****

**本科毕业论文**

**（2022届）**

|  |  |
| --- | --- |
| **题 目** | **基于知识图谱的github软件生态分析系统** |
| **学 院** | **计算机学院** |
| **专 业** | **计算机科学与技术** |
| **班 级** | **18052318** |
| **学 号** | **18051801** |
| **学生姓名** | **金倩婷** |
| **指导教师** | **陈溪源** |
| **递交日期** | **2022年6月** |

**摘要**

随着开源软件的规模迅速扩大以及其在各个领域的广泛应用，互联网技术的更迭速度有了显著的增长。但是由于其代码公开、易获取、可重用的特点，一旦软件出现安全漏洞，将会给开发团队带来无可估量的损害。Github作为最热门的面向开源和私有软件项目的托管平台，拥有超过900万开发者用户，包含的大量开源项目被应用于实际的开发中。

本研究设计实现了基于知识图谱的Github软件生态分析系统，利用爬虫、Neo4j图数据库、图嵌入算法等技术，实现知识获取、知识表示与存储以及可视化。为了获得数据信息以构建知识图谱，我们利用Github提供的接口对数据进行分析，得到其中的热门Java项目信息。此外，提取出项目依赖，并在Maven库中找到该依赖是否存在漏洞并获取相应的漏洞编号。接着，将数据导入到Neo4j中得到可视化展示，通过Cypher语言能够查询到项目信息以及项目与依赖之间的关系信息。同时，利用图嵌入算法及机器学习算法得到与当前项目最为相关的前五个项目，实现相关项目推荐功能。最后，使用Vue.js及D3.js框架实现系统的构建。

关键词：知识图谱；图数据库；爬虫；软件生态

**ABSTRACT**

With the rapid expansion of open source software and its widespread use in various fields, the pace of technology change on the Internet has grown significantly. Github, the most popular hosting platform for open source and private software projects, has more than 9 million developer users and contains a large number of open source projects that are used in actual development.

This study designs and implements a Github software ecology analysis system based on knowledge graphs, using technologies such as crawlers, Neo4j graph database, and graph Embedding algorithms to achieve knowledge acquisition, knowledge representation and storage, and visualization. In order to obtain data information to build the knowledge graph, we analyze the data using the interface provided by Github to get the information of popular Java projects in it. In addition, the project dependencies were extracted, and the Maven repository was used to find out whether the dependencies had vulnerabilities and to obtain the corresponding vulnerability numbers. Then, the data was imported into Neo4j to get a visual display, and the Cypher language was used to query the project information and the relationship information between the project and the dependency. At the same time, the graph Embedding algorithm and machine learning algorithm are used to get the top five most relevant projects to achieve the relevant project recommendation function. Finally, Vue.js and D3.js framework are used to build the system.

**Keywords**: knowledge graph；graph database；crawler；software ecology

**目录**

[1 引言 1](#_Toc103012632)

[1.1 研究背景和意义 1](#_Toc103012633)

[1.2 研究现状 2](#_Toc103012634)

[1.3 研究主要内容和论文结构 2](#_Toc103012635)

[2 相关研究基础 4](#_Toc103012636)

[2.1 知识图谱 4](#_Toc103012637)

[2.1.1 知识图谱的发展与概念 4](#_Toc103012638)

[2.1.2 知识图谱的构建 4](#_Toc103012639)

[2.1.3 知识库产品 5](#_Toc103012640)

[2.2 Python爬虫 6](#_Toc103012641)

[2.2.1 HTML语言 6](#_Toc103012642)

[2.2.2 Python爬虫技术 7](#_Toc103012643)

[2.3 Neo4j图数据库 7](#_Toc103012644)

[2.3.1 数据库基础 7](#_Toc103012645)

[2.3.2 Neo4j图数据库介绍 8](#_Toc103012646)

[2.4 JS框架 9](#_Toc103012647)

[2.4.1 Vue.js框架 9](#_Toc103012648)

[2.4.2 D3.js框架 9](#_Toc103012649)

[3 构建基于Neo4j的Github软件生态知识图谱 10](#_Toc103012650)

[3.1 知识获取 10](#_Toc103012651)

[3.1.1 获取项目信息 10](#_Toc103012652)

[3.1.2 获取项目关键文件 11](#_Toc103012653)

[3.1.3 获取项目依赖 12](#_Toc103012654)

[3.1.4 获取漏洞数据 13](#_Toc103012655)

[3.2 Neo4j图数据库存储表示 14](#_Toc103012656)

[3.3 相关项目推荐 16](#_Toc103012657)

[3.3.1 知识图谱推荐算法 17](#_Toc103012658)

[3.3.2 图嵌入算法 17](#_Toc103012659)

[3.3.3 图嵌入算法比较 18](#_Toc103012660)

[3.3.4 KNN算法 20](#_Toc103012661)

[4 Github软件生态分析系统 21](#_Toc103012662)

[4.1 系统功能设计 21](#_Toc103012663)

[4.2 系统实现 22](#_Toc103012664)

[4.2.1 登录界面 22](#_Toc103012665)

[4.2.2 项目信息展示界面 22](#_Toc103012666)

[4.2.3 整体3D展示界面 23](#_Toc103012667)

[4.2.4 知识展示查询界面 24](#_Toc103012668)

[4.3 系统测试 25](#_Toc103012669)

[4.3.1系统健壮性测试 25](#_Toc103012670)

[4.3.2系统功能测试 25](#_Toc103012671)

[5 结语 27](#_Toc103012672)

[5.1 工作总结 27](#_Toc103012673)

[5.2 展望 28](#_Toc103012674)

[致谢 29](#_Toc103012675)

[参考文献 30](#_Toc103012676)

# 1 引言

## 1.1 研究背景和意义

近年来，开源代码在代码库中的比重不断增加，凭借其灵活性、稳定性和相对较低的费用获得了不少公司和企业的青睐。开源软件始于上世纪70年，拥有代码公开、易获取、可重用的特点，近年来热度不断升高。在软件开源化的趋势下，开源软件已逐渐形成庞大规模，并在整个软件供应链中占有举足轻重的地位。但是，随着开源软件在各行业的应用越来越广泛，如果出现安全问题，那么将给开发团队，以及各大安全团队带来严峻的挑战。

奇安信实验室发布的《2021中国软件供应链安全分析报告》显示，安全漏洞存在于超过百分之九十的软件项目中；超过40%的开源项目存在覆盖率最广的安全漏洞；15年前的安全漏洞在当前多个开源项目中依然存在。RiskSense发布的报告显示，2019年公布了968个开源CVE漏洞，超过往年峰值的两倍，同时2020年新增的安全漏洞数也居高不下。

美国国家漏洞库（National Vulnerability Database，NVD）[1]针对开源软件的安全漏洞管理具有很多缺陷，例如收录漏洞的时间相对较晚，漏洞从发现到被公布平均需要花费54天时间，甚至最多需要近五年，黑客很有可能会在这个窗口期开发并部署漏洞应用软件。虽然例如 Apache Tomcat、 Magento和 JBoss等开源软件针对漏洞危害做了一定的武装，但是也经常会因为没有及时接收到 NVD的安全警告而导致公司处于完全的安全危险境地中。由此可见，开放源码安全问题广泛存在。此外，开源软件的安全漏洞增长速度迅猛，且其自身的追踪能力受到限制，很难得到及时的修补，所以开源软件的安全漏洞检测研究具有重大意义。

如今，软件体系不再是各自为政，各软件形成了一个具有多样性、复杂性、开放性、健壮性和可持续性的软件生态体系[2]。软件生态系统是一个由软件和开发人员以及它们之间的关系共同演化出来的复杂的社会技术体系，在这个生态系统中，软件和开发人员、软件、软件的相互关系是我们必须要考虑的问题。软件生态的研究正是本课题的核心所在，所以选用一个恰当的数据模型来描述这种对象及对象之间的关系是十分必要的。

在智能问答[3]、语义搜索[4]、个性化推荐[5]等方面，知识图谱为人们提供了更好的组织、管理和理解网络信息的方法。与其它数据模型相比，知识图谱更注重实体间的关联，有助于在现实生活中挖掘关联的特性，而这种特性恰好满足了生态系统对数据模型的需求。

综上，本课题期望从宏观的角度即软件生态来研究当前软件之间的依赖关系及发展趋势，并搭建开源软件生态系统知识库，借助知识图谱的特性进行深层次的软件生态系统研究，为提高开源软件行业的安全性做出力所能及的贡献。

## 1.2 研究现状

随着软件产业的不断发展，安全问题日益受到人们的重视，而在企业中，最主要的问题还是代码的安全问题。当前，维护软件安全的主要方式有两种，一种是将安全保护机制嵌入到软件的开发中，另外一种是对现有的软件进行安全扫描。

在软件源码安全保护的实践中，微软公司的SDL和高德纳公司的 DevSecOps[6]具有典型性，两者都注重软件生命周期的安全性。其中， SDL的中心思想是针对软件开发的各个环节，采取相应的安全措施来减少软件中的漏洞并将漏洞危害降到最低。DevSecOps则提升了安全监控、跟踪和分析的能力，并将安全任务分配至各个团队，使其效率大大提升。安全扫描工具主要有两种：一是通过代码级的检查来检测潜在的危险，比如 SQLMap, Wfuzz, Arachni；另外一种方法是扫描依赖关系后，与漏洞库中的已公布的漏洞信息进行比较，从而获得目前项目的所包含的漏洞个数与相关情况，比如Dependency-Track、WhiteSource、Synopsys、Snyk等。

此外，各公司开发了相应的开源代码漏洞库[7]帮助用户检测和定位漏洞信息，例如Open Source漏洞数据库旨在更好地对开源漏洞进行分类、存储和查询，做到对其项目更好地维护和管理。技术上采用二分法来定位漏洞版本并提供修复版本；WhiteSource漏洞数据库旨在为使用者提供开源安全服务和许可证安全管理服务，并提供自动报警功能，技术上使用软件组成分析来发现开源组件中的漏洞。

