软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(12):5801−5820 [doi: 10.13328/j.cnki.jos.007372] [CSTR: 32375.14.jos.007372]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

ReproLink: 面向可复现性的科研数据管理系统*

黄小龙[1,2], 杨婧如[2,3], 柳 熠[2,3], 马 郓[2,4], 景 翔[2,5], 黄 罡[1,2]

[1](北京大学 计算机学院, 北京 100871)
[2](数据空间技术与系统全国重点实验室, 北京 100091)
[3](北京大数据先进技术研究院, 北京 100195)
[4](北京大学 人工智能研究院, 北京 100871)
[5](北京大学 软件与微电子学院, 北京 102600)
通信作者: 杨婧如, E-mail: okiyang@pku.edu.cn; 柳熠, E-mail: liuyi14@pku.edu.cn

摘 要: 科研成果的可复现性是科学研究可靠性的基本保证, 更是科学技术进步的基石. 然而, 当前学术界面临着严峻的可复现性危机, 大量在顶级期刊和会议上公开发表的科研成果无法复现. 在数据科学领域, 成果的可复现性面临着科研数据多源异构、计算流程复杂、计算环境复杂等挑战. 针对这些问题, 提出面向可复现性的科研数据管理系统 ReproLink. ReproLink 提出对科研数据的统一建模, 将科研数据抽象为包含标识、属性集、数据实体三要素的科研数据对象; 通过对于复现流程的细粒度建模, ReproLink 建立一种对多步骤复杂复现流程的精确描述方法. 通过代码和运行环境的一体化建模, ReproLink 消除不同环境中代码执行行为的不确定性给成果复现带来的影响. 对 ReproLink 的性能测试和实例分析表明, ReproLink 在百万级的数据规模下具有较好的性能表现, 在论文复现、复现相关数据的溯源等现实场景中具有实用价值. ReproLink 系统技术架构已集成到国内唯一专门面向科研院所的一体化综合管理与服务平台-科南软件, 支持国内数百家科研机构的成果复现需求.

关键词: 科研数据管理; 可复现性; 数字对象架构; 数据语用; 数据共享
中图法分类号: TP311

中文引用格式: 黄小龙, 杨婧如, 柳熠, 马郓, 景翔, 黄罡. ReproLink: 面向可复现性的科研数据管理系统. 软件学报, 2025, 36(12): 5801–5820. http://www.jos.org.cn/1000-9825/7372.htm
英文引用格式: Huang XL, Yang JR, Liu Y, Ma Y, Jing X, Huang G. ReproLink: Reproducibility-oriented Research Data Management System. Ruan Jian Xue Bao/Journal of Software, 2025, 36(12): 5801–5820 (in Chinese). http://www.jos.org.cn/1000-9825/7372.htm

ReproLink: Reproducibility-oriented Research Data Management System

HUANG Xiao-Long[1,2], YANG Jing-Ru[2,3], LIU Yi[2,3], MA Yun[2,4], JING Xiang[2,5], HUANG Gang[1,2]

[1](School of Computer Science, Peking University, Beijing 100871, China)
[2](National Key Laboratory of Data Space Technology and System, Beijing 100091, China)
[3](Advanced Institute of Big Data Technology, Beijing 100195, China)
[4](Institute for Artificial Intelligence, Peking University, Beijing 100871, China)
[5](School of Software and Microelectronics, Peking University, Beijing 102600, China)

Abstract: The reproducibility of scientific research results is a fundamental guarantee for the reliability of scientific research and the cornerstone of scientific and technological advancement. However, the research community is currently facing a serious reproducibility crisis, with many research results published in top journals and conferences being irreproducible. In the field of data science, the reproducibility of research results faces challenges such as heterogeneous research data from multiple sources, complex computational

* 基金项目: 北京市科技新星计划 (Z211100002121159); 数据空间技术与系统全国重点实验室资助项目
收稿时间: 2024-08-20; 修改时间: 2024-10-10; 采用时间: 2024-12-01; jos 在线出版时间: 2025-05-22
CNKI 网络首发时间: 2025-05-23

processes, and intricate computational environments. To address these issues, this study proposes ReproLink, a reproducibility-oriented research data management system. ReproLink constructs a unified model of research data, abstracting it into research data objects that consist of three elements: identifier, attribute set, and data entity. Through fine-grained modeling of the reproduction process, ReproLink establishes a precise method for describing multi-step, complex reproduction processes. By integrating code and operating environment modeling, ReproLink eliminates the uncertainties caused by different environments affecting code execution. Performance tests and case studies show that ReproLink performs well with data scales up to one million records, demonstrating practical value in real-world scenarios such as paper reproduction and data provenance tracking. The technical architecture of ReproLink has been integrated into Conow Software, the only integrated comprehensive management and service platform in China specifically designed for scientific research institutes, supporting the reproducibility needs of hundreds of such institutes across the country.

Key words: scientific data management; reproducibility; digital object architecture (DOA); data provenance; data sharing

科研成果的可复现性 (reproducibility) 是科学研究的基石, 它不仅是研究结论可验证的基本保证, 也是研究人员在现有工作基础上推进新研究的基本要求[1−3]. 然而, 许多科研领域当前存在研究成果的可复现性较差的问题, 严重影响了科学研究的可靠性, 这一现象被称为可复现性危机 (reproducibility crisis)[4−7].

随着大数据时代的到来, 数据科学成为越来越重要的一种研究范式[8,9]. 在这一趋势下, 包括医学、社会科学、地球科学在内的各个领域的多种学科都迎来了研究范式的重要转变[9−12]. 在数据科学的语境下, 可复现性一般指计算可复现性 (computational reproducibility), 也就是“在相同的分析条件下, 使用相同输入数据、相同计算方法, 得到与原论文一致的计算结果”[1]. 和传统研究的复现相比, 对于数据科学研究的复现具有独特的意义: 在复现的基础上, 研究人员可以在新的场景下诱发数据集、代码等发挥新的价值. 例如百度人工智能团队在复现 BERT[13] 的基础上通过新增中文数据集并且修改模型掩码模块使其适配中文, 最终形成了百度的文心一言大语言模型[14]. 文心一言复用了 BERT 的大部分数据集、模型及代码, 在这一过程中, 科研人员从现有的数据集、代码中挖掘出了新的价值, 极大促进了 BERT 所使用的 English Wikipedia 和 BookCorpus 等数据集以及 BERT 源代码的价值释放.

当前, 数据科学研究的可复现性面临着诸多挑战. 首先, 数据科学研究中使用的数据是多源异构的, 这使得研究者获取复现研究成果所需科研数据的过程更加复杂. 大量科研数据分散式地存储在不同平台中, 这些平台之间在数据访问协议和数据格式上存在异构性, 缺乏对科研数据资源的统一建模. 这阻碍了对科研数据的自动化处理, 增加了复现过程中获取数据的难度. 其次, 数据科学研究的复现流程是复杂的. 对一篇论文的复现往往需要经过多阶段的数据处理, 其使用的数据集和代码之间具有复杂的逻辑关系. 在大部分的数据科学研究中, 研究者对于数据的计算流程以自然语言的形式分散在论文或论文代码的文档中, 这种记录方式无法精确刻画对数据的处理过程, 复现者理解的细微偏差也可能导致最终得出和原作者不同的结果. 最后, 数据科学研究的代码执行环境是复杂的, 其中往往存在大量的第三方依赖. 要复现某项成果, 必须先复现其执行环境. 然而, 现有的科研代码共享形式无法较好地支持对执行环境的复现. 例如一项对 Harvard Dataverse 上开源的 R 语言科研代码的研究显示, 38% 的代码由于执行环境不适配而无法执行[15]. 总而言之, 数据科学研究的复现性因数据的多源异构性、复现流程的复杂性以及执行环境复杂性而面临严峻挑战.

针对这些问题, 本文提出了一个面向可复现性的科研数据管理系统 ReproLink. ReproLink 对论文、代码、数据集这 3 类科研数据建立了统一模型, 基于数据语用理论 (data pragmatics)[16,17]提出了复现单元的概念. 在此基础上, ReproLink 实现了对于存储在分散式数据仓库中的科研数据的统一管理, 从而缓解了数据的多源异构性带来的影响. 在 ReproLink 中, 论文、代码和数据集被抽象为包含标识、属性集、数据实体三要素的科研数据对象. 其中标识为一个全局唯一的字符串, 用于定位和获取科研数据对象; 属性集为一组键值对的集合, 包括科研数字对象的各种基本属性, 主要用于描述和发现科研数据对象; 数据实体为科研数据对象包含的数据本身, 例如论文 PDF, 数据集压缩文件, 源代码等. 其次, 为了支持对复杂复现过程的精确描述, ReproLink 将复现过程中的单个运算步骤建模为复现单元. 复现单元关联了作为计算场景的论文对象, 作为计算过程的代码对象, 以及若干作为计算输入的数据集对象, 本质上刻画了这些数据集对象在此数据科学研究场景中的数据语用. 通过多个复现单元与论文、代码、数据集对象之间构成的网络, ReproLink 提供了一种对于复杂复现流程的精确描述方法, 使得多阶段的复现流程可

© 中国科学院软件研究所 http://www.jos.org.cn

以被一键执行. 最后, 为了缓解复现环境复杂性的影响, ReproLink 对代码和运行上下文环境进行了一体化建模, 为论文作者和复现者提供了一种共享复现环境的机制, 避免了不同的上下文运行环境对代码执行行为的影响, 从而提升了代码的可复现性. 此外, ReproLink 将科研数据之间的关系一阶实体化, 在统一的框架中对科研数据之间多种不同类型的关系进行了表征, 并在多种类型的科研数据对象和数据关系之上建立了科研数据关系图, 支持基于路径模式匹配的多跳数据查询和基于元数据匹配的关键字查询, 进而实现成果复现相关科研数据的高效发现.

本文的主要贡献如下.

1) 提出了对科研数据的统一建模, 将论文、代码、数据集这 3 类数据抽象为包含标识、属性集、数据实体三要素的科研数据对象; 基于数据语用原理, 提出了复现单元模型, 通过多个复现单元相互关联形成的网络实现对复杂复现过程的精确描述; 对科研数据之间不同类型的关系进行了统一表征, 并进一步建立了科研数据关系图, 满足不同复现场景下的科研数据发现需求.

2) 设计并实现了一个面向可复现性的科研数据管理系统 ReproLink. ReproLink 基于数字对象架构 (digital object architecture, DOA)[18]实现了科研数据对象的统一建模与分散式存储管理, 通过代码和上下文运行环境的一体化建模增强了代码的可复现性, 并实现了基于路径匹配的多跳关系查询以满足不同复现场景下的科研数据发现需求.

3) 使用真实数据对 ReproLink 进行了性能测试, 实验结果表明 ReproLink 在百万级的数据规模下具有较好的性能, 且其性能具有良好的可扩展性; 对 ReproLink 进行了用户反馈研究, 用户普遍认为 ReproLink 具有较好的易用性和实用性; 在实际场景下对 ReproLink 进行了用例分析, 结果表明 ReproLink 在论文复现、复现相关数据资源的溯源等场景中具有实际的应用价值.

ReproLink 系统技术架构已集成到国内唯一专门面向科研院所的一体化综合管理与服务平台——科南软件, 支持国内数百家科研机构的成果复现需求.

