arXiv:2601.15432v1 [cs.DL] 21 Jan 2026

盒装梅德福：元数据描述语言的改进与未来方向

Polina Shpilker
计算机科学系
塔夫茨大学
梅德福，MA

Benjamin Stubbs
计算机科学系
塔夫茨大学
MEDFORD，MA

Michael Sayers
计算机科学与统计系
罗得岛大学
罗得岛州金斯敦

Yumin Lee
计算机科学系
塔夫茨大学
马萨诸塞州梅德福

Lenore Cowen
计算机科学系
塔夫茨大学
梅德福德，MA

Donna Slonim
计算机科学系
塔夫茨大学
MEDFORD，MA

Shaun Wallace
计算机科学与统计系
罗得岛大学
罗得岛州金斯敦
shaun.wallace@uri.edu

Alva Couch
计算机科学系
塔夫茨大学
马萨诸塞州梅德福
alva.couch@tufts.edu

Noah M. Daniels计算机科学与统计系
罗得岛大学 罗得岛州金斯敦 noah
daniels@uri.edu

摘要

科学研究元数据对于确保研究工作的有效性、可重用性和成本效益至关重要。此前已介绍 MEDFORD 元数据语言，旨在简化非程序员编写和维护元数据的过程。然而，入门门槛和可用性方面的障碍依然存在，包括自动验证功能有限、数据传输困难，以及用户不熟悉文本文件编辑。为解决这些问题，我们推出了 MEDFORD-in-a-Box（MIAB），这是一个旨在促进研究人员采用并更早开展元数据采集的文档生态系统。MIAB 包含多项改进，包括更新后的 MEDFORD 解析器、更完善的验证例程以及 BagIt 导出功能。MIAB 还包括改进后的 VS Code 扩展，通过可视化 IDE 支持这些变更。通过简化元数据生成，这一新工具支持创建正确、一致且可重用的元数据，最终提升研究可重复性。

关键词 元数据、研究可访问性、FAIR 数据、标记语言、交叉引用、MEDFORD、科学研究、可重复性、可用性、元数据生成。

1 引言

人工智能时代的算法不仅需要数据，也渴求有关数据的数据，以提供上下文信息。尽管人们提出了许多合理且成功的 FAIR（可发现、可访问、可互操作和可重用）元数据标准， [2, 17], 但在生产环境中使用这些标准通常会被放在次要位置，直到研究完成，或资金方、出版规范要求提供文档时才开始考虑。我们认为，如果元数据更易于维护，尤其是以可发现的方式进行维护，就会有更多、更完善的文档更早生成。
一种现有的元数据维护方案是资源描述框架（RDF），最初由万维网联盟为此目的开发 [8, 15, 16]。RDF 相较于更适合机器读取的 XML 元数据格式，提供了一种更易于人类阅读的替代方案 [13]。需要注意的是，RDF 本身并不是一种语言，而是一个框架。多种语言可用于生成符合 RDF 标准的元数据，这些元数据通常采用图结构，并依靠 RDF 的三元组（实体、关系、实体）来定义图中的边。这些语言包括 Turtle [4], N3 [5], RPV [7], 和 Triple [23]。然而，尽管这些更高层级的语言以人类可读的形式呈现，但它们并不是为没有编程或 RDF 结构培训的学者设计的。

开放珊瑚礁数据元数据格式（MEDFORD）[20, 21]语言是一种此前提出的元数据描述语言，专门用于帮助非计算科学领域的研究人员以 FAIR 方式描述其研究过程和数据。MEDFORD 提供了一套易于理解的词汇，可进行扩展以满足用户需求，并提供能够提升可复用性和数据质量的元数据模式。MEDFORD 生成纯文本格式的文件，易于共享和编入索引，从而提高可发现性、可访问性以及与其他系统的互操作性。
本文的贡献有两方面。首先，我们介绍了对 [22, 19], 中提出的 MEDFORD 语言扩展的部分实现，并对宏和外部引用的拟议语法作了相应更新。其次，我们介绍了一套开源软件包，名为 MEDFORD-in-a-Box（MIAB），其中包含在 Visual Studio Code 中轻松编写和检查 MEDFORD 元数据文件所需的全部工具。MIAB 包括更新后的 MEDFORD 解析器、Visual Studio Code 扩展（最早在 [24] 中介绍）、MEDFORD语言服务器协议模块、全面的测试套件、数据验证资源和外部引用，同时还讨论了即将对拟议的宏语法进行的更新。我们相信，MIAB 将使非程序员也能自信地编写 MEDFORD 元数据文件。

