软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2020,31(1):47−66 [doi: 10.13328/j.cnki.jos.005893]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

代码知识图谱构建及智能化软件开发方法研究*

王 飞[1], 刘井平[2], 刘 斌[1], 钱铁云[1], 肖仰华[2], 彭智勇[1]

[1](武汉大学 计算机学院,湖北 武汉 430072)

[2](复旦大学 计算机科学技术学院,上海 201203)

通讯作者: 钱铁云, E-mail: qty@whu.edu.cn; 肖仰华, E-mail: shawyh@fudan.edu.cn; 彭智勇, E-mail: peng@whu.edu.cn

摘 要: 智能化软件开发正在经历从简单的代码检索到语义赋能的代码自动生成的转变,传统的语义表达方式无法有效地支撑人、机器和代码之间的语义交互,探索机器可理解的语义表达机制迫在眉睫.首先指出了代码知识图谱是实现智能化软件开发的基础,进而分析了大数据时代智能化软件开发的新特点以及基于代码知识图谱进行智能化软件开发的新挑战;随后回顾了智能化软件开发和代码知识图谱的研究现状,指出了现有智能化软件开发的研究仍然处于较低水平,而现有知识图谱的研究主要面向开放领域知识图谱,无法直接应用于代码领域知识图谱.因此,从代码知识图谱的建模与表示、构建与精化、存储与演化管理、查询语义理解以及智能化应用这 5 个方面详细探讨了研究新趋势,以更好地满足基于代码知识图谱进行智能化软件开发的需要.

关键词: 智能化软件开发;知识图谱;代码大数据

中图法分类号: TP311

中文引用格式: 王飞,刘井平,刘斌,钱铁云,肖仰华,彭智勇.代码知识图谱构建及智能化软件开发方法研究.软件学报,2020,31(1):47−66. http://www.jos.org.cn/1000-9825/5893.htm

英文引用格式: Wang F, Liu JP, Liu B, Qian TY, Xiao YH, Peng ZY. Survey on construction of code knowledge graph and intelligent software development. Ruan Jian Xue Bao/Journal of Software, 2020,31(1):47−66 (in Chinese). http://www.jos.org.cn/1000-9825/5893.htm

Survey on Construction of Code Knowledge Graph and Intelligent Software Development

WANG Fei[1], LIU Jing-Ping[2], LIU Bin[1], QIAN Tie-Yun[1], XIAO Yang-Hua[2], PENG Zhi-Yong[1]

[1](School of Computer Science, Wuhan University, Wuhan 430072, China)

[2](School of Computer Science, Fudan University, Shanghai 201203, China)

Abstract: The intelligent software development is migrating from simple code retrieval to semantic empowered automatic code generation. Traditional semantic representation cannot effectively support the semantic interaction among people, machines, and code. It becomes an urgent task to design a set of machine-readable semantic representation. In tThis paper, westudy firstly points out that code knowledge graph forms the basis to realize the intelligent software development, and then analyzes the new features and new challenges of intelligent software development based on code knowledge graph in the era of big data. Next, we review the research progress is reviewed both in intelligent software development and in code knowledge graph. It is noted that the current research of intelligent software development is still at a preliminary stage. Existing studies of knowledge graph mainly focus on open-domain knowledge graph, and they cannot be directly applied to code and software development domain. Therefore, we discuss the new research trends of code knowledge graph are discussed in detail from five aspects, including namely modeling and representation, construction and refinement,

* 基金项目: 国家重点研发计划(2018YFB1003400); 国家自然科学基金(61572376); 中央高校基本科研业务费专项资金(2042019k10278)

Foundation item: National Key Research and Development Program of China (2018YFB1003400); National Natural Science Foundation of China (61572376); Fundamental Research Funds for the Central Universities (2042019k10278)

收稿时间: 2019-01-14; 修改时间: 2019-06-24; 采用时间: 2019-09-16; jos 在线出版时间: 2019-11-06

CNKI 网络优先出版: 2019-11-06 11:49:26, http://kns.cnki.net/kcms/detail/11.2560.TP.20191106.1149.012.html

storage and evolution management, semantic understanding, and intelligent application, which are essential to meet the various types of demands of the intelligent software development.

Key words: intelligent software development; knowledge graph; big code

基于知识图谱的智能化软件开发是进一步推动软件行业发展的重大历史机遇.

一. 软件产业日益成为社会经济发展的战略性基础产业.进入 21 世纪以来,信息技术已成为推动国民经济发展、促进社会生产力提升的强大动力,作为与国民经济和社会发展紧密相关的基础战略型先导产业,信息产业受到了越来越多国家和地区的重视.得益于政策法规的鼓励和支持,我国软件行业在“十一五”期间保持快速增长态势,年均收入平均增速高达 30%以上.根据工信部发布的《2017 年软件业经济运行情况》报告显示,2017 年度中国软件行业共实现业务收入 5.5 万亿,比上年增长 13.9%,成为经济转型和产业升级的支柱[1];

二. 软件行业发展仍然面临诸多严峻挑战.尽管软件产业正处于蓬勃发展时期,但总体来看,我国软件企业竞争力仍然较弱.根据普华永道 2016 年度的“全球软件百强企业”报告,全球软件重点企业美国占比为 75%,而中国仅有一家企业入围.开发成本高昂、生产效率低下、移植运维困难等因素,成为制约我国软件行业深入发展的主要瓶颈[2];

三. 基于知识图谱的认知智能是突破软件产业发展瓶颈的重大机遇.2012 年谷歌推出知识图谱以来,知识图谱技术发展迅速,产生了日益广泛的社会、经济效益,成为发展人工智能战略的重要内容.知识图谱是一种大规模语义网络,表达了实体/概念及其之间的各种语义关系.知识图谱为机器语言认知提供了丰富的背景知识,使得机器语言认知成为可能[3],进而使得文本自动化处理、智慧搜索、精准推荐、自然人机交互、深度解释等一系列智能化应用成为可能.知识图谱是认知智能的核心,是软件行业智能化转型的使能器(enabler);

四. 软件行业大数据使得知识图谱的构建与应用成为可能.代码大数据为知识图谱的构建提供了数据来源,而基于大数据的算法(如深度神经网络)为知识图谱的自动构建提供了辅助.目前,全球存在许多开源且高度结构化的平台,例如 GitHub.这些开源平台为大规模代码知识图谱的构建提供了数据源.截止 2018 年,GitHub 已经集聚了全球 3 100 万开发者、210 万组织以及 9 600 万代码仓库,且每个仓库都包含高度结构化的信息,如编程语言、代码库名称、代码库描述等[4].

目前,知识图谱已在金融投资、风险控制、科学教育等诸多行业得到了较为广泛的应用.但在软件行业,代码知识图谱的应用还不多见.但是,知识图谱是认知智能的关键技术,展现出了解决智能化软件开发过程中各种瓶颈问题的巨大潜力.

代码知识图谱将成为智能化软件开发的利器:首先,知识图谱能够为理解用户意图提供强大的背景知识,支持知识和事实的逻辑推理,有助于实现语义层面的搜索泛化;其次,基于知识图谱的代码推荐具有较强的可解释性,帮助用户理解推荐结果,支持推荐规则和流程的优化;最后,知识图谱的引入能够为代码生成提供模板,通过融合模板和深度生成模型,能够为用户生成满足意图的代码.

例 1:以用户查询“如何将 tex 文件格式转换成 pdf 文件格式”为例,说明代码知识图谱在代码搜索和推荐中的优势.基于对查询短文本的语义理解和分析,如果现有的知识图谱中有一个实体名为 tex2pdf converter(输入为 tex 文件格式,输出为 pdf 文件格式),则系统将该实体对应的软件实例返回给用户.如果查询不到满足要求的代码实体,但是已有的知识图谱中存在如图 1 所示的其他相关代码实体.通过查询 tex 和 pdf 实体,以及两个实体之间的路径,可以发现 tex→tex2dvi converter→dvi2ps converter→ps2pdf converter→pdf 的路径,从而可以同时将这 3 个 converter 实体及对应的实例推荐给用户.进而,根据已有 3 个 converter 实体的 isA 关系,可以获得具体的软件包名和具体用法,作为辅助说明信息一并返回给用户.

传统的软件开发主要以人工编程为主,简单的代码自动构造为辅.传统的代码自动构造技术主要有模型驱动的软件开发[5]和基于逻辑规约[6]的程序合成这两种方式.传统的人工编程环境相对简单、代码自动纠错困难、编程接口有限,而代码自动构造技术又存在着模型设计复杂、通用性差以及可扩展性弱的缺陷.现阶段的软件

开发呈现出两个显著的新状况.

(1) 随着软件生态和网络技术的快速发展,不仅产生了大量的开源软件代码,还涌现了众多围绕开源软件服务的各种开源社区.这些开源软件和开源社区的各种资源信息构成了所谓的代码大数据;

(2) 人工智能技术的快速发展给软件智能化开发注入了全新的活力,以深度学习为代表的机器学习和知识图谱为代表的知识工程是现阶段人工智能技术发展的两个主要驱动因素.

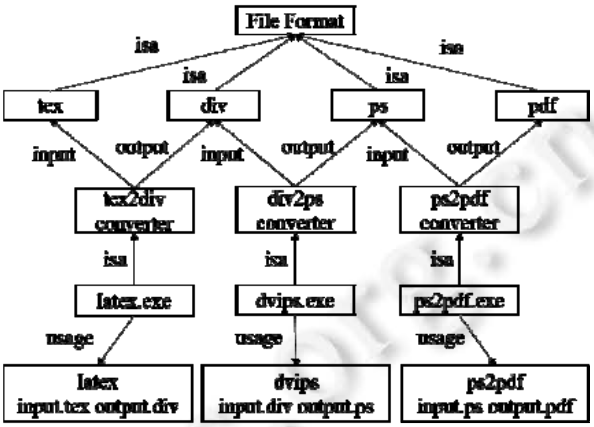


Fig.1 An example of code search based on knowledge graph

图 1 基于知识图谱的代码搜索样例

不同于传统的软件开发技术,代码大数据和人工智能技术是智能化软件开发的主要特征,贯穿于智能化软件开发的所有环节.智能化软件开发主要是指以代码大数据为基础,面向软件开发的全流程,构建代码知识图谱,训练机器学习模型,以可信而有效的方式提升软件需求、代码搜索、相似性量化、推荐、合成、缺陷检测、测试以及运维等软件开发每一个环节的智能化水平.但是受限于现有工作和分析的需求,本文将聚焦于代码搜索、相似性量化、推荐、合成等环节最新成果的脉络梳理,呈现现有基于机器学习技术的智能化软件开发的进展和瓶颈,阐明引入知识图谱技术的必要性,进而探讨代码知识图谱的构建、管理以及智能化应用前景等方面的诸多研究课题.

图 2 给出了智能化软件开发框架.

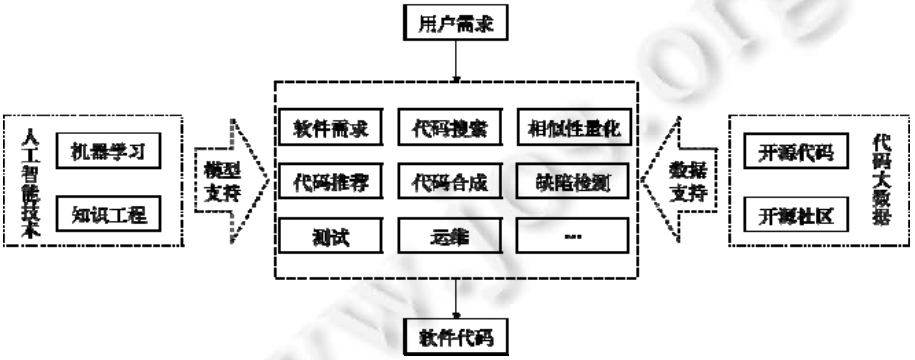


Fig.2 Intelligent software development framework

图 2 智能化软件开发框架

由于程序软件的各种资源信息存在统计上的规律性,因此,机器学习借助于代码大数据已经在智能化软件开发的诸多领域产生了广泛的研究成果.尽管基于统计特征为基础的机器学习在智能化软件开发的许多方面

是有效的,但是这种方式却存在着缺乏语义、不可解释、拟合性强和可移植性差等不足.因此,研究者尝试引入知识图谱技术来解决智能化软件开发面临的各种挑战.现阶段的知识图谱技术仅在智能化软件开发的部分子领域发挥一些辅助性作用,处于初始的起步阶段,未来极有可能和机器学习融合发展,提升智能化的软件开发的水平.