本文第 1 节介绍与本文研究相关的工作. 第 2 节介绍科研数据的统一建模并对相关概念进行形式化定义. 第 3 节介绍 ReproLink 的系统架构和几个主要模块的设计. 第 4 节对 ReproLink 的原型系统进行性能测试, 用户反馈研究和实例分析. 第 5 节总结全文的主要工作.

1 相关工作

1.1 科研数据共享平台

Papers with Code[19]是 Meta 发起的一个对机器学习论文、代码、数据集进行共享的平台, 它以众包的方式收集论文相关的代码和数据集信息. 分析论文和任务之间的关系. Hugging Face[20]主要关注模型和数据集的开放共享, 通过数据集和模型的元数据, 用户可以在平台上对数据集和模型进行细粒度的筛选. Hugging Face 还支持代码的在线执行功能, 在线执行的代码可以直接使用平台上的数据, 而不需要把平台上的模型或数据下载到本地. GitHub[21]、Gitee[22]、GitLab[23]等开源代码平台提供了代码的开放访问, 可以通过 HTTP、SSH、Git 等多种传输协议进行代码的传输. 许多论文作者将自己论文中所使用的代码上传到这些开源代码平台上, 以便相关领域的研究者使用. 然而, 这些平台的主要设计目标是数据的开放共享, 并不将科研成果的可复现性作为其主要目标. 例如, Papers With Code 中仅将论文和代码仓库进行简单关联, 并不对代码的执行环境进行建模, 也不对一次论文复现过程中的多次代码执行进行细粒度的建模. 本文的工作将单个论文的复现流程拆分为细粒度的复现单元, 提供了一种对复杂复现过程进行精确描述的方式, 并通过代码和运行环境的一体化建模消除了不同执行环境对论文复现的影响, 是对现有科研数据共享平台的重要补充.

1.2 数字对象架构

数字对象架构 (DOA)[18]是由 Kahn 等人提出的一种新型网络架构. DOA 将网络中的数据抽象为包含标识、元数据、数据实体三要素的数字对象 (digital object, DO). 在 DOA 中, 数字对象的标识由标识解析系统进行发放和解析, 元数据由数字对象注册表进行索引以提供对数字对象的检索服务, 数据实体则保存在数字对象所有者本

© 中国科学院软件研究所 http://www.jos.org.cn

地的数字对象仓库中. 通过数据实体和元数据分离的方式, DOA 使得没有权限访问数据实体的数据使用者也可以通过元数据发现数据. 通过对现有互联网上分散的异构信息系统进行统一管理、协同操作, DOA 旨在消除这些系统之间的数据孤岛, 实现异构、异地、异主数据的互通互联和互操作[24].

ReproLink 基于数字对象架构实现, 将论文、代码、数据集这 3 类科研数据对象实现为 DO, 并将属性集实现为 DO 的元数据. ReproLink 为每个科研数据对象注册全局唯一的标识, 并将它们的数据实体保存在数字对象仓库中. ReproLink 使用标识解析协议 (identifier resolution protocol, IRP)[25]和 DOA 中的标识解析系统通信, 并通过数字对象接口协议 (digital object interface protocol, DOIP)[26]和数字对象仓库通信.

1.3 科研数据关系挖掘

科研数据之间的关系蕴含着丰富的信息, 具有重要的价值. 许多工作使用统计学、图学习等领域的算法对这些科研数据之间的关系进行挖掘, 从中得出了有意义的结论. Small[27]基于两篇论文引用文献的相似度, 提出了一种新的衡量文档相似度的度量标准. Nayyeri 等人[28]提出了一种嵌入模型 Trans4E, 可以根据现有的科研数据之间的关系预测新的关系. Effendy 等人[29]通过分析计算机领域的论文引用关系和论文关键词, 对计算机领域的研究趋势进行了分析和预测. 通过科研数据间关系的一阶实体化, ReproLink 为科研数据之间的关系挖掘提供了一个统一的框架, 从而促进了这些对科研数据之间关系进行分析的工作.

Microsoft academic graph (MAG)[30]是一个由科研数据及他们之间关系构成的知识图谱, 其中的实体包括文章、专利、代码、数据集等. SemOpenAlex [31]中囊括了论文、作者、期刊、概念等多种实体, 汇聚了超过 2600 万条表示数据间关系的三元组. Linked Papers with Code[32]是基于 Papers with Code 数据集, 其中包括 40 万个机器学习领域的论文, 以及这些论文与他们对应的代码之间的关系. ReproLink 使得创建面向可复现性的类似形式的数据集成为可能, 在未来也可以利用这些现有数据集中的关系来加强 ReproLink 中基于数据之间的关系进行数据发现的能力.

2 概念定义

在数据科学研究的复现过程中, 论文、代码和数据集是 3 类必不可少的科研数据. 论文是复现的整体场景, 其中包含了研究的背景和方法论, 若去掉论文, 研究的复现便退化成为单纯的代码执行, 失去了其本来的意义. 代码是复现的具体计算步骤, 若没有代码, 即使有论文中的方法论和算法描述, 也难以准确复现研究中的实现细节. 数据集是用于复现的基础材料, 若没有研究的原始数据集, 实验就无法基于相同的数据进行, 从而无法验证结果与原论文是否一致. 而若能成功获取这 3 类数据, 复现者便可将原始数据集作为输入, 运行复现代码, 并将最终结果与论文比对, 验证复现是否成功.

ReproLink 将论文、代码和数据集建模为包含标识、属性集以及数据实体 3 类数据要素的科研数据对象, 在统一的框架下实现了对异构科研数据的表征. 其中, 标识为一个全局唯一的字符串, 数据使用者可以通过标识定位和访问科研数据对象. 属性集为一组键值对的集合, 包含了科研数据对象的基本信息, 通过对于属性集的匹配查询, 可以实现对科研数据对象的检索和发现. 数据实体则为科研数据对象包含的数据本身.

科研数据对象的模型建模了科研数据的语义信息, 其元数据和数据实体包含了科研数据本身固有的, 不受外界因素影响的含义. 然而, 科研数据对象的模型并不包含数据在某个特定的复现场景中表现出的信息. 为了表征这一信息, ReproLink 基于数据语用原理[16]对科研数据在具体复现场景中的含义进行了建模. 数据语用即“数据在特定的语境中的含义”[16], 指的是数据与应用的具体结合方式. ReproLink 通过复现单元来刻画科研数据的数据语用. 复现单元为一类特殊的科研数据对象, 代表了复现中的单个计算步骤, 为一次复现的最小组成单位. 复现单元通过关系对象与论文、代码、数据集以及其他复现单元之间联系, 每个复现单元关联了作为计算场景的论文、作为计算过程的代码对象, 以及若干作为输入数据的数据集对象或其他复现单元. 对于一篇论文的完整复现过程由一系列复现单元以及他们之间的关系对象构成.

定义 1 (**科研数据对象**). 具备以下特征的对象被称为科研数据对象.

1) 科研数据对象 $O=(I,P,E)$, 其中 I 为标识, P 为属性集, E 为数据实体. 设全体科研数据对象的集合为 $\mathbb{O}$, 全

体标识的集合为$\mathbb{I}$, 全体属性集的集合为$\mathbb{P}$, 全体数据实体的集合为$\mathbb{E}$, 函数$\pi_I:\mathbb{O}\to\mathbb{I}$, $\pi_P:\mathbb{O}\to\mathbb{P}$, $\pi_E:\mathbb{O}\to\mathbb{E}$分别为科研数据对象到其标识、属性集、数据实体的映射. 即对于科研数据对象$O=(I,P,E)$, 有$\pi_I(O)=I$, $\pi_P(O)=P$, $\pi_E(O)=E$.

2) 标识$I\in\mathbb{I}$为一个字符串, 用于唯一标识和寻址$\mathbb{O}$中的每个科研数据对象. $\forall O_1,O_2\in\mathbb{O}$, $O_1\neq O_2$满足$\pi_I(O_1)\neq\pi_I(O_2)$. 且$\forall I\in\mathbb{I},\exists O\in\mathbb{O}$满足$\pi_I(O)=I$, 即$\pi_I(O)$为双射. 设$\pi_I$的反函数为$\pi_I^{-1}:\mathbb{I}\to\mathbb{O}$. π_I^{-1}表示了使用唯一标识定位和访问科研数据对象的过程.

3) 属性集P为一个键值对的集合, 对于属性$p_i\in P$, 有$p_i=(k_i,v_i)$, 其中k_i为该属性的键, v_i为该属性的值, 满足$\forall i,j,k_i\neq k_j$. 科研数据对象O属性的键集合和值集合分别记作K_O,V_O, 有$K_O=\{k_i\mid\exists v_i,(k_i,v_i)\in\pi_P(O)\}$, $V_O=\{v_i\mid\exists k_i,(k_i,v_i)\in\pi_P(O)\}$. 函数$\phi_O:K_O\to V_O$为科研数据对象$O$的属性映射函数, $\forall k\in K_O$, $\phi_O(k)=v$当且仅当$(k,v)\in\pi_P(O)$. 对于类型为t科研数据对象O^t, 存在一组必要属性集K_t, 表示这种类型的科研数据对象必须包含的属性, 满足$\forall O^t$, $K_t\subset K_{O^t}$. 除必要属性集规定的属性外, 用户还可通过增加可选属性的方式扩展系统的表示形式, 属性集P主要包括用于描述和发现科研数据对象的相关元数据.

4) 所有科研数据对象的属性集中都包含其类型, $\forall O\in\mathbb{O},type\in K_O$. 称$\phi_O(type)$为科研数据对象$O$的类型, $\phi_O(type)\in\{code,paper,dataset,reproduction\}$.

5) 数据实体E表示科研数据对象所包含数据资源的实际内容, 为一个字节序列, 例如论文的 PDF 文件、数据集文件、源代码文件等.

图 1 中的对象①–⑧即为科研数据对象. 以对象③即文献 [33] 所对应的科研数据对象为例. 图 1 右侧的示例展示了对象③的标识、属性集和数据实体. 其标识为文献 [33] 的 DOI, 属性集包括此论文的标题、作者等信息, 数据实体即为此论文的 PDF 文件.

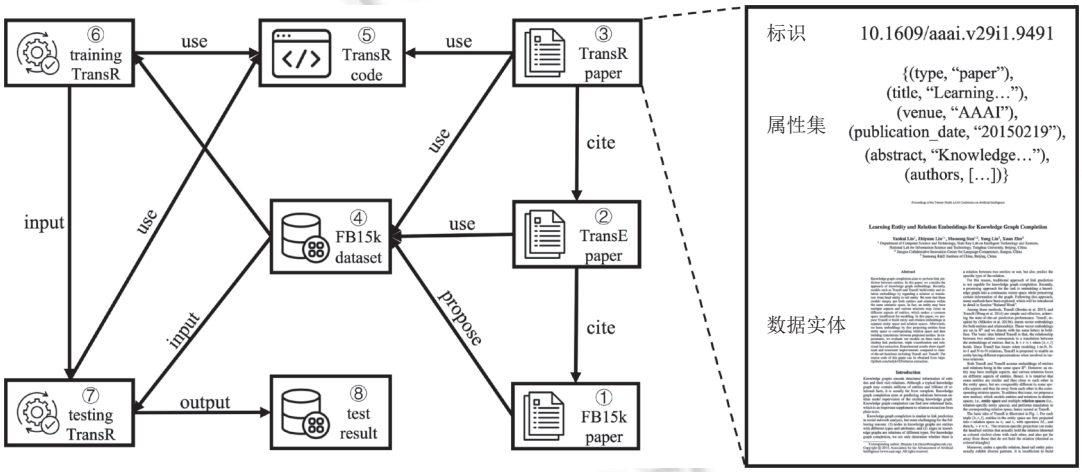

图 1 科研数据关系图示例

本文的研究主要关注科研中使用的论文、代码、数据集这 3 类数据, 因此科研数据对象包括论文对象、代码对象和数据集对象这 3 种类型. 为了一致地表示复现单元和其他科研数据对象之间的关系, 本文将复现单元也建模为一类特殊的科研数据对象. 实际上, 也可以将模型扩展到其他类型的科研数据, 以满足不同领域的科研数据管理需求. 下面依次给出这些科研数据对象的定义.

