软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007456] [CSTR: 32375.14.jos.007456]

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

融合中文需求文本语义特征的 UML 活动图自动生成方法*

袁中锦[1], 黄　翰[1], 向　毅[1], 刘方青[2], 郝志峰[3]

[1](华南理工大学 软件学院, 广东 广州 510006)

[2](广东工业大学 管理学院, 广东 广州 510520)

[3](汕头大学 数学与计算机学院, 广东 汕头 515063)

通信作者: 黄翰, E-mail: hhan@scut.edu.cn; 向毅, E-mail: xiangyi@scut.edu.cn

摘　要: UML 活动图是软件需求分析的重要工具. 实现由需求文本生成 UML 活动图流程的自动化有助于缩短软件开发时间, 降低人力成本. 现有的 UML 活动图自动生成方法通过人工编写或数据驱动的方式来构建规则, 从需求文本中提取活动图图元素及其关系. 然而, 这些方法通常只考虑到需求文本的语法特征, 忽略了需求文本的语义特征, 导致自动生成的 UML 活动图中可能出现图元素错误或冗余. 因此, 提出一种融合中文需求文本语义特征的 UML 活动图自动生成方法. 该方法结合需求文本与 UML 活动图的相关性、需求文本的时序性提取 UML 活动图图元素及其关系, 弥补了现有方法易受需求文本中无关信息干扰、难以正确识别并表示多种类型业务活动的缺陷. 在 100 个工业界实际应用案例上的消融和对比实验结果验证了该方法在 UML 活动图的完整性、正确性和冗余性方面较其他主流方法的优越性.

关键词: UML 活动图自动生成; 需求分析; 中文需求文本; 语义特征; 智能化软件工程

中图法分类号: TP311

中文引用格式: 袁中锦, 黄翰, 向毅, 刘方青, 郝志峰. 融合中文需求文本语义特征的UML活动图自动生成方法. 软件学报. http://www.jos.org.cn/1000-9825/7456.htm

英文引用格式: Yuan ZJ, Huang H, Xiang Y, Liu FQ, Hao ZF. Automated UML Activity Diagram Generation Method Combining Semantic Features of Requirements Statements in Chinese. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7456.htm

Automated UML Activity Diagram Generation Method Combining Semantic Features of Requirements Statements in Chinese

YUAN Zhong-Jin[1], HUANG Han[1], XIANG Yi[1], LIU Fang-Qing[2], HAO Zhi-Feng[3]

[1](School of Software Engineering, South China University of Technology, Guangzhou 510006, China)

[2](School of Management, Guangdong University of Technology, Guangzhou 510520, China)

[3](School of Mathematics and Computer Science, Shantou University, Shantou 515063, China)

Abstract: Since UML activity diagrams are essential tools for software requirements analysis, automating the process of generating UML activity diagrams helps reduce development time and labor costs. Existing approaches directly extract the elements and their relations of a UML activity diagram from unstructured natural language requirements either by manually constructing extraction rules or adopting data-driven approaches. However, these approaches typically consider only the syntactic features of software requirements statements while neglecting the semantic features. This leads to potential errors or redundancies in the automatically generated UML activity diagrams. Therefore, this study proposes an automated approach for constructing activity diagrams that combines the semantic features of software requirements statements in Chinese. This approach integrates the relevance between software requirements statements and UML activity

* 基金项目: 中央高校基本科研业务费专项资金 (93K172024K24); 国家自然科学基金 (62276103); 广东省普通高校创新团队项目 (2023KCXTD002); 广东省基础与应用基础研究基金 (2023B1515120020, 2024A1515030022); 广东省哲学社会科学规划 2023 年度审计理论研究专项重点项目 (GD23SJZ09); 惠州市重点领域研发项目 (2024BQ010011)
收稿时间: 2024-08-01; 修改时间: 2024-09-23, 2025-03-04; 采用时间: 2025-04-29

diagrams, as well as the temporal properties of the software requirements statements, to extract UML activity diagram elements and their relations. It compensates for the shortcomings of existing approaches, which are easily disrupted by irrelevant information in the requirements and struggle to correctly represent various business activities. Experimental results on 100 industrial cases validate the superiority of the proposed approach over state-of-the-art approaches in terms of the completeness, correctness, and redundancy of automatically generated UML activity diagrams.