2 MEDFORD 解析器的扩展

MEDFORD 解析器是 MEDFORD系统的核心。它会检查 MEDFORD 语法、验证 MEDFORD 文件，并创建用于共享的归档文件。除了报告语言错误外，MEDFORD 解析器还能识别各种导出文件类型所需的属性，并帮助确保研究人员提供这些属性。例如，美国国家科学基金会资助的珊瑚研究必须提交给 BCO-DMO[6], ，而 BCO-DMO 要求提供 @Contributor 或 @Cruise 等元数据信息。

MEDFORD 解析器有助于确保所提供的元数据格式正确。在最新版本中，MEDFORD 为若干基本主令牌数据类型提供了验证例程，例如电子邮件、数字和文本。这些数据类型与基本操作和比较值相结合，从而可以轻松定义常见的数据验证任务，例如 Version 是数字 >0 ，或电子邮件地址看起来合理。

这些规则旨在采用公式化的形式，使我们能够在未来为用户提供一种方法，让他们指定自己的自定义主令牌和验证例程，而无需编写 Python 方法。我们设想用户可以使用简化的文本格式（例如 YAML）来定义令牌和验证规则。

例如，考虑描述新实体“机构”的 YAML，其中我们定义了几个次要令牌，例如地址、城市和统一资源标识符，并为它们指定验证规则：

机构：- 类型：字符串 - 必填 #此输出模式要求提供该标记。必须存在，否则会报错 - 多个 #此标记可以有多个实例 - 内容：- 地址：#次级标记 - 类型：字符串 #此处的类型表示由验证器调用的验证子程序 - 建议提供 #这意味着我们希望提供次级标记，但只警告，不报错

- 城市：- 类型：字符串 - 建议提供 - 省份：- 建议提供 - 国家：- 类型：字符串 - 统一资源标识符：- 类型：统一资源标识符 - 建议提供 - 电话：- 类型：电话 - 建议提供

随后可以导入这些 YAML 文件，以提供额外的数据模型。高级用户也可以为这些新数据模型配套自定义验证子程序，前提是遵循 IO 模板，并在 MEDFORD 验证映射文件 MEDFORD.mvd 中将其关联起来。

MEDFORD语言在设计时也考虑了数据传输；借助一种名为 BagIt 的技术， [14], MEDFORD 解析器可以将 MEDFORD 元数据文件及其引用的所有研究数据文件完整打包到一个 ZIP 文件中，如 BagIt 格式所述。这个 ZIP 文件包含哈希值，以确保文件传输无误，并且可以标明相关引用，其中也可能包含远程引用。具体来说，Primary 和 Copy 标签对应的文件会存储在 ZIP 文件中，其哈希值则记录在清单文件中；但 Ref 标签提及的引用既不会存储在 ZIP 文件中，其哈希值也不会记录在清单文件中。这样一来，MEDFORD 文件便可与其所描述的研究数据一同传输，使研究人员只需传输一个文件，就能向协作者或投稿数据库提供完整的数据描述。此外，这种方式也保留了引用外部数据源的灵活性，例如基因组数据集等可能过大而无法实际装入 BagIt ZIP 文件传输的数据源。

最新版的 MEDFORD 解析器包含更新后的 BagIT 实用工具，重点关注数据质量和完整性。BagIT 归档文件现在有自己的验证例程，会检查其中的每个文件，确保其具有关联标签，并根据 MEDFORD 文件验证所有预期文件均已存在，然后再进行归档。