定义 2 (论文对象). 对于$O^P\in\mathbb{O}$, 若$\phi_{O^P}(type)=paper$, 则称O^P为论文对象. 论文对象的数据实体$\pi_E(O^P)$为论文 PDF 文件. 定义论文对象的必要属性集为$K_{paper}=\{title,venue,abstract,publication_date,authors\}$, 对于任意的论文对象$O^P$, 都有$K_{paper}\subset K_{O^P}$, 并且有:

1) $\phi_{O^P}(title)$为论文的标题;

2) $\phi_{O^P}(venue)$为论文发表的期刊或会议;

3) $\phi_{O^P}(abstract)$ 为论文的摘要;

4) $\phi_{O^P}(publication_date)$ 为论文发表的时间;

5) $\phi_{O^P}(authors)$ 为论文的作者列表.

图 1 中的对象①、对象②和对象③分别为文献 [34]、文献 [35] 以及文献 [33] 对应的论文对象. 文献 [34] 提出了 FB15k 数据集; 文献 [35] 提出了一种对于知识图谱中结点的嵌入算法 TransE, 并在 FB15k 数据集上进行测试; 文献 [34] 基于 TransE 提出了另一种嵌入算法 TransR. 其中对象③的标识、属性集和数据实体展示在右侧. 其属性集合包括 *type* 属性和上述 5 种必要属性.

定义 3 (代码对象). 对于 $O^C \in \mathbb{O}$, 若 $\phi_{O^C}(type) = code$, 则称 O^C 为代码对象. 代码对象的数据实体 $\pi_E(O^C)$ 包括代码的运行环境和代码自身, 可以表示为 $E^C = (environment, code)$, 其中 *environment* 表示代码的运行环境, 为一个 Docker 镜像 (dockerimage), *code* 表示代码文件, 为一个 zip 压缩包. 定义代码对象的必要属性集为 $K_{code} = \{description, authors\}$, 对于任意的代码对象 O^C, 都有 $K_{code} \subset K_{O^C}$, 并且有:

1) $\phi_{O^C}(description)$ 为对代码内容的自然语言描述,

2) $\phi_{O^C}(authors)$ 为代码的贡献者列表.

设 π_{EE} 为代码对象到其环境的映射, π_{EC} 为代码对象到其代码文件的映射. 即对于 $O^C \in \mathbb{O}$, 若 $\pi_E(O^C) = (environmnet, code)$, 则 $\pi_{EE}(O^C) = environment$, $\pi_{EC}(O^C) = code$.

图 1 中的对象⑤为 TransR 算法的代码实现对应的代码对象, 此代码对象实现了论文对象③中的算法. 对象⑤的运行环境为一个包括 Python 和相关依赖包的 Docker 镜像, 其代码文件为实现了 TransR 算法的一组 Python 程序文件组成的压缩包.

定义 4 (数据集对象). 对于 $O^D \in \mathbb{O}$, 若 $\phi_{O^D}(type) = dataset$, 则称 O^D 为数据集对象. 数据集对象的数据实体 $\pi_E(O^D)$ 表示数据集内容, 为一个 zip 压缩包. 定义数据集对象的必要属性集 $K_{dataset} = \{name, description, authors, date\}$, 对于任意的数据集对象 O^D, 都有 $K_{dataset} \subset K_{O^D}$, 并且有:

1) $\phi_{O^D}(name)$ 为数据集的名称;

2) $\phi_{O^D}(description)$ 为对数据集内容的自然语言描述;

3) $\phi_{O^D}(authors)$ 为数据集的作者列表;

4) $\phi_{O^D}(date)$ 为数据集的发布日期.

图 1 中的对象④为 FB15k 数据集对应的数据集对象. 此数据集对象由论文对象①提出, 在论文对象②和论文对象③中被使用. 对象⑧为算法 TransR 在 FB15k 数据集上进行测试的输出结果构成的数据集对象.

定义 5 (复现单元). 对于 $O^R \in \mathbb{O}$, 若 $\phi_{O^R}(type) = reproduction$, 则称 O^R 为复现单元. 复现单元是数据实体为空的特殊科研数据对象. 定义复现单元的必要属性集 $K_{reproduction} = \{name\}$, 对于任意的复现单元 O^R, 都有 $K_{reproduction} \subset K_{O^R}$, $\phi_{O^R}(name)$ 表示复现单元的名称.

图 1 中的对象⑥和对象⑦为复现单元, 分别表示论文③的复现过程中的训练阶段和测试阶段. 在对象⑥对应的训练阶段中, TransR 代码对象 (对象⑤) 以 FB15k 数据集 (对象④) 的训练数据集为输入, 运行 TransR 训练算法, 并输出训练得到的模型参数; 在对象⑦对应的测试阶段中, TransR 代码对象以 FB15k 数据集的测试数据集和训练阶段输出的模型参数作为输入, 运行 TransR 测试算法, 并输出最终的测试结果.

前文对各类科研数据对象的内容进行了详细定义, 接下来将对这些科研数据对象之间的关系进行建模. ReproLink 使用关系对象表示科研数据对象之间的关系.

定义 6 (关系对象). 关系对象 $r = (u, v, t)$, 其中 u 为关系起点对象的标识, v 为关系终点对象的标识, $u, v \in \mathbb{I}$. t 为关系对象的类型, $t = (t_s, t_u, t_v)$, 其中 t_s 为语义类型, 表示这个关系的语义; t_u, t_v 别为起点类型和终点类型, 分别表示起点对象和终点对象的类型, 其值分别为 $\phi_{\pi_I^{-1}(u)}(type)$ 和 $\phi_{\pi_I^{-1}(v)}(type)$.

表 1 列出了 ReproLink 中所有关系对象的类型及其含义. 设全体关系对象的集合为 $\mathcal{R}$, 所有关系对象的类型的集合为 $\mathcal{T}$, 函数 $\pi_u : \mathcal{R} \to \mathbb{I}, \pi_v : \mathcal{R} \to \mathbb{I}, \pi_t : \mathcal{R} \to \mathcal{T}$ 分别为关系对象到其起点、终点、类型的映射. 即对于关系对象 $r = (u, v, t)$ 满足 $\pi_u(r) = u$, $\pi_v(r) = v$, $\pi_t(r) = t$.

© 中国科学院软件研究所 http://www.jos.org.cn

表 1 关系对象的类型

语义类型	起点对象类型	终点对象类型	含义
use	*paper*	*code*	论文u使用代码v
use	*paper*	*dataset*	论文u使用数据集v
cite	*paper*	*paper*	论文u引用论文v
propose	*paper*	*dataset*	论文u提出数据集v
use	*reproduction*	*code*	复现u中使用了代码v
input	*dataset*	*reproduction*	数据集u为复现v的输入数据集
input	*reproduction*	*reproduction*	复现u的输出为复现v的输入数据集
output	*reproduction*	*dataset*	复现u的输出数据集为数据集v

图 1 中对象①–⑧之间的有向边为关系对象, 由于空间有限, 图中仅展示了关系的语义类型. 代码对象⑤是对论文对象③的一个实现, 因此他们之间存在类型为 $(use, paper, code)$ 的关系对象; 论文对象②和论文对象③使用了数据集对象④, 因此他们之间存在类型为 $(use, paper, dataset)$ 的关系对象; 论文对象③引用了论文对象②、论文对象②引用了论文对象①, 因此他们之间存在类型为 $(cite, paper, paper)$ 的关系; 数据集对象④在论文对象①中被提出, 因此他们之间存在类型为 $(propose, paper, dataset)$ 的关系. 复现单元⑥和复现单元⑦均使用了数据集对象④作为输入, 因此它们之间分别存在类型为 $(input, dataset, reproduction)$ 的边. 复现单元⑥和复现单元⑦均使用了代码对象⑤作为输入, 因此它们之间存在类型为 $(use, reproduction, code)$ 的边. 复现单元⑦的另一个输入为复现单元⑥中训练得到的数据模型, 因此它们之间存在类型为 $(input, reproduction, reproduction)$ 的边. 最后, 复现单元⑦的输出为数据集⑧, 因此它们之间存在类型为 $(output, reproduction, dataset)$ 的边.

将科研数据对象看作节点, 关系对象看作边, 可以得到科研关系图.

定义 7 (科研数据关系图). 对于一个科研数据对象的集合 $N \subset \mathbb{O}$ 和一个关系对象的集合 $R \subset \mathcal{R}$, 若其满足 $\forall r \in R$ 有 $\pi_I^{-1}(\pi_u(r)) \in N$, $\pi_I^{-1}(\pi_v(r)) \in N$, 则称二元组 $\mathcal{G} = (N, R)$ 为一个科研关系图.

图 1 中所有的科研数据对象和关系对象构成了一个科研关系图 $\mathcal{G}_1$. 对象④–⑧以及它们之间的关系也构成了一个科研关系图 $\mathcal{G}_2$. 其中 $\mathcal{G}_2$ 表示了对论文对象③的一次完整的复现流程, 包含两个复现单元: 复现单元⑥表示论文复现中的训练过程, 其使用数据集④中的训练数据和验证数据, 调用代码对象⑤中的训练代码, 输出训练完成的模型; 复现单元⑦表示论文复现中的测试过程, 其使用数据集④中的测试数据以及复现单元⑥输出的模型, 调用代码对象⑤中的测试代码, 并最终输出测试结果数据集⑧.

3 ReproLink 系统设计

ReproLink 向用户提供科研数据对象访问、科研数据对象检索、科研成果复现执行这 3 类服务. 本节首先对 ReproLink 的主要功能和总体架构进行了介绍, 接着详细介绍了其中的数据管理模块、对象检索引擎、复现执行引擎、分散式对象存储、标识解析服务等模块.

3.1 总体架构

ReproLink 的总体架构如图 2 所示, 包括以下几个模块.

1) 控制器: 接收用户的请求, 根据请求的类型将其分发至对象检索引擎、数据管理模块、复现执行引擎, 并在请求执行完毕后向用户返回结果.

2) 数据管理模块: 实现了对科研数据对象的创建、访问、修改和删除等功能, 向用户以及复现执行引擎提供服务. 数据管理模块将科研数据对象的标识和他们的存储位置、状态信息存储在本地的标识索引中. 当用户请求的标识不在标识索引中时, 数据管理模块通过标识解析器向全局标识解析服务发出标识解析请求, 并将返回结果保存在标识索引中. 在解析了数据对象的存储位置后, 数据管理模块通过操作执行器操作分散式数据对象存储中的数据.

© 中国科学院软件研究所 http://www.jos.org.cn

3) 对象检索引擎: 向用户提供关键字搜索和路径匹配搜索功能, 关键字搜索使用基于科研数据对象属性建立的关键字索引实现, 路径匹配搜索使用基于关系对象建立的关系索引实现.