综上，纵览当前软件安全性维护措施与工具，绝大部分的关注点都仅在于当前软件。这虽然能够提供一定的安全性保障，有效降低当前软件的开发成本，但却忽略了软件之间的相互依赖关系是引发当前软件危机最主要的原因。因此从宏观的角度来维护软件的安全性有着积极的意义，而软件生态的研究正是本课题的核心所在。

## 1.3 研究主要内容和论文结构

本文研究的主要内容是基于知识图谱的Github开源软件生态知识图谱。从Github中爬取项目信息，经过数据处理、数据存储以及可视化后，将项目以及它们之间的依赖关系通过三元组样式进行展示，并将所依赖的部分与漏洞数据库进行匹配，使用户清晰的把握该项目依赖存在的漏洞。本文着眼于开源软件之间的依赖关系，设计基于Neo4j图数据库的知识图谱，将项目以及项目之间的关系进行统一存储，设计并完成Github软件生态分析系统。主要工作如下： 第一：数据获取与处理，根据Github提供的接口爬取热门Java项目，并使用网络爬虫爬取项目主文件夹及子文件夹下的pom.xml文件，使用Maven指令获取项目文件中的依赖树文件，得到该项目与其他项目及构件的依赖关系。接着在Maven漏洞库中得到相应漏洞信息，为 Github软件的生态知识图谱的建立提供了数据支撑。第二： 通过数据收集得到项目依赖关系以及漏洞信息后，将其导入Neo4j图表数据库，使用 Cypher语法进行快速准确查询。第三：使用基于图嵌入的Node2Vec算法进行图特征学习，实现项目相关性推荐。第四：使用Vue.js和D3.js框架进行可视化展示。

全文由五章组成，其主要内容和结构如下所示：

第一章、引言

本章概述了本文的研究背景及意义，分析了开源软件的发展过程及其伴随出现的安全漏洞风险，提出了软件生态分析的重要性，解释了使用知识图谱作为数据模型的原因。接着，介绍了开源软件安全漏洞技术与产品的研究现状。最后，概括了本文研究的主要内容和组织结构。

第二章、相关研究基础

本章论述了本文主要涉及的基本技术原理，包括知识图谱的发展与应用，爬虫获取数据及其处理的方式，Neo4j图数据库的数据存储和Cypher语法的数据查询，以及Vue.js框架和D3.js数据可视化框架技术。

第三章、构建基于Neo4j的Github软件生态知识图谱

本章主要分析软件生态知识图谱的构建过程，包括数据的来源和处理，Neo4j数据节点的设计和存储，相关项目推荐算法的分析，以及使用Cypher语言对源数据的查询与展现。

第四章、Github软件生态分析系统

本章详述了系统构建的目标以及整体框架的设计，包括数据库的设计和系统界面的设计。实现了基于知识图谱的Github软件分析可视化展示。

第五章、结语

本章作为总结，对工作内容进行了概述，分析了其中的优点与特色，同时指出了目前存在的问题并明确了今后的改进方向。

# 2 相关研究基础

在第一章介绍了开源软件发展过程及检测漏洞技术的研究现状之后，本章将介绍该系统所涉及的相关技术原理，包括知识图谱、网络爬虫、Neo4j图数据库、Vue.js框架和D3.js框架。

## 2.1 知识图谱

### 2.1.1 知识图谱的发展与概念

知识图谱（Knowledge Graph）[8]的起源可以追溯至语义网络、人工智能、数据库、知识表示和推理等不同研究领域的发展与进步。其中，人工智能和语义网络的发展大大推动了知识图谱的出现与进步。在人工智能领域，人们希望计算机能够像人一样“思考”，具有对信息分析、推理、预测的能力。而知识图谱能帮助实现这一目的，它能够更好地理解信息，并可设计相关算法对数据进行预测推理。在语义网领域，计算机技术的快速发展使得互联网中的数据量急速增长。虽然数据的存储和检索技术也在持续地向前推进，但仅仅只是满足了对数据的低级操作，其复杂工作例如推理、预测等工作一直没有得到显著的发展。对于企业来说，获取数据后仍需耗费大量的时间和财力来进行数据的分析与处理。在这种情况下，利用原始的数据以支持推理分析等复杂工作的需求大大提升，而知识图谱能够很好地实现这个目标。

知识图谱核心概念为结构化的语义知识库，自2012年谷歌推广以来，得到了学术界和业界的广泛关注。知识图谱用图形和符号来表示信息概念及其之间的联系，以“实体-关系-实体”三元组为主体。它由实体、关系、属性组成，实体则为真实事件中概念、人、事物，关系则为实体与实体联系起来的媒介，属性可蕴含于实体以及实体间的关系中。知识图谱概念的提出，使得组织、管理和理解数据信息更加准确快速，在语义搜索、智能问答、信息检索、语义解析等方面都有着广泛的应用。

### 2.1.2 知识图谱的构建

知识图谱的构造方法包括自顶向下和自底向上。其中，自顶向下的构建是指先定义好本体和数据模式，再从诸如网页等结构化的知识库所提供的优质数据中提取本体和其他相关信息，并将其纳入到知识库中。而自底向上[9]构建，则是指利用特定的技术，从已有的公开数据中抽取出实体信息，从中选择优质部分并将其纳入到知识库中，最后定义顶层的本体和数据模式。

其中，自底向上构建应用最为广泛，包括信息抽取、知识融合、知识加工三个步骤[10]，如图2.1所示。其中，信息抽取是从不同的资料来源中抽取出实体、属性和联系，从而形成知识表示，包括实体抽取、关系抽取、属性抽取三个步骤；知识融合是在获得实体、关系以及实体的属性的基础上，将从文本中抽取得到的实体对象，使用相似度计算与相应的知识库实体对象进行关联，并消除矛盾和歧义。例如同一个实体有不同的表达，或者某个表达对应了多个不同的实体，包括实体链接和知识合并两个步骤；事实并不等于知识，所以还需要进行知识加工，从中选取优质的部分加入到知识库中，包括本体构建、知识推理和三个步骤。

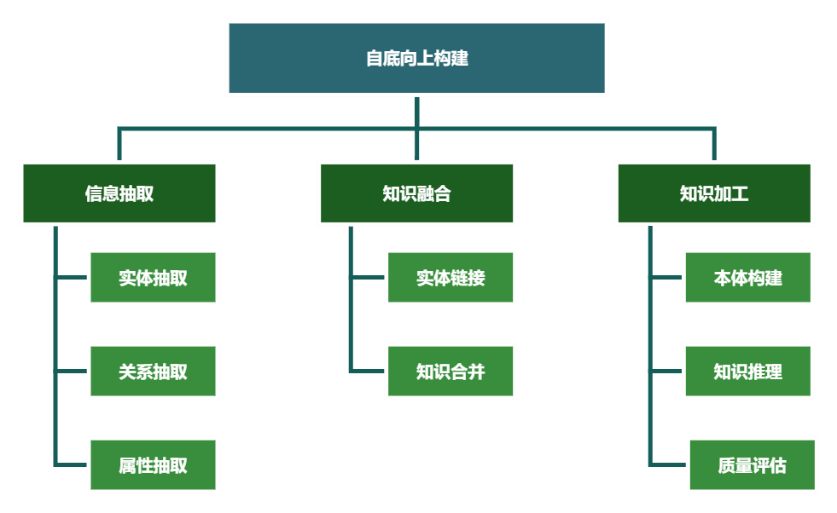


图2.1 自底向上构建流程图

对于一般的知识图谱构造来说，如果实体表达单一，且知识源优质无歧义，则可以将构造知识图谱的流程分为知识获取、知识表示与存储以及知识可视化三个步骤。

### 2.1.3 知识库产品

目前，知识图谱的研究是网络信息领域的一个热点。表2.1列出了目前最受欢迎的知识库产品和类似的应用， WolframAlpha知识库拥有10兆多个实体，规模巨大；谷歌目前拥有5亿个实体，350亿条关联，且其规模还在持续扩大；微软的 Probase是目前拥有最多概念的知识库，总数已达数千万，包括苹果 Siri、谷歌 Now在内的热门智能助手软件都是基于该知识库。近年来，互联网巨头如 Facebook，苹果， IBM等，也纷纷加入到知识图谱的研究中去，但由于现有的技术与规范尚不成熟，知识图谱仍处在不断演变的过程中。

表2.1 知识图谱及相似产品

|  |  |  |
| --- | --- | --- |
| **Knowledge Base** | **Products** | **Data Source** |
| Knowledge Vault | Google Search Engine/Google Now | Wikipedia, Freebase, Web Open Data |
| Wolfram Alpha | Apple Siri | Methematica |
| Satori/Probase | Bing Search Engine/Microsoft Cortana | Wikipedia, Web Open Data |
| Watson KB | IBM Watson System | Web Dictionaries, The World Book Encylopedia |
| DBpendia KB | DBpendia | Wikipedia |
| YAGO KB | YAGO | Wikipedia |
| NELL KB | NELL | Web Open Data |
| Facebook KB | Shopycat | Social Network Data |
| Zhilifang KB | Sougou Search Engine | Web Open Data |
| Zhixin KB | Baidu Zhixin Platform | User Generated Content |
| Cross-Lingual KB | XLORE | Chinese/English Enclopedia, Wikipedia |
| Zhishi.me KB | Zhishi.me | Chinese Encyclopedia |

## 2.2 Python爬虫

### 2.2.1 HTML语言

Web网页的制作使用HTML（Hyper Makeup Language，超文本标记语言）,于1990年创立，如今已发展到HTML5，与其他的Web技术结合，便能制作出功能完备的网页。

