一种基于特定大分子序列比对及自动序列筛选技术的系统发生树绘制与物种演化路径推断系统

朱璟瑞*[a], 胡高远[a], 邓钰潼[a]

[a]河南科技大学附属高级中学(周山校区), 471031

2024 年 9 月 30 日

摘要

近些年来，随着分子生物学的发展，基因序列与蛋白质序列越来越多地被作为绘制系统发生树的重要参考依据，从中诞生了很多相应的技术。但是，这些技术在实践过程中，仍存在着一些问题。比如参考序列选择比较困难等，这些都给绘制出的系统发生树的准确性与严谨性带来了一定的影响。在本文中，笔者搭建了一套用于绘制系统发生树的计算机应用软件，这套系统集成了多种常用的比对与建树算法，同时，该系统还综合了近十年的相关科研数据，使用了人工智能(AI)技术以对建成的系统发生树进行综合分析，从而得出最优异的、最能代表真实物种演化路径的系统发生树，这一突破，能够使得研究人员无需拘泥于参考序列的选择，可缩短科研周期，提升科研效率。同时，本系统包括了插件系统，这使得用户可以自定义需要的序列比对与建树算法，可扩展性更强。该系统的综合能力显著优于目前市面上常用的几款解决方案，使物种演化路径的推断过程变得更加便利、准确，对相关的科研工作可能会起到巨大的推进作用。

关键词: 分子生物学, 序列比对, 系统发生树, 计算机应用软件, 人工智能

目录

*通讯作者: 870239526@qq.com

1 序列比对与系统发生树构建的相关算法简介

作为一种物种演化路径推断系统，序列比对算法与系统发生树构建算法自然是重中之重，下面笔者将介绍本系统中所使用的几种相关算法：

1.1 序列比对方法

序列比对是将两个或多个核酸序列或蛋白质序列排列在一起，以揭示它们之间的相似性和差异性的过程。本系统的序列比对过程，采用的是Needleman-Wunsch 算法，这一算法效率高、准确性好，是目前业内最常用的序列比对算法之一。

1.1.1 Needleman-Wunsch算法

Needleman-Wunsch算法是将动态规划算法应用于生物序列的比较的最早期的几个实例之一。该算法是由 Saul B. Needlman 和 Christian D. Wunsch 两位科学家于1970年发明的 [2]。该算法的状态转移方程如下：

$$F(0,0) = 0$$

$$F(i,j) = \max \begin{cases} F(i-1,j-1) + s(x_i, y_j) \\ F(i-1,j) + d \\ F(i,j-1) + d \end{cases}$$

在生物序列比对算法中，该算法拥有很多优良的特性，如准确度较高、适用于多种序列的比对等。故本项目使用的默认算法便是本算法。

1.2 系统发生树构建方法

常见的系统发生树构建方法有很多，比如最小进化法、最大似然法等等。受开发时间限制，本系统只自带了非加权组平均法与邻接法的建树算法。关于其他方法，用户可以通过本系统的插件系统自行添加。

1.2.1 非加权组平均法

非加权组平均法（UPGMA，unweighted pair-group method with arithmetic means）是一种常用的聚类分析方法。其算法思想基于聚类分析，其核心在于通过计算不同对象之间的距离来确定它们之间的亲缘关系。其具体的算法流程分为以下三步：

首先，需要计算所有对象（如物种、基因序列等）之间的距离。

找出距离最小的两个对象（OTU，Operational Taxonomic Units，操作分类单元），将它们聚为一个新的OTU。新的OTU的分支点位于这两个OTU间距离的1/2处。计算新的OTU与其他OTU之间的平均距离。重复上述步骤，找出距离最小的两个OTU（可以是原始的OTU或之前聚类形成的新OTU）进行聚类。如此反复，直到所有的OTU都聚到一起，形成一个完整的系统发生树。

通过上述聚类过程，最终可以得到一个表示物种间亲缘关系的系统发生树。在系统发生树中，每个节点代表一个OTU，节点之间的连线表示它们之间的亲缘关系，连线的长度则反映了它们之间的进化距离。

1.2.2 邻接法

邻接法（Neighbor-Joining，NJ）是由Naruya Saitou和Masatoshi Nei于1987年首次提出的一种用于构建系统发育树的聚类方法 [5]。邻接法的算法思想基于最小进化原理，即在整个进化过程中，树的分支长度之和达到最小。该算法从一个完全未解析的树开始，然后通过迭代过程逐步构建出完整的系统发育树。具体算法流程如下：