Key words: automated UML activity diagram generation; requirements analysis; requirements statement in Chinese; semantic feature; intelligent software engineering

需求工程 (requirement engineering, RE) 是软件开发生命周期 (software development life cycle, SDLC) 的首要阶段[1]. 在需求分析 (requirements analysis) 阶段产生的误解可能会对后续的软件开发阶段造成不利影响, 导致软件维护成本大幅增加[2]. 对象管理组织 (object management group, OMG) 发布的统一建模语言 (unified modeling language, UML) 可以缩短软件客户与需求分析人员之间对需求理解的差距[3]. 活动图 (activity diagram) 作为 UML 支持的图表之一, 能够实现软件开发过程中工作流和业务过程的可视化[4], 促使需求分析人员和客户就待开发软件的组织结构及动态特征达成一致意见, 对提升需求分析的效率起着重要作用.

然而, 即使 UML 活动图功效显著, 自然语言表述固有的模糊性和不确定性仍可能导致人工绘制活动图耗费大量的时间和人力资源, 尤其是对于复杂的大型系统[5]. 对于开发人员而言, 为了正确理解客户需求而在学习领域知识方面的过多投入也是一个棘手的问题[6]. 由此可见, 研发自动化或半自动化的 UML 活动图生成方法对于缩短软件开发时间、提高软件开发效率至关重要.

从本质上讲, UML 活动图自动生成问题是一个从非结构化的需求文本数据中提取具有与活动图生成直接相关文本向量的问题. 根据输入格式, 现有的 UML 活动图自动生成方法可以分为两类: 基于结构化输入的方法和基于非结构化输入的方法. 基于结构化输入的方法要求软件客户严格按照特定的格式来描述需求[7−10]. 这类方法要求客户掌握相应的表达技巧和表示方法, 以撰写出符合标准和规范的需求文本. 这在实际的软件开发过程中往往难以实现[11]. 自然语言处理 (natural language processing, NLP) 技术面世后, 计算机开始具备分析和处理自然语言的能力, 基于非结构化输入的方法应运而生[12−16]. 这些方法利用 NLP 技术分析非结构化的自然语言需求文本, 通过结合分析结果设计的启发式规则从需求文本中提取用于构建 UML 活动图的信息. 但这些方法在构建 UML 活动图图元素及其关系的提取规则时, 通常只考虑到需求文本的语法特征, 对需求文本的语义特征探究较少. 这可能会在自动生成 UML 活动图的过程中引发误差或产生遗漏. 例如, 图 1 中需求文本 1 的首句在语法上遵循“主-谓-宾”结构. 根据文献 [15] 中的 UML 活动图自动生成方法, 该句的内容应该被视作活动图的一个节点. 然而从语义的角度看, 该句并未描述具体的软件业务活动, 不应该出现在最终生成的活动图中. Maatuk 等人[15]与 Kumar 等人[16]提出的 UML 活动图自动生成方法均认为分号后的语句仅用于为分号前的语句提供附加信息, 主张直接舍弃分号后的语句. 但在需求文本 2 中, 分号后面的语句内容属于待开发软件的系统行为, 与业务活动直接相关, 应该出现在最终生成的活动图中.

需求文本1: 登录模块是账号功能模块的基础。系统管理员点进登录界面, 输入账号和密码。如果系统校验成功, 那么系统管理员就可以成功登入系统首页。如果系统校验失败, 那么系统会提示 “账号或密码错误”。