HTML使用不同的标签进行网页布局，一般也分为三类。第一类属于网页的布局标签，用于将网页布局分为多个模块，方便对各个区域进行局部布局，比如<div>、<span>、<li>等；第二类属于链接标签，使得页面之间能够相互联系，以及方便各种资源的相互传递，包括<a>标签中的“href”连接到网页地址，用户只需点击该HTML展示部分便能跳转到指定页面，又比如<img>标签中的“src”，将src地址中所指向的图片传递到该HTML页面中，实现网页图片的展示；第三类属于特征标签，能够辅助显示HTML元素的表示特征，比如<strong>标签用于加粗文本，达到强调的作用。

### 2.2.2 Python爬虫技术

互联网上存在着大量有用的信息，只需通过搜索引擎就能获取到成千上万的信息，对于研究和学习来说，总能找出其中的有用信息并加以利用。但对于其中一些研究来说，所需要的数据并非只有一个网页中的少数部分，可能需要成百上千条，甚至以亿为单位。此时通过人工方式来搜寻、整理数据往往需要耗费大量的时间精力。此时，如何正确快速的提取出所需的信息成为了一大难点，网络爬虫技术应时而生。

虽然Java、C、C++等语言都可以实现网络信息的爬取，但是大部分的网络爬虫项目都是以Python为基础的。相较其他语言，Python所提供的抓取接口使用更加简单、功能更加全面。其次，爬虫的原理就是模仿浏览器进行搜索的过程，而Python提供的第三方包可以帮助完成这一行为，包括模拟登录，存储设置等。此外，Python的Beautiful Soup以简洁的语言和表现形式，帮助用户在大量冗余信息中，对其感兴趣的数据进行提取。

## 2.3 Neo4j图数据库

### 2.3.1 数据库基础

数据库可分为关系型数据库和非关系型数据库，表2.2详细对比了以Mysql[11]为代表的关系型数据库和以Neo4j[12]为代表的非关系型数据库。其中，关系型数据库以数据库表形式存储于硬盘中，灵活度欠缺，而非关系型数据库可以以包括键值对在内的多种形式存储于网络缓存中，使用灵活，应用场景广泛。故非关系型数据库的查询速度远远快于关系型数据库。此外，关系型数据库使用传统SQL语言进行查询，而非关系型数据库提供的查询语言更为简单。此外，在大量数据的读取和写入方面，关系数据库比较起来稍逊一筹。

表2.2 关系型数据库与非关系型数据库对比

|  |  |  |
| --- | --- | --- |
|  | **关系型数据库（Mysql）** | **非关系型数据库（Neo4j）** |
| 存储形式 | 数据库表 | 键值对 |
| 事件处理 | 与外部程序无法通信 | 外部事件可监听 |
| 查询 | 索引SQL | Cypher语言 |
| 查询速度 | 慢 | 快 |
| 数据扩展 | 纵向扩展 | 横向扩展 |
| 存储位置 | 硬盘 | 网络缓存 |
| 成本 | 高，需购买 | 低，开源 |

### 2.3.2 Neo4j图数据库介绍

图数据库（Graph Database，GDB）对于操作大量复杂且结构灵活的数据有着天然的优势。对于非结构化的数据来说，相比于使用关系型数据库复杂的操作，图数据库使用节点、边和属性来表示和存储数据，节点关系查询无需冗长的递归查询和多表联合查询，因此更适合用来解决复杂的关系问题。此外，图数据库支持多种图挖掘算法，在分布式存储上也有很大的优势。

而Neo4j是图数据库中的最为优秀的代表之一，始于2003年，经过多年的发展，已经被广泛地运用于生产环境中。它拥有很多优点，包括支持完整的ACID（原子性，一致性，隔离性和持久性）规则，可靠性高；能够以更自然地存储和管理数据，且不受数据类型和数量的限制；拥有嵌入式、高性能、轻量级特征。

Neo4j提出全新的Cypher查询语言，简化了查询语句。Cypher是一种声明式图查询语言，受到了如SQL、SPARG的启发，表现能力强，能够有效地对图表进行查询与更新。一个Cypher 查询语言包括四部分：表达式、图模式、子句和查询。针对一个属性图而言，Cypher 语句既包括查询也包括数据更新和操作等功能。Cypher中包含数值、字符串、布尔、节点、关系、列表六种基本形式，使用一对圆括号（）来表示节点，使用CREATE 命令用于创建节点和关系，使用MATCH 命令用于从数据库中获取节点，关系的信息，使用WHERE 子句来过滤 MATCH 的查询结果。

## 2.4 JS框架

### 2.4.1 Vue.js框架

Vue.js是一个渐进式的 JavaScript框架，同时也是一个轻量、高性能的MVVM（Model–view–viewmodel）库，将视图和业务逻辑分开，做到能够方便快捷地构建用户界面和传递数据。此外，Vue.js框架设计为能够从底部到顶部一层一层地进行开发，即增量开发。此外，相比于其他框架的学习，Vue内核小巧、注重于视图层的开发，学习起来相对容易。

Vue.js框架具有两大特征：第一点为响应式的数据绑定，将数据与结构分离，使开发者不用再去操作DOM对象，做到更加注重业务逻辑的开发；第二点为组件化，将各个功能模块划分为可循环使用的组件，大大提高了开发效率。同时，整个项目的可维护性和团队协作开发能力得到了显著的提升。

### 2.4.2 D3.js框架

随着时代的发展，数据量呈爆炸式增长，而由大量文字所描述的信息数据显得过于晦涩和冗余，寻找规律及进行预测需要人工进行大量的阅读和数据分析。此时，数据可视化应运而生。数据可视化即对拥有的大量信息进行可视化处理，用图表、图形各种方式来展现事物的概念、发展、规律。近年来，可视化越来越受到人们的欢迎，应用范围也越来越广，各种各样的数据可视化工具也在飞速发展，D3.js框架便是领域中非常强大的存在。

D3（Data-Driven Documents）, 是一个JavaScript的函数库，遵循了当前的 Web标准，无需其它框架就能在当前的浏览器中独立运作，并与强有力的可视组件一起实现 DOM操作，且提供了多种易于使用的功能，使 JavaScript的数据处理变得更加容易。虽然使用原生JavaScript同样可以做到D3.js能够做到的一切，但是使用D3能够极大减少工作量，特别是在数据可视化上，D3把生成可视化的繁琐过程简化，只需输入一些简单的操作命令，就可以将其变成优雅的图像。

D3还有一点明显的优势，即可以直接操作矢量图形（Scalable Vector Graphics, SVG）。SVG使用XML（Extensible Markup Language，可扩展标记语言）格式来定义图形，使得图像质量不会因为放大或者更改大小而变差，同时可以为图像增加动态效果。对于图数据库中成千上万的节点与关系来说，SVG图形可以使数据信息更加直观立体地进行展现，方便用户观察得更具体，对数据的把控也更准确。

# 3 构建基于Neo4j的Github软件生态知识图谱

上一章中介绍了构建基于Neo4j的Github软件生态知识图谱所需要了解的相关理论，例如知识图谱、网络爬虫、Neo4j图数据库等。而构建一个知识图谱，需要经历知识获取、知识表示与存储以及知识可视化三个步骤。我们将重点关注Github项目中所包含的依赖关系，利用爬虫技术进行知识的获取，利用Neo4j图数据库进行知识的表示与存储，并进行可视化展示。同时，使用图嵌入算法，增加相关项目推荐功能，构建一个系统、优质的软件生态知识图谱。

## 3.1 知识获取

### 3.1.1 获取项目信息

针对数据源分析，本论文将Github中的Java项目数据作为数据来源。开始尝试使用Python爬虫爬取Github项目的网页内容，但由于Github 设置了反爬机制，导致常常出现连接超时、读取超时的错误。同时使用代理连接虽然能够达到目的，但是需要隔一段时间才能进行一次爬取，效率较慢，故需要一个快速便捷的方法得到所需数据。最终使用Github api进行数据获取任务。

Github提供了公开的api接口使得用户可以进行多样的Github操作，是与Github交互的优秀工具。用户可通过该api获取用户、项目等信息。支持两种验证访问方式，第一种是使用token认证，第二种是使用auth认证。相比于后者，前者更加方便，只需在个人用户界面进行申请，对于生成的token还可以进行访问权限范围的设置。在构造的http请求的header中增加上述token认证，即可进行访问。

因为只需获取Github的Java项目中前1000个项目文件，故采用api所提供的“repository\_search\_url : https://api.Github.com/search/repositories?q={query} {&page,per\_page,sort,order}”查询方法，得到包含所有Java项目的相关信息，并筛选出其中的有用部分，包括项目id、url、名称、star数、fork数、描述信息、创建时间及更改时间，如图3.1及图3.2。



图3.1 获得项目网址（部分）



图3.2 获得项目信息（部分）

### 3.1.2 获取项目关键文件

在IDEA中，点击Maven展开项目就能看到项目引入的jar包，再点击Show Dependencies ，即可打开项目依赖树图。这种方法对于已经下载到本地的单个文件来说非常方便快捷，但是对于我们的项目数量要求来说，成百上千个文件的项目依赖获取如果采用这种人工的方法，那么将耗费巨大的时间和精力。所以我们采用提取项目pom.xml关键文件并依靠该文件生成依赖树的方法，实现信息获取自动化。以下介绍如何提取项目pom.xml关键文件。

Pom（Project Object Model, 项目对象模型）通过XML格式进行保存，是Maven 的配置文件，用以描述项目的各种信息，包括源代码、配置文件、开发者信息、项目依赖关系等等。其中的项目依赖关系，正是我们需要提取的关键信息。此外，一个完整的pom.xml文件，放置在项目的根目录下。一般包括基本配置、Maven坐标、依赖配置、构建配置、项目信息、环境配置等等，其中的依赖配置是一大重点，包括dependencies、parent、dependencyManagement、modules、properties模块。