1 智能化软件开发的特点与挑战

针对软件行业服务对象及服务方式的多样性和复杂性,结合当前软件开发工具平台持续演进、团队协作不可或缺、软件复用日渐普遍的发展趋势,本节将以代码大数据在数据演变、数据管理和智能化软件开发等方面的新特点为基础,分析代码大数据在代码知识图谱建模、代码知识图谱管理和代码知识图谱应用等方面面临的全新挑战.

1.1 智能化软件开发的新特点

- 代码数据特点正在经历从单一同质到多元异质的演变.

早期的代码大多为个人或组织私有,如今的开源社区已成为代码的主要信息源,其他还包括软件网站、开发工具包等来源.代码数据形式和相互关系变得极其复杂多样:首先,数据内容和种类变得更加丰富,不仅包括源代码、二进制/中间码文件、第三方 API 文件等核心代码数据,还覆盖代码编写者、代码开发平台、代码功能、版本历史、修正记录、评价记录等附属信息;其次,数据之间的关系也变得更加多元,实体、属性不仅有实体关联网络,还会涉及实体-属性、属性-属性等复杂的关联网络;最后,数据分布不均问题日益突出,例如在 GitHub 网站上,以 javascript 编写的代码约有 2.3M,而以 C++作为编程语言的代码仅有 413K.

- 代码数据管理需要实现从静态稳定到动态演化的变更.

有效而准确的信息反馈,是智能化软件开发服务追求的重要目标.传统的代码规模相对比较小,涵盖的程序设计语言相对比较单一,涉及的内容也相对比较稳定,通常需要相当长的时间才会出现代码和文档数据的演化,继而需要进行数据的更新.但是随着软件行业规模的快速发展,代码和相关文档的规模急剧膨胀;适合同一应用场景的程序设计语言更加多元化,使得不同程序设计语言编写的具有相同或者相似功能的代码数据和相关文档呈现交叉融合的态势;软件行业规范的不断成熟,使得代码数据及相关开发文档的结构特性越来越良好,代码的更新周期也变得越来越短;开源软件的风气盛行,使得代码数据质量和相关文档的完整性呈现层次性,代码数据呈现了一个非常显著的特点,即代码数据从静态而稳定的状态向动态而不断演化的状态转变.这些全新的变化使得代码数据成为真正具有大容量、多类型、变化快、低价值的 4V 特征的代码大数据,给面向代码大数据的管理以支持更全、更及时的智能化软件服务提出了更高、更新的要求.

- 智能化软件服务需要完成从简单搜索到深度智能应用的演进.

传统的代码服务主要体现在简单的代码搜索,但是随着代码及相关文档的结构越发良好以及代码规模的急剧膨胀,给代码的深度智能化服务提供了无限的遐想空间.智能化代码服务需要提供对深度智能应用的有效支撑,主要涵盖以下几个智能应用:(1) 构建具有复杂语义表达能力的智能化搜索;(2) 基于用户行为和语义理解的个性化代码推荐;(3) 支持高度可交互的定制化代码生成.代码搜索是开发者最为频繁的行为之一,智能化软件服务需要构建具有复杂语义表达能力的语义理解机制,实现对象从单一代码扩充到专家、概念等复杂多元对象,甚至可能涉及到跨领域的联合搜索;代码推荐需要实现单纯的从基于行为建模用户需求的推荐发展到基于用户行为特征和代码功能语义相互融合来建模的个性化推荐;代码自动生成需要进一步扩展自动化生成复杂代码的能力,采用自然的方式进行人机交互来提升人机交互水平,降低软件开发需求的理解壁垒,是实现高层级代码自动生成的必由之路.

1.2 基于代码知识图谱进行软件开发的新挑战

正是由于这些新的特点,现有通用知识图谱的构建方法很难适用于代码领域,而智能化软件开发的诸多环节也需要探索知识图谱全新的应用实践.为了支持智能化软件开发全过程的深度智能应用,需要解决知识表

示、图谱管理、智能化服务这 3 个方面提出的全新挑战.

(1) 挑战 1:知识表示方面的挑战.

三列表(主语-谓语-宾语)是知识图谱事实上的表示形式,其中,主语和宾语可归结为实体.针对代码领域的知识图谱构建需要一定的深度与完备性,对准确度要求非常高,有严格与丰富的数据模式,而开源代码数据的动态、异质、倾斜分布特性,以及代码成分之间的复杂结构语义关系,决定了代码知识图谱的建模与表示学习是一项极具挑战性的工作.这种挑战性主要体现在以下两个方面.

- 一个是代码知识图谱中实体和关系的界定问题.在知识图谱建模中,文本数据中实体和关系的界定是相对清晰的,但是代码数据中实体和关系的界定就显得非常模糊,实体和关系的选择以及抽象程度都需要非常专业的背景知识;
- 另一个则是知识图谱中实体和关系的挖掘工作.文本数据中实体和关系可以通过高频度进行挖掘,而代码数据则存在明显的长尾特性,绝大部分实体出现频度低、样本少的问题,数据结构异常复杂,实体之间的关系也不同于文本数据.如何为这些代码实体习得有效表示,也是一个亟待解决的问题.

(2) 挑战 2:图谱管理方面的挑战.

代码开发是知识密集型活动,代码版本和开发平台的更新都十分常见,这种更新对知识图谱通常都是受限的、被动获知的,即原生数据的更新不会形成对知识图谱更新的通知.此外,代码的更新通常具有复杂的形态,例如函数定义模块中,参数个数的变化将带来所有调用模块的变化.如何在有限的数据获取能力条件下,搭建适应复杂数据演化的管理模式,是代码知识图谱管理面临的严峻挑战.此外,代码知识图谱的分析强调语义关联,如何了解用户对于代码知识的查询需求,并且通过知识图谱构造出富含语义的查询结果,是图谱管理需要完成的另外一个重要任务.

(3) 挑战 3:智能服务方面的挑战.

为了正确理解和判断用户的搜索及编写意图,合理地进行代码搜索的查询扩展,准确反馈以及生成用户需要的代码片段,其核心均涉及到“如何用知识图谱去理解代码语义”这一挑战性的研究.一方面,需要解决用户描述的语义解释问题,了解用户想要的核心代码结构,例如针对“draw picture python”这个搜索,如果缺乏背景知识,很难理解用户希望得到的是“以 python 语言实现的画图程序”而不是“画一张巨蟒的图”;另一方面,知识库中的代码片段也要做对应的语义映射,查找语义层面上相似的代码进行推荐或者自动生成代码.代码知识图谱中的实体大多数都是代码片段而非现实中的概念,更加抽象,因而对原始代码片段的语义解析更加困难,这也为研究带来了新的挑战.

小结:代码知识图谱是实现智能化软件开发的一条可行之路.基于代码知识图谱进行智能化软件开发的挑战主要体现在 3 个方面: (1) 异质多源的数据特点,使得进行代码知识图谱的建模和构建变得非常困难;(2) 代码数据通常是分散存储、协同编辑的,具有频繁的动态演化特点,因此,代码知识图谱及时感触数据变化并适时调整图谱数据是另一个挑战;(3) 构建用户需求表示模型和代码图谱理解模型之间的桥梁是支撑整个智能化软件服务的核心性问题.

2 智能化软件开发现状

人工开发软件程序是一个耗时费力的过程,因此,智能化软件开发引起了研究人员的高度关注.随着开源软件的快速发展,互联网上聚集了大量可以被程序员复用的代码和代码仓库.一方面,通过代码推荐和代码检索,代码片段和 API 接口等软件资源可以有效地应用于开发新项目;另一方面,这些软件资源可以用于训练代码生成模型,自动化地生成项目需要的代码片段甚至代码模块.因此,如何帮助程序员有效地搜索相关的代码片段和 API 接口、生成满足特定功能的代码模块,已成为软件工程领域的一个重要研究课题.

基于概率统计为基础的机器学习已经在智能化软件开发的需求分析、程序搜索、相似性量化、推荐、合成、缺陷检测、测试和运维等多个方面拥有广泛的研究工作并有相关文献进行综述[7,8].不同于现有的综述工作,本文选取了代码搜索、相似性量化、代码推荐以及代码合成等在智能化软件开发中发挥关键作用且依靠机

器学习实现重要技术进步的研究领域进行介绍,突出研究脉络的梳理,挖掘基于概率统计为基础的机器学习在用户需求语义和查询语义建模方面的优点和不足,为进一步讨论以知识图谱为代表的知识工程进行用户需求语义和查询语义建模支持智能化软件开发奠定讨论基础.

2.1 智能化代码搜索和相似性量化

- 代码搜索

代码搜索是实现代码复用的关键,也是智能化软件开发各项应用得以运行的基础.现在大量研究者投入到开源代码社区的建设中,涌现了许多拥有广泛用户基础的开源代码搜索引擎,例如,Ohloh、Krugle、SearchCode、NerdyData、Symbol Hound、Merobase 等.这些代码搜索引擎大多采用传统的文本检索技术,将代码作为纯文本进行索引,通过字符匹配目标代码的方式进行代码检索.现有的代码搜索根据搜索类型可以分为基于关键词搜索、基于代码结构搜索以及基于测试用例搜索.基于关键词搜索是指根据代码需求者提供的关键词从托管代码资源库中寻找匹配的代码或者代码片段的过程.尽管基于关键词搜索的准确性不高,但是 Google[9]、Baidu[10]等通用型搜索引擎和 Krugle[11]、SearchCode[12]等商业型代码搜索引擎都提供了面向关键词的代码搜索支持,是代码搜索领域应用最为广泛的一种搜索类型.基于代码结构搜索是指根据代码数据属性类型或者 API 的调用序列为匹配基础的代码搜索过程.PARSEWeb[13]是一个典型的基于代码结构的搜索类型,利用输入输出数据类型的映射关系对进行代码模块的初步检索,基于获得的代码块 API 调用序列分析代码之间的关联性,判断满足需要的代码模块.基于测试用例搜索是根据用户提供的输入输出数据的映射关系实施的一种匹配性检索过程.Stolee 等研究者[14]提出了一种基于规约求解的测试用例搜索方法,先进行代码的编码用以加速处理,再将用户提供的输入输出数据对和代码转换成约束条件,使用约束求解器进行代码的匹配.

由于代码数据在很大程度上体现了代码开发人员的个人风格,例如,变量的命名方式、循环和判断的编写方式等,所以代码数据和文本数据存在一定的差异性.此外,大部分开源社区共享的代码缺乏统一格式的注释,甚至部分代码没有注释,这就导致基于字符串匹配的搜索效果非常不理想.因此,研究者开始挖掘用户查询和开源代码之间各种其他特征和语义关系来增强搜索的效果,主要有代码语义扩展、查询语义扩展和查询语义变换这 3 种方式.Keivanloo[15]和 Chatterjee 等研究者[16]利用 API 描述文档进行代码 API 的语义注释来丰富代码语义,而 Nie 等研究者[17]则利用伪反馈技术从 Stack Overflow 的 QA 信息中抽取与用户查询相关的术语词汇进行查询语义的扩展.由于自然语言和代码之间存在天然的语义鸿沟,Lü[18]和 Rahman 等研究者[19]分别利用 API 描述文档和 Stack Overflow 的 QA 信息将自然语言的查询转换成关联的 API,基于转换的 API 实施代码的检索.随着深度学习的快速发展,神经网络技术开始应用于代码语义的深度搜索.由于基于文本相似性的代码搜索很难理解代码语义,Gu 等研究者[20]提出了一个深度神经网络模型将代码片段和自然语言描述联合嵌入高维向量空间,使得代码片段和相应描述信息具有相似的向量表示,这样就可以基于自然语言实现代码片段的深度关联检索.需要指出的是,基于知识图谱进行代码语义增强的研究将在第 3 节给出相关介绍.