4) 复现执行引擎: 实现了在线代码执行和论文复现. 调用数据管理模块在一次论文复现中汇聚分散式存储的多个数据对象, 为代码执行分配算力, 并持续监控代码执行状态.

5) 分散式对象存储: 科研数据对象以分散式存储的形式保存在全球各地的数据仓库中. 这些数据仓库可以由数据所有者维护或第三方机构维护. 这些数据仓库中的科研数据对象由数据管理模块进行管理.

6) 标识解析服务: 标识解析服务负责全体科研数据对象的标识分配和寻址. 在创建科研数据对象时, 由标识解析服务为新创建的科研数据对象分配一个全局唯一的标识; 在对科研数据对象进行访问、更新、删除等操作时, 标识解析服务通过给定的标识定位目标科研数据对象, 并返回其所在的数据仓库.

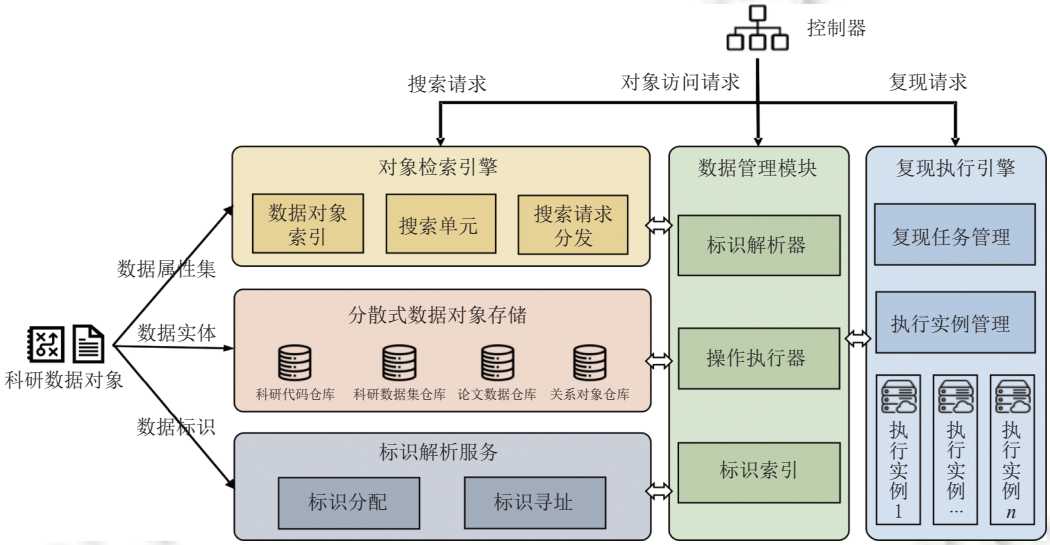

图 2 ReproLink 总体架构

3.2 数据管理模块

ReproLink 中的数据分散式地存储于数据源的数据仓库中, 在用户访问数据、复现过程中使用数据时, 由数据管理模块和各个数据仓库通信, 以汇聚来自不同主体的多源异构的数据. 除此之外, 数据管理模块还负责对系统中的科研数据对象进行创建、修改、删除等操作. 数据管理模块包含 3 个子模块, 分别是标识索引、标识解析器、操作执行器.

ReproLink 中的科研数据对象分散式地存储于各个数据仓库中, 为了对给定标识的科研数据对象执行某项操作, 需要先获得这个科研数据对象所在仓库的地址, 这一操作通过标识索引完成. 标识索引是一个关系型数据库, 其中每一行的内容为一个科研数据对象的标识、类型、所在仓库地址, 表 2 给出了标识索引的一个示例. 这个数据库在标识列上建立索引, 可以根据标识快速定位数据行. 数据管理模块收到对科研数据对象的访问、修改、删除请求时先使用请求中的标识在本地的标识索引中查询此标识对应的仓库地址和基本属性, 接着使用操作执行器向该仓库地址发送对此科研数据对象的请求.

若在本地的标识索引中不存在所请求的标识, 数据管理模块会通过标识解析器使用 IRP 协议向外部的标识解析服务发出标识解析请求. 若这一请求同样失败, 则说明数据管理模块收到的请求中的标识是非法标识, 因此返回错误信息. 若标识解析服务成功返回了请求对象的地址和基本属性, 则数据管理模块将这一标识对应的地址和基本属性存入标识索引, 从而后续对于同一数据对象的操作可以直接使用本地的标识索引.

© 中国科学院软件研究所 http://www.jos.org.cn

表 2 标识索引示例

id	标识	地址	类型
1	10.1109/5.771073	tcp://52.25.150.45:4021	*paper*
2	10.1038/nphys1170	udp://198.51.100.14:8765	*paper*
3	10.18632/oncotarget.22303	tls://143.198.47.108:6443	*paper*
4	10.6084/m9.figshare.12345678	ws://203.0.113.77:5050	*code*
5	10.5281/zenodo.9999991	tcp://95.85.31.123:2222	*dataset*

在获得了科研数据对象所在的仓库地址和基本信息后, 数据管理模块中的操作执行器向数据仓库发送对科研数据对象的操作请求. 操作执行器可以执行科研数据对象的创建、获取、修改、删除这 4 种操作. 它和数据仓库之间的通信使用 DOIP 协议. DOIP 协议支持多种传输层协议, 在执行请求时, 操作执行器根据在本地保存的仓库信息选择合适的传输层协议, 并向数据仓库发出对数据的操作请求.

3.3 对象检索引擎

对象检索引擎采用分层架构, 分为请求分发模块、搜索单元、数据对象索引这 3 部分. 图 3 展示了对象检索引擎的基本架构, 并以一次路径匹配搜索为例, 展示了对象检索服务的执行过程.

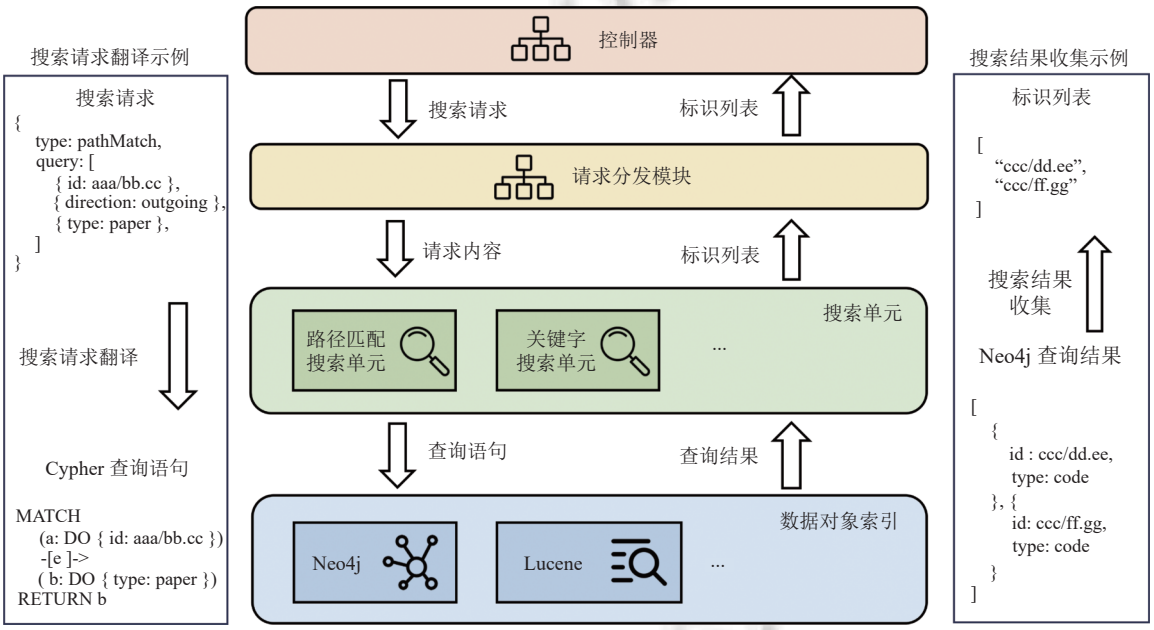

图 3 对象检索引擎和关系索引

3.3.1 请求分发模块

在收到客户端发来的搜索请求后, 控制器将这一请求发至对象检索引擎, 请求的数据格式如图 3 左侧的搜索请求翻译示例所示, 包括一个表示搜索请求类型的 type 域和一个表示搜索请求内容的 query 域. 根据不同的搜索类型, 搜索请求的内容可以具有不同的格式. 对于路径匹配搜索, 搜索请求的内容为一个数组, 其中交替存放路径中结点和边需要满足的条件; 对于关键字搜索, 搜索请求的内容为一组关键字. 根据搜索请求的类型, 请求分发模块将搜索请求的内容转发至不同的搜索单元.

3.3.2 搜索单元

ReproLink 支持关键字搜索、路径匹配搜索等搜索方式, 对于每种搜索方式, 对象检索引擎都包含一个处理这种搜索请求的搜索单元. 在收到请求分发模块转发的请求后, 搜索单元根据请求的内容生成与之匹配的数据库查

询语句, 并使用数据对象索引中的对应数据库执行搜索请求, 这一过程称为搜索请求翻译. 图 3 左侧的示例展示了对于一个路径匹配搜索请求进行翻译的过程, 路径匹配搜索单元根据搜索请求中的路径模式生成了一个 Cypher 查询语句, 并在数据对象索引中的 Neo4j 数据库中执行.

在搜索执行完毕后, 搜索单元得到从数据对象索引返回的搜索结果. 由于底层数据库的不同或查询语句返回语句的差异, 不同类型的搜索单元会从数据对象索引得到不同格式的搜索结果, 因此搜索单元还需要将这些搜索结果转换为一个统一的格式. 这一过程称为搜索结果收集. 图 3 右侧的示例展示了对路径匹配搜索请求返回的结果进行结果收集的过程. 在执行示例中的 Cypher 查询语句后, Neo4j 数据库返回的结果中将每个数据对象表示为包含了其全部属性的字典, 在结果收集的过程中, 路径匹配搜索单元从每个字典中提取出数据对象的标识, 并向上层模块返回一个标识列表. ReproLink 可以通过增加搜索单元的方式方便地扩展, 从而支持新的搜索请求类型. 表 3 展示了 ReproLink 目前支持的搜索请求的类型. 在搜索结果收集完成后, 搜索单元层将统一格式的搜索结果返回至请求分发模块.

表 3 ReproLink 支持的搜索类型

搜索类型	*type*域	*query*域
关键字搜索	keywordSearch	一组关键字
路径匹配搜索	pathMatch	路径模式

3.3.3 数据对象索引

为了支持关键字搜索和路径匹配搜索等搜索方式, ReproLink 在本地维护了一个由异构数据库组成的数据对象索引. 目前此模块包含一个支持关键字搜索的关键字索引和一个支持路径匹配搜索的关系索引. 关键字索引使用 Apache Lucene 实现, 在科研数据对象的属性集上建立索引. 关系索引使用 Neo4j 图数据库实现, 将科研数据对象建模为节点, 关系对象建模为边, 为路径匹配搜索提供支持.