进入Github官网，观察项目根目录下的pom.xml文件，点击Github为文件提供的raw标签，可进入raw.Githubusercontent.com资源库，使用Python3中的urllib.request模块提供的urlretrieve()函数，可直接将远程数据下载到本地。故将所得的项目Github地址中的“https://Github.com/”更换为 “https://raw.Githubusercontent.com/”，并添加“/master/pom.xml”路径，即可得到该项目根目录的pom.xml资源库地址，使用urlretrieve()函数即可下载。

但仅有根目录的pom.xml文件并不够，项目中并非只有一个pom.xml文件，其子文件夹中的子项目也引用了不少依赖，相应的也会有该文件，而寻找相应的子文件也是一大挑战。观察主目录下的pom.xml文件，可以发现在依赖配置的modules模块中，有着相关子文件的信息。以JakeWharton的ActionBarSherlock项目为例，其所包含的modules列表如图3.3所示，包含actionbarsherlock、actionbarsherlock-fest、actionbarsherlock-i18n、actionbarsherlock-samples四个文件夹，目录下都有pom.xml文件，与实际包含依赖文件夹相符。

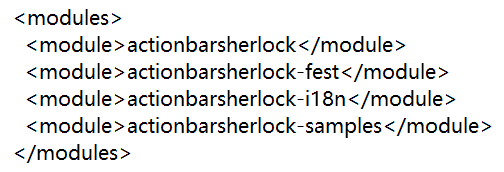


图3.3 ActionBarSherlock项目子modules

得到子modules之后，将目标url网址设为该子文件夹所在路径，使用urlretrieve()函数下载目录下的pom.xml文件后，递归查询其是否存在子模块依赖。遍历完整个项目文件后，得到项目包含pom.xml文件的所有子文件夹并将其下载的pom.xml存储到所属文件夹路径之下。

### 3.1.3 获取项目依赖

获得项目所包含的全部关键文件后，会发现所得的pom.xml文件散落在各个文件夹中，虽然每个pom文件都有详细的依赖包信息，但是分别收集、存储需要耗费很大的精力，故采用Maven提供的命令生成依赖树，清晰地获取项目所包含的依赖信息。

Maven是 Java开发人员使用的一种依赖管理和代码构造的标准，最近几年成为最受欢迎的软件包管理工具。已知项目主文件夹下的pom.xml，Maven会将其包含的依赖引入，并对该pom.xml文件下的依赖进行递归解析，最终获得一种类似树的依赖结构。

具体实现流程如下：在cmd命令行下，进入项目主pom.xml所在的目录，执行“mvn dependency:tree”命令，便可获得项目依赖树。以JakeWharton的ActionBarSherlock项目为例，生成的依赖树如图3.4所示。

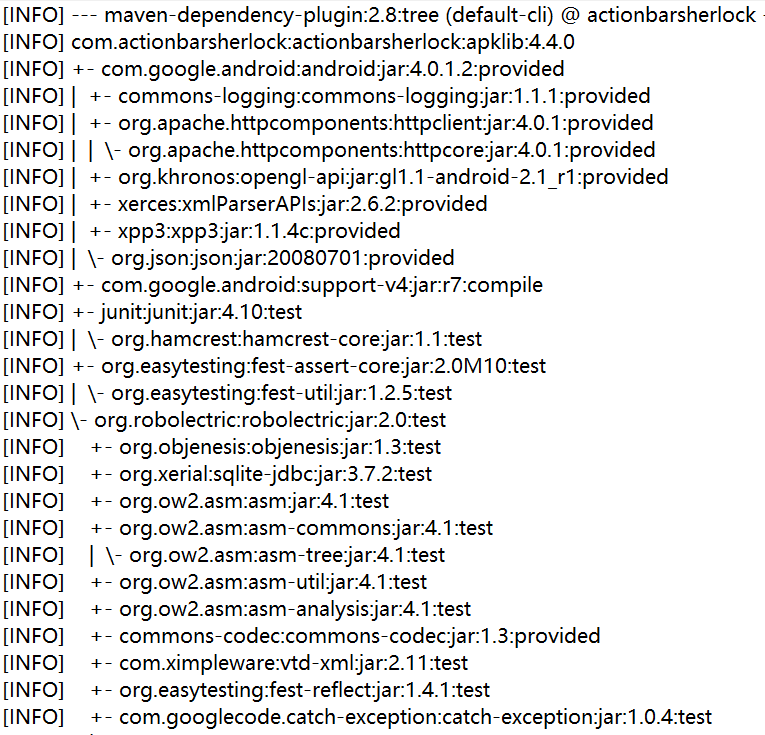


图3.4 ActionBarSherlock项目依赖（部分）

### 3.1.4 获取漏洞数据

获取项目依赖后，下一步需要判断有关依赖是否存在漏洞。Maven中央仓库中提供了漏洞查找功能，用来展示各个依赖版本的漏洞信息，其中描述了依赖版本的证书、分类、描述信息，发布的各类版本和发布时间。对于依赖的各个版本，都提供了其漏洞编号，点击即可在漏洞库中查找对应的漏洞信息，方便开发者维护项目安全。Maven库中的依赖漏洞信息可通过网络爬虫技术进行获取。

爬虫技术是一种根据特定规则，模拟浏览器打开网页，对网页信息进行批量自动抓取的技术。一般来说，使用Python爬取页面信息包括四个步骤，包括请求网页、解析网页、提取网页数据、存储数据。具体操作流程如下：

1）请求网页

利用功能强大的Python3所提供的第三方库Request，向目标网址发起请求，即利用Request.get()方法，可包含headers、data等信息，获取网站响应信息。此步骤为模拟在浏览器中输入网址并提交的过程。

2）解析网页

服务器正常响应后可收到包含所有网页内容的Response，其中包括HTML、Json字符串、图片视频等等。随即使用BeautifulSoup工具解析网页，获取网页HTML信息。

3）提取网页数据

解析网页后，可以发现所得HTML网页内容存在大量冗余信息，故需要对感兴趣的数据进行提取，方法包括使用正则表达式、BeautifulSoup等。我们使用BeautifulSoup进行信息提取。通过所提供的soup.find()函数定位到漏洞标签所在的class类，分别提取<a>标签中的文字信息，如图3.5，即可得相应依赖的CVE漏洞编号。同样的，通过该方法可以获取漏洞的描述信息、发布时间、最后修改时间、风险等级、url地址以及影响版本等有用信息。

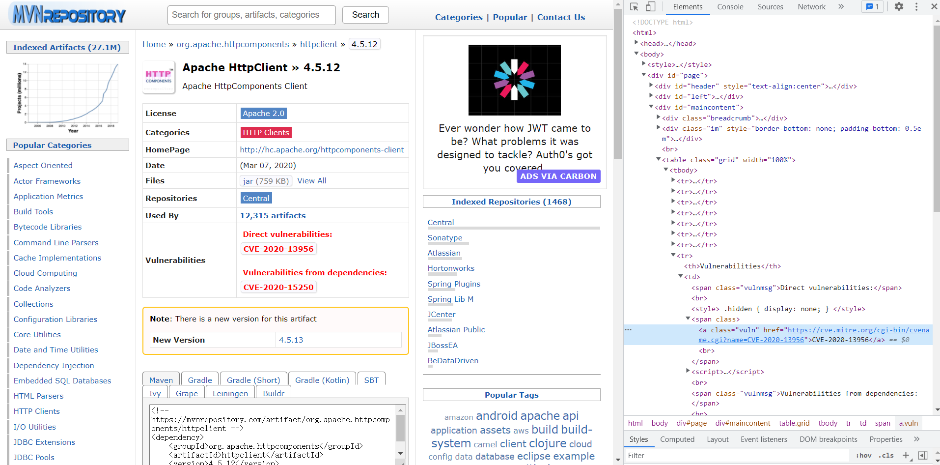


图3.5 Maven漏洞网页

（4）存储数据

可以将所需要的信息进行存储在文件中，也可以直接将数据信息导入数据库中。

## 3.2 Neo4j图数据库存储表示

通过上述步骤，我们得到了项目信息及漏洞信息。该数据信息以三元组的形式保存到Neo4j图数据库中。以图3.6为例，在本Neo4j图数据库中，设计了两种类型的节点，不同类型的节点有不同的颜色。紫色节点为Repository类型的“Joda-time”和“Storm”项目节点，两者的边关系分别为支持与依赖；橙色节点为Vulnerability类型的“CVE-2022-23302”漏洞节点，指向紫色节点Repository的单向关系为影响，反之指向的单向关系为被影响。

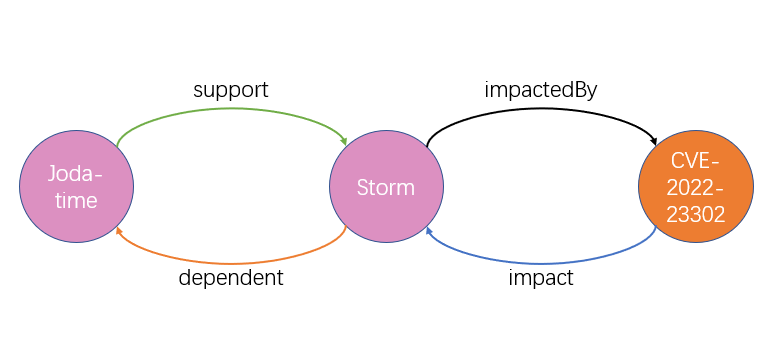


图3.6 实体关系图（部分）

在存储的数据信息中，Repository项目类型属性包括项目id、url、名称、拥有者、star数、fork数、描述信息、创建时间及更改时间等属性；Vulnerability漏洞类型包括漏洞CVE编号、描述信息、发布时间、最后修改时间、风险等级、url地址以及影响版本等属性。两者之间的关系为单向关系，方便进行单向查询。通过Cypher的Match、Merge语言将节点与关系导入图数据库后，所构建的Github软件分析知识图谱如图3.7所示。从构建的知识图谱中看，能清晰地判断项目与漏洞节点，以及项目与项目之间和项目与漏洞之间的关系。开发者可凭此了解到githuub热门Java项目之间的依赖关系，以及漏洞影响的范围。此外，对于安全团队来说，漏洞风险的评估属性也可以帮助他们发现并首先处理高危漏洞，降低高危漏洞可能带来的巨大影响。同时，得到的漏洞影响项目数量信息也可以帮助其优先解决那些存在于广大项目中的漏洞，降低因为依赖关系所可能导致的大范围项目安全风险。