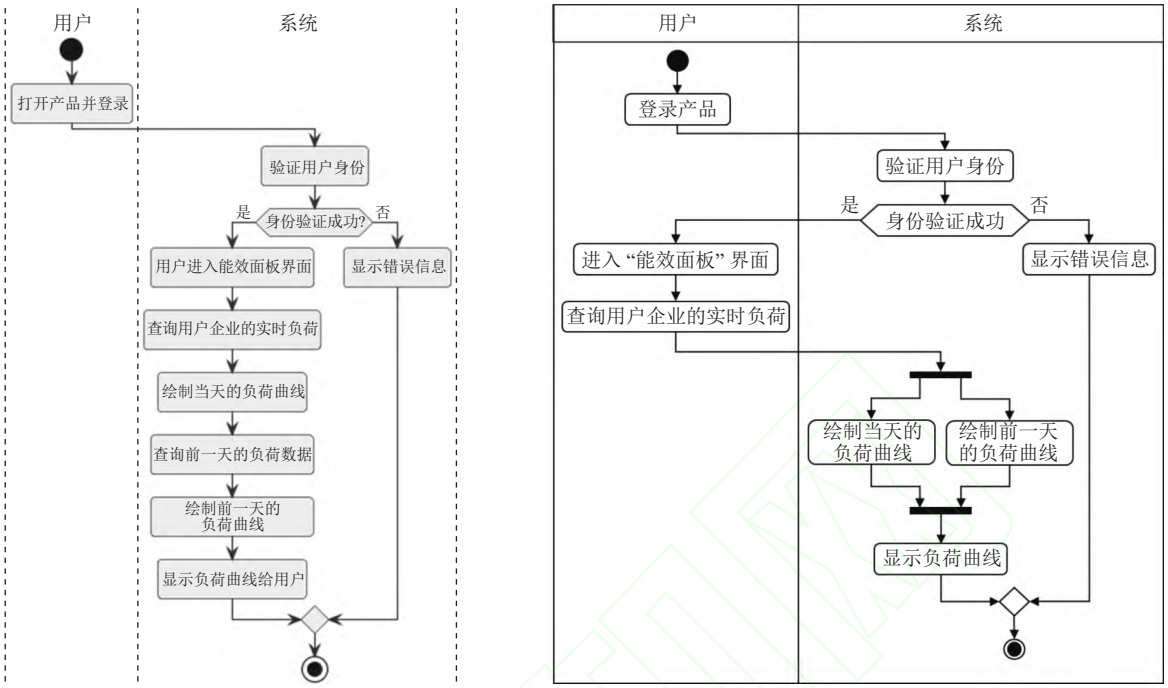
需求文本2: 用户进入APP首页, 如果各项有数据, 将在本期用能情况模块看到商户本期用电量、用水量、用电量同比、用水量同比; 如果各项无数据, 将看到界面数据显示 “- -”。

需求文本3: 用户登录产品, 进入 “能效面板” 界面。系统根据用户信息查询出用户企业的实时负荷, 绘制当天的负荷曲线, 同时绘制前一天的负荷曲线。

图 1 一个软件开发项目中的部分需求描述

此外, 近年来以 ChatGPT 为代表的自然语言生成大模型 (以下简称大模型) 已经在软件工程的多项活动中为软件开发人员提供了一定程度的帮助[17]. 将图 1 中的需求文本 3 作为输入, 使用提示语句“根据以下软件需求文本生成 UML 活动图, 用 PlantUML 表示: [需求文本 3]”, 要求 ChatGPT4 生成对应的 PlantUML 脚本, 由 PlantUML 工具绘制的 UML 活动图如图 2(a) 所示. 相较于图 2(b) 人工绘制的活动图, 这种半自动化的方法能较好地捕捉句式结构等需求文本的语法特征, 但忽视了需求文本在语义上具备时序性, 无法正确表示两个或多个并发进行的业

务活动. 结合以上分析, 在 UML 活动图自动生成过程中对需求文本的语义特征加以利用有助于提高自动生成活动图的完整性和正确性, 降低自动生成活动图的冗余性. 如何融合需求文本的语义特征以提升自动生成 UML 活动图的准确性是一个亟待解决的重要问题.