3.4 复现执行引擎

ReproLink 支持用户进行对论文的在线复现, 这一功能通过复现执行引擎完成. 一次完整的在线论文复现包括多次复现单元所关联代码的在线执行. 一次代码的在线执行涉及一个代码对象和若干个数据集对象. 设待执行代码的标识为 I^C, 输入数据集的标识分别为 $I_1^D, I_2^D, \ldots, I_n^D$, 代码对象为 O^C, 数据集对象为 $O_1^D, O_2^D, \ldots, O_n^D$ 一次代码的执行主要包括以下几个步骤.

1) 算力匹配: 根据论文复现需要的算力资源, 匹配合适的硬件环境. 后续的代码执行都在该环境中执行.

2) 创建任务: 在服务器上创建任务进程. 任务进程负责管理代码执行的全生命周期.

3) 获取复现所需数据: 任务进程根据代码对象的标识 I^C 和数据集对象的标识 $I_1^D, I_2^D, \ldots, I_n^D$, 调用数据管理模块将代码 O^C 和数据集 $O_1^D, O_2^D, \ldots, O_n^D$ 下载到任务进程所在机器的本地硬盘.

4) 创建复现环境: 使用代码对象的运行环境 $\pi_{EE}(O^C)$ 创建一个 Docker 容器作为此次复现的软件环境. 在创建容器时分别挂载代码对象的代码文件 $\pi_{EC}(O^C)$ 和数据集对象的数据实体 $\pi_E(O_1^D), \ldots, \pi_E(O_n^D)$.

5) 执行复现: 在复现环境中执行复现所使用的代码, 由任务进程全程监控复现的输出并实时向复现执行引擎返回日志.

6) 收集复现结果: 使用输出数据的内容作为数据实体, 调用数据管理模块创建数据集对象 $O^{D'}$.

7) 返回复现结果: 代码执行结束, 任务进程将输出数据集的标识 $\pi_I(O^{D'})$ 返回服务器.

算法 1 描述了在代码执行服务器上的任务进程在代码执行过程中的操作逻辑, 也就是上述的第 3–7 步. 基于以上代码执行函数, ReproLink 实现了包含多个复现单元的多阶段复现流程的一键复现, 其流程见算法 2. 算法 2 的 ExecuteReproductionUnits 函数首先调用 GetDependentObjects, 使用深度优先搜索获取待执行的复现所依赖的全部复现单元. 维护 Ready 集合和 Done 集合, 分别表示输入数据已经就绪的复现单元和已经执行完成的复现单元. 每次循环从 Ready 集合中取出一个就绪的复现单元, 调用 ExecuteOneReproductionUnit 函数执行此复现单元,

© 中国科学院软件研究所 http://www.jos.org.cn

在复现执行完成后将其加入 Done 集合, 并更新 Ready 集合, 从而以拓扑序依次执行每个复现单元. 伪代码中的 GetCode、GetDatasets 和 GetInputReproductionUnits 函数均使用对象检索引擎基于单跳路径匹配的搜索实现. 在每个复现单元执行完成后, 复现执行引擎都会调用数据管理模块创建结果数据集对象和复现单元之间的关系对象. 在复现完成后, 其他用户就可以使用对象检索引擎从新数据集检索到输入数据集, 或根据输入数据集查询衍生的新数据集, 从而实现了基于关系的数据溯源和数据发现.

算法 1. 代码执行.

1. **function** ExecuteCode(I^C, I_1^D,…, I_n^D)
2. $O^C \leftarrow$ DoManager.RetrieveDo(I^C)
3. **for** i=1 **to** n **do**
4. $O_i^D \leftarrow$ DoManager.RetrieveDo(I_i^D)
5. **end for**
6. *container* $\leftarrow$ CreateContainer($\pi_{EE}(O^C)$, $\pi_{EC}(O^C)$, $\pi_E(O_1^D)$,…, $\pi_D(O_n^D)$)
7. StartContainer(*container*)
8. WaitAndMonitor(*container*)
9. $D' \leftarrow$ GetResults(*container*)
10. $O^{D'} \leftarrow$ DoManager.CreateDo(D')
11. **return** $O^{D'}$
12. **end function**

算法 2. 复现执行.

1. **function** GetDependentObjects(r)
2. $R \leftarrow \{r\}$
3. **for** $r \in$ GetInputReproductionUnits(r) **do**
4. $R \leftarrow R \cup$ GetDependentObjects(r)
5. **end for**
6. **return** R
7. **end function**
8. **function** ExecuteOneReproductionUnit(r, *Results*)
9. $C \leftarrow$ GetCode(r)
10. $D \leftarrow$ GetDatasets(r)
11. **for** $r' \in$ GetInputReproductionUnits(r) **do**
12. $D \leftarrow D \cup \{Results[r']\}$
13. **end for**
14. *output* $\leftarrow$ ExecuteCode(C, D)
15. **return** *output*
16. **end function**
17. **function** ExecuteReproductionUnits(*reproductionUnit*)
18. $R \leftarrow$ GetDependentObjects(*reproductionUnit*)
19. $Results, Ready, Done \leftarrow \varnothing, \varnothing, \varnothing$
20. **for** $r \in R$ **do**

© 中国科学院软件研究所 http://www.jos.org.cn

```
21.     if GetInputReproductionUnits(r) = ∅ then
22.         Ready ← Ready ∪ {r}
23.     end if
24.   end for
25.   while Ready ≠ ∅ do
26.     Get an element r from Ready
27.     Ready ← Ready − {r}
28.     result ← ExecuteOneReproductionUnit(r, Results)
29.     DoManager.CreateRelationship(r, result, (output, reproduction, dataset))
30.     Results[r] ← result
31.     Done ← Done ∪ {r}
32.     for r′ ∈ {r′|r ∈ GetInputReproductions(r′)) do
33.       if GetInputReproductions(r′) ⊂ Done then
34.         Ready ← Ready ∪ {r′}
35.       end if
36.     end for
37.   end while
38. end function
```

3.5 分散式数据对象存储和标识解析服务

ReproLink 中的科研数据对象以分散式存储的形式保存于世界各地的科研数据仓库中, 数据所有者可以在本地建立自己的科研数据仓库以保存其拥有的科研数据对象, 也可以将属于自己的数据对象托管在第三方的科研数据仓库中. 科研数据仓库基于 DOA 中的数字对象仓库实现, 使用 DOIP 协议对外提供服务. DOIP 协议支持 TCP、UDP、TLS 等多种不同的传输层协议, 在异构协议的基础上对外提供了一个统一的接口.

标识解析服务基于 DOA 中的标识解析系统实现. 标识解析服务维护了这些科研数据对象的一个中心化索引. 在创建科研数据对象时, 数据管理模块首先通过标识解析服务注册该科研数据对象, 标识解析服务会为这数据对象分配一个全局唯一的数据标识. 在对数据对象进行获取、更新、删除操作时, 数据管理模块先向标识解析服务请求该数据对象所在的数据对象仓库以及此仓库支持的数据交换协议, 再使用合适的协议和存储该数据对象的仓库通信.

4 系统测试与应用验证

本节对 ReproLink 的数据管理模块和对象检索引擎进行了性能测试, 测试结果表明 ReproLink 具有较高的吞吐率和较低的延迟, 且在节点数量增加时具有良好的可扩展性. 接着, 本节对 ReproLink 进行了用户体验研究, 用户普遍认为 ReproLink 具有较好的易用性和实用性. 最后, 本节在 3 个领域的实际场景中对 ReproLink 进行了实例分析, 体现了 ReproLink 对于不同的科研领域具有一定的普适性.

4.1 性能测试

本节的性能测试主要关注 ReproLink 的数据管理模块和对象检索引擎的性能. 对于复现执行引擎, 由于其功能为执行第三方代码, 吞吐率主要由第三方代码的运行时间决定, 因此本节不对复现执行引擎进行性能测试.

4.1.1 测试设置

本节的测试使用阿里云 ecs.g7.large 型实例作为测试环境, 其具体配置见表 4. 在可扩展性实验中, 测试使用最多 32 台阿里云 ecs.g7.large 服务器; 在其他实验中, 测试使用两台阿里云 ecs.g7.large 服务器.

© 中国科学院软件研究所 http://www.jos.org.cn

表 4 测试环境信息

测试环境	环境描述
硬件环境	型号: 阿里云ecs.g7.large CPU: 双核 Intel(R) Xeon(R) Platinum 8369B CPU @ 2.70 GHz 内存: 8 GB 硬盘: 120 GB SSD 网络: 25 Mb/s Ethernet
软件环境	操作系统: Ubuntu 22.04.3 LTS 64 bit Docker 24.0.7 Eclipse Temurin JDK 17.0.11+9
环境部署	ReproLink v1.0.1

测试使用的数据为 Papers with Code 平台公开发布的数据集[19]. 对于除了可扩展性测试以外的其他性能测试, 在初始化时随机从 Papers with Code 平台的数据中抽取共计 1 万个论文、代码或数据集, 根据他们的属性在系统中创建相应的科研数据对象. 接着在这些科研数据对象之间随机选择起点和终点, 创建 2 万个随机生成的关系对象. 在对可扩展性测试初始化时, 在每个数据仓库中都插入 10 万个随机抽取的论文、代码或数据集对象, 数据总量最多为 160 万个科研数据对象.

4.1.2 数据管理模块测试

首先对单个数据仓库的情形进行测试. 对于数据管理模块, 对其创建、查询、修改、删除科研数据对象和关系对象的性能进行测试. 在对查询操作、修改操作进行性能测试时, 操作的目标从系统中所有的科研数据对象或关系对象之中均匀随机地选择. 在对删除操作进行测试时, 操作的目标从当前未被删除的科研数据对象或关系对象中均匀随机地选择. 测试中使用 8 个客户端线程向服务器发送请求. 经验证, 此配置可以使服务器达到其最大吞吐率.

● 科研数据对象操作性能测试. 表 5 展示了对于科研数据对象的创建、查询、修改、删除 4 项功能的测试结果. 科研数据对象的查询操作吞吐率显著大于剩余的 3 类操作, 达到了 4.8 万的吞吐率, 满足大规模场景下的科研数据对象查询需求. 剩余的 3 类操作在执行时均需要更新对象检索引擎中的关键字索引和关系索引, 从而保证关键字搜索和关系搜索的性能. 在实际情况中, 大部分请求为查询科研数据对象, 对于科研数据对象的创建、修改和删除操作相对较少, 因此 ReproLink 主要针对查询操作进行优化.

● 关系对象操作性能测试. 表 6 展示了对于关系对象操作的性能测试结果, 和科研数据对象类似, 关系对象查询操作的吞吐率显著高于剩余 3 类操作, 这是因为在对关系对象进行创建、修改、删除时需要更新对象检索引擎中的关系索引. 实际情况中对于关系对象的操作主要为查询操作, 因此这样的性能可以满足实际情况的需要.

● 可扩展性测试. 接着, 对 ReproLink 的可扩展性进行测试. 测试使用最多 16 台服务器, 每个服务器作为一个独立的数据对象仓库接入 ReproLink. 在测试之前, 对于每个仓库, 从 Papers with Code 的数据中随机选取 10 万个科研数据对象存入. 在测试中, 使用 16 台服务器作为客户端, 每个客户端在所有数据对象仓库中的数据对象中均匀随机地选择操作目标, 并对其进行查询操作.