3 Visual Studio Code 扩展更新

Visual Studio Code [9] （VS Code）是业界标准的集成开发环境（IDE），拥有活跃的在线社区，致力于通过称为扩展的外部软件包来扩展编辑器的功能。尽管自 2022 [24], 以来，VS Code 市场就已提供 MEDFORD 扩展，但 MIAB 代表着一次重大更新，也标志着我们未来发展方向的转变。

微软率先推出了语言服务器协议（LSP），旨在让代码检查、语法高亮和补全建议等语言功能能够在不同编辑器之间移植（严格来说，给定 M 个编辑器和 N 种语言，LSP 可将一个 $M \times N$ 问题转化为一个 $M + N$ 问题）。这是因为，不必让每个编辑器都需要 N 个语言插件（$M \times N$），而只需 M 个 LSP 插件和 N 个语言服务器。请注意，对于某些编辑器，为了支持其他功能（例如不常见的构建命令或格式化命令），可能还需要为某种语言安装额外的轻量级插件。
VS Code 扩展旨在成为使用 MEDFORD 的便捷入门途径。安装 VS Code 后，用户只需执行几条命令，即可安装 MEDFORD 解析器、MEDFORD 语言服务器和扩展本身，并立即开始使用 MEDFORD。该扩展会将 MEDFORD 文件关联到 .mfd 文件类型，方便打开。扩展还为 MEDFORD 语言提供语法高亮，以提升可读性并减少错误。扩展启用期间，语言服务器会在后台运行，提供实用的交互式反馈，例如补全次要标签。

此新版本已更新为使用最新版本的 pydantic 和 pyglas，并经过重构以简化更新和提升可复用性；同时，它还集成了新版 MEDFORD 解析器，以支持陆续发布的新数据模型、验证和导出模式。

我们已将 VS Code 扩展中的 MEDFORD 解析器、BagIT 实用工具和语言服务器组件解耦，以便独立开发并减少代码重复。例如，语言服务器本身并不依赖 VS Code 应用程序；将其分离后，我们便可将其集成到其他技术中，例如 Monaco 这类基于 Web 的 IDE，或支持语言服务器协议的其他编辑器，例如 neovim 或 Zed。通过

从扩展中移除数据模型信息后，解析器便成为数据模型的唯一事实来源。这样我们就能轻松开发并发布新模型，从而推动实现上文所述的未来功能：可导入的用户数据模型和验证任务。

4 MEDFORD语言规范的扩展

我们还升级了 MEDFORD语言的正式规范，以实现所需的可用性。本节将详细介绍其中几个关键问题和扩展。

4.1 元数据互连性

我们最初设计 MEDFORD 时采用了自包含的区块结构。一个 MEDFORD区块代表一个实体，该实体的所有元数据都包含在其中。最初，实体之间并无上下文或关系的概念。然而，这种设计很快就被证明不足以描述有机研究数据。

我们最初的用户测试邀请了一组珊瑚研究人员，他们希望使用 MEDFORD 来记录自己的研究样本。要追踪一种珊瑚物种，所需的基本信息包括其学名、当前的（研究时）基因组构建，以及该物种的研究地点——珊瑚礁。有关这些珊瑚研究样本的详细信息可在 MEDFORD 中描述如下：

```
@Species P.Dam @Species_珊瑚礁 新喀里多
尼亚堡礁 @Species_珊瑚礁-坐标（坐标）
```

这些研究人员还需要记录采集样本的珊瑚礁信息。珊瑚礁通常需要记录地理坐标、研究日期以及珊瑚礁中存在的珊瑚物种等信息，因此 MEDFORD 表示法如下：

```
@reef 新喀里多尼亚堡礁 @reef-坐标
（坐标） @reef-coral_物种 鹿角珊瑚
@reef-coral_物种 鹿角珊瑚
```

但是，珊瑚物种的描述与珊瑚礁之间存在很强的依赖关系：要完整描述其中一方，就必须对另一方也做出一些描述。只需将一个主令牌设为另一个令牌的次级标记，便可解决这一问题。

```
@Species P.Dam @Species_珊瑚礁 新喀里多
尼亚堡礁 @Species_珊瑚礁-坐标（坐标）

@Species P.Acuta @Species_珊瑚礁 新喀里多尼亚堡礁
@Species_珊瑚礁-坐标（坐标，有一处拼写错误）
```