- 代码相似性量化

在进行代码搜索时,需要度量查询与代码、需求与代码之间的匹配程度.现有的度量相似性研究主要用于程序代码抄袭检测,常规的检测方式分为两类:基于属性计数(attribute counting,简称 AC)和基于结构度量(structure metrics,简称 SM).Halstea[21]率先提出了基于属性计数的相似性度量技术,其核心思想是将软件代码的属性信息向量化,采用相关对比分析的方法进行衡量.由于这种方法忽视了软件代码的语义特征,导致其检测效果并不理想.目前主流的方法是基于结构的相似性度量技术,几乎所有的软件代码相似性检测都采用基于结构的相似性度量技术,或者是采用属性和结构相结合的方法.Cosma[22]提出了基于 LSA 的源代码文档潜在语义相似性检测方法,着重分析了不同 LSA 参数在代码抄袭检测中的重要程度,并以可视化的形式呈现了代码之间的相似性,而 Đurić[23]则设计和开发了剽窃检测的源代码相似性系统,通过算法和模板的个性化扩展来支持属性和结构的相似性检测.随着深度学习的发展,基于神经网络的代码表示开始应用于代码的相似性计算.Alon 等研究者[24]引入神经网络来训练代码的分布式表示,将代码片段表示为一个固定长度的代码向量,用于评估代码的相似性,主要包括两个步骤:(1) 将代码分解为抽象语法树的路径集合;(2) 设计神经网络学习每个路径和路径

集合的向量表示.需要指出的是,上述代码相似性计算方法是以统计特征为基础,没有融合代码的实际语义关系,因此,这些代码相似量化主要用于代码抄袭检测,无法准确理解代码本身的语义,更难以有效地支撑基于语义的代码搜索等高级应用.

小结:现有的代码检索和相似性量化主要引入了开源社区资源和文档描述数据等辅助信息丰富源代码和查询的语义信息,提升反馈的源代码和用户查询之间的匹配程度,而代码的语义相似性则主要基于代码含有的属性和结构的关联性以及代码和描述的嵌入表示距离信息进行度量.整体来看,其思路和方法相对比较直观,都是以代码的分布统计关联建模语义关联.因此,现有的代码检索和相似性量化方法很难真实地建模用户需求和代码之间的语义关联,在丰富代码语义建模、感触用户需求表达以及进行代码语义和用户需求匹配检索等方面存在非常大的局限性,因此,代码的检索和相似性量化有很大的提升空间.

2.2 智能化代码推荐和合成

- 代码推荐

代码推荐是指根据代码上下文的情景信息智能化地为软件开发者推荐合适的代码片段的功能,主要有API示例代码推荐和代码补全两个应用场景.在进行软件开发时,Bing Developer Assistant[25]能够根据代码情景信息从开源代码库中选择合适的代码片段来辅助软件开发者进行软件开发,提升软件代码的健壮性.为了提升代码推荐的有效性,Niu 等研究者[26]依据简洁、完整和易于理解的原则设计了代码特征,构建了 learning-to-rank 排序模型用于检索获得的示例代码的排序.由于程序代码有着类似于自然语言的统计特性,因此,自然语言的许多研究成果应用于代码补全的代码情景建模.Raychev 等研究者[27]从开源代码库中抽取代码 API 的调用序列,基于 *n*-gram 语言模型将 API 调用序列的预测问题建模成自然语言句子的生成模型,用于编程环境下实时的代码补全问题.Tu 等研究者[28]也认为代码序列具有重复性和规律性,不同之处在于,这种重复性和规律性具有很强的局部特性,因此采用缓存的局部代码序列取代全局代码序列来训练生成模型.Neuyen 等研究者[29]则将 *n*-gram 模型与主题模型相结合,挖掘更加丰富的语义信息,提升情景预测的准确性.尽管 *n*-gram 在代码补全方面是有效的,但是一些研究者仍然察觉到了自然语言和程序代码之间的差异性.程序代码中,单一功能的 API 调用确实呈现一定的局部有序性,但是在程序功能复杂、多种不同功能的 API 混合调用时,程序代码和自然语言的局部有序性就会呈现较大的差异性.因此,一些研究者[30]又提出了基于程序代码的抽象语法结构特征来建模代码 API 的调用模型,以局部的抽象语法结构子图代替具体的代码语句建模代码的统计特征,进一步提升了代码补全的实际效果.Li 等研究者[31]提出了将代码结构化为程序设计语言相关的抽象语法树(abstract syntax tree,简称AST),利用深度优先的方法进行语法树的序列化,接着融入 LSTM 和指针网络设计深度神经网络结构,训练代码序列的向量化表示用于代码的补全.由于代码 API 的参数信息也是代码补全需要考虑的重要因素,因此,Zhang 等研究者[32]提出了针对开源代码库中 API 进行建模,构建了基于 API 抽象使用实例的参数使用数据库,并根据函数调用的参数信息进行代码补全.

- 代码合成

代码合成是指根据用户的应用需要自动化地选择相关的程序子模块来合成支持特定应用场景的代码程序,强调智能化的用户需求感知与分解、功能模块的选择与合成.根据代码合成应用场景的差异性,分为特定领域的代码合成和开放领域的代码合成.特定领域的代码合成是指代码合成的子功能领域空间是有限的,合成代码支持的模块功能具有显著的领域特色,典型的应用场景是表格填充、字符串编辑等功能.Gulwani 等研究者[33]提出了基于输入输出实例来选择子功能模块完成代码合成操作.为了提升代码合成的准确性和可用性,Desai 等研究者[34]设计了自然语言到子功能模块的关联模型,先将用户自然语言表达的需求映射成子功能模块,再执行代码模块的合成操作.Raza 等研究者[35]将上述两种方法进行有机结合,利用关联模型来选择子功能模块完成代码合成操作,再通过输入输出实例来验证程序的功能.开放领域的代码合成主要研究基于代码 API 进行用户需求的分解和代码程序的合成.由于用户需求和程序的子功能空间异常庞大,开放领域的代码合成通常和代码检索密切关联.Raghothaman 等研究者[36]提出了基于代码搜索的方式将用户需求转换成相应的代码 API,通过开源软件的模式挖掘获取相关的调用模式,用于这些代码 API 和关联代码的合成,最终形成相应的代码程序.为了

进一步提升开放领域代码合成的可用性,Wang 等研究者[37]不仅设计了代码接口的匹配原则来缩小代码的搜索空间,还引入测试用例来提升合成代码的有效性.基于神经网络的代码合成是以程序的输入输出对以及相应的代码片段作为神经网络的输入,借助梯度下降法来训练神经网络模型,用于代码片段或者代码 API 的预测,再借助代码规约技术进行代码合成[38].

小结:代码推荐、补全和合成构成了代码自动化生成的技术环节.现有的代码自动化生成技术主要是依据代码的情景统计特征进行代码语句的分布统计建模,推荐、补全和合成的代码仅仅具有概率上的合理性.尽管现有工作基于输入输出数据进行了概率合理的代码验证,但是这种代码仍然很难达到生产环境下的可用性.此外,代码补全仅仅支持有限的连续步骤,而代码合成也仅能实现简单功能的代码整合.究其原因,主要还是归结到统计语义和用户语义、代码语义之间巨大的语义鸿沟,因此,现有的代码自动生成技术与真正根据业务需要来智能化生成源代码还有非常遥远的距离.

3 知识图谱研究现状

知识图谱是一种重要的知识表示形式,能够打破不同应用场景下的数据隔离.现有的知识图谱研究主要面向通用领域或金融医疗等领域,涵盖知识图谱的建模与表示、知识图谱的管理与应用等两个方面的研究.

3.1 知识图谱的建模与表示

知识图谱的建模与表示主要包括知识图谱的建模、构建、表示、补全、纠错等方面的研究工作.现分别介绍如下.

在知识图谱建模与构建方面,传统知识库一般基于本体建模方法,近年来,尽管知识图谱相关研究很多,有一些研究工作尝试加入时间空间约束,但总体上知识图谱建模仍然进展甚微.在知识图谱构建方面,按其数据源区分,通常可以分为两类:一是 Web 网页,二是相对结构化的在线百科.以 Web 网页为数据源构建的知识图谱主要包括 KnowItAll[39]、TextRunner[40]、Probase[41].KnowItAll 提出了基于规则模板的实体/概念关系抽取方式.然而,人工定义的规则模板虽然在精度上较好,但无法进行大规模的泛化.TextRunner 提出了自监督学习方法构建知识图谱.Probase 则借助 Hearst 模板,通过迭代方法抽取了大量的 isA 关系.以在线百科为数据源构建的知识图谱主要包括 YAGO[42]和 DBPedia[43]等,是基于维基百科中的结构化信息自动化构建的大规模知识图谱.开源软件通常会发布程序源代码、缺陷报告、邮件列表和问答文档等软件开发过程的资源信息.在代码知识图谱构建方面,研究者不仅从缺陷报告、邮件列表和问答文档中挖掘实体,还构建了涵盖包、类、接口、域以及方法等传统的代码实体,定义了这些实体之间各种复杂的关联关系.研究者进行了方法名称、返回类型以及参数类型的建模,并从这些命名方式中挖掘实体来增强方法作用的语义表示能力[44].HDSKG[45]是一款领域知识图谱的构建工具,能够从 Web 网络上抽取候选三元组,并通过机器学习模型来评估候选三元组的领域关联性.研究者将 HDSKG 应用于 Stack Overflow 网站上的软件知识图谱构建,共抽取了 35 279 条三元组、44 800 个概念及 9 660 个动词短语,其准确性和召回率要远远高于 OpenIE.为了构建实体为中心的知识图谱来辅助理解和修改软件 bug[46],有研究者提出了一种融合条件随机场模型和嵌入模型特征的开源软件 Bug 实体判别方法.

在知识图谱表示方面,现有的工作是将知识图谱中离散符号化的实体、概念和关系转化为连续数值表示[47,48],主要分为以下 3 类:第 1 类是基于翻译的表示学习[49−51],即认为关系 r 是从头实体向量 h 到尾实体向量 t 的一个翻译操作,即 $h+r\approx t$;第 2 类是基于张量分解的表示学习[52,53],即将三元组看成张量中的元素,通过张量分解对实体、概念和关系进行表示学习;第 3 类是基于空间分布的表示学习[54,55],即通过拟合实体、概念和关系的空间特征,通过高维向量建模隐式空间,实现实体、概念和关系的空间分布与知识图谱的结构特征保持某种隐式的对应关系.

在知识图谱补全方面,为了避免对大规模标注数据的依赖,Mintz 等研究者[56]在 2009 年首次将远程监督方法应用于知识图谱建模,进行知识图谱的关系补全.Lin 等研究者[57]针对基于远程监督方法建模关系时存在的严重噪声问题,提出了句子级别的注意力模型来提升关系建模的准确性.Bao 等研究者[58]针对数据稀疏问题提出了基于结构和文本联合表示的知识图谱补全方法.目前,关系补全的研究热点是基于表示学习的关系推理,

即:首先将知识图谱的节点和边表示为低维稠密的向量[59,60],然后基于最大似然估计的方法推断实体对的潜在关系.此外,有部分研究者利用迁移学习模型,从数据集相对充分的知识图谱学到数据特征迁移到实体或关系相对稀疏的知识图谱,进行知识图谱的关系的补全[61].

