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[1,*], 胡高远[2], 邓钰潼[3]

[1,2,3,*]河南科技大学附属高级中学(周山校区), 471031

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)技术以自动选择合适的比对序列并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。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

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 [?]。该算法的状态转移方程如下：

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

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

1.2 系统发生树构建方法

1.2.1 最大似然法

1.2.2 非加权组平均法

1.2.3 邻接法

1.2.4 最小进化法

1.3 常见的解决方案及其弊端

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 [?]。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
│   │   ├── .metadata ..... 序列元数据
│   │   └── ...
│   ├── alignment ..... 对比结果
│   │   ├── .ali ..... 序列对比结果
│   │   ├── .ali.dat ..... 对比得分数据
│   │   └── ...
│   ├── tree ..... 系统发生树
│   │   ├── .tree ..... 系统发生树文件
│   │   └── ...
│   └── correction ..... AI修正结果
│       └── ...
├── input ..... 导入文件夹
│   └── ...
└── output ..... 导出文件夹
    └── ...
├── plugins ..... 插件文件夹
│   ├── ...
│   └── plugin_list.dat ..... 插件目录
├── training ..... AI训练文件夹
│   └── ...
├── setting.dat ..... 项目设置
└── setting.json ..... 命名空间
```

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：

```
with  open(os.path.join(projectpath, "setting.json
    "), "r") as f:
pjset = json.load(f)

for i in pjset:
    if i in ["sequence_list", "tree_list", "
        sketch_list", "alignment_list"]:
        for j in pjset[i]:
            namespaces[j] = pjset[i][j]
```