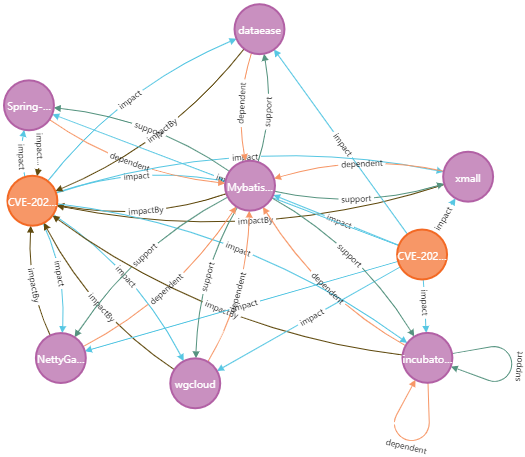


图3.7 Github软件分析知识图谱（部分）

以“漏洞”为查询条件，使用Match进行节点与关系的可视化展示，可以得到每个漏洞所影响的项目范围图。以CVE编号为“CVE-2022-23302”的漏洞为例，使用Cpyther语言查询漏洞节点和与之相关联的项目，所用查询语言为：MATCH p=(n:vulnerability{name: “name: “CVE-2022-23302””}) --() RETURN p。最终呈现结果如图3.8所示，可以看到橙色节点Vulnerability即为指定的CVE编号为“CVE-2022-23302”的漏洞节点，与之相关联的紫色节点即为与之相关的Repository类型的项目节点。通过该节点的属性展示，可以清楚的观察到其描述信息、发布时间以及最后修改时间等相关信息。通过Neo4j自带的数量统计，发现该漏洞影响了237个项目节点。其中，“impact”与“impactedBy”关系边各有237条，“support”与“supported”关系边各有837条，数量惊人。值得一提的是，该项目漏洞风险等级为高危。高风险的漏洞却存在于数量如此巨大的项目中，足以可见解决开源软件安全问题刻不容缓。

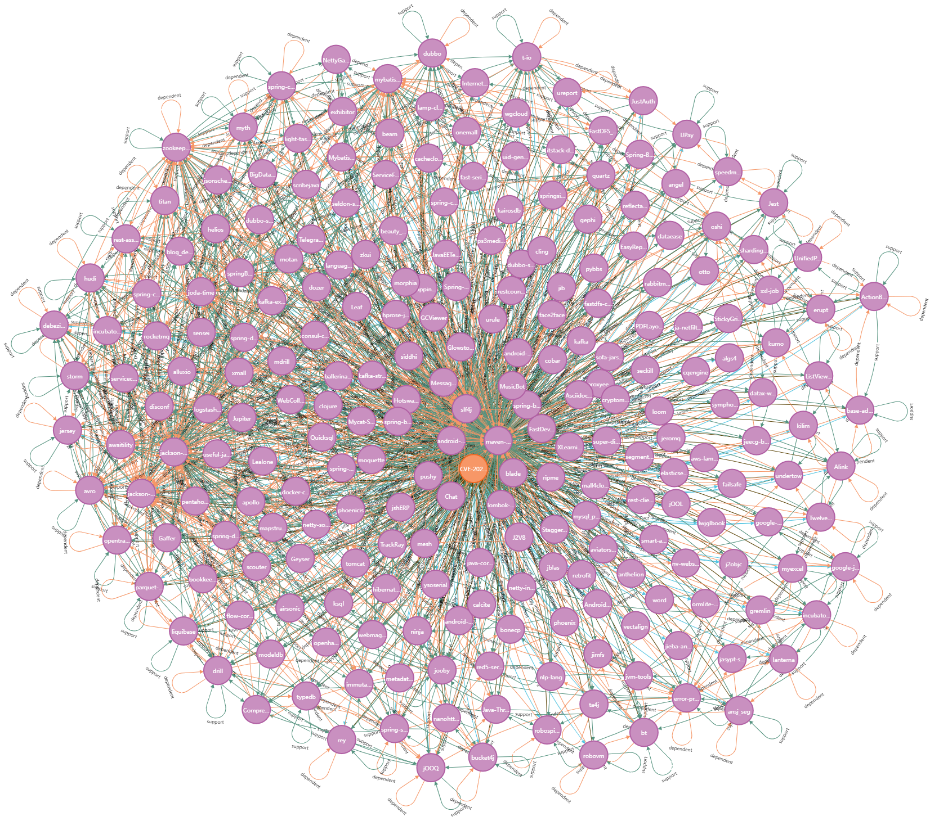


图3.8 漏洞关系图

## 3.3 相关项目推荐

在获取到的海量数据信息中，人们常常难以选择其中的有用信息，导致信息过载问题非常严重。推荐系统作为一种能够缓解信息过载的关键性工具，通过对庞大数据的学习，能够从大量的信息中提取出优质的可用信息，高效地为用户筛选过滤数据，已经应用到购物、社交、工业等各个领域之中，成为近年来的热门研究方向。例如，在社交网络中的好友推荐功能、购物软件中的猜你喜欢功能、视频音乐软件的每日推荐功能以及各类阅读、求职、就医、工农业所覆盖的日常生活和生产实践中，推荐系统都发挥着它独有的功能。

对于该文来说，Github软件生态知识图谱已经成功构建，项目与依赖之间的关系已经得到了清晰地展现，同时项目信息与漏洞信息也能够通过Cypher语句进行有效地获取与展现，但是对于如此庞大的数据来说，用户想要在其中得到与当前项目相关的项目是非常困难的。于是，推荐算法的使用解决了这个问题。通过推荐算法的应用，用户能够清晰地得到与当前项目相关性最高的项目信息，从而快速进行下一步的数据分析。

### 3.3.1 知识图谱推荐算法

基于知识图谱的推荐算法有三类，分别是基于路径、基于嵌入、基于混合的推荐算法[13]。基于路径的推荐算法将知识图谱中的节点联系做到了充分利用，依靠节点间的联系实现预测和推荐；基于嵌入的推荐算法需要对知识图谱中的节点与关系进行从高维度到低维度的映射，降低知识图谱的高维性，并通过图嵌入算法对节点及其关系进行表征；基于混合的推荐算法将路径和嵌入的思想进行结合，首先通过连接的思想提取出节点之间的联系，再通过嵌入的思想进行低维特征表示，相比于前两者的算法来说，其表现更为突出。

对于基于路径的算法来说，需要手动设计原路径，且应用范围有限；对于基于混合的算法来说，其模型较为复杂，计算量大。故最终选择了应用最广泛的基于嵌入的推荐算法，虽然无法挖掘实体间的高阶语义，但对于该文所涉及到的功能实现来说，并不要求对关系进行深度挖掘。此外，基于嵌入的推荐算法很好地降低了图的维度，提高了知识图谱的灵活性，减轻了工作量。

### 3.3.2 图嵌入算法

图嵌入[14]（Graph Embedding）算法分为两种类型：第一类为顶点嵌入，每一个节点都用向量进行编码表示，从节点维度上进行预测分析；第二类为图嵌入，每张图用一个向量进行编码表示，从图的维度上进行预测分析。对于该文来说，所要实现的功能为节点之间的相关推荐，因此使用顶点嵌入方法。该方法的核心在于通过降低维度来抽取节点邻域的信息，将其转换为节点嵌入（Node Embedding）。然后将节点嵌入应用到后续的机器学习系统中以完成推荐任务。当前主流图嵌入算法有三种不同的思想，分别是降维思想、序列思想以及动态思想。

降维思想是指对节点的表示进行降维，将节点属性降到一个更小维度的向量中进行表示。例如我们知道，图由节点和边组成，使用邻接矩阵的数据结构便能轻松表示。矩阵的每一行和每一列表示各个节点，矩阵中的值用来表示两个节点之间是否相连。若有100个节点，则需要100\*100的矩阵进行表示。对于节点数目较少的图来说，是完全可以使用邻接矩阵的表示方法，但一旦节点数目增加一个数量级，那么用该方法表示大型图的特征是完全没有必要也是难以实现的。此时，降维思想在存储节点信息的同时，起到了节省存储空间的作用。

序列思想是将相互连接着的各个相邻节点看做序列，类似于文字表达。得到这些序列常常使用随机游走（Random Walk）的方法。随机游走是指在已知图和起始节点的情况下，随机选择一个新的节点（走到各个邻居节点的概率相同），以该节点为当前节点，并重复进行以上操作。使用随机游走产生的从初始节点到最终节点的概率，即可看做两个节点的相似度。

动态思想考虑了新节点的加入所造成的改变。上述的降维思想和序列思想将各个节点的Embedding值看做固定的值，但当有新的节点加入图中时，会产生新的边，即新的关系，而这些关系的终点与旧的节点相连。此时，旧节点的Embedding值理所应当应该发生变化。故动态思想考虑了这一点，根据节点的增删变化情况做出了相应的处理。

### 3.3.3 图嵌入算法比较

Neo4j提供了三种图嵌入的方法：FastRP（Fast Random Projection，随机投影）算法、Node2Vec（Node to Vector，节点向量）算法以及GraphSAGE（Graph Sample and Aggregate，图聚合）算法分别对应了降维思想、序列思想、动态思想三种思想。三种图嵌入算法的的比较如表3.1所示。