表 5 科研数据对象操作性能测试结果

操作	吞吐率 (Ops/s)
创建科研数据对象	1 256
查询科研数据对象	48 720
修改科研数据对象	1 254
删除科研数据对象	1 223

表 6 关系对象操作性能测试结果

操作	吞吐率 (Ops/s)
创建关系对象	690
查询关系对象	56 661
修改关系对象	693
删除关系对象	630

ReproLink 采用分散式存储架构, 具有良好的可扩展性. 在数据规模扩大时, 可以通过增加数据对象仓库的数量来满足扩大的数据规模. 图 4 展示了 ReproLink 的总吞吐率随数据仓库数量不同而变化的曲线. 随着节点数量

的增加, ReproLink 的总吞吐率大致以线性增长. 在节点数量为 16 时, 总数据量达到了 160 万, 此时 ReproLink 的总吞吐率达到了每秒 57 万次操作, 展现了良好的可扩展性.

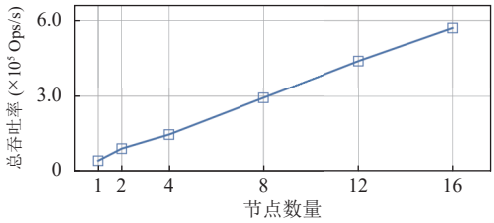

图 4 ReproLink 吞吐率随节点数量的变化趋势

● 相关工作对比测试. 最后, 为了进一步验证 ReproLink 能够适应现实场景的工作负载, 将 ReproLink 的性能与现有的科研数据管理平台进行对比测试. 测试使用 Papers with Code 作为比较对象. Papers with Code 具有庞大的用户基数, 并且被数据科学研究者广泛地使用, 因此与其的性能对比具有一定的代表意义. 在测试中, 通过不断增加客户端的数量对两个平台进行压力测试, 并记录其吞吐率和平均延迟. 图 5 和图 6 分别展示了 ReproLink 和 Papers with Code 的吞吐率-延迟曲线. ReproLink 的延迟在吞吐率小于 4 万的情况下均稳定在较低水平, 并且在吞吐率超过 4.5 万时仍然能保持小于 10 ms 的延迟; 而 Papers with Code 在吞吐率超过 800 后便出现延迟的急剧上升, 且其平均延迟均大于 200 ms.

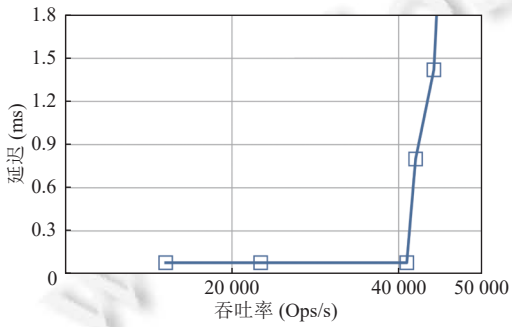

图 5 ReproLink 吞吐率-延迟曲线

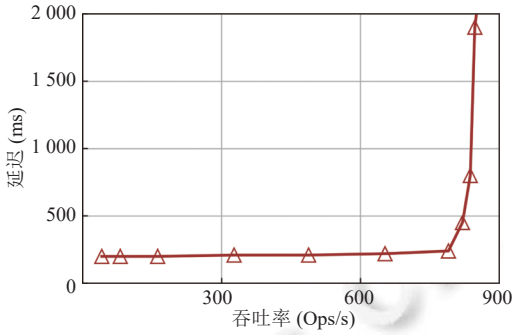

图 6 Papers with Code 吞吐率-延迟曲线

4.1.3 对象检索引擎测试

对象检索引擎的性能测试包括路径查询和关键字查询. 在路径查询测试中, 在系统中均匀随机地选择一个科研数据对象作为起点, 分别对单跳、两跳、三跳、四跳的路径模式匹配搜索进行性能测试. 在进行关键字查询测试前, 先对系统中科研数据对象的标题和摘要进行词频统计, 在测试时根据词频随机抽取搜索词, 每个词被抽中的概率和其词频成正比. 在测试中使用 8 个客户端线程向服务器发送搜索请求. 表 7 展示了对对象检索引擎进行性能测试的结果. 对于路径查询, 随着路径长度的增加, 搜索吞吐率逐步下降. 在实际情况中, 大部分请求为单跳或两跳的路径查询, 关键字查询和长度更长的路径查询的频率较低, 因此这样的吞吐率可以满足目前的系统需要. 随着未来系统规模的扩大, 可以通过创建分布式索引的方式提高吞吐率, 以满足进一步增长的需求.

表 7 对象检索引擎性能测试结果

操作	吞吐率 (Ops/s)	操作	吞吐率 (Ops/s)
单跳路径查询	14377	四跳路径查询	6539
两跳路径查询	11316	五跳路径查询	4301
三跳路径查询	9087	关键字查询	1121

© 中国科学院软件研究所 http://www.jos.org.cn

4.2 用户体验研究

为了进一步探讨 ReproLink 的用户体验, 本节对 ReproLink 进行了用户体验研究. 我们向 10 位 ReproLink 的用户发放了问卷, 问卷包含 3 个评分题和一个简答题. 其中, 3 个评分题分别请用户从用户界面、易用性以及实用性 3 个方面对 ReproLink 进行打分. 表 8 展示了问卷中这 3 道题的具体描述, 每道题的打分范围为 1–5 分. 图 7 展示了打分的具体结果, 用户普遍对 ReproLink 的用户界面和易用性较为满意, 并认为 ReproLink 具有较好的实用性.

表 8 调查问卷内容

评价角度	问题描述
用户界面	您认为ReproLink的界面的用户友好程度如何? (满分5分, 1分代表不满意, 5分代表非常满意)
易用性	您认为ReproLink的操作便利程度如何? (满分5分, 1分代表操作繁琐, 5分代表操作非常便利)
实用性	您认为ReproLink能在多大程度上加快科研人员的复现? (满分5分, 1分代表不能加快复现, 5分代表极大程度加快复现)

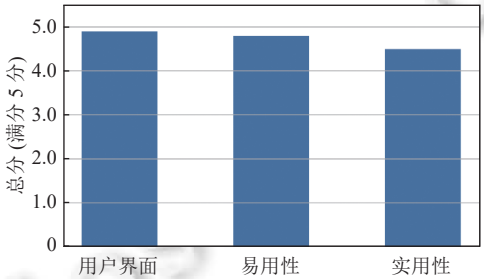

图 7 ReproLink 用户评分

问卷中的简答题请用户分享其在使用 ReproLink 过程中的感受与想法. 表 9 展示了 3 个具有代表性用户反馈, 用户普遍对 ReproLink 中对多源异构数据的聚合及代码和运行环境的一体化建模给出了积极的反馈. 这些设计减少了用户在数据获取和运行环境配置上耗费的时间, 让他们可以更多地关注于论文的方法, 提升科研效率.

表 9 ReproLink 用户反馈

用户	用户反馈
用户1	ReproLink简化了复杂实验的复现流程, 使数据科学研究者可以更加专注于研究本身. 具体来说, ReproLink集成了多种数据仓库的访问渠道, 使得我的研究可以更多的关注于方法而不是数据的获取和部署, ReproLink预配置的代码环境解决了传统复现过程中的环境兼容问题, 减少了我手动配置的时间成本
用户2	在科研过程中我经常需要复现大量的论文实验, 复现时经常需要进行繁琐的环境配置、包管理、数据处理等工作, ReproLink提供了一个简洁易用的用户界面, 帮助我一键运行实验, 并且可以复用已经完成的复现工作, 学习成本也比较低, 大大提高了我的科研效率
用户3	ReproLink能够帮助我快速实现相关工作的复现流程. ReproLink不仅能够根据文献资料迅速帮我获取相关数据集, 并且提供我相应的代码以及运行环境. 这样我可以较为便利地复现相关论文, 大大节省了找数据和配置代码运行环境的时间, 提升我的科研效率

4.3 案例 1: 机器学习领域论文复现

ReproLink 使得论文的读者可以方便地在线重复论文的计算过程. 对于一个论文对象, 若系统中已经存在一组描述此论文复现过程的复现单元, 那么用户只需找到最终输出复现结果的复现单元, 调用复现执行引擎在线执行该复现单元, 则可实现简单地一键在线复现; 若此论文与其相关的代码和数据集之间尚未建立关系对象, 则用户需要首先使用多跳路径查询找到此论文使用的代码和数据集, 接着创建若干复现单元以描述论文的复现过程, 再进行论文的在线复现. 在这一过程完成后, 后续想要复现此论文的其他用户都可以简单地一键复现. 图 8 展示了使用 ReproLink 进行论文复现的过程中系统的各个界面.

作为示例, 本节使用 ReproLink 复现了论文“Translating embeddings for modeling multi-relational data”[35]. 本节首先模拟了查找和创建论文、代码、数据集等科研数据对象以及他们之间的关系对象的过程, 接着模拟复现者在线复现论文的过程.

© 中国科学院软件研究所 http://www.jos.org.cn

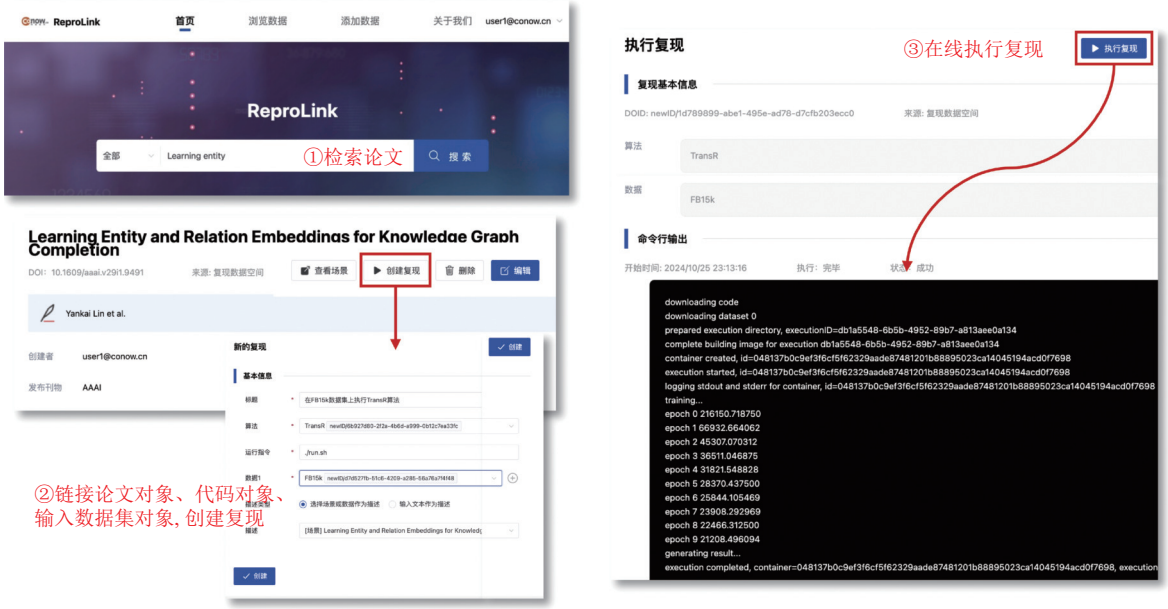

图 8 ReproLink 系统界面

查找科研数据对象和关系对象的过程按以下步骤进行.

1) 使用关键字搜索功能, 在系统中找到需要复现的论文“Translating embeddings for modeling multi-relational data”[35], 设其为 O^P.

2) 在系统中查找代码对象 TransE. 本文使用的复现代码在论文作者开源的代码的基础上进行微调, 设其为 O^C.