但这会带来更多复杂问题。在这里，同一珊瑚礁上的每个物种实例都会重复包含珊瑚礁元数据。这种重复增加了工作量，提高了转录错误的发生率，还给收集到的数据强加了层级结构。我们将物种视为主要实体，这种选择合理，但并非必然。然而，如果研究人员关注的是不同珊瑚礁之间的差异，他们可能更希望@Species是@Reef的子令牌。这个例子表明，我们需要一种既不重复也不具层级性的元数据对象关系概念。

4.2 命名元数据对象

最初，MEDFORD的设计中没有唯一标识符这一概念；没有唯一标识符，实体之间的关联就难以实现。不过，在MEDFORD文档中，第一行通常被称为元数据对象的“描述”或desc行。

```
@Species P. Dam   <-- 这是描述 @Species-Construction 澄黄滨珊瑚基因
组 v1.0
```

MEDFORD 解析器不会使用这一行；它通常作为元数据对象的一行描述，供用户查看，该对象由后续的次级令牌行描述。这类似于注释次级令牌的功能，后者允许用户为给定的元数据对象创建不会经过处理的自定义注释。每个区块都必须包含一条 desc行，该文件才会被视为有效的 MEDFORD，因此解决方案显而易见：可以将 desc行重新定义为名称行。MEDFORD区块的第一行不再是自由格式的注释，而是被视为该区块的名称。

与名称类似，desc可以由任意ASCII字符组合构成，也可以包含标准的MEDFORD宏。关键的是，desc可作为跨对象引用的唯一标识符。在上面的示例中，MEDFORD中有一个名为P. dam的@Species对象。定义名称后，我们现在可以提出一种语法：

@Species P.Dam @Species-Construction 澄黄滨珊瑚基因组
v1.0 @Species-Reef 新喀里多尼亚堡礁

4.3 对象名称冲突

这种对desc语法的新改进使我们可以定义名称。我们知道名称必须具有唯一性以避免冲突，但唯一性需要达到什么程度?

请看以下内容：

@Species P.Acuta @Species-Construction NCBI BioProject
PRJNA812628

@Photo P.Acuta
@Photo-Type JPEG

在这里，研究人员描述了一种珊瑚物种，并附上了旅途中拍摄的照片。两者使用相同的描述或名称，因此会对应同一个标识符。利用 MEDFORD 的区块特性，我们可以推断它们是不同的实体，因为它们具有不同的主令牌。研究人员编写 MEDFORD 区块时，该区块会立即与一个主令牌关联。主令牌定义了所声明对象的类型。因此，我们不仅可以根据上下文重复使用名称，还可以通过主令牌与名称的组合，对所引用元数据的类型作出预期。因此，我们调整了语法，要求用户声明主令牌：

@Reef-Species @Species P.Acuta

因此，我们决定，MEDFORD 名称只需在共享同一主令牌的区块中保持唯一。也就是说，如果用户要描述一个 @Species，其名称只需与其他 @Species 的名称有所区别。这种折中方案符合合理的命名指南。使用唯一的标签/名称组合有助于用户避免误解，并减少后续错误。

4.4 引用其他来源的元数据

虽然这些新的语法特性有助于拓展 MEDFORD 的用途，但由于 MEDFORD 引用必须包含在当前文档中，其应用仍受到限制。
设想多个研究小组正在研究同一组珊瑚物种。我们不希望要求所有人都复制参考文件以便在内部引用，从而造成记录重复。为满足这类用户需求，我们扩展了所提出的引用功能，使其能够引用存储在外部 MEDFORD 文件中的元数据。

这需要对 MEDFORD 语法再做两项补充：一是能够声明被引用元数据存在于外部文件中，二是能够声明将被引用的外部文件的位置。
我们选择通过 MEDFORD 导入来实现这一点，该导入会摄取被引用的整个文件。尽管从计算资源的角度来看，这种做法似乎不够理想，但它对用户体验的提升足以弥补这一点。要求用户手动导入每个被引用的元数据对象，会严重损害 MEDFORD 作为用户友好系统的效能，并增加用户错误的可能性。