表3.1 图嵌入算法比较

|  |  |  |  |
| --- | --- | --- | --- |
|  | **FastRP** | **Node2Vec** | **GraphSAGE** |
| 思想 | 降维思想 | 序列思想 | 动态思想 |
| 学习方法 | 线性代数 | 神经元网络 | 神经元网络 |
| 特征 | 可扩展、稀疏随机投影 | 对节点同质性和结构性可侧重选择生成Embedding | 动态预测节点Embedding |
| 优点 | 速度快、并行化 | 采样高效、可处理两类相似性 | 可应用于大规模的图结构数据中 |
| 缺点 | 超参数调优需大量实验 | 无法考虑边的权重 | 计算量大 |

FastRP[15]算法的核心在于不像大多数嵌入算法将任务分为构造节点相似矩阵和对矩阵降维两个部分，而是将节点相似矩阵和稀疏随机投影结合起来，其优点在于允许节点嵌入的迭代运算，不需要重复地构建相似性矩阵，使得FastRP的速度达到质的提升。其中，随机投影起源于高维向量的欧式距离计算，假如当前两个10000维的文本进行相似度计算，而计算维度上限为100，那么构造的随机投影矩阵为10000\*100，因为1 \* 10000 dot 10000 \* 100 = 1 \* 100。

Node2Vec[16]算法的特征在于将图中的节点相似性形态归纳为两类：第一类为同质性，即相邻节点之间的相似性；第二类为结构性，即图中拥有相似结构的节点之间的相似性。为了体现出图中节点的同质性，随机游走偏向于使用DFS（Depth First Search，深度优先搜索），因为DFS能够跳出局部，获得更大感受野，找出每一部分节点簇的边界，使得该节点与簇内节点的联系比簇外节点更加紧密；为了体现出图中节点的结构性，随机游走偏向于使用BFS（Breadth First Search，广度优先搜索），因为BFS更能够学习微观上的局部特征并感知节点所处的局部结构，使得相似功能的节点之间的联系更为紧密。也就是说，DFS所得到的效果是相邻节点的Embedding更接近，而BFS所得到的效果是相似结构节点的Embedding更相近。具体算法中可更改相关参数，用于控制采用DFS和BFS的比例值。

GraphSAGE[17]算法的关键在于它不需要为每一个节点进行独立的 Embedding训练，而是通过学习一个对邻近结点聚合表示的函数，从而生成某个中心结点的特性表示，是一种能利用节点属性信息产生未知节点Embedding的归纳式算法。也就是说，每个节点的Embedding是根据其邻居关系的变化而变化，且对图进行采样以学习可以预测嵌入的函数而不是直接嵌入，即不需要在图中节点或关系有变化后对图进行全面重新学习。其中，聚合函数包括Mean Aggregator、GCN Aggregator、Pooling Aggregator、LSTM Aggregator等。具体算法为：首先对图中每个节点的邻居节点进行采样，接着使用聚合函数聚合邻居节点信息，最后各节点产生Embedding映射函数并得到各节点的向量表示。

从以上分析可知，FastRP算法以及GraphSAGE算法本质上期待得到的结果都是距离相近的两个节点尽可能的相似，也就是Node2Vec中提到的同质性相似形态。该文所要实现的是相关项目推荐功能，而我们知道，两个相邻项目虽然之间的“距离”相近，但或许只是用到了相同的插件或相同的HTML代码文件，两者所要实现的功能效果可能完全不相关。所以这时候如果期待距离相近的节点尽可能相似是完全没有意义的。也就是说，实现该相关项目推荐功能并不注重节点同质性，而是更加注意节点的结构性。这也是为什么虽然FastRP的速度更快、GraphSAGE能动态预测节点却最终没有成为选择的原因。

### 3.3.4 KNN算法

至此，我们已经充分利用了图的拓扑信息，得到了所有项目节点的Embedding信息，接下来需要得出与中心节点相关度最高的项目节点。KNN（K-Nearest Neighbors，K个最近的邻居）是机器学习算法中最经典的算法之一，可通过所提供的不同特征值之间的距离进行分类任务。在本文任务中，不用对于其进行分类，只需找出与当前项目节点距离最接近的前五个项目节点。

KNN算法的具体实现使用sklearn.neighbors模块，将n\_neighbors参数设为6，表示查询6个最近邻的节点，其中的一个节点为当前项目节点本身。将用过Node2Vec图嵌入得到的每个节点的Embedding信息作为fit函数的输入来训练算法，最终结果得到每个节点最近的5个节点以及相应的距离指数，使用Cpyther语言将其添加到节点的属性中。

# 4 Github软件生态分析系统

至上一章已经成功构建了Github软件生态知识图谱，在Neo4j图数据库将数据进行了清晰的展示。此外，采取了Node2Vec图嵌入算法，实现了相关项目推荐功能。本章将基于上述所得，实现Github软件生态展示系统，采用Vue.js和D3.js框架进行可视化开发。

## 4.1 系统功能设计

当前开源软件安全问题亟待解决，而软件项目之间的复杂且多样依赖关系，大大增加了漏洞的影响范围。构建Github软件生态展示系统，其需求分析主要是针对Github中的开源项目信息和依赖关系，设计出一款方便查看和查询的系统。目的是帮助用户和开发者准确了解项目信息并对项目之间的关系做到准确定位，并以结构化的方式进行知识图谱的展示。

该软件生态分析系统功能设计主要分为四个部分，第一部分为登录界面，用户通过账户密码进行登录；第二部分为项目信息具体展示界面，在本界面中能观察到所选项目的信息，包括依赖信息、相关项目信息，帮助用户实现对该项目的全面了解。同时，采用动态图表及动画的形式，增加系统的可观赏性；第三部分为整体3D展示界面，在本界面中能观察到该知识图谱的整体情况，将其通过3D方式展现出来，实现对整个图数据库信息的了解与把控；第四部分为知识展示查询界面，用户可以输入不同的Cypher语言进行查询，提交后能够在页面中显示出相关信息，提升系统的可交互性。具体功能设计如图4.1。

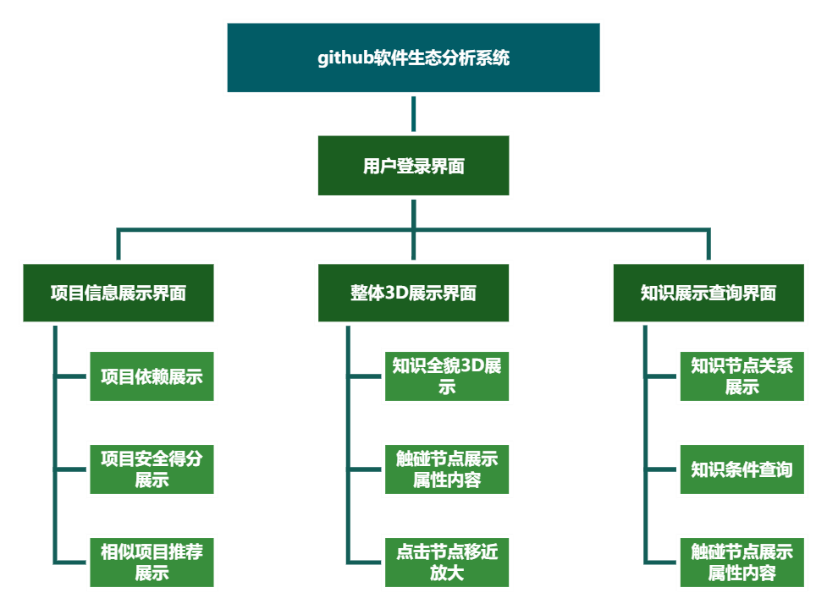


图4.1 系统功能图

## 4.2 系统实现

### 4.2.1 登录界面

Github软件生态展示系统的首界面即为用户登录界面，通过输入用户名和密码或重新注册即可成功登录。用户登录界面如图4.2所示。

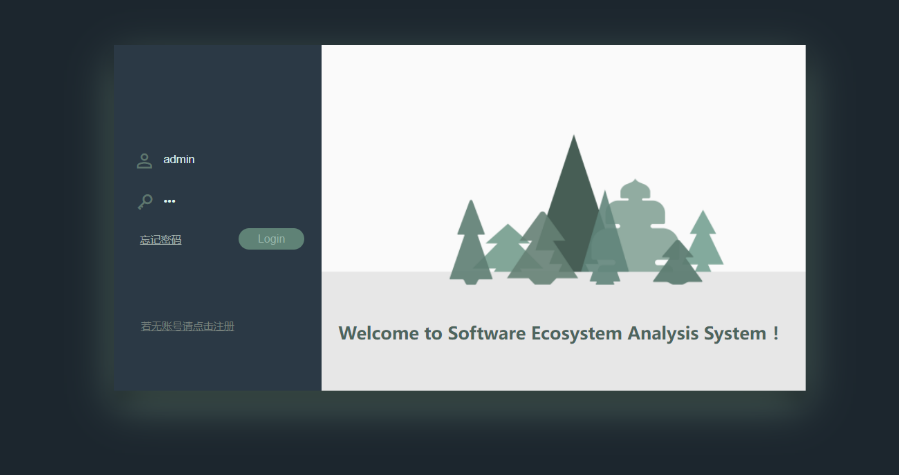


图4.2 系统登录界面

### 4.2.2 项目信息展示界面

该界面用于展示部分项目的相关信息，帮助用户及开发者更全面的了解该项目。所需数据信息使用Cypher语言查询Neo4j图数据库得到，包括项目依赖信息以及相关项目信息。