3) 使用基于路径的多跳查询查找论文对象 O^P 引用的论文中提出的所有数据集对象, 并在其中人工查找和论文内实验相符的数据集. 在查询时, 发送给对象检索引擎的搜索请求为查找论文对象 O^P 经过任意跳类型为 $(cite, paper, paper)$ 的关系, 接着经过恰好一跳类型为 $(propose, paper, dataset)$ 关系的路径所能达到的所有终点数据集. 查询得到 FB15k, WN18 等数据集. 本节的模拟测试中使用 FB15k 数据集作为输入数据, 设其为 O^D.

4) 创建复现单元 O^R.

5) 创建关系对象, 起点为 O^R, 终点为 O^C, 类型为 $(use, reproduction, code)$.

6) 创建关系对象, 起点为 O^D, 终点为 O^R, 类型为 $(input, dataset, reproduction)$.

在创建复现单元和关系对象后, ReproLink 即可实现论文的一键在线复现. 后文图 9 展示了此次模拟论文复现完成后的系统界面截图.

4.4 案例 2: 医学领域论文复现

除了计算机领域的研究之外, ReproLink 也可用于支持其他领域的科学研究中计算过程的复现. 本节对疗影像分割的经典论文 UNet++[36]进行了复现, 此论文提出了医疗影像分割模型 UNet++, 并在 6 个医疗影像分割数据集上对 UNet++进行测试. 该论文使用的数据集分别来自 6 个不同的数据提供者, 并且论文中并未直接提供访问数据集的链接, 而只引用了部分提出数据集的论文. 研究者在复现论文时必须在互联网上分别寻找这些数据集, 并经过各不相同的步骤在各自的提供者网站上获取实验数据集. 在论文复现的过程中, 获取多源异构数据的过程消耗了研究人员大量的时间, 极大地降低了论文复现的效率.

在 ReproLink 中, 数据集对象保存在数据所有者各自的数据仓库中. 这些仓库可以使用不同的协议对外提供服务, ReproLink 支持多种不同的底层协议, 基于这些异构的仓库向上层用户提供了对于科研数据对象的统一访问接口, 从而缓解了数据的多源异构性对复现的影响. 本节模拟了在 ReproLink 上使用 UNet++开源代码[37]复现论文中在 BRATS 数据集[38]上的实验. 具体步骤如下.

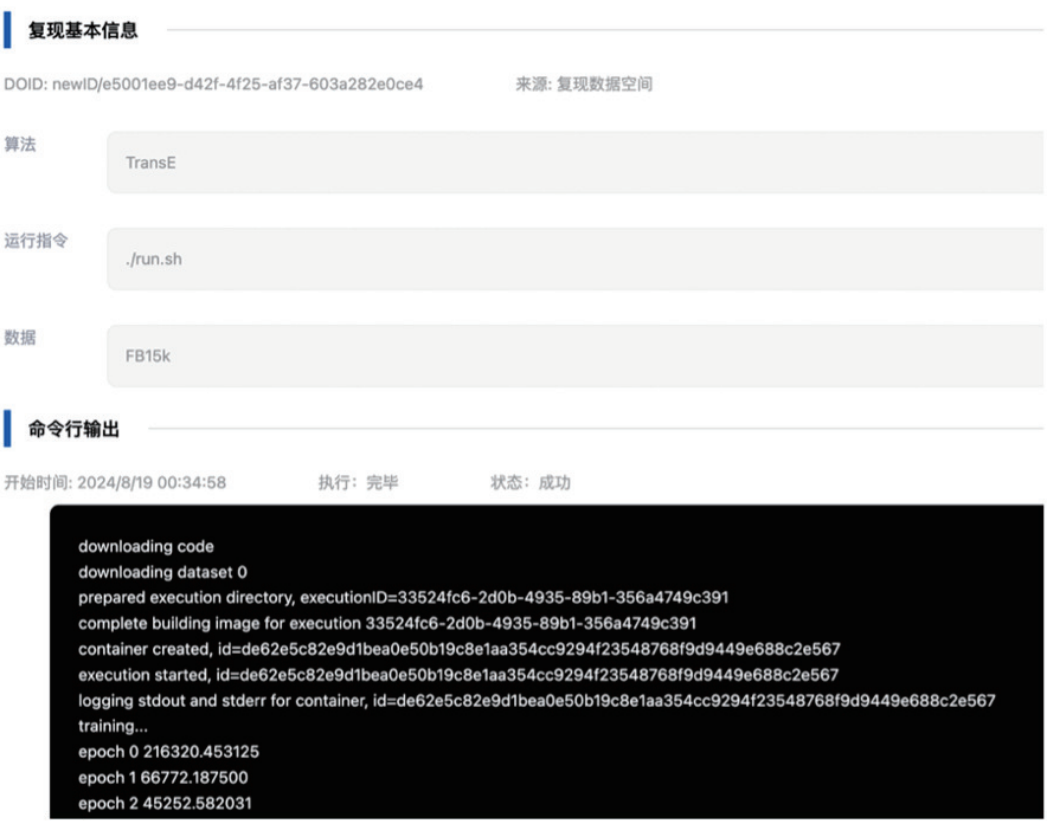

图 9 论文复现完成后的系统界面

1) 使用关键字搜索功能, 找到需要复现的论文 UNet++: Redesigning skip connections to exploit multiscale features in image segmentation[36]. 设其为 O^P.

2) 使用基于关系的多跳路径查询, 从此论文对象出发, 查询经过一跳类型为 $(cite, paper, paper)$ 的关系, 再经过一跳类型为 $(propose, paper, dataset)$ 的关系, 所能到达的所有终点数据集对象. 设其为 O^D.

3) 使用关键字搜索, 在系统中查找 UNet++代码对象. 设其为 O^C.

4) 以 O^P 为论文, O^C 为代码, O^D 为输入数据集, 创建复现单元 O^R. O^P、O^C 和 O^D 来自不同的数据仓库, 这些仓库可以使用异构的底层存储, 并以不同的协议对外提供数据. 但在科研数据对象的统一建模框架下, ReproLink 以复现单元为媒介, 基于科研数据对象的标识建立了它们之间的联系.

5) 在线执行复现单元. 在此过程中, ReproLink 使用对应的协议和数据存放的各个数据仓库通信, 在一次复现单元的执行过程中汇聚来自不同主体的科研数据对象.

使用 ReproLink, 可以大幅减少科研人员在复现过程中获取数据的时间, 从而提升论文复现的效率.

4.5 案例 3: 数据溯源

在 ReproLink 的计算框架下, 科研人员可以在处理科研数据的同时实现对这些科研数据的数据溯源 (data provenance). ReproLink 在复现单元执行的过程中自动记录了复现单元和输出数据集之间的关系. 根据这一关系对象以及复现单元与输入数据之间的关系对象, 其他科研人员可以使用路径匹配搜索根据输出数据集查询其源数据集以及数据集作者对源数据集进行的数据处理流程.

通过在线代码执行, ReproLink 使得数据集作者可以方便地根据已有的数据集创建新的数据集. 当数据集作者希望在已有数据集上做出改动时, 他可以通过复现执行引擎在已有数据集上运行一份转换代码, 并在代码运行完

© 中国科学院软件研究所 http://www.jos.org.cn

成后将代码运行的输出文件创建为一个新的数据集对象. 作为示例, 本文使用 ReproLink 复现了 Google 的 C4 数据集 (colossal clean crawled corpus)[39]的创建过程中的一个步骤. C4 数据集由 Google 于 2019 年发布, 包含的文档总数超过 3 亿 6 千万, 数据总量超过 300 GB. C4 中的语料全部来自爬虫项目 Common Crawl[40]. 对于 Common Crawl 中的数据, C4 进行了包括去重、去除非英文文档、去除不良内容、去除 Lorem ipsum 占位符等一系列过滤操作, 最终得到适合大语言模型训练的数据集. 以 C4 数据集为基础, Google 训练了大语言模型 T5[39].

在发布 C4 数据集时, Google 并未提供数据集本身的访问链接, 而仅提供其对 Common Crawl 语料的处理代码, 这使得需要使用 C4 数据集的用户需要自己进行对 Common Crawl 的预处理操作. 然而, 由于 Common Crawl 数据集的容量巨大, 对 Common Crawl 的预处理代码需要大约 335 CPU-日 (即 335 个 CPU 同时运行 1 天) 的计算量, 这增加了使用 C4 数据集进行研究的门槛. 为了方便 C4 数据集的共享, 艾伦人工智能研究院对 C4 数据集进行了预处理, 并将其发布在 Hugging Face 等平台上[41]. 但这也带来了新的问题: 新发布的 C4 数据集和 Common Crawl 数据集之间的关系并没有被显式地表达, C4 数据集的使用者无法方便地对其中的数据进行溯源.

ReproLink 可以在一定程度上解决这一问题: 若 Common Crawl 已经作为一个数据集对象被上传到系统中, C4 数据集的发布者只需要将对 Common Crawl 进行数据过滤的代码上传至系统, 以 Common Crawl 为输入数据集, 此代码为运算过程, 创建复现单元, 并通过在线复现功能使用此代码对 Common Crawl 数据集进行过滤, 待运行完成后获取结果数据集即可. 在这个过程中, ReproLink 会自动创建 C4 结果数据集对象和复现单元之间的关系. C4 数据集和 Common Crawl 数据集之间存在长度为 2 的间接关系, 可以基于它们之间的关系对象进行数据溯源.

本节对 C4 数据集的创建过程中“去除网页无效内容”这一步骤进行了复现. 由于 Common Crawl 数据容量过大, 处理无法在短时间内完成, 本文进行的复现只使用了 Common Crawl 中的部分数据. 具体的步骤如下.

1) 创建 Common Crawl 数据集对象, 设其为 O_1^D.

2) 根据 C4 的数据清洗算法实现去除网页无效内容的代码, 并将其创建为代码对象 O^C.

3) 创建复现单元 O^R, 建立 O^R 到 O^C 类型为 $(use, reproduction, code)$ 的边, 建立 O_1^D 到 O^R 的类型为 $(input, dataset, reproduction)$ 的边.

4) 调用复现执行引擎, 在线执行 O^R.

5) 执行完成后, 得到复现的输出数据集 O_2^D. O_2^D 即为输出的 C4 数据集对象.

在复现执行过程中, ReproLink 会自动维护 O^R 和 O_2^D 之间的关系, 创建类型为 $(output, reproduction, dataset)$ 的关系对象. 在代码执行完成后, 用户可以使用路径搜索功能, 查询从 O_2^D 出发, 经过两跳类型分别为 $(output, reproduction, dataset)$、$(input, dataset, reproduction)$ 的关系的数据对象, 即可搜索到 O_1^D, 从而实现了数据溯源.

5 总结与展望

本文设计并实现了一个面向可复现性的科研数据管理系统 ReproLink. ReproLink 提出了对科研数据的统一建模, 将论文、代码、数据集这 3 类科研数据抽象为包含标识、属性集、数据实体三要素的科研对象. 通过将多阶段的论文复现流程拆分为若干复现单元, ReproLink 为科研人员提供了一种精确表示研究成果的复杂复现流程的方式. 通过代码和上下文运行环境的一体化建模, ReproLink 增强了代码的可复现性. 对 ReproLink 的性能测试和实例分析表明 ReproLink 具有较好的性能和可扩展性, 且在实际场景中具有一定的应用价值.