在知识图谱纠错方面,目前的研究主要包括实体类型纠错、分类体系纠错以及实体关系纠错.实体(type)类型冲突问题是异构网络的常见问题[62].Nickle 利用矩阵因子分解(matrix factorization)预测 YAGO 中实体的类型[63].Nuzzolese 等研究者利用 Wikipedia 中的链接,结合 KNN 算法预测知识图谱中实体的类型[64].Paulheim 等研究者提出利用条件概率确定实体的类型[65].Sleeman 等研究者利用 SVM 为 DBpeida 和 Freebase 中的实体确定其类型[66].在实体关系纠错方面,比较有代表性的工作是基于矩阵分解的知识库纠错[67]和基于社交网络背景的错误链接发现[68,69].Liu 等研究者利用词嵌入技术(word embedding)预测实体之间是否存在一定的关系[70].Dong 等研究者将实体之间的关系看成是一个分类问题,利用机器学习的方法判断两个实体之间的关系是否正确[71].Paulheim 利用统计学习的方法发现知识图谱中存在的错误关系[72].Liang 等研究者[73]基于推断的思路在 Probase 知识库上实现了错误事实的纠错.也有一些研究利用了第三方语料库信息,例如,Lange[74]和 Wu[75]等研究者学习 Wikipedia 中存在的模式,并利用条件随机场建立实体之间的关系.此外,还有基于远程监督的方法同样利用 Wikipedia 建立实体间的关系,例如 Aprosio[76]等.West 等人利用搜索引擎建立实体之间的关系[77],而 Ritze 则是利用整个 HTML 网页[78].还有一些方法针对属性值的纠错,如 Wienand 提出的针对数值型属性纠错的离群点检查算法[79].

小结:尽管通用知识图谱的建模与表示取得了重大的突破,然而面向特定领域的建模与表示的研究仍十分有限,特别是面向代码大数据的知识图谱需要一定的深度与完备性、较高的准确度以及严格与丰富的数据模式.此外,代码知识图谱通常存在数据不均衡和长尾特性,导致这些模型在学习时不能充分学习和理解这些实体的特性,从而无法有效地对错误事实进行纠错.因此,代码知识图谱的建模与表示学习具有很大的挑战性.

3.2 知识图谱的管理与应用

知识图谱的管理与应用主要包括知识图谱的演化更新、存储组织和查询处理这 3 个方面,现分别介绍如下.

知识图谱的演化更新,是维持知识图谱数据时效性和可用性的必然要求.现有进行知识图谱的演化更新工作可以分为两类:一类是对知识图谱进行全量更新,另外一类则是对知识图谱进行增量更新.由于构建知识图谱的原生数据体量庞大、结构复杂,研究者更多地使用增量更新的方式保持知识图谱的时效性和可用性.根据知识图谱数据的原生数据源的使用方式,知识图谱数据的增量更新可分为开放式增量更新[56,57]和受限式增量更新[58].开放式增量更新是指通过原生数据源定期开放的更新接口对知识图谱进行增量式数据更新,但由于大多数原生数据源并不会提供数据更新接口,其应用有限.受限式增量更新是指通过网络方式获取原生数据源用于知识图谱数据的更新,但由于网络获取方式的局限性,原生数据源的获取能力通常是受限的.

在知识图谱存储组织方面,目前的研究工作主要分为泛化存储和关联存储两种类型.泛化存储通过牺牲数据关联性以达到统一数据存储建模的目的,典型的存储方式包括三列表方式[80]、聚类方式[81]、谓词方式[82]和实体方式[83,84]这 4 种存储组织方法,但它们均存在各自的局限性.三列表方式直接面向资源描述框架(RDF),构建一个包含 3 个列的数据表用于存储主语、谓语和宾语,尽管能实现完美的泛化存储,但是割裂了所有的数据关联.聚簇方式将具有相同数据模式的数据聚簇到同一张表中组织存储,实现了一定程度的关联存储,但数据的无模式特性导致空间浪费.谓词方式为每一种谓词创建一个单独的三列表,实现按照谓词分割的三列表存储,但它需要涉及更多的 JOIN 操作,且仅仅支持绑定谓词的查询.最近也有研究者提出面向实体的存储方式[83,84],将三列表中的主语或宾语作为键值,剩余的三列表属性作为属性对与键值进行关联存储.这种存储方式增强了局部的关联性,减少了数据查询时的关联构建,但是由于属性组采用 HASH 方式映射存储位置,给数据的实际读取带来了很大的不确定性.关联建模方式则是数据的存储直接面向特定的查询,不再纠结于统一的数据存储模型,以图模型[85,86]组织的存储方式最为典型,以图中的所有节点作为键值,所有与该键值关联的其他节点和关系标签作为该键值的属性值,建立起所谓的 Key-Value 的存储架构,但这种存储方式的维护代价较大.代码知识图谱主要有两种离散型的知识存储建模方式:一种是基于资源描述框架的方式,将代码知识图谱表示为〈主语,谓语,

宾语〉或者〈主语、属性、属性值〉,存储到 Jena 和 gStore 等通用的三元组存储系统中,提供 SPARQL 语言的查询机制;另一种则是建模成属性图的形式,包含节点、有向边和属性,节点使用带有标签有向边进行关联,节点和有向边可以拥有键值对的属性,使用 Neo4j、Virtuoso 等图数据库的底层实现模型来组织存储,提供 Cypher 或者类 SQL 的数据操作和访问机制.研究者以开源软件 Lucene-Core 为例构建了代码知识图谱,验证了基于属性图架构的存储管理方式能够有效地支持代码补全和检索功能[44].此外,Wang 等研究者[87]设计了自然语言接口来访问代码知识图谱,从代码知识图谱中抽取元模型来构建面向查询的推理子图,用于自动化地将自然语言查询转换成结构化的 Cypher 查询,再调用图数据库查询引擎获取相应的查询答案.

在知识图谱查询处理方面,现有的研究工作主要集中在以下几个方面:实体查询、实体关系查询、关键词查询、自然语言查询.在进行实体查询时,一个必要的步骤是计算实体之间的相似性[88].早期的方法主要基于语料库中文字分布的相似度来衡量[89],但这种方法无法衡量文本的语义信息.由于知识图谱对实体的表述更加具体和准确,后续学者研究利用知识图谱中基于实体间的最短距离[90]进行相似度计算,但该方法无法处理不同分级密度的知识图谱.为了克服这种局限性,Zhu 等研究者在最短距离的基础上,提出结合信息量概念来更准确地衡量实体的相似性[91].此外,也有部分学者基于图的结构信息来计算实体的相似度[92].但这些方法无法区分路径上的属性,因此,文献[93,94]提出了利用 meta-path 的概念来计算实体的相似性.文献[95]研究了实体推荐,并基于概率模型给出了推荐的解释.目前的实体相似查询算法的复杂度大都高于线性,因此算法的可扩展性仍有待继续提高[96].在实体关系查询方面,近期的研究开始考虑实体关系的属性因素.文献[97]研究了实体对在 meta-path 上的关系,返回在这种关系下联系最紧密的实体对.文献[98]研究了在异质网络中,属性不完善时的节点聚类.自然语言查询即给定一个查询语句,在知识图谱上返回与查询语句密切相关的结果.文献[99]提出了一种从知识库中学习解答一系列问题的系统,利用单词和知识库成分的低维嵌入返回相应的子结构.文献[100]提出了基于模板的 KBQA 系统,将自然语言查询转化为提问模板.文献[101]研究了多个查询语句和 SPARQL 查询模板的匹配,并提出了几种结构和概率的剪枝技术来提高效率.关键词查询的主要目的是找到与查询关键词相关的子结构.文献[102]将非结构化、半结构化和结构化的数据建模为图,以便自适应地处理异构数据上的关键字查询.文献[103]利用摘要技术生成 top-k 结构化查询语句供用户选择.文献[104]研究了 RDF 结构上基于回溯算法的关键字查询检索模型.文献[105]提出用覆盖率的概念来定义查询结果汇总的缺失程度.文献[106]利用信息增益来评估搜索目标存在的概率,以更好地理解用户的搜索意图.文献[107]将关键字匹配的消歧工作整合进查询图构建中,组合成可以高效表示用户查询目的查询图.

随着代码大数据和人工智能技术的日益成熟,智能化的方式进行软件开发得到了越来越多的关注.研究者提出了一个由数据聚合、知识获取和智能辅助组成的智能化软件开发框架,其中,智能辅助则是以知识图谱为基础来支撑软件构建、测实验证、协助组团以及软件演化和维护等环节[108].但是,在现阶段还很难实现统一而全面的代码知识图谱建模,许多代码知识图谱是直接面向相关应用查询进行构建的.研究发现,Stack Overflow 网站上许多问题的答案都引用了 API 文档,有的甚至直接链接到 API 文档,这说明许多软件开发者没有能够完全掌握使用文档中已经阐明的各种 API 使用方法.研究者设计了语法模式从使用文档中抽取警告语句,应用共引解析技术和启发式方法将句子中的各种代词转化成相关实体,并将这些警告语句链接到 API 实体形成 API 警告知识图谱来提升 API 的可使用性,避免软件的警告信息,加快软件开发效率[109].代码检索是智能化软件开发的基础,而利用代码术语知识图谱进行查询重写,能够有效地增强代码检索能力.查询重写主要使用两种类型的代码术语知识图谱:一种是 Wordnet 等通用知识图谱涵盖的软件领域的上下位词汇[110],另一种则是软件领域自行构建的代码术语关联的知识图谱.这种构建方式通常是从软件的结构关系中获取代码实体名称蕴含的代码术语,挖掘这些代码术语之间的上位、下位、近义、反义、整体、局部等术语关系[111].

小结:总的说来,现有知识图谱管理存在如下局限性:(1) 数据源的获取能力有限,使得现有增量更新方式范围受限;(2) 现有的存储组织方式容易导致语义割裂和维护困难的问题,无法适应知识图谱的复杂结构演化;(3) 现有的查询处理方法从查询表达到结果反馈依然存在巨大的语义鸿沟,需要进一步探索准确理解用户意图的方法.在软件工程领域,人员组织结构会根据业务需要和进度情况灵活安排,软件模块之间的功能划分也会随

着开发的推进合理调整,并且,程序语言的多态、重载、继承等特性导致代码知识图谱中属于同一类的实体在文本上经常会高度重复.这些特性驱使我们去寻求新的模式,以支持代码知识图谱的存储、增量更新和查询处理.

4 基于代码知识图谱的智能化软件开发研究趋势

智能化软件开发是人们实现高级能动智能的重要手段,但是人与机器、机器与代码之间存在巨大的语义鸿沟.基于知识图谱实现人、机器和代码之间的语义关联,能够为代码搜索、代码推荐和代码自动生成等智能化软件开发应用提供基础性支撑.现有的研究仍然处于初始的起步阶段,将从代码知识图谱的建模与表示、构建与精化、存储与演化管理、查询语义理解以及智能化应用这 5 个方面全面介绍代码知识图谱赋能智能化软件开发的研究新趋势.

4.1 代码知识图谱的建模与表示

为了实现多源异质代码数据的整合,将离散分布的代码数据连通为全局统一的知识库,为软件智能化开发提供基于全网数据的技术支持,研究代码知识图谱的建模方法,包括代码大数据中概念、实体、属性的抽象方法,以及代码领域实体间的关系映射规则,形成具有属性的实体/概念通过关系链接而成的知识图谱模型;研究知识图谱数据的表示方法,特别是代码数据、评价记录、修正记录、功能描述等不同异质数据的统一表示方式,以及软件领域知识图谱中长尾实体的向量学习方法.