项目信息展示界面如图4.3所示。用户选择相应的项目文件，界面中心位置便能呈现出相应的项目依赖文件信息，包括依赖名、依赖的url地址、漏洞信息、更新时间和主要使用语言。点击某行依赖，下方会展示该依赖的详细介绍信息。右侧上方为D3.js所绘制的软件得分信息SVG图，该得分通过项目依赖漏洞数所占项目依赖比例得出，并能够在项目切换时实现动态增减。右侧下方同样是通过D3.js所绘制的相关项目推荐SVG图，在项目切换时同样能进行动态绘制，可观赏性强。此外，点击上方导航栏，便能跳转到整体3D展示界面以及知识展示和查询界面。

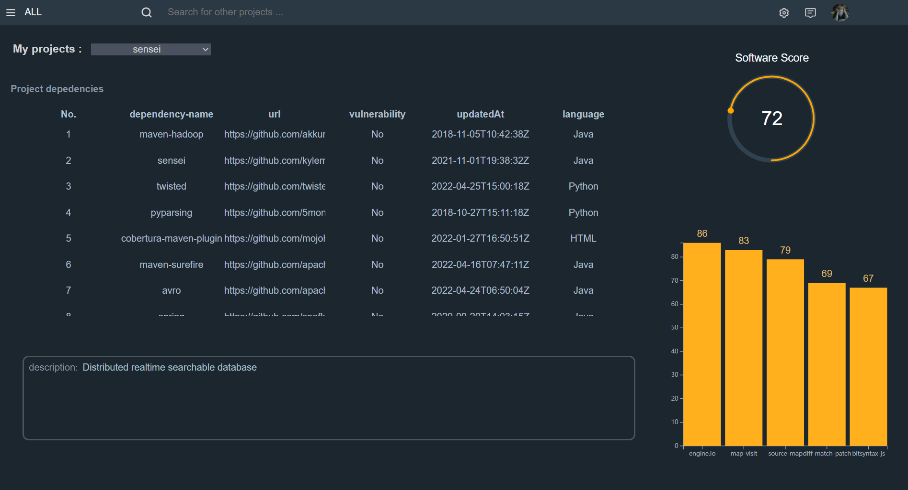


图4.3 项目信息展示界面

### 4.2.3 整体3D展示界面

该界面用于对Neo4j中的节点以及其关系进行3D展示，为用户展现该知识图谱所包含的节点与关系，使用D3.js提供的3d-force-graph（力导图）模块编写，观赏效果强。

在Vue中声明所需要的Neo4j图数据库的配置信息，便能从其中读取所有数据，包括节点和关系信息以及所有属性和标签。对返回的数据信息进行存储及渲染后，配置节点和边，设置基本样式。同时，对该图加入动态操作配置，包括放大、缩小、旋转等动态操作，帮助用户更好地查看知识。用户可以通过放大、缩小操作，改变该3D关系图的远近以及方向位置。点击某个节点便能将其以动画的形式移动到最前方。此外，将光标移至节点或关系上，能够观察到节点以及关系的属性信息。

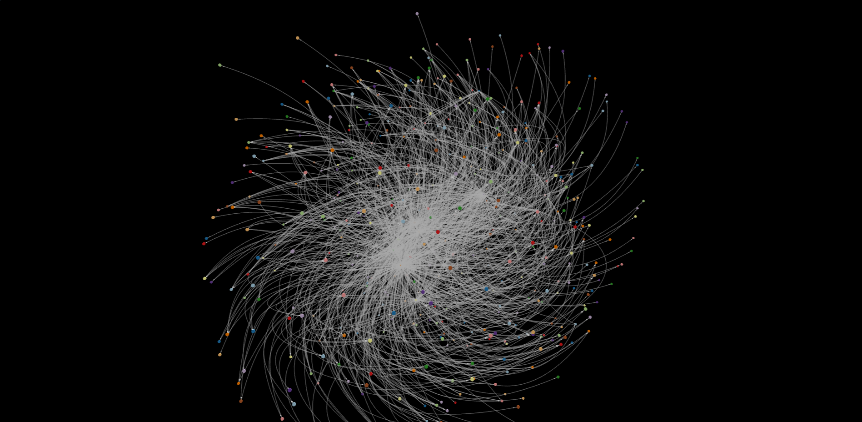


图4.4 整体3D展示界面图

### 4.2.4 知识展示查询界面

展现Github软件生态知识图谱是系统中的一项主要功能，该界面可以将Neo4j图数据库中的节点以及关系进行可视化展示。在下方的输入框中输入相应的Cypher查询语句，便能连接所提供的Neo4j数据库并将所得的结果通过SVG图的效果展现出来，默认初始查询语句为"MATCH P=(n)--() RETURN P LIMIT 100"。

界面使用基于D3库的Neovis.js库进行可视化编写，具体展示效果如图4.5所示。Neovis.js在D3.js的基础上，将JavaScript与Neo4j数据库紧密连接起来，且该框架基于Neo4j属性图构建，故两者数据模式匹配一致，使用简洁明了的语句即可进行知识图谱的相关信息展示。同时，能够进行缩小、放大、移动节点等操作。此外，同3D展示界面一样，将鼠标光标移至节点或关系处，可以观察到其属性信息，如图4.6所示。

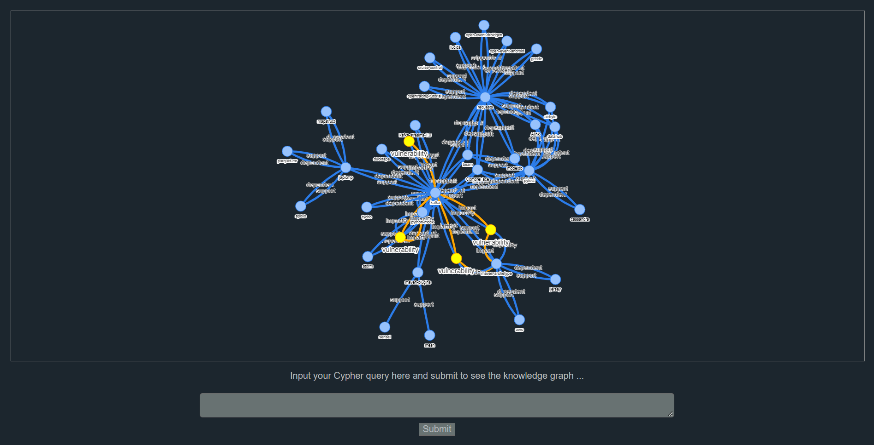


图4.5 知识展示查询界面

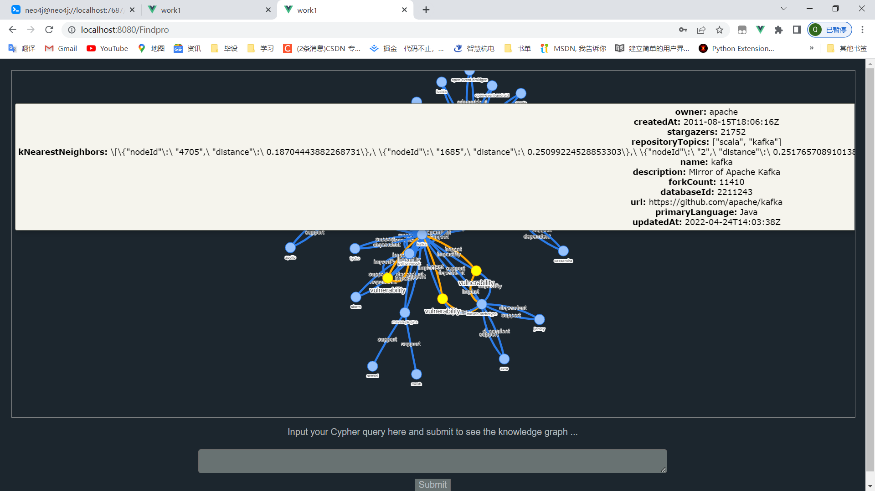


图4.6 节点属性展示图

## 4.3 系统测试

进过上述操作后，系统构建已经基本完成，但在软件工程开发周期中，最终的测试阶段是最基础也最重要的阶段。在系统运行期间，往往会发生与预想不同的情况，而测试的目的就是以最小的代价，找出系统中尽可能多的错误，尽量减小在系统正式上线后出现错误而造成的损失。测试一般主要包括健壮性测试和功能测试。健壮性测试则为检验该软件系统是否能在特殊情况下做到正确运行并且做到能够正确按照预期运行；功能测试即对该软件系统所实现的各项功能进行检验。

### 4.3.1系统健壮性测试

该系统基于Web端搭建，故判断系统健壮性最主要的就是测试该系统在各个浏览器中能否做到正确运行。本文采取了市面上流通最广的浏览器，包括Chrome浏览器、Firefox浏览器、Microsoft Edge浏览器等。表4.1即为系统健壮性测试结果。

表4.1 系统健壮性测试结果

|  |  |  |
| --- | --- | --- |
| **浏览器名称** | **网页运行情况** | **是否兼容** |
| Chrome | 正确运行、页面布局正确 | 是 |
| Firefox | 正确运行、页面布局正确 | 是 |
| Microsoft Edge | 正确运行、页面布局正确 | 是 |
| Internet Explore | 正确运行、页面布局正确 | 是 |

### 4.3.2系统功能测试

系统功能测试又称黑盒测试，在测试过程中，将程序看做无法看清内部结构的黑盒子，着眼于程序外部结构，对系统界面的各项功能进行一一测试，判断各个部分能否做到顺利运行、各个功能能否正常使用。本文将系统的各个界面、各个组件所涉及的功能进行测试，表4.2即为可视化系统功能测试用例。