‘@宏定义甚至更多定义

@Note ‘@Macro @Note 第二次使用 {‘@Macro}

图1：在 MEDFORD语言 v1.0 中定义和使用宏的示例。使用宏有两种不同的语法，其中一种允许在宏名称前后直接添加文本，因为花括号可以明确标示宏的名称。

然而，在解决一个问题的同时，这一修正不幸带来了新的问题：外部引用的数据可能彼此不一致。例如，假设两个不同的MEDFORD文件都定义了一个 @Species P. dam，但其中一个引用的基因组构建比另一个更早。在这种情况下，MEDFORD语言必须提供一种区分来源的方法，避免让解析器承担判断如何处理冲突的责任。

为了解决这个新问题，我们利用刚刚引入的新特性：MEDFORD中的每个对象都必须有一个名称，因此，定义外部MEDFORD文件引用的语法也必须包含一个名称。

4.5 外部引用语法

@Reef 新喀里多尼亚堡礁 @Reef-Species 来自
CoralsMFD：@Species P.Dam

这种语法旨在通过最大限度地发挥可供性并与用户预期保持一致，提升可用性。例如，即使没有使用 Python 或 MEDFORD 的经验，新读者也很可能会推断出 @Species P.Dam 存在于名为 CoralsMFD 的对象中。这种语法选择使用户能够在需要时更深入地嵌套溯源信息，同时保持语法可解析且可预测。这一设计选择应能提升用户快速学习并在不同 MEDFORD文件中复用语法的能力。因此，MEDFORD 能够在用户和领域之间灵活扩展，同时保持一套简单的核心语法和规则 [12, 18]。
为形式化这种关系，我们还纳入了用于明确命名空间到文件的解析规则。请看：

@导入 CoralsMFD @导入文件 ~/shared/corals_
metadata.mfd

该示例声明 应从文件 corals metadata.mfd 导入这些 MEDFORD元数据，以便 e
可供随时参考 在 MEDFORD 文件中的任何位置使用昵称 CoralsMFD。

4.6 重新设计宏语法

读者可能已经注意到，反复重新输入“新喀里多尼亚堡礁”很不方便。这是一个很好的例子，说明了 MEDFORD 为何设计了宏功能：较短的名称可以展开为较长的内容。最初设计 MEDFORD语言时，宏采用了图1所示的类编程语法。宏通过一行定义，该行以特殊字符 ‘@ 开头，后面紧接着宏的名称。宏名称只能包含字母和数字，不能包含空格或特殊字符。名称后接一个空格，之后直到遇到换行符的所有内容都属于宏的内容。要使用宏，需在某一行的载荷中写入宏头字符和宏名称。此外，宏名称和宏头可以用花括号括起来，使研究人员能够在宏内容前后添加其他字符。图2展示了这一点的重要性。

然而，这种语法遇到了一个重大问题：MEDFORD 允许载荷延伸到新行。这是为了支持作者署名，使研究人员能够将数据分隔到多行中，从而更好地适应标准语法（例如地址）或提高可读性（例如较大的描述性注释块）。请考虑以下情况：两名研究人员属于同一机构，但分别隶属不同部门，如图3所示。

理论上，可以强制要求始终使用前文所述的花括号语法来引用宏，从而解决这一问题。不过，花括号格式是为更高级的用户设计的，我们不能指望所有用户刚开始接触 MEDFORD 时就能理解它。我们不能依赖花括号语法，因为研究表明，复杂的宏语法会降低新用户的采用意愿并增加学习难度 [10]。因此，这种歧义表明有必要修订当前语法。

```
‘@CoralPhoto _pdam.png ‘
@CoralPhotoDir ./data/photos/Jul27/

@照片 01{‘@CoralPhoto} @照片文件 {‘
@CoralPhotoDir}01{‘@CoralPhoto}

@Photo 02{‘@CoralPhoto} 照片文件 {‘
@CoralPhotoDir}02{‘@CoralPhoto}

# 解析为：@Photo  01_pdam.png @Photo-File ./
data/photos/Jul27/01_pdam.png

@Photo  02_pdam.png @Photo-File ./
data/photos/Jul27/02_pdam.png
```