- 代码知识图谱的建模.

知识图谱的目标在于描述真实世界中存在的实体、概念以及关系.通用知识图谱模型难以精确表达代码独有的复杂语义关系,而软件工程建模方法大多关注单一项目中实体间的简单关系,没有考虑多源异构数据中其他事实的影响以及它们的全局结构关系.为了实现面向开源项目中代码知识图谱建模,需要实现对代码数据、评价记录、修正记录、功能描述等不同类型的软件资源进行实体、概念、属性以及相互关系的识别和抽取.此外,为了更好地表达代码语义,研究代码内部类节点和函数节点之间的关系定义方法,根据节点语义构建节点之间的关联关系.

- 异质数据的知识图谱表示.

随着互联网技术的快速发展,代码数据的表现形式层出不穷,开发人员获取重用代码的方式变得极为丰富.构建代码知识图谱的原始数据非常广泛,既有来自源代码数据的函数、模块、项目的功能描述,也有代码模块的评价记录和修正记录.异质数据中实体和关系也变得更加丰富,以深度学习为例,ConvNetJS、TensorFlow、Theano 和 Keras 等都是支持深度神经网络应用的框架,TensorFlow、Theano 和 Keras 支持 Python 和 C++等高级编程语言,而 ConvNetJS 则支持 Javascrpts 的脚本运行;ConvNetJS、TensorFlow 和 Theano 是独立运行框架,而 Keras 则需要以 TensorFlow 和 Theano 为底层支撑框架.异质数据中实体和关系的统一表示形式,是代码知识图谱研究需要解决的一个关键问题.

- 代码知识图谱中长尾实体的向量化学习.

尽管开源项目代码数据丰富多样,软件代码领域仍存在大量长尾数据.例如,快速排序、外排序和插入排序等排序算法为大多数研究人员所重用的代码,而内存溢出等代码信息相对较少.由于代码数据分布上的严重不均,长尾数据的样本极为匮乏,从而很难为其习得有效的向量表示.一个不容忽视的现象是,不同代码数据在版本控制、评价记录、功能描述以及模块划分等方面具有诸多相似之处,因此,如何联合丰富代码领域的知识图谱数据进行匮乏代码领域知识图谱中长尾节点的向量化学习,是一个亟需解决的问题.

4.2 代码知识图谱的构建与精化

开源环境下的代码数据存在分布不均、数据稀疏、质量参差不齐等特性,决定了代码知识图谱构建的复杂性.为了提高代码知识图谱质量,纠正数据中蕴含的错误事实,增强对软件领域描述的覆盖范围,需要研究代码知识图谱的构建与精化方法,包括软件新实体的识别、代码图谱中已有软件实体对的关系抽取、软件领域的概

念图谱上下位关系的补全以及基于众包的筛选和纠错.

- 代码知识图谱的命名实体识别.

代码知识图谱的核心单元是实体,因此,软件新实体的发现是知识图谱构建的关键步骤.目前,主流的实体识别技术是基于统计学习的方法,但软件领域通常缺乏充足的训练数据,直接利用机器学习方法很难精准地识别出新实体.如何基于少量领域训练数据来训练高精准的命名实体识别模型,即领域数据稀疏条件下的软件新实体识别.研究如何充分利用开放领域学习好的特征(比如相同的词汇,相同的句法)来帮助训练软件领域实体识别模型.

- 样本匮乏环境下的关系抽取.

代码知识图谱依赖语义关系将图谱中的实体关联起来,因此,实体间的关系抽取是知识图谱的关键技术之一.软件领域关系抽取的重要形式之一是从软件领域的非结构化文本中抽取出软件相关的关系三元组.目前,主流的关系抽取任务是基于统计学习方法,但构建领域关系抽取的领域训练数据通常较为缺乏.研究领域数据稀疏背景下的关系抽取方法.考虑到不同领域关系通常有不同规模的训练数据量,借助递进学习(curriculum learning)的思路,通过安排不同关系的训练顺序来充分、可靠地抽取信息.此外,样本的语义描述包括语法标记、实体类型等信息也对关系三元组抽取有极大的帮助,如何在训练模型中融合这些辅助信息,是一个值得研究的问题.

- 代码知识图谱的补全.

知识图谱的规模有限,因此任何图谱都很难覆盖所有的知识,也即,基于手工或半自动方法构建的代码知识图谱通常是不完备的.例如,“C++”是一种“面向对象的程序设计语言”,但该事实可能未被已有的知识库所涵盖.因此,知识图谱的补全任务在图谱的完善和精化中具有不可替代的作用.研究代码知识图谱的补全技术,特别是代码知识分类体系下的上下位事实补全,能够有效地支持知识图谱的泛化查询.从代码数据中抽取的知识通常具有较高的可信度,因此,研究如何基于已有知识事实推断未知事实,是进行高质量代码知识图谱补全的可行途径.

- 代码知识图谱的筛选和纠错.

知识图谱通常由手工或半自动化的方法构建,但由于其大规模的特性,通常存在部分错误的三元组事实,因此,代码知识图谱的筛选和纠错是提升代码知识图谱服务水平的有效举措.领域图谱的纠错很难依赖规模较小的领域训练数据完成,可借助其他人工智能项目(如 ImageNet)的成功经验.采用众包技术进行知识图谱错误事实筛选和纠错,是实现海量数据的高质量标注的有效途径.但是不同于通用知识图谱,软件领域的知识图谱对于非软件行业的从业人员来说可能是一个完全陌生的领域.因此,在一个不确定环境下,如何构建高质量的众包系统进行候选三元组事实的标注、设计众包奖励制度、融合众包结果以及评估和监控众包质量,则是一个极具挑战性的问题.

4.3 代码知识图谱的存储与演化管理

为了保证知识图谱数据的时效性和可用性,满足用户不断演变的数据查询和分析需要,则要研究代码知识图谱数据的演化更新机制和存储组织方法.主要包括:研究知识图谱数据区域演化侦测机制,构建知识图谱数据与演化知识图谱数据的融合更新策略;研究知识图谱数据统一而泛化的存储组织方法,适应知识图谱数据不断增长和变化的实际情况.

- 代码知识图谱的演化更新策略.

软件是一个复杂的实体,涉及到类、方法、函数、测试用例、开发人员、文档和数据,这些实体之间存在紧密的耦合关系.以 SVM 算法为例,根据损失函数中的正则项,可以分为 L1-SVM 和 L2-SVM,根据优化方法,又可以分为梯度下降、指数梯度下降和 TRON 方法.一个 SVM 算法发生了变化,必然导致类、接口以及函数之间各种关系发生变化.因此,代码知识图谱中的演化不是一个实体的演化,而是一个区域内实体和关系的演化.代码知识图谱的演化管理主要包括以下几个研究问题:(1) 如何建模代码知识图谱的区域演化,构建实体和关系之间的关联发生机制和拓扑依赖关系;(2) 在代码知识图谱发生演化时,如何确定代码知识图谱的演化区域,并

实施相应的演化动作.

- 代码知识图谱的关联存储机制.

传统的知识图谱存储主要关注知识管理,但是代码知识图谱不仅包含知识数据,还包含大量的元知识数据.代码知识图谱的元知识信息不仅能够指明代码数据的版本和运行条件,还能够表明代码知识的起源和元知识构建时间等信息.因此,带有元知识的代码知识图谱数据既包含软件开发人员、代码模块、函数等离散型实体属性,又涵盖数字、时间等连续性字面属性.这些元知识信息丰富了代码知识图谱的信息维度,有助于提升代码知识图谱用于智能化软件开发的服务水平.代码知识图谱的关联存储机制主要包含以下几个研究问题:(1) 如何建模带有元知识的知识图谱查询语义,降低语义表达所带来的操作代价;(2) 如何建模带有元知识的代码知识图谱存储,通过存储机制表达代码知识图谱中实体之间关联性,以此来提升代码知识图谱的检索性能;(3) 考虑到代码知识图谱中普遍存在的连续属性,如何有效地支持面向代码知识图谱中连续属性的查询.

4.4 代码知识图谱的查询语义理解

在代码知识图谱上进行语义计算和查询处理,是实现代码知识图谱有效利用的重要方式,正确地理解用户发起的面向代码知识图谱查询的语义就显得格外重要.查询语义理解的研究主要涵盖以下内容:准确理解用户发起不同查询类型的查询语义,支持满足要求的查询结果返回;借助辅助信息理解用户的查询语义,实现代码知识图谱的扩展检索;优化代码知识图谱的查询性能,提升代码知识图谱查询结果的可解释性.

- 查询语义的精准理解和分析.

在代码知识图谱查询中,用户通常以词组或者语句的形式表达一个查询.一般情况下,一个词组或者语义是具有歧义的.这样,当用户查询转换为内部查询时,一个语句可能转换成多种结构化的表达形式.因此,用户的输入查询与知识图谱中实体和关系的表达形式存在巨大的语义鸿沟.如何理解和表达用户的查询意图,便成为知识图谱中查询研究的技术难点.另一方面,在代码知识图谱中,由于程序设计语言的多态、继承、重载等特性,不同实体也可能存在高度的文本相似性,给查询语义的理解增加了难度.查询语义的精准理解和分析主要包含以下研究问题:(1) 如何理解用户的查询意图,实现用户查询到结构化查询的关联映射;(2) 如何设计用户不同查询意图的结构化表达形式,包括实体查询、可达性查询、最短路径查询、正则表达式查询以及模式查询等.

- 查询语义的协同理解与计算.

传统的关于代码语义的协同分析往往是结合用户行为的语义理解与计算,忽略了查询与软件领域中其他信息的语义关系.代码知识图谱是一种软件领域信息结构化的巨型语义网,关联了代码领域中所有的实体、概念、属性以及各种关联关系.用户的行为暴露了用户的偏好,为理解用户的查询意图提供了情景信息,而代码知识图谱包含了所有的领域知识,实际上限定了代码知识图谱支持智能化软件开发的能力.因此,结合用户行为和知识图谱进行代码语义理解和计算是一个亟待解决的难题.查询语义的协同理解与计算主要包含以下研究问题:(1) 如何进行用户历史行为的建模,实现用户行为的结构化语义表示;(2) 如何融合用户历史行为信息和代码知识图谱数据来理解查询语义,实现更好的关联查询结果的反馈.

- 查询结果的高效获取和高质量返回.

代码知识图谱本质上是一个异质网络,其上的诸多查询可以理解为图上的相关操作.由于图数据固有的结构复杂性,其查询常常具有很高的复杂度.此外,由于程序设计语言具有多态、重载、继承等特性,代码知识图谱中属于同一类的实体在文本上经常会高度重复,这就会导致查询结果的大量重叠.最后,基于代码知识图谱的查询不仅应该关注结果的精确性,还应该关注结果的可解释性和可使用性,需要为用户返回与查询相关的背景知识,例如支撑代码的文档、代码背景、代码开发者等信息,以便更好地协助用户理解和使用代码.查询结果的高效获取和高质量返回主要包含以下研究问题:(1) 研究最小化图类型查询代价的通用方法,设计通用的索引类型来提升不同查询类型的执行效率;(2) 研究查询结果的汇总表达,减少查询结果的高度重复;(3) 研究面向相关性、多样性、易理解性等不同查询偏好的结果排序机制,保证能够有效地返回用户感兴趣的结果.

4.5 代码知识图谱的智能化应用

智能化软件开发就是建模软件开发的业务流程,通过代码复用及代码自动化生成来减轻人工编写代码的程度,降低软件开发成本,提升软件开发效率和质量.代码知识图谱能够辅助开发人员把握软件的功能需求,理解代码语义,搜索和匹配语义相似的代码片段,因此,代码知识图谱是实现智能化软件开发的利器.代码知识图谱描述了属性与概念、实体与概念之间的关联性,而概念层面的关联相比于实体或者属性层面的关联更加抽象和本质,这将有助于量化更高层面的实体关联性.因此,代码知识图谱的智能化应用研究将引入概念关联性来研究代码语义相似性计算、代码语义搜索、代码语义推荐以及代码自动化生成.