(a) ChatGPT4 绘制的 UML 活动图　　(b) 人工绘制的 UML 活动图

图 2　根据同一需求文本分别由 ChatGPT4 和人工绘制的 UML 活动图

针对 UML 活动图自动生成任务, 本文提出一个融合中文需求文本语义特征的 UML 活动图自动生成方法 (automated UML activity diagram generation combining semantic features of software requirements statements in Chinese, AutoADGC). 该方法将需求文本的语义特征提取问题建模为一个多标签文本分类问题, 利用需求文本与自动生成活动图的相关性识别出与活动图生成相关的句子, 利用需求文本的时序性确定活动图中的节点类型和相邻节点之间的关系, 结合需求文本的语法特征和语义特征在需求文本相关句中提取活动图图元素及其关系, 完成 UML 活动图自动生成任务. 在 100 个工业界实际应用案例上的实验结果表明, 需求文本的语义特征有助于减少需求文本中无关信息的干扰, 正确处理并发活动, 优化自动生成的 UML 活动图, 使其完整性和正确性更高, 冗余性更低. 综上, 本文的主要贡献如下.

(1) 设计了一种语义多标签需求文本分类模型来提取需求文本的相关性语义特征和时序性语义特征. 该模型通过不同的网络结构分别提取文本与标签的全局、局部特征来增强文本与标签之间的关系. 所获取的需求文本语义特征对于提升自动生成活动图的完整性和正确性具有积极作用, 并能有效减少自动生成活动图中的冗余成分.

(2) 提出了一种融合中文需求文本语义特征的 UML 活动图自动生成方法 AutoADGC, 结合需求文本的语义特征和语法特征实现 UML 活动图的自动生成. 与多种现有的 UML 活动图自动生成方法相比, 本文方法自动生成的活动图在完整性、正确性和冗余性上表现更好.

(3) 本文在由实际软件需求文本构成的数据集上对 AutoADGC 方法的性能进行了全面的评估, 验证了该方法的有效性和实用性. 为了便于其他研究者复现和应用本文方法, 本文基于 AutoADGC 方法实现了一款 UML 活动图自动生成工具. 该工具的源代码及所使用的数据集已在 GitHub 上共享 (https://github.com/yzj-hub/ISES).

本文第 1 节介绍 UML 活动图自动生成的国内外相关研究工作. 第 2 节阐释 UML 活动图自动生成问题的数

学建模以及融合中文需求文本语义特征的 UML 活动图自动生成方法. 第 3 节通过对比实验验证了所提方法的有效性, 并通过消融实验验证了需求文本的语义特征对自动生成活动图准确度的贡献. 第 4 节讨论研究过程中潜在的有效性威胁, 并提出具有针对性的解决方案. 第 5 节总结全文, 并对未来的研究工作进行初步探讨.

1 UML 活动图自动生成相关工作

鉴于 UML 活动图是需求分析中业务活动动态工作流程建模的重要工具, 当前已有不少研究围绕高质量、高效率的活动图自动生成方法展开. 根据输入格式的区别, 这些方法可以被划分为基于结构化输入的方法和基于非结构化输入的方法两类.

基于结构化输入的方法通过限制用户需求的表述形式降低自然语言需求文本的模糊性与二义性. 例如, Gutiérrez 等人[7]为描述每个用例制定了特定模板, 然后使用查询/视图/转换 (query/view/transformation, QVT) 标准 (http://www.omg.org/spec/QVT/1.1/PDF/) 将这些规范化的用例自动转换为 UML 活动图. Yue 等人[8]提出限制性用例建模 (restricted use case modeling, RUCM) 方法, 使用一个用例模板和一系列限制规则对用户的自然语言需求描述进行限制, 并基于 RUCM 方法研发了一个名为 aToucan 的工具[10], 用于支持活动图、顺序图等 UML 模型的自动生成. 此外, 商业词汇和规则语义 (semantics of business vocabularies and business rules, SBVR)[18]也是一种用结构化业务规则来表示用户需求的框架. Iqbal 等人[9]开发了一个应用工具 SBVR2ACTIVITY, 自动解析使用 SBVR 规范撰写的用户需求, 根据句法和语义分析结果选择其中的名词、动词等作为 UML 活动图的组成成分, 自动生成可视化的活动图图形. 虽然这些基于结构化输入的方法能够成功生成符合用户需求的 UML 活动图, 但它们对需求表述的严格限制要求客户花费时间和精力学习相关表达技巧和表示方法, 以编写符合标准和规范的需求文本, 这在实际软件开发过程中是不易实现的.