初始化一个完全未解析的树，其拓扑结构对应于星形网络，所有分类单元都从一个中心节点出发。

在每一步迭代中，基于当前的距离矩阵计算一个称为Q的矩阵。Q矩阵的元素Q(i,j)表示将分类单元i和j连接到一个新的中间节点时，树的分支长度之和的增加量。在Q矩阵中找到最小值Q(i,j)，将对应的分类单元i和j连接到一个新的中间节点u。计算每个分类单元到新节点u的距离，并更新距离矩阵。迭代上述过程，用新节点替换连接的邻居对，并使用前一步计算的距离更新距离矩阵。

当所有分类单元都被合并到树中，且树的拓扑结构完全解析时，迭代过程终止。

1.3 常见的解决方案及其弊端

系统发生树构建与物种演化路径推断，现如今市面上已经有了几种常见的解决方案，但它们都有各自的弊端。

2 本系统的基础架构

Python由荷兰国家数学与计算机科学研究中心的 Guido van Rossum 于1990年代初设计 [4]。Python提供了高效的高级数据结构，还能简单有效地面向对象编程。 Python语法和动态类型，以及解释型语言的本质，使它成为多数平台上写脚本和快速开发应用的编程语言，随着版本的不断更新和语言新功能的添加，逐渐被用于独立的、大型项目的开发。基于Python的许多优良的特性以及其良好的可扩展性、丰富的社区内容，我们决定使用Python构建本系统。

本系统的软件架构（除数据外）共分为三个部分：算法包、用户系统、插件系统。本系统的软件目录树（源代码）如下：

```
AITD System
├── aitd ..................... aitd 算法包
│   ├── __init__.py ..... 主文件
│   ├── error.py ..... 错误类文件
│   └── xlist.py ..... 算法目录文件
├── data ..... 软件数据
│   ├── fonts ..... 字体文件夹
│   │   └── ...
│   ├── icons ..... 图标文件夹
│   │   └── ...
│   ├── setting.dat ..... 软件设置数据
│   ├── en.yml ..... 语言数据（英文）
│   └── zh_CN.yml ..... 语言数据（中文简体）
├── licenses ..... 许可证文件夹
│   └── ...
├── aitd_c.py ..... 命令提示符端
├── aitd_w.pyw ..... 窗口端
└── LICENSE ..... 开源许可证
```

2.1 算法嵌入、项目文件夹及命名空间系统

2.1.1 算法嵌入

本系统自带的序列比对、系统发生树构建等算法的源代码均在 `aitd\__init__.py` 中。所有算法（除人工智能外）被分为以下五种：

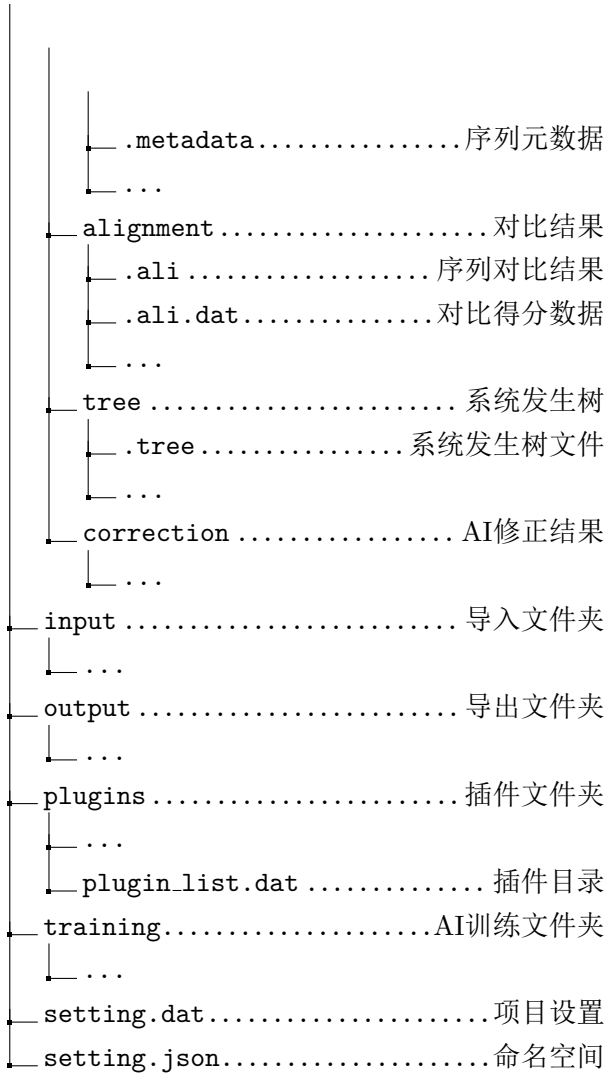
1) 文件解析器(Parser)：解析序列文件格式并存储序列信息；
2) 序列比对(Comparator)： 比对两个或多个序列；
3) 比对加载(Processor)：根据比对结果判断两个序列的差异性；
4) 建树(TreePlanter)：根据多个序列之间的差异性绘制系统发生树；
5) 草图绘制(Display)： 将系统发生树绘制成图像。