- 代码语义相似性计算.

代码语义相似性计算是代码搜索、推荐和复用的基础.现有工作主要量化了不同程序代码相关性,更多关注的是程序代码的结构相似性,例如,token 和 string 的相似性、代码行相似性等.通过更改函数命名方式或者调整程序的执行流程,可以轻易地改变程序代码的结构相似性,因此,简单的结构分析难以准确地度量代码片段间真实的相似程度.代码语义相似性计算主要涵盖以下研究内容:(1) 如何为代码数据进行横向的知识建模,实现代码数据中实体、属性以及关系的结构性关联;(2) 如何为代码数据进行纵向的泛化建模,实现代码数据中实体和属性的概念化关联以及关系的复合化关联;(3) 如何融合代码数据的描述信息,量化代码数据的语义相似性.

- 代码语义搜索.

代码语义搜索主要研究人机信息交互的同质化,是实现智能化软件开发的关键环节.传统的代码搜索往往是基于关键字的模糊匹配,该方法只考虑了字符串的字面信息,而忽略了实体和属性之间更为本质的语义关联,这将导致基于字符串的代码搜索很难获取完整、有效的匹配结果.另外,代码数据在很大程度上反映了程序员个人的代码编写风格,例如命名方式和代码结构等,即大部分开源软件数据存在长尾效应,这样,面向长尾数据的搜索结果往往是非常贫乏的.因此,设计一种兼顾语义关联度和语义丰富度的搜索方案就显得尤为重要.代码知识图谱是一个大型语义网,涵盖了软件概念、实体和属性之间的丰富关系,因此,代码知识图谱能够有效地支持代码数据的语义搜索.代码语义搜索主要涵盖以下研究内容:(1) 如何转换软件开发人员搜索词条的结构化表示,将用户输入的单词、语句或者代码片段等搜索词条转换为代码知识图谱上的查询;(2) 如何融入代码知识图谱的层次化信息,实现对结构化查询中实体、属性以及关系的概念化重写;(3) 如何建模代码查询和检索结果的关联度与丰富度,提供调节两种度量值的平衡机制.

- 代码智能推荐.

代码智能推荐是根据用户的代码需求,推荐完全或者部分匹配的软件代码供用户选择.这就需要做好两件工作:一件是感知用户的行为偏好和代码需求,另一件则是匹配用户需求与推荐代码之间的关联性.传统的代码推荐往往是基于开发人员的行为来感知用户需求,但是这种方法仅仅考虑了用户的历史信息,而忽视了其历史信息之间的关联性.例如,一位用户的历史信息表明该用户习惯 Python 语言,并需要使用 SVM 算法进行分类任务,如果代码库中没有 Python 版本的 SVM 代码,那么传统的推荐机制可能无所适从,但是基于代码知识图谱的推荐可以为其推荐 Python 版本的贝叶斯算法.做出这样智能推荐的原因是 SVM 和贝叶斯算法在代码知识图谱的概念层上都属于分类算法.因此,代码智能推荐主要涵盖以下研究内容:(1) 如何挖掘软件开发人员的历史行为,建模软件开发人员的代码需求和行为偏好;(2) 如何构建一个公共的隐式空间,将软件开发人员的代码需求和代码知识图谱投影到同一个维度空间;(3) 如何融合软件开发人员的行为偏好,在不同的维度上为代码用户推荐高质量的代码.

- 代码自动生成.

代码自动生成是指根据软件开发需要或者用户的需求,自动化地生成代码框架和代码片段的过程.在软件行业高速发展的今天,仍然有许多软件开发人员采用手工方式编写代码,这种开发方式往往效率较低且需要大量重复的劳动力.如何快速、准确地理解用户需求,并且为其自动化生成代码,已经成为刻不容缓的研究问题.代码知识图谱是集成了软件源代码、程序 API 代码接口、评价记录、修正记录以及功能描述等不同类型资源信息的网状知识库,因此,代码知识图谱有提升智能化理解软件开发需求的能力,是自动化软件开发的基础.现有

的代码自动生成研究主要面向简单而独立的功能模块,研究成果的可用性和有效性存在巨大的局限性.因此,代码自动生成主要涵盖以下研究内容:(1) 建模软件开发需求,进行软件需求文档、代码接口要求、代码功能需求等不同软件需求文档的统一结构化表示,实现机器与软件需求之间的高效交互;(2) 构建具有一定泛化能力的代码自动生成模型,支持具有复杂功能的代码框架和代码片段的自动化生成,以最优化的方式满足代码的开发需求;(3) 设计融合软件开发者的代码偏好和编程环境来评估自动化生成的代码框架或代码片段的完整性程度,为项目负责人和软件开发者提供多种代码生成选择方案,作为进一步软件开发的基础.

5 总 结

本文指出了知识图谱是实现人、机器和代码等异质数据间语义联通的桥梁,分析了大数据时代智能化软件开发的特点与挑战,代码数据正在从单一同质到多元异质、静态稳定到动态演化而发生变化,人们对于智能化软件开发的期望也从简单的代码检索到复杂的代码自动化生成的提升,代码知识图谱的表示、管理和语义建模应用是基于知识图谱进行智能化软件开发面临的重要挑战.尽管现有工作在代码搜索、代码推荐和代码自动生成等领域进行了相关研究,但是代码搜索和推荐的有效性以及代码自动生成的程度和范围都有很大的局限性,软件智能化开发有着巨大的提升空间.虽然目前已有大量面向开放领域的知识图谱建模与表示、知识图谱管理与应用的研究,但是这些研究成果很难直接移植到代码知识图谱这样的领域知识图谱.最后,详细讨论了代码知识图谱在图谱的建模与表示、构建与精化、存储与演化管理、查询语义理解以及智能化应用等方面研究的新趋势,以更好地满足基于代码知识图谱的智能化软件开发的需要.虽然知识图谱技术呈现出具有提升智能化软件开发的巨大潜能,但是现阶段以代码知识图谱为基础来支撑智能化软件开发的工作还处于初始的起步阶段,研究工作还不是很多,体系性也比较弱,因此本文所述工作更多的还是基于作者和团队自身的理解,受限于水平,不足之处还请谅解.

References:

[1] Economic operation of software industry in 2017. 2017. http://www.miit.gov.cn/n1146285/n1146352/n3054355/n3057656/n5340637/c6040371/content.html

[2] PWC global 100 software leaders. 2016. https://www.pwc.com/gx/en/technology/publications/global-software-100-leaders/assets/global-100-software-leaders-2016.pdf

[3] Liu L. Industrial application and future development of knowledge graph. The Internet Economy, 2018,(4):16−21 (in Chinese with English abstract). [doi: 10.19609/j.cnki.cn10-1255/f.2018.04.003]

[4] GitHub. 2018. https://octoverse.github.com

[5] Schmidt DC. Model-driven engineering. IEEE Computer, 2006,39(2):25−31.

[6] Manna Z, Waldinger RJ. Toward automatic program synthesis. Communications of the ACM, 1971,14(3):151−165. [doi: 10.1145/362566.362568]

[7] Liu BB, Dong W, Wang J. Survey on intelligent search and construction methods of program. Ruan Jian Xue Bao/Journal of Software, 2018,29(8):2180−2197 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5529.htm [doi: 10.13328/j.cnki.jos.005529]

[8] Allamanis M, Barr ET, Devanbu P, Sutton C. A survey of machine learning for big code and naturalness. ACM Computing Surveys, 2018,51(4):1−37. [doi: 10.1145/3212695]

[9] Google search. http://www.google.com

[10] Baidu search. http://www.baidu.com

[11] Krugle. http://www.krugle.com

[12] Search code. http://www.searchcode.com

[13] Thummalapenta S, Xie T. PARESEWeb: A programmer assistant for reusing open source code on the Web. In: Proc. of the 22nd IEEE/ACM Int'l Conf. on Automated Software Engineering. ACM Press, 2007. 204−213. [doi: 10.1145/1321631.1321663]

[14] Stolee KT. Finding suitable programs: Semantic search with incomplete and lightweight specifications. In: Proc. of the 34th Int'l Conf. on Software Engineering. 2012. 1571−1574. [doi: 10.1109/icse.2012.6227034]

[15] Keivanloo I, Rilling J, Zou Y. Spotting working code examples. In: Proc. of the 36th Int'l Conf. on Software Engineering. 2014. 664−675. [doi: 10.1145/2568225.2568292]

[16] Chatterjee S, Juvekar S, Sen K. SNIFF: A search engine for java using free-form queries. In: Proc. of the Int'l Conf. on Fundamental Approaches to Software Engineering. Berlin, Heidelberg: Springer-Verlag, 2009. 385−400. [doi: 10.1007/978-3-642-00593-0_26]

[17] Nie LM, Jiang H, Ren ZL, Sun ZY, Li XC. Query expansion based on crowd knowledge for code search. IEEE Trans. on Services Computing, 2016,9(5):771−783. [doi: 10.1109/tsc.2016.2560165]

[18] Lü F, Zhang HY, Lou JG, Wang SW, Zhang DM, Zhao JJ. CodeHow: Effective code search based on API understanding and extended Boolean model. In: Proc. of the 30th IEEE/ACM Int'l Conf. on Automated Software Engineering. IEEE, 2015. 260−270. [doi: 10.1109/ase.2015.42]

[19] Rahman MM, Roy CK, Lo D. RACK: Code search in the IDE using crowdsourced knowledge. In: Proc. of the 39th Int'l Conf. on Software Engineering Companion. IEEE, 2017. 51−54. [doi: 10.1109/icse-c.2017.11]

[20] Gu XD, Zhang HY, Kim S. Deep code search. In: Proc. of the 40th Int'l Conf. on Software Engineering. IEEE, 2018. 933−944. [doi: 10.1145/3180155.3180167]

[21] Halstead MH. Elements of Software Science (Operating and Programming Systems Series). New York: Elsevier Science Inc., 1977.

