
# 基于 Neo4j 的 Python 代码提示
# 项目需求说明书

1 组

2023 年 7 月

# 目 录

## 一、项目概述

本项目基于 Neo4j 实现 Python 代码提示，解决了传统编译器由于数据类型不明确、第三方库函数等问题导致的没有代码提示的问题，帮助程序员省去借助 type()函数实现代码提示的麻烦过程。将自定义类或函数加入到 Neo4j 数据库中，实现了根据上下文进行变量类型的推测，进而达到代码提示的目的。

项目选择人工智能方向，主要使用 Django 架构、Neo4j 图数据库支持、Echarts 可视化图表、Ace-editor、Jquery、AST 等技术点，使用 Python 语言借助 Visual Studio Code 完成开发。

## 二、工作范围

本项目使用爬虫技术爬取 Python 官方的库函数说明文档，文档中未涉及的用法，本项目中未作拓展。由于时间条件限制，本项目中第三方库仅选取常用的第三方库进行爬取、存入 Neo4j 数据库，并未包含全部。

本项目选定 Python 语言为目标语言，暂不支持包括 C++、Java 等的其他语言。

## 三、功能需求

前端搭建一个简单的代码编辑器，接收输入的代码信息，并对其进行简单的分析，将其分析后得出的代码信息打包发送到后端进行处理。经后端处理后的信息返回到前端，以列表的形式呈现给用

户，即实现代码提示功能。

后端接收前端发送的 json 语句，将代码编辑器中输入的代码进行语法分割，分析得到已输入代码的数据类型等相关信息。然后查询 Neo4j 数据库，基于已得到的关系，得到后续可能的方法、命名空间、类、属性等，将其打包发送到前端，实现代码提示功能。如果基于已知信息无法查询得到可能的后续代码，将返回 None。

数据的爬取及处理上，爬取了 Numpy 官方文档以及 Pands 官方文档，得到了两个第三方库拥有的 namespace 以及 method 和 parameters 还有 method 返回的 class。由于两个文档的布局、编写不统一，得到的数据较为杂乱，固需要在爬取时加入多重筛选以及在爬取后进行人工二次处理，如将返回值类型统一、修改 csv 文件方便数据库写入等。

在进行可视化页面设计时，需要在 Django 中创建一个可视化页面，用来展示数据绘制图，具体做法是利用 Echarts 集成，具体做法是：在 web 页面中使用 Echarts 的 JavaScript 库来创建和配置不同类型的图表，其中关键点是要实现数据交互和动态更新，做法是通过 Ajax 从服务器请求新的数据，并将其更新到图表中，以实现实时或动态的数据展示。

## 四、非功能需求

在进行可视化页面设计时，为了实现性能的优化，对数据进行异步加载，对于耗时的数据处理，采用异步加载的方式，确保页面

快速加载，并在后台进行数据处理；对图表进行优化，使用合适的图表类型和数据粒度，避免加载过多或不必要的数据；压缩和合并前端资源（如 JavaScript 和 CSS 文件），减少页面加载时间。

## 五、数据来源及处理

进行项目爬取时，需要爬取 Numpy 和 Pandas 官方文档，并将其中的信息整理成五个部分：namespace 及其隶属关系部分、method 部分、property 部分、return 部分，然后导入 Neo4j 数据库中的节点和关系。由于这两个文档的布局和编写不统一，则需要在爬取过程中加入多重筛选，并在爬取后进行人工二次处理，以确保数据的准确性和规范性。

**具体流程如下：**

1.爬取与筛选

在进行爬取时，针对 Numpy 和 Pandas 文档的不同布局和格式进行多重筛选。将从文档中提取 namespace 及其隶属关系、method、property 和 return 信息。然后，将把这些信息保存为结构化的数据，方便后续处理。

