、范围、局限与影响。《心智与机器》 30，4（2020），681–694。[6] César A. Galindo-Legaria。1994。作为析取的外连接。载于 SIGMOD 1994 会议。ACM，348–358。https://doi.org/10.1145/191839.191908[7] Aamod Khatiwada、Grace Fan、Roee Shraga、Zixuan Chen、Wolfgang Gatterbauer、Renée J Miller 和 Mirek Riedewald。2023。SANTOS：基于关系的语义表联合搜索。《ACM 管理数据汇刊》1，1（2023），文章 9。https: //doi.org/ 10 1145 3588689 。 / Aamod Khatiwada、Roee Shraga、Wolfgang [8] Gatterbauer 和 Renée J. Miller。2022。集成数据湖表。《VLDB 汇刊》16，4（2022），932–945。https://doi.org/ 10 14778 3574245 3574274 。 / 。Fatemeh Nargesian、Erkang Zhu、[9] Ken Q. Pu 和 Renée J. Miller。2018。开放数据上的表联合搜索。《VLDB 汇刊》11，7（2018），813–825。https: //doi.org/ 10 14778 3192965 3192973 。 / 。Matteo Paganelli、Domenico Beneventano、[10] Francesco Guerra 和 Paolo Sottovia。2019。全析取计算并行化。《大数据研究》17（ 2019 18 31 10 1016 2019 07 002 ），–。https://doi.org/ 。 /j.bdr. 。[11] Anand Rajaraman 和 Jeffrey D. Ullman。 1996 。通过外连接和全析取集成信息（扩展摘要）。载于 PODS 1996。ACM。[12] Raghu Ramakrishnan 和 Johannes Gehrke。2003。数据库管理系统（第 3 版）。麦格劳-希尔。[13] Roee Shraga、Haggai Roitman、Guy Feigenblat 和 Mustafa Canim。 [n.d.]。使用多模态深度学习进行网页表格检索。载于 SIGIR 2020 会议。ACM。[14] Erkang Zhu、Dong Deng、Fatemeh Nargesian 和 Renée J. Miller。2019。JOSIE：在数据湖中查找可连接表的重叠集合相似性搜索。载于 SIG MOD 会议 *2019* 847 864 10 1145 3299869 3300065 。 ACM，–。[15] https://doi.org/ 。 / 。Erkang Zhu、Fatemeh Nargesian、Ken Q. Pu 和 2016 Renée J. Miller。 。LSH Ensemble：互联网规模域搜索。《VLDB 汇刊》9，12（2016），1185–1196。https://doi.org/10.14778/2994509.2994534