[22] Cosma G, Joy M. An approach to source-code plagiarism detection and investigation using latent semantic analysis. IEEE Trans. on Computers, 2012,61(3):379−394. [doi: 10.1109/TC.2011.223]

[23] Đurić Z, Gašević D. A source code similarity system for plagiarism detection. The Computer Journal, 2013,56(1):70−86. [doi: 10.1093/comjnl/bxs018]

[24] Alon U, Zilberstein M, Levy O, Yahav E. Code2vec: Learning distributed representations of code. Proc. of the ACM on Programming Languages, 2019,40(3):1−29. [doi: 10.1145/3290353]

[25] Zhang HY, Jain A, Khandelwal G, Kaushik C, Ge S, Hu WX. Bing developer assistant: Improving developer productivity by recommending sample code. In: Proc. of the 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. ACM Press, 2016. 956−961. [doi: 10.1145/2950290.2983955]

[26] Niu HR, Keivanloo I, Zou Y. Learning to rank code examples for code search engines. Empirical Software Engineering, 2016,22(1): 259−291. [doi: 10.1007/s10664-015-9421-5]

[27] Raychev V, Vechev M, Yahav E. Code completion with statistical language models. ACM SIGPLAN Notices, 2014,49(6):419−428. [doi: 10.1145/2594291.2594321]

[28] Tu ZP, Su ZD, Devanbu P. On the localness of software. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. ACM Press, 2014. 269−280. [doi: 10.1145/2635868.2635875]

[29] Nguyen TT, Nguyen AT, Nguyen HA, Nguyen TN. A statistical semantic language model for source code. In: Proc. of the 9th Joint Meeting on Foundations of Software Engineering. 2013. 532−542. [doi: 10.1145/2491411.2491458]

[30] Nguyen AT, Nguyen TN. Graph-based statistical language model for code. In: Proc. of the 37th IEEE/ACM Int'l Conf. on Software Engineering. 2015. 858−868. [doi: 10.1109/icse.2015.336]

[31] Li J, Wang Y, Lyu MR, King I. Code completion with neural attention and pointer networks. arXiv Preprint arXiv:1711.09573, 2017. http://arxiv.org/abs/1711.09573

[32] Zhang C, Yang JY, Zhang Y, Fan J, Zhang X, Zhao JJ, Ou PZ. Automatic parameter recommendation for practical API usage. In: Proc. of the 34th Int'l Conf. on Software Engineering. IEEE, 2012. 826−836. [doi: 10.1109/ICSE.2012.6227136]

[33] Gulwani S. Automating string processing in spreadsheets using input-output examples. ACM SIGPLAN Notices, 2011,46(1): 317−330. [doi: 10.1145/1925844.1926423]

[34] Desai A, Gulwani S, Hingorani V, Jain N, Karkare A, Marron M, Sailesh R, Roy S. Program synthesis using natural language. In: Proc. of the 38th Int'l Conf. on Software Engineering. ACM Press, 2016. 345−356. [doi: 10.1145/2884781.2884786]

[35] Raza M, Gulwani S, Milic-Frayling N. Compositional program synthesis from natural language and examples. In: Proc. of the 24th Int'l Joint Conf. on Artificial Intelligence. 2015. 792−800. https://dl.acm.org/citation.cfm?id=2832359

[36] Raghothaman M, Wei Y, Hamadi Y. SWIM: Synthesizing what I mean: Code search and idiomatic snippet synthesis. In: Proc. of the 38th Int'l Conf. on Software Engineering. ACM Press, 2016. 357−367. [doi: 10.1145/2884781.2884808]

[37] Wang YP, Feng Y, Martins R, Kaushik A, Dillig I, Reiss S P. Hunter: Next-generation code reuse for Java. In: Proc. of the 24th ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. ACM Press, 2016. 1028−1032. [doi: 10.1145/2950290.2983934]

[38] Cambronero J, Li HY, Kim S, Sen K, Chandra S. When deep learning met code search. arXiv Preprint arXiv:1905.03813, 2019.

[39] Etzioni O, Cafarella M, Downey D, Kok S, Popescu AM, Shaked T, Soderland S, Weld D S, Yates A. Web-scale information extraction in knowitall: (Preliminary results). In: Proc. of the 13th Int'l Conf. on World Wide Web. 2004. 100−110. [doi: 10.1145/988672.988687]

[40] Yates A, Banko M, Broadhead M, Cafarella M, Etzioni O, Soderland S. Textrunner: Open information extraction on the Web. In: Proc. of the Human Language Technologies: The Annual Conf. of the North American Chapter of the Association for Computational Linguistics: Demonstrations. 2007. 25−26. [doi: 10.3115/1614164.1614177]

[41] Wu WT, Li HS, Wang HX, Zhu KQ. Probase: A probabilistic taxonomy for text understanding. In: Proc. of the 2012 ACM SIGMOD Int'l Conf. on Management of Data. 2012. 481−492. [doi: 10.1145/2213836.2213891]

[42] Suchanek FM, Kasneci G, Weikum G. Yago: A core of semantic knowledge unifying wordnet and wikipedia. In: Proc. of the 16th Int'l Conf. on World Wide Web. 2007. 697−706. [doi: 10.1145/1242572.1242667]

[43] Auer S, Bizer C, Kobilarov G, Lehmann J, Cyganiak R, Ives Z. DBpedia: A nucleus for a Web of open data. In: Proc. of the 6th Int'l Semantic Web Conf. 2008. 722−735. [doi: 10.1007/978-3-540-76298-0_52]

[44] Li WP, Wang JB, Lin ZQ, Zhao JF, Zou YZ, Xie B. Software knowledge graph building method for open source project. Journal of Frontiers of Computer Science and Technology, 2017,11(6):851−862 (in Chinese with English abstract). [doi: 10.3778/j.issn.1673-9418.1609026]

[45] Zhao XJ, Xing ZC, Kabir MA, Sawada N, Li J, Lin SW. HDSKG: Harvesting domain specific knowledge graph from content of webpages. In: Proc. of the 24th IEEE Int'l Conf. on Software Analysis, Evolution, and Reengineering. IEEE, 2017. 56−67. [doi: 10.1109/SANER.2017.7884609]

[46] Zhou C, Li B, Sun XB, Guo HJ. Recognizing software bug-specific named entity in software bug repository. In: Proc. of the 26th Conf. on Program Comprehension. ACM Press, 2018. 108−119. [doi: 10.1145/3196321.3196335]

[47] Liu K, Zhang YZ, Ji GL, Lai SW, Zhao J. Representation learning for question answering over knowledge base: an overview. Acta Automatica Sinica, 2016,42(6):807−818 (in Chinese with English abstract). [doi: 10.16383/j.aas.2016.c150674]

[48] Shi C, Sun YZ, Yu PS. Research status and future development of heterogeneous information network. Communications of the CCF, 2017,13(11):35−40 (in Chinese with English abstract). https://www.ccf.org.cn/c/2017-11-15/619584.shtml

[49] Bordes A, Usunier N, Garcia-Duran A, Weston J, Yakhnenko O. Translating embeddings for modeling multi-relational data. In: Proc. of the 26th Int'l Conf. on Neural Information Processing Systems. 2013. 2787−2795. http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-rela

[50] Xiao H, Huang ML, Yu H, Zhu XY. TransA: An adaptive approach for knowledge graph embedding. arXiv Preprint arXiv:1509.05490, 2015. http://arxiv.org/abs/1509.05490

[51] Xiao H, Huang ML, Zhu XY. TransG: A generative model for knowledge graph embedding. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics. 2016. 2316−2325. [doi: 10.18653/v1/P16-1219]

[52] Nickel M, Tresp V, Kriegel HP. A three-way model for collective learning on multi-relational data. In: Proc. of the 28th Int'l Conf. on Machine Learning. 2011. 809−816. http://dl.acm.org/citation.cfm?id=3104584

[53] He WQ, Feng YS, Zou L, Zhao DY. Knowledge base completion using matrix factorization. In: Proc. of the 18th Asia Pacific Web Conf. 2015. 256−267. [doi: 10.1007/978-3-319-25255-1_21]

[54] He SZ, Liu K, Ji GL, Zhao J. Learning to represent knowledge graphs with gaussian embedding. In: Proc. of the 24th ACM Int'l on Conf. on Information and Knowledge Management. 2015. 623−632. [doi: 10.1145/2806416.2806502]

[55] Xiao H, Huang ML, Zhu XY. From one point to a manifold: Knowledge graph embedding for precise link prediction. arXiv Preprint arXiv:1512.04792, 2015. http://arxiv.org/abs/1512.04792

[56] Mintz M, Bills S, Snow R, Jurafsky D. Distant supervision for relation extraction without labeled data. In: Proc. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th Int'l Joint Conf. on Natural Language Processing of the AFNLP, Vol. 2. 2009. 1003−1011. [doi: 10.3115/1690219.1690287]

[57] Bao KF, Gu JZ, Yang J. Knowledge graph completion method based on jointly representation of structure and text. Computer Engineering, 2018,44(7):205−211 (in Chinese with English abstract). [doi: 10.19678/j.issn.1000-3428.0047598]

[58] Lin YK, Shen SQ, Liu ZY, Luan HB, Sun MS. Neural relation extraction with selective attention over instances. In: Proc. of the 54th Annual Meeting of the Association for Computational Linguistics. 2016. 2124−2133. [doi: 10.18653/v1/P16-1200]

[59] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv Preprint arXiv:1301.3781, 2013. http://arxiv.org/abs/1301.3781

[60] Xie RB, Liu ZY, Jia J, Luan HB, Sun MS. Representation learning of knowledge graphs with entity descriptions. In: Proc. of the 30th AAAI Conf. on Artificial Intelligence. 2016. 2659−2665. [doi: 10.1016/j.patrec.2016.09.005]

[61] Hamaguchi T, Oiwa H, Shimbo M, Matsumoto Y. Knowledge transfer for out-of-knowledge-base entities: A graph neural network approach. arXiv Preprint arXiv:1706.05674, 2017. http://arxiv.org/abs/1706.05674

[62] Naiman CF, Ouksel AM. A classification of semantic conflicts in heterogeneous database systems. Journal of Organizational Computing and Electronic Commerce, 1995,5(2):167−193. [doi: 10.1080/10919399509540248]

[63] Nickel M, Tresp V, Kriegel HP. Factorizing YAGO: Scalable machine learning for linked data. In: Proc. of the 21st Int'l Conf. on World Wide Web. 2012. 271−280. [doi: 10.1145/2187836.2187874]

[64] Nuzzolese AG, Gangemi A, Presutti V, Ciancarini P. Type inference through the analysis of Wikipedia links. In: Proc. of Linked Data on the Web, Vol.937 of CEUR Workshop Proc. 2012. http://ceur-ws.org/Vol-937/ldow2012-paper-13.pdf

[65] Paulheim H, Bizer C. Type inference on noisy RDF data. In: Proc. of the Int'l Semantic Web Conf. 2013. 510−525. [doi: 10.1007/978-3-642-41335-3_32]

[66] Sleeman J, Finin T. Type prediction for efficient coreference resolution in heterogeneous semantic graphs. In: Proc. of the 7th IEEE Int'l Conf. on Semantic Computing. 2013. 78−85. [doi: 10.1109/ICSC.2013.22]

[67] Socher R, Chen DQ, Manning CD, Ng AY. Reasoning with neural tensor networks for knowledge base completion. In: Proc. of the 26th Int'l Conf. on Neural Information Processing Systems. 2013. 926−934. http://papers.nips.cc/paper/5028-reasoning-with-neural-tensor-networks-for-knowledge-base-completion

[68] Fouss F, Pirotte A, Renders JM, Saerens M. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. IEEE Trans. on Knowledge and Data Engineering, 2007,19(3):355−369. [doi: 10.1109/tkde.2007.46]

[69] Backstrom L, Leskovec J. Supervised random walks: Predicting and recommending links in social networks. In: Proc. of the 4th ACM Int'l Conf. on Web Search and Data Mining. 2011. 635−644. [doi: 10.1145/1935826.1935914]

[70] Xie RB, Liu ZY, Jia J, Luan HB, Sun MS. Representation learning of knowledge graphs with entity descriptions. In: Proc. of the 30th Conf. on Artificial Intelligence. 2012. 2659−2665. [doi: 10.1016/j.patrec.2016.09.005]

[71] Dong XL, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, Strohmann T, Sun SH, Zhang W. Knowledge vault: A Web-scale approach to probabilistic knowledge fusion. In: Proc. of the 20th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2014. 601−610. [doi: 10.1145/2623330.2623623]

[72] Paulheim H, Bizer C. Improving the quality of linked data using statistical distributions. Int'l Journal on Semantic Web and Conf. Systems, 2014,10(2):63−86. [doi: 10.4018/ijswis.2014040104]

[73] Liang JQ, Xiao YH, Wang HX, Zhang Y, Wang W. Probase+: Inferring missing links in conceptual taxonomies. IEEE Trans. on Knowledge and Data Engineering, 2017,29(6):1281−1295. [doi: 10.1109/TKDE.2017.2653115]

[74] Lange D, Böhm C, Naumann F. Extracting structured information from wikipedia articles to populate infoboxes. In: Proc. of the 19th ACM Int'l Conf. on Information and Knowledge Management. 2010. 1661−1664. [doi: 10.1145/1871437.1871698]

[75] Wu F, Hoffmann R, Weld DS. Information extraction from Wikipedia: Moving down the long tail. In: Proc. of the 14th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2008. 731−739. [doi: 10.1145/1401890.1401978]

