基于 Node.js 的 Python 脚本调用方法

吴晓一

（厦门大学嘉庚学院，福建 漳州 363105）

摘　要： Node.js 自从问世以来，由于其高并发，语言一致等优点，深受开发者的青睐。特别是在前后端分离逐渐成为业界主流的今天，Node.js 更是作为整个技术栈的核心部分而存在。然而，受 Node.js 的自身局限，在后端业务涉及到自然语言处理等人工智能相关领域时，远不如 Python 语言便捷、高效。提出了一种基于 Node.js 的 Python 脚本调用方法，并以一个在线中文分词系统的实例，表明该方法的有效性。

关键词： Node.js 语言；Python 语言；前后端分离；在线中文分词

1　概述

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境，随着前后端分离架构逐步成为 Web 端开发的主流，Node.js 也成为各种技术栈的核心所在。另一方面，Python 编程语言则在科学计算和数据分析领域被广泛应用，伴随着近年来人工智能的火热，更是在机器学习和自然语言处理等方面发挥着难以替代的作用。

在 Web 端应用日益人性化、智能化的今天，后端业务逻辑不可避免地涉及到协同过滤等机器学习算法，以及分词、检索等自然语言处理技术。这方面并非 Node.js 所长，从零开始实现这些算法，（1）实现成本过高，（2）实际运行效率低下。

因此，提出了一种基于 Node.js 的 Python 脚本调用方法，可以在后端业务中调用 Python 写成的脚本，并将 Python 脚本的处理结果输出给前端，以达到智能化 Web 端服务的目的。

2　实现原理

2.1　前后端分离

从软件工程的角度，开发工作可拆分为两种基本分工：前端（front-end）和后端（back-end）。所谓前端，简单来说，就是用户看到的用户交互界面和各种数据的展示。而后端则是具体业务逻辑的实现和数据的持久存储。

理想的前后端开发理应与主从式架构（client-server model）保持高度的一致和统一。即，前端只负责客户端的用户界面，后端只负责在服务端提供服务。客户端向服务端提交调用某种服务的请求，服务端就做出响应并将结果返回给客户端。

基于这个思路，在后端只需准备好各种类似设计的接口，每当客户端的请求方法和请求 URI 发过来，就调用符合该请求的接口，返回相应的状态码和以 JSON 格式承载的响应体就可以了。这类符合 REST（REpresentational State Transfer，表述性状态转移）设计风格的程序接口就可以称为 RESTful API。基于 RESTful API，前、后端在开发过程中就可以实现彻底分离：后端只实现 API 接口，前端只专注于设计用户界面。如此一来，不仅实现了开发者的职责分离，也实现了前后端技术上的分离。

2.2　Python 脚本调用

基于前后端分离架构，实际业务逻辑尽数被封装在 Node.js 实现的接口之中。而涉及到人工智能相关的业务，可以进一步分离在 Python 脚本里，并被接口所调用。调用方面可以借助 Node.js 的第三方库 python-shell 来实现，利用这个库，能够有效地实现 Node.js 和 Python 脚本之间的通信。具体方法如下。

2.2.1 安装与引用

和其他 Node.js 的第三方库一样，使用 Node.js 的包管理器 NPM（Node Package Manager）进行安装。打开任意命令行工具（比如 Windows 的 PowerShell 或CMD），输入如下指令执行即可。

```
$ npm install python-shell
```

并在接口文件中，从该库引出 PythonShell 类。

作者简介： 吴晓一（1981-），男，博士，讲师，研究方向：Web 前后端开发、语料库语言学、机器学习、自然语言处理。

```
import {PythonShell} from 'python-shell';
```

2.2.2 **实例化** python-shell

在向 Python 发送信息之前，需要先将 PythonShell 类实例化。实例化时提供 Python 的脚本名作为参数。

```
const pyshell = new PythonShell(脚本名);
```

2.2.3 Node.js **向** Python **发信**

在实例化 PythonShell 之后，可以使用实例的 send 方法，向实例化时所提供的脚本发送任意信息。

```
pyshell.send(信息);
```

2.2.4 Python **接收来自** Node.js **的信息**

因为 PythonShell 实例的 send 方法传出的信息是经由控制台的，因此，使用 Python 内置的 input() 函数接收该信息即可，input() 函数通过发出提示信息，获取来自控制台的一切输入。比如使用

```
input(提示信息)
```

就可以获取到 2.2.3 中传过来的任意信息。

2.2.5 Python **向** Node.js **发信**

与 input() 获取控制台输入类似，在发信方面也使用Python 的内置函数 print() 即可，但为了和 RESTful API 接口所使用的格式对接，往往需要采用 JSON 格式。具体操作是：先引用 Python 的 json 内置库，再使用 json.dumps 将词典格式转为字符串承载的 JSON 格式，并打印输出。

```
print(json.dumps(词典))
```

2.2.6 Node.js **接收来自** Python **的信息**

监听来自 Python 的信息，需要使用 PythonShell 实例的 on 方法。该方法需要两个参数：一个是名为“message”的监听事件，可以获取到来自控制台的标准输出流，亦即来自 Python 脚本的信息；另一个是具体处理该信息的回调函数（Callback Function）。

```
pyshell.on('message', (message) => {具体处理语句});
```