图2：使用宏替换数据中重复部分的示例。在此处，宏用于存储文件名和文件路径中保持一致的部分。请注意，本示例中的每次宏使用都被花括号包围；这样可以将宏的名称与其后的字符区分开来，否则任何解析器都无法从字符串‘@CoralPhotoDir01‘@CoralPhoto中识别出宏名称 CoralPhotoDir。

```
    ‘@机构 大学机构 42 索默街 普莱斯维
尔，ST 02020

@贡献者 John Doe @贡献者-隶属部门 匿名部 ‘@机构


@贡献者 Arthur Word @贡献者-隶属部门 写作与修辞部 ‘@机构
```

图3：宏定义与宏用法重叠的示例。由于 MEDFORD负载可以跨越多行，研究人员将‘@机构宏用法放在新行上，就像实际的机构地址通常另起一行书写一样。然而，这会产生以宏头字符‘@开头的行，因此也就形成了宏定义行。解析器无法区分这两种情况。

我们目前正在考虑几种新的宏语法方案，重点是将宏定义和宏调用的语法区分开来。几个提案包括使用 >@ 定义宏、使用 <@ 调用宏，或者使用 ‘@ 定义、使用 “@ 调用。我们的目标是确保宏语法易于快速输入，同时又不会显得过于“像代码”，以免吓退潜在的新用户。我们计划开展用户研究实验，以确定这些方案或其他可能的候选方案中，哪些最易于上手且效率最高。

4.7 MEDFORD实用工具：EXIF

如果能够提取原生文件类型中已存储的元数据，例如照片日期或可用的 GPS 数据（如图 4 所示），那么上述详述的外部引用语法还可以扩展出更多功能。例如，现代数码相机拍摄的照片会自动以一种名为 EXIF 的格式存储元数据。我们可以使用导入块将这些元数据关联到 MEDFORD 实体。这样，导入操作便会引用照片文件中的 EXIF，而无需研究人员重新输入数据。随着我们不断更新 MEDFORD 内部和外部引用，我们还将把这项功能扩展到高价值文件类型。

与此同时，我们创建了一个用于小型 MEDFORD 工具的代码仓库：https://github.com/TuftsBCB/medfordTools，并加入了一个小脚本，通过使用 Python exif 软件包解析图像文件的 EXIF 数据，为其创建 MEDFORD文件。这些导入工具将作为导入工具集成到未来版本的 VS Code 扩展中。