2.人工二次处理

由于文档的多样性，无法完全依赖自动爬取来确保数据的准确性和一致性。因此，会在爬取后进行人工二次处理，包括统一返回值类型，修改 CSV 文件以便更好地导入数据库。

3.整理成五个部分

为了将数据导入 Neo4j 数据库中的节点和关系，会将整理后的数据划分成五个节点部分，分别是：

·namespace 及其隶属关系部分：将不同 namespace 及其之间的隶属关系提取出来，以便在数据库中建立节点和关系。

·method 部分：提取每个 method 的名称和参数信息，并与对应的 namespace 关联。

·property 部分：提取每个 property 的名称和相关信息，并与对应的 namespace 关联。

·return 部分：提取每个 method 的返回值类型信息，并与对应的 method 关联。

分成五个关系部分，分别是：

·has_method：确定 namespace 中包含的 method

·has_namespace：确定 namespce 是否包含另外的 namespace

·has_property：确定 namespace 中包含的 property

·has_class：确定 namespace 中包含的 class

·return_class：确定 method 和 property 的返回值属于的 class

4.导入 Neo4j 数据库

经过整理和处理后，将数据导入 Neo4j 数据库中。在数据库中建立合适的节点和关系，使得可以对 Numpy 和 Pandas 的信息进行查询和分析。

## 六、数据库

通过使用爬虫技术爬取 Python 官方的库函数说明文档，建立 Neo4j 数据库，如图 6.1 所示，该数据库中包含的节点有 namespace（命名空间）、property（属性）、method（方法）、class（类）、root（根节点），其中的 root 是指导入的包，有 numpy、pandas、一些内置包（List、Dict、Set、Str）。如图 6.2 所示，包含的关系有 has_class（所包含的类）、has_method（所包含的方法）、has_property（所包含的属性）、has_namespace（所包含的命名空间）、return_class（返回值的类型）。