[76] Aprosio AP, Giuliano C, Lavelli A. Automatic expansion of dbpedia exploiting Wikipedia cross-language information. In: Proc. of Extend Semantic Web Conf. 2013. 397−411. [doi: 10.1007/978-3-642-38288-8_27]

[77] West R, Gabrilovich E, Murphy K, Sun SH, Gupta R, Lin DK. Knowledge base completion via search-based question answering. In: Proc. of the 23rd Int'l Conf. on World Wide Web. 2014. 515−526. [doi: 10.1145/2566486.2568032]

[78] Ritze D, Lehmberg O, Bizer C. Matching HTML tables to dbpedia. In: Proc. of the 5th Int'l Conf. on Web Intelligence, Mining and Semantics. 2015. 10. [doi: 10.1145/2797115.2797118]

[79] Wienand D, Paulheim H. Detecting incorrect numerical data in DBpedia. In: Proc. of the European Semantic Web Conf. 2014. 504−518. [doi: 10.1007/978-3-319-07443-6_34]

[80] Neumann T, Weikum G. Rdf-3x: A RISC-style engine for RDF. VLDB Endowment, 2008,1(1):647−659. [doi: 10.14778/1453856.1453927]

[81] Chong EI, Das S, Eadon G, Srinivasan J. An efficient SQL-based RDF querying scheme. In: Proc. of the 31st Int'l Conf. on Very Large Data Bases. 2005. 1216−1227. http://dl.acm.org/citation.cfm?id=1083734

[82] Abadi DJ, Marcus A, Madden SR, Hollenbach K. Scalable semantic Web data management using vertical partitioning. In: Proc. of the 33rd Int'l Conf. on Very Large Data Bases. 2007. 411−422. http://dl.acm.org/citation.cfm?id=1325900

[83] Sun W, Fokoue A, Srinivas K, Kementsietsidis A, Hu G, Xie GT. SQLGraph: An efficient relational-based property graph store. In: Proc. of the 2015 ACM SIGMOD Int'l Conf. on Management of Data. 2015. 1887−1901. [doi: 10.1145/2723372.2723732]

[84] Bornea MA, Dolby J, Kementsietsidis A, Srinivas K, Dantressangle P, Udrea O, Bhattacharjee B. Building an efficient RDF store over a relational database. In: Proc. of the 2013 ACM SIGMOD Int'l Conf. on Management of Data. 2013. 121−132. [doi: 10.1145/2463676.2463718]

[85] Zou L, Özsu MT, Chen L, Shen XC, Huang RZ, Zhao DY. gStore: A graph-based SPARQL query engine. The Int'l Journal on Very Large Data Bases, 2014,23(4):565−590. http://link.springer.com/article/10.1007/s00778-013-0337-7 [doi: 10.1007/s00778-013-0337-7]

[86] Shen XC, Zou L, Özsu MT, Chen L, Li YH, Han S, Zhao DY. A graph-based RDF triple store. In: Proc. of the 31st IEEE Int'l Conf. on Data Engineering. 2015. 1508−1511. [doi: 10.1109/ICDE.2015.7113413]

[87] Wang M, Zou YZ, Cao YK, Xie B. Searching software knowledge graph with question. In: Proc. of the 18th Int'l Conf. on Software and Systems Reuse. Springer-Verlag, 2019. 115−131. [doi: 10.1007/978-3-030-22888-0_9]

[88] Zhang FZ, Yuan NJ, Lian DF, Xie X, Ma WY. Collaborative knowledge base embedding for recommender systems. In: Proc. of the 22nd ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining. 2016. 353−362. [doi: 10.1145/2939672.2939673]

[89] Pennington J, Socher R, Manning CD. GloVe: Global vectors for word representation. In: Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing. 2014. 1532−1543. [doi: 10.3115/v1/D14-1162]

[90] Li YH, Bandar ZA, Mclean D. An approach for measuring semantic similarity between words using multiple information sources. IEEE Trans. on Knowledge and Data Engineering, 2003,15(4):871−882. [doi: 10.1109/TKDE.2003.1209005]

[91] Zhu GG, Iglesias CA. Computing semantic similarity of concepts in knowledge graphs. IEEE Trans. on Knowledge and Data Engineering, 2017,29(1):72−85. [doi: 10.1109/TKDE.2016.2610428]

[92] Chakrabarti S. Dynamic personalized pagerank in entity-relation graphs. In: Proc. of the 16th Int'l Conf. on World Wide Web. 2007. 571−580. [doi: 10.1145/1242572.1242650]

[93] Yu X, Sun YZ, Norick B, Mao TC, Han JW. User guided entity similarity search using meta-path selection in heterogeneous information networks. In: Proc. of the 21st ACM Int'l Conf. on Information and Knowledge Management. 2012. 2025−2029. [doi: 10.1145/2396761.2398565]

[94] Yu X, Ren X, Sun YZ, Gu QQ, Sturt B, Khandelwal U, Norick B, Han JW. Personalized entity recommendation: A heterogeneous information network approach. In: Proc. of the 7th ACM Int'l Conf. on Web Search and Data Mining. 2014. 283−292. [doi: 10.1145/2556195.2556259]

[95] Cui WY, Xiao YH, Wang HX, Song YQ, Hwang SW, Wang W. KBQA: Learning question answering over QA corpora and knowledge bases. VLDB Endowment, 2017,10(5):565−576. [doi: 10.14778/3055540.3055549]

[96] Paulheim H. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic Web, 2017,8(3):489−508. [doi: 10.3233/SW-160218]

[97] Wang CG, Sun YZ, Song YL, Han JW, Song YQ, Wang LD, Zhang M. RelSim: Relation similarity search in schema-rich heterogeneous information networks. In: Proc. of the 2016 SIAM Int'l Conf. on Data Mining. 2016. 621−629. [doi: 10.1137/1.9781611974348.70]

[98] Sun YZ, Aggarwal CC, Han JW. Relation strength-aware clustering of heterogeneous information networks with incomplete attributes. VLDB Endowment, 2012,5(5):394−405. [doi: 10.14778/2140436.2140437]

[99] Jayaram N, Gupta M, Khan A, Li CK, Yan XF, Elmasri R. GQBE: Querying knowledge graphs by example entity tuples. In: Proc. of the 30th IEEE Int'l Conf. on Data Engineering. 2014. 1250−1253. [doi: 10.1109/TKDE.2015.2426696]

[100] Zheng WG, Cheng H, Zou L, Yu JX, Zhao KF. Natural language question/answering: Let users talk with the knowledge graph. In: Proc. of the 2017 ACM on Conf. on Information and Knowledge Management. 2017. 217−226. [doi: 10.1145/3132847.3132977]

[101] Diaz G, Arenas M, Benedikt M. SPARQLByE: Querying RDF data by example. VLDB Endowment, 2016,9(13):1533−1536. [doi: 10.14778/3007263.3007302]

[102] Li GL, Ooi BC, Feng JH, Wang JY, Zhou LZ. EASE: An effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In: Proc. of the 2008 ACM SIGMOD Int'l Conf. on Management of Data. 2008. 903−914. [doi: 10.1145/1376616.1376706]

[103] Tran T, Wang HF, Rudolph S, Cimiano P. Top-*k* exploration of query candidates for efficient keyword search on graph-shaped (RDF) data. In: Proc. of the 25th IEEE Int'l Conf. on Data Engineering. 2009. 405−416. [doi: 10.1109/ICDE.2009.119]

[104] Elbassuoni S, Blanco R. Keyword search over RDF graphs. In: Proc. of the 20th ACM Int'l Conf. on Information and Knowledge Management. 2011. 237−242. [doi: 10.1145/2063576.2063615]

[105] Wu YH, Yang SQ, Srivatsa M, Iyengar A, Yan XF. Summarizing answer graphs induced by keyword queries. VLDB Endowment, 2013,6(14):1774−1785. [doi: 10.14778/2556549.2556561]

[106] Shan Y, Li MD, Chen Y. Constructing target-aware results for keyword search on knowledge graphs. Data & Knowledge Engineering, 2017,110:1–23. [doi: 10.1016/j.datak.2017.02.001]

[107] Han S, Zou L, Yu JX, Zhao DY. Keyword search on RDF graphs—A query graph assembly approach. In: Proc. of the 2017 ACM Conf. on Information and Knowledge Management. 2017. 227–236. [doi: 10.1145/3132847.3132957]

[108] Lin ZQ, Xie B, Zou YZ, Zhao JF, Li XD, Wei J, Sun HL, Yin G. Intelligent development environment and software knowledge graph. Journal of Computer Science and Technology, 2017,32(2):242–249. [doi: 10.1007/s11390-017-1718-y]

[109] Li HW, Li S, Sun J, Xing ZC. Improving API caveats accessibility by mining API caveats knowledge graph. In: Proc. of the 2018 IEEE Int'l Conf. on Software Maintenance and Evolution. IEEE Computer Society, 2018. 183–193. [doi: 10.1109/ICSME.2018.00028]

[110] Lu ML, Sun XB, Wang SW, Lo D, Duan YC. Query expansion via wordnet for effective code search. In: Proc. of the 22nd Int'l Conf. on Software Analysis, Evolution and Reengineering. IEEE, 2015. 545–549. [doi: 10.1109/saner.2015.7081874]

[111] Lemos OAL, de Paula AC, Zanichelli FC, Lopes CV. Thesaurus-based automatic query expansion for interface-driven code search. In: Proc. of the 11th Working Conf. on Mining Software Repositories. ACM Press, 2014. 212–221. [doi: 10.1145/2597073.2597087]

附中文参考文献:

[3] 刘柳.知识图谱的行业应用与未来发展.互联网经济,2018,(4):16–21. [doi: 10.19609/j.cnki.cn10-1255/f.2018.04.003]

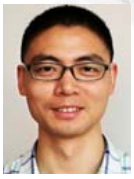
[7] 刘斌斌,董威,王戟.智能化的程序搜索与构造方法综述.软件学报,2018,29(8):2180–2197. http://www.jos.org.cn/1000-9825/5529.htm [doi: 10.13328/j.cnki.jos.005529]

[44] 李文鹏,王建彬,林泽琦,赵俊峰,邹艳珍,谢冰.面向开源软件项目的软件知识图谱构建方法.计算机科学与探索,2017,11(6):851–862. [doi: 10.3778/j.issn.1673-9418.1609026]

[47] 刘康,张元哲,纪国良,来斯惟,赵军.基于表示学习的知识库问答研究进展与展望.自动化学报,2016,42(6):807–818. [doi: 10.16383/j.aas.2016.c150674]

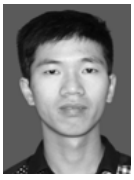
[48] 石川,孙怡舟,菲利普·俞.异质信息网络的研究现状和未来发展.中国计算机学会通讯,2017,13(11):35–40. https://www.ccf.org.cn/c/2017-11-15/619584.shtml

[57] 鲍开放,顾君忠,杨静.基于结构和文本联合表示的知识图谱补全方法.计算机工程,2018,44(7):205–211. [doi: 10.19678/j.issn.1000-3428.0047598]



王飞(1989－),男,江苏连云港人,博士生,主要研究领域为知识图谱,常识挖掘,推荐系统.

钱铁云(1970－),女,博士,教授,博士生导师,CCF 专业会员,主要研究领域为 Web 挖掘,数据管理,自然语言处理.



刘井平(1991－),男,博士生,主要研究领域为知识图谱,常识挖掘,推荐系统.

肖仰华(1980－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为大数据管理和挖掘,图数据库,知识图谱.

刘斌(1975－),男,博士,讲师,CCF 专业会员,主要研究领域为复杂数据管理,数据挖掘.

彭智勇(1963－),男,博士,教授,博士生导师,CCF 会士,主要研究领域为复杂数据管理,可信数据管理,Web 数据管理.