通过以上步骤，即可实现 Node.js 与 Python 脚本间的通信，该通信机制如图 1 所示。

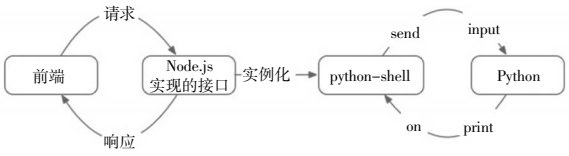

图 1　Node.js 与 Python 脚本间的通信机制

2.3　**防乱码处理**

由于 Windows 控制台的默认编码并非 UTF-8，因此在通信过程中传输中文信息时容易导致乱码。为了防止出现乱码，可以在 Node.js 实现的接口中使用函数 encodeURI() 将字符串编码后再发送，并在 Python 脚本接收后再使用函数 unquote() 对其解码。这就能够彻底避免通信过程中可能造成的乱码问题。

3　实际用例

接下来将基于上述通信机制，实现一个 Node.js 和 Python 脚本间通信的实际用例。

3.1　系统概述

分词（Tokenization），是自然语言处理（Natural Language Processing）的基本技术，具体任务是将一个句子切成一个个单词的序列。比如，“我们都是好孩子”这句话，经过分词处理之后，就变成了“我们”、“都是”、“好孩子”这 3 个词。

如果说欧美诸语（比如说英语）尚可以粗略地用空格作为切割符实现不甚精确的分词，那么像日文、中文这种句中没有空格的语言就需要使用特殊的分词算法了。

在分词方面，Python 语言有许多非常优秀的库，jieba 就是其中之一。基于 Node.js 的 Web 应用调用 Python 程序，并获取其执行结果，再返回给前端显示。

3.2　Python **脚本**

在命令行工具中安装 jieba 分词库，由于这是 Python 而非 Node.js 的库，因此需使用pip 安装而非 npm：

```
$ pip install jieba
```

接下来新建 Python 脚本文件 seg.py：

```
import jieba # 引用 jieba 库
import json # 引用 json 库
    from urllib.parse import unquote # 引用解码函数
#unquote()
if _name_ == '_main_':
    sen = input('请输入一句话：') # 获取用户输入句子
    sen = unquote(sen) # 使用函数 unquote()对该句子
# 解码
    words = jieba.cut(sen) # 使用 jieba 对该句子分词
    result = {"result": ' '.join(words)} # 将分词结果拼接
# 为字符串
    print(json.dumps(result)) # 输出以 json 格式表示的
# 字符串
```

这是一个非常简单的 Python 脚本，总体流程就是经由控制台获取来自 Node.js 的输入，并使用 unquote() 函数对其解码，解码以后使用 jieba 库进行分词，将分

词结果使用空格拼为字符串，并作为 JSON 格式输出到标准输出流。

3.3 Node.js 接口实现

接口方面，在获取用户请求体后做如下处理：

```
...
    const { sentence } = req.body;//从请求体中获取用
//户输入的句子
     const pyshell = new PythonShell('seg.py');//实例
//化一个 Python Shell
     pyshell.send(encodeURI(sentence));//把用户输入
//的句子编码后传给 Python 脚本
       pyshell.on('message', (message) => {//获取到
//Python 脚本的输出
        const output = JSON.parse(message);//将输出
//解析为真正的 json 格式
      return res.json({//返回响应体给前端
        message: '分词成功',
        data: output.result,
      });
    });
      pyshell.end((err) => { if (err) throw err; });//结束
//Python 进程
...
```

接口的主要处理流程如上面代码所示，先从用户传来的请求体中解构出待分词的句子，然后实例化出一个 PythonShell 的实例，利用该实例，向控制台发送经过 URI 编码的用户输入句子，同时监听来自 Python 的标准输出流，一旦获取到 Python 的处理结果，就将其解析为真正的 JSON 格式，返回给前端。

3.4 实际运行效果

在前端随意输入一句话提交，如图 2 所示，借助 Python 的分词脚本，可轻松获取到分词结果，也表明了 Node.js 和 Python 脚本通信成功。

图 2 在线分词系统调用 Python 分词脚本

4 结语

Python 是近年来非常热的一门编程语言，特别是在人工智能、数据分析以及自然语言处理领域，发挥着不可或缺的重要作用。因此，如何将 Node.js 在 Web 开发上的便利性与 Python 在数据处理上的优越性有机地结合起来是一个非常有意义的研究方向。

提出了一个基于 Node.js 的 Python 脚本调用方法，详细阐述了二者的通信机制，并借助一个在线中文分词系统的小型案例，证明了该方法的有效性。

文中所提出的方法，可以在 Web 应用中实现更复杂的数据处理，或更智能的数据展示，以期进一步提高用户体验。

参考文献

[1] Fielding RT. Architectural styles and the design of network-based software architectures [D] . Irvine: University of California, 2000.

[2] Nicolas Mercier. python-shell 文档. https://github.com/extrabacon/python-shell, 2020.

[3] Subramanian V. Pro MERN Stack : Full Stack Web App Development with Mongo, Express, React and Node. 美国：Apress, 2017.

(上接第 11 页)

[4] HUANG, Zhiheng; XU, Wei; YU, Kai. Bidirectional LSTM-CRF models for sequence tagging. arXiv preprint arXiv:1508.01991, 2015.

[5] DEVLIN, Jacob, et al. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[6] BACHLECHNER, Thomas, et al. ReZero is All You Need: Fast Convergence at Large Depth. arXiv preprint arXiv: 2003.04887, 2020.

[7] VASWANI, Ashish, et al. Attention is all you need. In: Advances in neural information processing systems. 2017. p. 5998-6008.