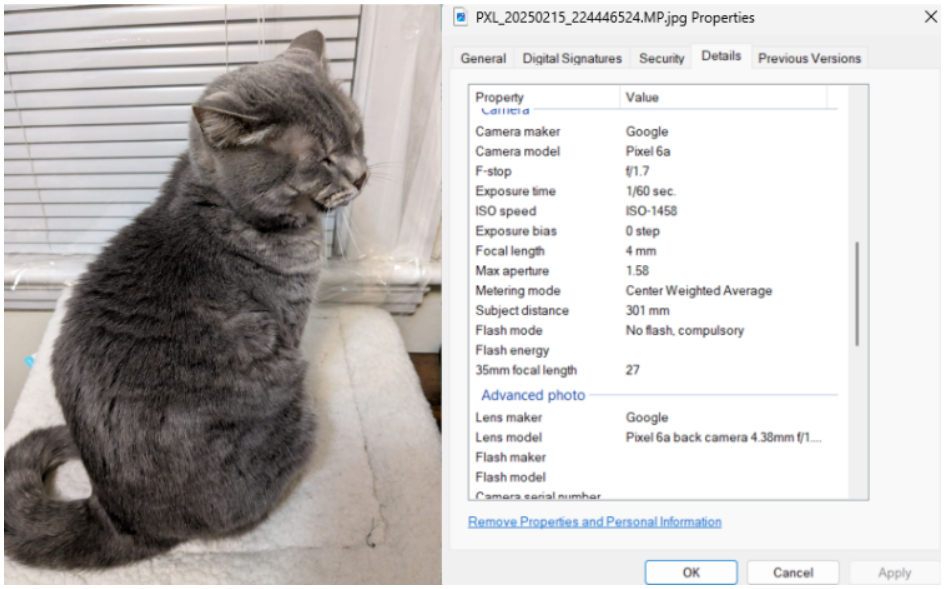


图 4：示例照片及其 EXIF 元数据。

5 结论与未来工作

我们提出了以连通性和可用性为核心的 MEDFORD语言扩展方案，以确保能够在 MEDFORD 中准确记录内在关联的研究数据。这些新的语法规则使 MEDFORD文件能够引用其他 MEDFORD文件，从而实现数据重用并减少转录错误。为支持这些变更，并为用户生成的数据规范日益增多做好准备，我们加强了 MEDFORD 中的数据建模和验证例程，并更新了 VS Code 扩展和语言服务器模块。我们还计划进行一次非正式测试，以确定新的宏语法，随后将予以实现。

延续这一主题，我们未来的工作也将聚焦于互联互通和可用性。我们正在开发教学材料和视频，为建立 MEDFORD 用户社区奠定基础。此外，各种导入插件可以从数据文件（如图像和基因组测序仪文件）中提取或关联元数据，这可能会很有用。我们还计划开展正式的用户测试，以确定如何更好地服务科学界、探索新的主／次版本标签要求，并研究新兴技术（如智能代理）如何利用 MEDFORD 等工具提供的上下文来改善用户体验。我们还在研究开发一个用于计算实验（例如基准测试）的日志记录库，使其能够直接输出到 MEDFORD。

我们的试点应用领域是珊瑚礁研究，因为当时我们正从事这一领域的工作，也正是在这个领域中，我们意识到需要 MEDFORD 这样的工具。不过，MEDFORD 显然也适用于许多其他领域。在生物信息学和生物医学的其他领域，轻松收集实验和测序运行的元数据将很有价值。对于涉及算法及其实现的基准测试和计时的计算实验而言，借助 MEDFORD 轻松收集结果，将有助于使计算结果更精简、组织更有序。在网络安全领域，存在多种用于描述威胁和入侵的格式 [11, 1, 3]。然而，缺乏统一标准或格式之间的互通性这一问题，可以通过 MEDFORD 加以解决。

我们的目标是，这些新功能（即外部引用、灵活验证和改进后的 VS Code 扩展）以及未来的功能（教学材料、正式用户测试和其他后续想法）将有助于确保 FAIR 原则能够轻松应用于研究数据。

6 可用性

MEDFORD 解析器和编辑器支持可在以下地址免费获取：

- https://github.com/TuftsBCB/medford

- https://github.com/TuftsBCB/medford-language-server
- https://github.com/TuftsBCB/medford-vscode