目前, ReproLink 的部署尚处于初步阶段, 且其数据规模仍有扩大的空间. 因此本文的后续工作方向主要是基于用户在实际使用中反馈的体验和需求进一步优化 ReproLink 的功能, 并进一步扩大 ReproLink 系统的数据规模. 此外, 随着大模型智能体技术的发展, 大量软件工程领域的经典任务可以基于大模型实现自动化. 在后续工作中也将探索在 ReproLink 中集成大模型智能体以提升复现效率的可能性.

References:

[1] National Academies of Sciences, Engineering, and Medicine. Reproducibility and Replicability in Science. Washington: National Academies Press, 2019. [doi: 10.17226/25303]

[2] Krishnamurthi S, Vitek J. The real software crisis: Repeatability as a core value. Communications of the ACM, 2015, 58(3): 34–36. [doi: 10.1145/2658987]
[3] Barr E, Bell J, Hilton M, Mechtaev S, Timperley C. Continuously accelerating research. In: Proc. of the 45th IEEE/ACM Int'l Conf. on Software Engineering: New Ideas and Emerging Results. Melbourne: IEEE, 2023. 123–128. [doi: 10.1109/ICSE-NIER58687.2023.00028]
[4] Hutson M. Artificial intelligence faces reproducibility crisis: Unpublished code and sensitivity to training conditions make many claims hard to verify. Science, 2018, 359(6377): 725–726. [doi: 10.1126/science.359.6377.725]
[5] Baker M. 1,500 scientists lift the lid on reproducibility. Nature, 2016, 533(7604): 452–454. [doi: 10.1038/533452a]
[6] Gundersen OE, Kjensmo S. State of the art: Reproducibility in artificial intelligence. In: Proc. of the 32nd AAAI Conf. on Artificial Intelligence. New Orleans: AAAI, 2018. 1644–1651. [doi: 10.1609/aaai.v32i1.11503]
[7] Donoho DL, Maleki A, Rahman IU, Shahram M, Stodden V. Reproducible research in computational harmonic analysis. Computing in Science & Engineering, 2009, 11(1): 8–18. [doi: 10.1109/MCSE.2009.15]
[8] Li GJ, Cheng XQ. Research status and scientific thinking of big data. Bulletin of Chinese Academy of Sciences, 2012, 27(6): 647–657 (in Chinese with English abstract). [doi: 10.3969/j.issn.1000-3045.2012.06.001]
[9] Dhar V. Data science and prediction. Communications of the ACM, 2013, 56(12): 64–73. [doi: 10.1145/2500499]
[10] Reichstein M, Camps-Valls G, Stevens B, Jung M, Denzler J, Carvalhais N, Prabhat. Deep learning and process understanding for data-driven earth system science. Nature, 2019, 566(7743): 195–204. [doi: 10.1038/s41586-019-0912-1]
[11] Raghupathi W, Raghupathi V. Big data analytics in healthcare: Promise and potential. Health Information Science and Systems, 2014, 2(1): 3. [doi: 10.1186/2047-2501-2-3]
[12] Mi JN, Zhang CP, Li DY, Lin T. Fourth research paradigm: Transformation of social science research driven by big data. Academia Bimestris, 2018(2): 11–27 (in Chinese). [doi: 10.16091/j.cnki.cn32-1308/c.2018.02.003]
[13] Devlin J, Chang M W, Lee K, Toutanova K. BERT: Pre-training of deep bidirectional Tansformers for language understanding. In: Proc. of 2019 Conf. of the North American Chapter of the Association for Computational Linguistics. Minneapolis: ACL, 2019. 4171–4186. [doi: 10.18653/v1/N19-1423]
[14] Sun Y, Wang SH, Li YK, Feng SK, Chen XY, Zhang H, Tian X, Zhu DX, Tian H, Wu H. ERNIE: Enhanced representation through knowledge integration. arXiv:1904.09223, 2019.
[15] Trisovic A, Lau MK, Pasquier T, Crosas M. A large-scale study on research code quality and execution. Scientific Data, 2022, 9(1): 60. [doi: 10.1038/s41597-022-01143-6]
[16] Teng T, Huang G, Chen XR, Mei H. An approach to dynamically operating data pragmatics for internetware. Ruan Jian Xue Bao/Journal of Software, 2008, 19(5): 1160–1172 (in Chinese with English abstract). https://www.jos.org.cn/jos/article/abstract/20080508 [doi: 10.3724/SP.J.1001.2008.01160]
[17] Luo CR, Ma Y, Jing X, Huang G. Internet of data: A solution for dataspace infrastructure and its technical challenges. Big Data Research, 2023, 9(2): 110–121. [doi: 10.11959/j.issn.2096-0271.2023024]
[18] Kahn R, Wilensky R. A framework for distributed digital object services. Int'l Journal on Digital Libraries, 2006, 6(2): 115–123. [doi: 10.1007/s00799-005-0128-x]
[19] Papers with Code. 2024. https://paperswithcode.com/
[20] Hugging Face. 2024. https://huggingface.co/
[21] GitHub. 2024. https://github.com/
[22] Gitee. 2024. https://gitee.com/
[23] GitLab. 2024. https://about.gitlab.com/
[24] Zhang N, Liu Y, Ma XJ, Jiang HO, Wang L, Jing X, Huang G. Identifier resolution technology for human-cyber-physical ternary based on internet of data. Ruan Jian Xue Bao/Journal of Software, 2024, 35(10): 4681–4695 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6990.htm [doi: 10.13328/j.cnki.jos.006990]
[25] DONA. Digital object identifier resolution protocol specification 3.0. 2024. https://www.dona.net/specsandsoftware
[26] DONA. Digital object interface protocol specification 2.0. 2024. https://www.dona.net/specsandsoftware
[27] Small H. Co-citation in the scientific literature: A new measure of the relationship between two documents. Journal of the American Society for Information Science, 1973, 24(4): 265–269. [doi: 10.1002/asi.4630240406]
[28] Nayyeri M, Cil GM, Vahdati S, Osborne F, Rahman M, Angioni S, Salatino A, Recupero DR, Vassilyeva N, Motta E, Lehmann J. Trans4E: Link prediction on scholarly knowledge graphs. Neurocomputing, 2021, 461: 530–542. [doi: 10.1016/j.neucom.2021.02.100]
[29] Effendy S, Yap RHC. Analysing trends in computer science research: A preliminary study using the microsoft academic graph. In: Proc. of the 26th Int'l Conf. on World Wide Web Companion. Perth: ACM, 2017. 1245–1250. [doi: 10.1145/3041021.3053064]
[30] Wang KS, Shen ZH, Huang CY, Wu CH, Dong YX, Kanakia A. Microsoft academic graph: When experts are not enough. Quantitative

© 中国科学院软件研究所 http://www.jos.org.cn

Science Studies, 2020, 1(1): 396–413. [doi: 10.1162/qss_a_00021]
[31] Färber M, Lamprecht D, Krause J, Aung L, Haase P. SemOpenAlex: The scientific landscape in 26 billion RDF triples. In: Proc. of the 22nd Int'l Semantic Web Conf. Athens: Springer, 2023. 94–112. [doi: 10.1007/978-3-031-47243-5_6]
[32] Färber M, Lamprecht D. Linked Papers with Code: The latest in machine learning as an RDF knowledge graph. arXiv:2310.20475, 2023.
[33] Lin YK, Liu ZY, Sun MS, Liu Y, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In: Proc. of the 29th AAAI Conf. on Artificial Intelligence. Austin: AAAI, 2015. 2181–2187. [doi: 10.1609/aaai.v29i1.9491]
[34] Bollacker K, Evans C, Paritosh P, Sturge T, Taylor J. Freebase: A collaboratively created graph database for structuring human knowledge. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. Vancouver: ACM, 2008. 1247–1250. [doi: 10.1145/1376616.1376746]
[35] Bordes A, Usunier N, García-Durán A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. In: Proc. of the 27th Int'l Conf. on Neural Information Processing Systems. Lake Tahoe: Curran Associates Inc., 2013. 2787–2795.
[36] Zhou ZW, Siddiquee MR, Tajbakhsh N, Liang JM. UNet++: Redesigning skip connections to exploit multiscale features in image segmentation. IEEE Transactions on Medical Imaging, 2020, 39(6): 1856–1867. [doi: 10.1109/TMI.2019.2959609]
[37] Zhou ZW. UNetPlusPlus. 2018. https://github.com/MrGiovanni/UNetPlusPlus
[38] Menze BH, Jakab A, Bauer S, *et al*. The multimodal brain tumor image segmentation benchmark (BRATS). IEEE Trans. on Medical Imaging, 2015, 34(10): 1993–2024. [doi: 10.1109/TMI.2014.2377694]
[39] Raffel C, Shazeer N, Roberts A, Lee K, Narang S, Matena M, Zhou YQ, Li W, Liu PJ. Exploring the limits of transfer learning with a unified text-to-text Tansformer. The Journal of Machine Learning Research, 2020, 21(1): 140.
[40] Common Crawl. 2024. https://commoncrawl.org/
[41] allenai/c4. 2024. https://huggingface.co/datasets/allenai/c4

附中文参考文献:

[8] 李国杰, 程学旗. 大数据研究: 未来科技及经济社会发展的重大战略领域——大数据的研究现状与科学思考. 中国科学院院刊, 2012, 27(6): 647–657. [doi: 10.3969/j.issn.1000-3045.2012.06.001]
[12] 米加宁, 章昌平, 李大宇, 林涛. 第四研究范式: 大数据驱动的社会科学研究转型. 学海, 2018(2): 11–27. [doi: 10.16091/j.cnki.cn32-1308/c.2018.02.003]
[16] 滕腾, 黄罡, 陈兴润, 梅宏. 网构软件数据语用的一种动态支撑方法. 软件学报, 2008, 19(5): 1160–1172. https://www.jos.org.cn/jos/article/abstract/20080508 [doi: 10.3724/SP.J.1001.2008.01160]
[17] 罗超然, 马郓, 景翔, 黄罡. 数据空间基础设施的技术挑战及数联网解决方案. 大数据, 2023, 9(2): 110–121. [doi: 10.11959/j.issn.2096-0271.2023024]
[24] 张宁, 柳熠, 马新建, 姜海鸥, 王璐, 景翔, 黄罡. 面向人机物融合的数联网标识解析技术. 软件学报, 2024, 35(10): 4681–4695. http://www.jos.org.cn/1000-9825/6990.htm [doi: 10.13328/j.cnki.jos.006990]

黄小龙(2000－), 男, 博士生, CCF 学生会员, 主要研究领域为系统软件, 数联网, 数据空间.

马郓(1989－), 男, 博士, 研究员, 博士生导师, CCF 专业会员, 主要研究领域为智能化系统软件, Web 系统.

杨婧如(1995－), 女, 博士, 助理研究员, CCF 专业会员, 主要研究领域为数据治理技术与系统, 数据库系统.

景翔(1979－), 男, 博士, 副研究员, CCF 高级会员, 主要研究领域为操作系统, 智能计算.

柳熠(1993－), 男, 博士, 副研究员, CCF 专业会员, 主要研究领域为服务计算, 云计算, 数据空间, 数联网.

黄罡(1975－), 男, 博士, 教授, 博士生导师, CCF 会士, 主要研究领域为系统软件, 数联网, 数据空间.

© 中国科学院软件研究所 http://www.jos.org.cn