算法实装时，使用 `setattr` 函数将该算法的执行函数设置为 `aitd.xlist` 中对应的列表实例的成员函数，同时给算法赋予一个名字。当需要使用该种算法时，可直接使用 `getattr` 函数获取对应名字的成员函数。

2.1.2 项目文件夹

本系统会将用户的数据存储为自定义的项目文件夹。项目文件夹的目录如下：

```
<项目名>
├── cache ..... 缓存文件夹
│   ├── log ..... 运行日志
│   │   └── ...
│   ├── sketch ..... 建树草图
│   │   └── ...
│   └── ...
└── data ..... 数据文件夹
    └── sequence ..... 序列数据
        └── .seq ..... 序列内容
```



2.1.3 命名空间系统

本系统中的所有数据（算法，项目数据等）在软件运行时，均会被赋予一个唯一确定的名字，即命名空间名(Name in the namespace)，以便于进行定位与引用。每一个命名空间名由数据类型和名称标识两部分构成，两部分之间由两个半角冒号(::)连接。

在打开某一个项目后，系统会首先读取项目文件夹中的 `setting.json` 文件，这个文件中存储了所有项目数据的命名空间名以及项目文件的存储地址。随后，系统会将这些数据，以及系统自带的算法等数据全部存储进 `namespace : dict` 中。具体的存储代码如下：

2.2 基于Python的命令提示符+窗体化双端系统

3 系统发生树差别算法与人工智能的引入

3.1 系统发生树的差别分析

3.2 对相关科研成果的综合研判与“最优建树序列”的获取

3.3 人工智能的引入

4 本系统的优异性与目前存在的问题

4.1 系统的可扩展性与可移植性

4.2 系统对物种演化路径的准确判断及其便利性

4.3 目前存在的问题及可能的解决方案

5 致谢与作者贡献声明

5.1 作者贡献声明

朱璟瑞（第一作者）：项目的发起者，完成了项目整体框架的构建、窗口端与算法库的代码；

胡高远：负责了人工智能模型的构建；

邓钰潼：主导了人工智能训练用数据的寻找，完成了控制台端的代码；

尹家宝：参与了人工智能训练用数据的寻找；

白致远：提供了训练人工智能所必需的算力资源。

5.2 致谢

感谢华东师范大学计算机科学与技术学院的赵静博士，她为本项目人工智能模型的构建提了很多建设性的意见；

感谢马来西亚国立大学的刘东升，他帮助我们进行了提示文本与项目文档的英文翻译的审校工

作；

感谢河南科技大学附属高级中学（周山校区）的常易凡老师、陈洪武老师、朱晓东老师，作为我们的指导老师，他们为本项目的构建提供了一些帮助；

感谢河南科技大学附属高级中学（周山校区）的张剑辉老师、罗会杰老师，他们带领笔者走进了生物学和信息技术的大门，本项目的构建亦有他们间接的帮助；

谨以此项目献给河南科技大学附属高级中学（周山校区）的张丹丹老师，感谢您这两年对我们的谆谆教诲，祝愿您生下一个健康快乐的宝宝!

参考文献

[1] 彭焕文, 王伟. 基于分子数据的系统发生树构建[J]. 植物学报, 2023, 58(02): 261-273.

[2] Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins[J]. J Mol Biol, 1970 Mar, 48(3): 443-53.

[3] mmqwqf. 基于python的非加权分组平均法构造简单系统发生树（DNA）[EB/OL]. (2020-10-12), https://blog.csdn.net/mmqwqf/article/details/108988456..

[4] Python Software Foundation. History and License[EB/OL]. (2024-09-21), https://docs.python.org/3/license.html.

[5] Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees[J]. Mol Biol Evol. 1987 Jul; 4(4):406-25.