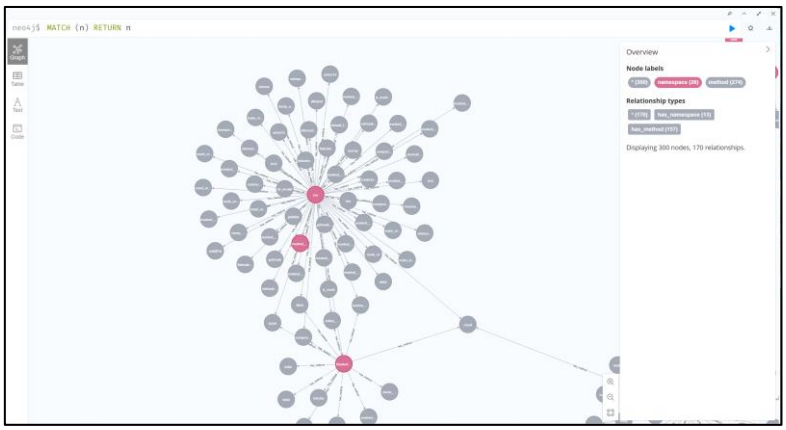

图 6.1 Neo4j 数据库结点

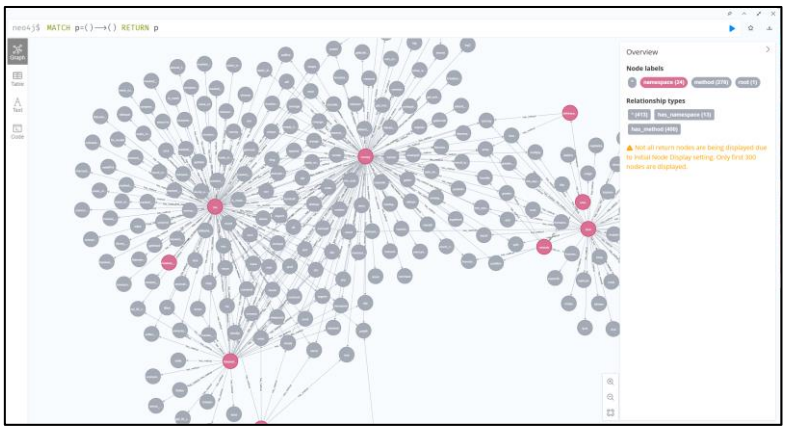

图 6.2 Neo4j 数据库关系

查询时，根据给定的结点限定条件，一步步查询得到给定代码的最后一个语句的下一结点，然后将其以列表的形式打包返回，实现代码提示功能。

## 七、界面需求

该项目的界面共有两个，一个是代码编写界面，一个是 API 查询界面。代码编写界面的网页背景采用了渐变色的效果，从上到下颜色从深到浅；页面顶部有一个导航栏，其中包含了网页的 logo、代码编写选项和 API 查询选项；在导航栏下方是一个代码编辑器的区域，通过 ACE 编辑器实现；页面使用了 Bootstrap 框架，因此具有响应式的设计，可以根据不同屏幕尺寸自动调整布局和样式，以提供更好的用户体验。

API 查询界面的网页背景使用了线性渐变，从顶部的深蓝色逐渐变成底部的浅蓝色；图表使用了 ECharts 库来绘制，展示了一个图形化的数据结构。图表中的节点可以通过点击进行交互，用户点击节点后，会在面包屑导航中动态更新显示点击的节点路径；初始状态下，面包屑导航显示"Home"、"Library"和"Data"三个路径，用于表示页面所处的位置。当用户点击图表中的节点时，面包屑导航会根据点击节点的路径动态更新，每次点击会添加一个新的路径。用户可以通过点击路径中的链接，回到之前点击过的节点；页面中提供了一个搜索框，用户可以在搜索框中输入节点名称，然后点击"Search"按钮进行查询，查询结果会在图表中标示出来，方便用户快

速定位目标节点；在图表显示上，初始状态，界面上呈现的就是所有 root 标签的节点：numpy、pandas、List、Dict、Set、Str，当点击其中的一个节点（以 numpy 为例），面包屑导航上显示路径‘numpy’，界面上显示的就是 numpy 指向的节点，是单路径，再点击 numpy 指向的任意节点，那么该节点就会写入上面的路径中，界面上会展示该节点关系到的其他节点，图表中会出现两级路径，之后再点击下一级节点就会继续写入路径中，界面上的路径级数进一步加深，当点击上面路径中的任意一个节点时，界面上的图表就会回到该级路径下的形式。

## 八、时间进度

| 日期 | 工作任务或里程碑 |
| --- | --- |
| 2023/7/17 | 1.同指导老师讨论项目各项分工<br>2.细化各项分工任务及完善<br>3.完成项目相关文档<br>4.完成项目初步 |
| 2023/7/18 | 1.完成数据的爬取及预处理<br>2.完成数据库基本框架的搭建<br>3.完成代码编辑器的初步搭建<br>4.完成 Python 后端语意分析<br>5.组内交流，确定后期任务 |

| | |
|---|---|
| 2023/7/19 | 1.根据前后端需求更新数据处理<br>2.完成 Python 后端与 Neo4j 的交互，实现语法分析与 Neo4j 的结合<br>3.结合后端模块完善前端编辑器<br>4.美化 Web 可视化 |
| 2023/7/20 | 1.完成各模块调试收尾<br>2.制作答辩展示 PPT<br>3.答辩演练 |
| 2023/7/21 | 1.项目答辩<br>2.技术能力测评 |

## 九、小组成员及分工

**朱子轩——前端开发**

搭建代码编辑器，接收外部输入代码，将所有的代码分块，每次将改变的代码块以 Post 请求方式发送给后端；根据后端返回分析变量类型，给出提示。

**刘修铭——后端开发**

调用 Python 的 AST 库，对代码块进行语法分析，分析代码中的变量，调用的类和方法；查询 Neo4j 数据库分析部分变量类型，返回部分变量类型、类和方法。

**梁晓储——数据爬取及预处理和数据库设计**

爬取 Python 官方说明文档以及 Numpy 和 Pandas 等第三方库官

方说明文档，得到方法名称、参数、返回值等数据并完成数据的预处理，为项目提供数据支持；设计数据库架构。

**陈佳卉——Neo4j 数据库构建和 Web 可视化**

按照已有的数据库架构完成数据库构建，并结合 Django 和 Echarts 完成 Web 界面的可视化设计。