本文讨论的这些工具的开发版本可在各代码仓库的“IJMSOrelease”分支中获取。

7 利益冲突声明

所有作者均声明不存在利益冲突。

参考文献

[1] M. S. Abu、S. R. Selamat、A. Ariffin 和 R. Yusof。网络威胁情报——问题与挑战。《印度尼西亚电气工程与计算机科学期刊》，10(1):371–379，2018。[2] A. Ball、J. Greenberg、K. Jeffery 和 R. Koskela。RDA 元数据标准目录工作组：最终报告。RDA 建议，研究数据联盟，2016。[3] S. Barnum。使用结构化威胁信息表达（STIX）标准化网络威胁情报信息。*Mitre* 公司，11(2012):1–22，2012。[4] D. Beckett、T. Berners-Lee、E. Prud’hommeaux 和 G. Carothers。RDF 1.1 Turtle。万维网联盟，25:18–31，2014。[5] T. Berners-Lee、D. Connolly、L. Kagal、Y. Scharf 和 J. Hendler。N3Logic：万维网的逻辑框架。《逻辑编程理论与实践》，8(3):249–269，2008。[6] 生物与化学海洋学数据管理办公室。BCO-DMO。https://www.bco-dmo.org。[7] T. Bray。RDF 的 RPV（资源／属性／值）语法。[8] K. S. Candan、H. Liu 和 R. Suvarna。资源描述框架：元数据及其应用。《*ACM SIGKDD* 探索通讯》，3(1):6–19，2001。[9] M. Corp。Microsoft Visual Studio Code，2025。[10] W. Crichton 和 S. Krishnamurthi。编程语言学习的剖析。《*ACM* 编程语言论文集》，8(OOPSLA1):29–54，2024。[11] P. Gao、X. Liu、E. Choi、S. Ma、X. Yang、Z. Ji、Z. Zhang 和 D. Song。ThreatKG：用于自动收集和管理开源网络威胁情报的威胁知识图谱。*arXiv* 预印本 *arXiv:2212.10388*，2022。[12] J. Greenberg。理解元数据与元数据方案。载于元数据，第 17–36 页。Routledge，2012。[13] O. Kaykova、O. Khriyenko、A. Naumenko、V. Terziyan 和 A. Zharko。RSCDF：面向工业资源的动态且上下文敏感的元数据描述框架。《东欧企业技术期刊》，3(2):55–78，2005。[14] J. Kunze、J. Littman、E. Madden、J. Scancella 和 C. Adams。BagIt 文件打包格式（v1.0）。RFC 8493，2018。[15] F. Manola、E. Miller、B. McBride 等。RDF 入门指南。万维网联盟建议，10(1-107):6，2004。[16] J. Z. Pan。资源描述框架。载于本体手册，第 71–90 页。施普林格，2009。[17] J. Qin、A. Ball 和 J. Greenberg。元数据的功能与架构要求：支持科学数据的发现与管理。都柏林核心与元数据应用国际会议，第 62–71 页，2012 年 9 月。[18] J. Riley。理解元数据。华盛顿特区，美国：美国国家信息标准组织 *(http://www. niso. org/publications/press/UnderstandingMetadata. pdf)*，23(2017):7–10，2017。[19] M. Sfakakis、E. Garoufallou、M. Damigos、A. Salaba 和 C. Papatheodorou。元数据与语义研究：第 *18* 届元数据与语义研究会议（*MTSR 2024*），希腊雅典，*2024*年*11*月*19*日至*22*日，修订版精选论文，第 2331 卷。施普林格·自然，2025。[20] P. Shpilker、J. Freeman、H. McKelvie、J. Ashey、J.-M. Fonticella、H. Putnam、J. Greenberg、L. Cowen、A. Couch 和 N. M. Daniels。MEDFORD：一种兼具人类可读性与机器可读性的元数据标记语言。《数据库》，2022，2022 年 1 月。

[21] P. Shpilker、J. Freeman、H. McKelvie、J. Ashey、J.-M. Fonticella、H. Putnam、J. Greenberg、L. Cowen、A. Couch 和 N. M. Daniels。开放珊瑚礁数据元数据格式（MEDFORD）。载于元数据与语义研究会议，第 206–211 页。施普林格，2022。[22] P. Shpilker、B. Stubbs、M. Sayers、L. Cowen、S. Wallace、A. Couch 和 N. M. Daniels。通过扩展 MEDFORD 语言实现元数据交叉引用。载于 M. Sfakakis、E. Garoufallou、M. Damigos、A. Salaba 和 C. Papatheodorou（编），元数据与语义研究，第 153–158 页，尚，2025。施普林格·自然瑞士。[23] M. Sintek 和 S. Decker。Triple——一种面向语义网的查询、推理与转换语言。载于国际语义网会议，第 364–378 页。施普林格，2002。[24] L. Strand、A. Powers、P. Shpilker、L. Cowen、A. Couch 和 N. M. Daniels。MEDFORD 元数据语言的上下文敏感编辑。载于元数据与语义研究，《计算机与信息科学通讯》，第 278–283 页。施普林格·自然瑞士，尚，2024。