表4.2 系统功能测试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试名称** | **Github软件生态知识图谱可视化系统测试用例** | | | |
| **系统页面** | **功能类型** | **具体操作** | **预期效果** | **实际效果** |
| 登录界面 | 系统登录 | 用户进行注册登录 | 输入账号密码成功注册登陆 | 输入账号密码成功注册登陆 |
| 项目信息展示界面 | 项目依赖展示 | 单击选择项目名称切换 | 项目依赖及描述信息正确展示 | 依赖及描述信息正确展示 |
|  | 项目得分展示 | 单击选择项目名称切换 | 项目得分正确切换 | 得分信息图动态切换绘制 |
|  | 相关项目推荐展示 | 单击选择项目名称切换 | 相关项目信息正确切换 | 相关项目图动态切换绘制 |
| 整体3D展示界面 | 界面变换 | 鼠标操作放大、缩小界面 | 3D效果图正确放大、缩小 | 效果图正确动态变化 |
|  | 项目属性展示 | 光标移至节点或关系处 | 节点或关系属性成功展示 | 属性正确展示 |
|  | 特定节点展示 | 点击某个节点 | 节点动态移至最前方 | 节点成功移至最前方 |
| 知识展示查询界面 | 项目属性展示 | 光标移至节点或关系处 | 节点或关系属性成功展示 | 属性正确展示 |
|  | Cypher语言查询 | 输入Cypher语言并提交 | 节点及关系指示图正确展示 | 知识正确展示 |

# 5 结语

## 5.1 工作总结

如今，互联网已覆盖生产生活的方方面面，而开源软件在其中起到了重要的作用。在如此环境下，开源软件的安全问题无疑是今后关注的一大重点。因此，本研究所提出的Github软件生态分析是一个值得研究的课题。本研究也通过构建Github软件生态分析系统，为开源软件使用者提供一定的参考，帮助开发人员提升其项目软件的安全性。本文具体工作情况如下：

1）对涉及到的背景和理论知识进行了归纳总结，包括开源软件背景和安全问题、知识图谱构建过程、相关网络爬虫技术、Neo4j图数据库发展与使用以及D3.js框架特性。

2）详细分析构建基于Neo4j的Github软件生态知识图谱的过程，包括所用到的项目信息来源分析、关键文件及项目依赖和漏洞获取，以及信息存储和Neo4j图数据库的数据导入并介绍Cypher语言的可视化查询，将项目与项目、项目与依赖之间的关系通过Neo4j图数据库进行可视化呈现。

3）详细介绍了相关度推荐的三个算法并进行比较，此外借助Node2Vec和KNN算法实现了相关项目推荐功能。

4）实现了软件生态知识图谱的系统搭建，使用Vue.js及D3.js框架进行展示，帮助用户了解该生态的项目具体信息及依赖关系，构建基于Neo4j的Github软件生态分析知识图谱的可视化查询平台。

本文的优势有两点：第一点在于采用无监督的方式，对项目信息及依赖树进行提取，将开源项目所依赖的包进行展示，并帮助开发者了解到其不安全的依赖，减少漏洞攻击窗口期；第二点在于实现了基于图嵌入的Node2Vec算法，分析了项目节点之间的结构性联系，实现了相关项目推荐功能，帮助开发者进行项目的优化与提升进步。

通过以上工作，我熟练掌握了网络爬虫的使用和知识图谱的构建过程，对于图算法方面也有了进一步的了解，并且能够灵活使用Neo4j图数据库，对前端搭建也有了更深的体会。

## 5.2 展望

虽然本研究设计并实施了基于Neo4j的Github软件生态分析系统，很好地帮助开发人员发现了项目依赖上的漏洞问题。但由于个人的时间和精力以及在开发上的不足，且对项目整体的把控不强，在许多地方有待完善和深入的探索。以下是本人发现的具体可改进之处：

1）在信息提取阶段，数据来源只针对于Github中的热门Java项目，而开源项目中，存在着大量C、C++、Python等其他语言项目，没有实现多种语言项目的覆盖，故还可以对其他语言项目进行深入钻研。

2）对于灵活多变的软件生态来说，该研究没有增加修改项目的功能，如果要增加或修改，则必须要重新进行数据提取、依赖提取、导入数据库等多项操作，灵活性不强，故在便捷添加新项目的功能方面还有很大的提升空间。

3）尽管研究对项目依赖及其所包含的漏洞进行了提取，给了开发者漏洞标号，能够帮助其进行后续的改进，但是并没有给出具体的改进方案，故还可以对产生漏洞的原因进行分析，帮助开发者更快地做出修改。

4）对于所实现的系统来说，前端与数据库相连接需将数据库的地址、用户名、密码显式地写在前端页面中，不够安全。同时，系统功能较为单一，且形式不够多样，与用户之间的交互性不强，故在功能设计与样式美化方面还可以再进一步地提升。

# 致谢

四年光阴如梭，走过千遍樱花盛开的小道，看过图书馆西边少有的粉红夕阳，匆匆赶往教室上早八的身影恍若昨日，我的大学生涯却已经到了尾声，即将离开杭电迈向我新的征程。在这四年中，我加入了大大小小的社团，参与了各类竞赛项目，组织了形形色色的班级活动，点滴成长，离不开尽责的老师和身边的同学们。

首先，我要感谢我的导师秦飞巍教授。一直以来，不管是在科研上，还是生活上，都给了我很大的鼓励与支持。秦老师学术严谨、循循善诱，帮助我以积极乐观的心态来面对学术道路上遇到的困难和挑战。正是因为秦老师的指导，让我在本科阶段不断成长，积极参加大创、新苗、挑战杯等竞赛以及项目，领会到不一样的新东西。

其次，要感谢我的毕业设计指导老师陈溪源老师，陈老师亲切友善随和，同时对我的毕业设计提出严格要求，并帮助我梳理设计的结构与脉络，让我不断改进与提升。感谢我的室友万婉宁、林颖瑜等同学，我们在学业上相互学习，在生活中互帮互助，分享欢乐，也共担责任。虽在不同的道路上，但我们相互鼓励支持，各自为我们的理想而奋斗努力。感谢你们的陪伴和温暖。

最后，感谢父母对我的包容与关爱， 感谢他们在我最无助、最焦虑的时候做我最坚强的后盾。也要感谢在我身边坚定陪伴我的男朋友，总是在我遇到困难挑战的时候，鼓励我，支持我，帮助我勇敢自信地面对一切已知未知的恐惧。在我项目停滞不前时，在我考研进入瓶颈时，在我陷入自卑焦虑时，一直陪在我身边。

感谢我的母校——杭州电子科技大学。很开心能在这个学校度过美好的大学生活，也感谢母校提供的浓厚的科研环境，以及各式各样的讲座、招聘会以及课外活动。不是在最美好的年纪遇见了你，而是因为有你，成就了更美好的我。

# 参考文献

1. Zhang,Ou,Caragea. Predicting Cyber Risks through National Vulnerability Database[J]. Information Security Journal: A Global Perspective,2015,24(4-6).
2. 张得光,李兵,何鹏,周华昱.基于软件生态系统的开源社区特性研究[J].计算机工程,2015,41(11):106-113.
3. 杜睿山,张轶楠,田枫,王梅,李婷玉,张蕾,程有为,赵崇志.基于知识图谱的智能问答系统研究[J].计算机技术与发展,2021,31(11):189-194.
4. 王昊奋,漆桂林,陈华钧.《知识图谱：方法、实践与应用》[J].自动化博览,2020,37(01):7.
5. Yang Xu,Huan Ziyi,Zhai Yisong,Lin Ting. Research of Personalized Recommendation Technology Based on Knowledge Graphs[J]. Applied Sciences,2021,11(15).
6. 戴启铭,毛润丰,黄璜,荣国平,沈海峰,邵栋.DevSecOps:DevOps下实现持续安全的实践探索[J].软件学报,2021,32(10):3014-3035.
7. 贾培养,孙鸿宇,曹婉莹,伍高飞,王文杰.开源软件漏洞库综述[J].信息安全研究,2021,7(06):566-574.
8. Chaudhri Vinay K.,Baru Chaitanya,Chittar Naren,Dong Xin Luna,Genesereth Michael,Hendler James,Kalyanpur Aditya,Lenat Douglas B.,Sequeda Juan,Vrandečić Denny,Wang Kuansan. Knowledge graphs: Introduction, history, and perspectives[J]. AI Magazine,2022,43(1).
9. Liang Zongwei,Yang Junan,Liu Hui,Huang Keju,Cui Lin,Qu Lingzhi,Li Xiang. HRER: A New Bottom-Up Rule Learning for Knowledge Graph Completion[J]. Electronics,2022,11(6).
10. 刘巍,陈霄,陈静,周觐,张斌.知识图谱技术研究[J].指挥控制与仿真,2021,43(06):6-13.
11. Bouamrane Karim,Matallah Houcine,Belalem Ghalem. Comparative Study Between the MySQL Relational Database and the MongoDB NoSQL Database[J]. International Journal of Software Science and Computational Intelligence (IJSSCI),2021,13(3).
12. Jeang-Kuo Chen,Wei-Zhe Lee. An Introduction of NoSQL Databases Based on Their Categories and Application Industries[J]. Algorithms,2019,12(5).
13. 朱冬亮,文奕,万子琛.基于知识图谱的推荐系统研究综述[J].数据分析与知识发现,2021,5(12):1-13.
14. 田萱,陈杭雪.推荐任务中知识图谱嵌入应用研究综述[J/OL].计算机科学与探索:1-31[2022-05-06].
15. Haochen Chen,Syed Fahad Sultan,Yingtao Tian,Muhao Chen,Steven Skiena. Fast and Accurate Network Embeddings via Very Sparse Random Projection.[J]. CoRR,2019,abs/1908.11512.
16. Grover Aditya,Leskovec Jure. node2vec: Scalable Feature Learning for Networks.[J]. KDD : proceedings. International Conference on Knowledge Discovery &amp; Data Mining,2016,2016.
17. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs[J]. Advances in neural information processing systems, 2017, 30.