自然语言处理 (NLP) 技术的出现为 UML 活动图自动生成方法提供了使用非结构化的自然语言文本作为输入的可能性. 例如, RETRANS[12]借助斯坦福提供的一组用 Java 编写的自然语言分析工具 CoreNLP[19]对需求文本进行词性分析, 接着通过关键字跟踪来检测需求中的参与者和用例, 从而自动生成 UML 用例图. 在用例图生成之后, 该方法将用例图中涉及的组件连接起来, 利用类库自动创建对应的 UML 活动图. 此外, 将 NLP 技术与启发式规则相结合是目前最为常见的一种 UML 活动图自动生成方式. Gulia 等人[13]针对需求文本的词性标注和句法分析结果构建了一系列规则. 这些规则能够直接从需求文本中提取构建 UML 活动图和序列图所需的内容. Alami 等人[14]研发了一种半自动化算法, 利用 NLP 工具从阿拉伯语用户需求中自动构建 UML 活动图. 他们使用一个名为 MADA+TOKAN[20]的阿拉伯语 NLP 应用工具包对输入语句进行拆分和标记, 并设计了一套启发式方法以从解析后的需求文本中寻找并提取用于生成活动图的素材. 与之类似, Maatuk 等人[15]也提出一种 NLP 技术与启发式规则相结合的 UML 活动图自动生成方法. 该方法可以根据英文自然语言需求文本自动生成对应的 UML 用例图和活动图. Sharma 等人[5]提出的方法利用 NLP 技术分析需求文本的词法和句法, 通过自定义的语法知识模式将需求文本中的语义信息存储在中间框架中, 实现 UML 活动图和序列图的自动化生成. Abdelnabi 等人[21]开发了一套预定义规则, 用于提取面向对象的概念, 如类、属性、方法和关系, 以便利用 NLP 工具预处理的需求文本自动构建 UML 类图, 并将该方法进一步推广到 UML 用例图和活动图的自动生成中[15]. 除此之外, 机器学习的高速发展激发了软件工程领域的研究者们将人工智能集成到软件和服务中的广泛兴趣[22]. 以 ChatGPT 为代表的大模型已经能够根据自然语言描述的需求文本自动生成与 PlantUML 工具兼容的文本脚本, 让用户通过 PlantUML 工具绘制 UML 活动图. 受上述工作的启发, 本文同样采用 NLP 工具完成对中文需求文本的语法分析, 以消除对输入格式的结构化限制, 贴合真实的需求调研场景.

尽管上述方法取得了一些积极的进展, 但它们在获取 UML 活动图图元素及其关系时仅重点关注了需求文本的语法特征, 而没有更深入地考虑需求文本的语义特征. 例如, 现有方法通常假定客户提供的需求文本均可用于生成 UML 活动图[5], 但需求文本中很可能同时包含与业务活动相关和无关的句子, 类似图 1 中的需求文本 1 和需求文本 2. 无关句容易导致自动生成的 UML 活动图出现节点冗余或节点遗漏的情况. 此外, 需求文本在语义层面的

时间活动关系与自动生成活动图的图元素类型、图元素关系紧密相连, 直接影响到活动图的完整性、正确性和冗余性. 因此, 研究如何将需求文本的相关性、时序性语义特征融入 UML 活动图自动生成过程, 以提高活动图生成的准确率, 是一项极具必要性和挑战性的工作.

2 融合中文需求文本语义特征的 UML 活动图自动生成方法

为了自动生成高完整性、高正确性和低冗余性的 UML 活动图, 本文提出了融合中文需求文本语义特征的 UML 活动图自动生成方法 AutoADGC. 该方法的基本思想是将中文需求文本的相关性、时序性语义特征提取转化为多标签中文需求文本分类, 利用中文需求文本的语法特征和语义特征实现 UML 活动图图元素及其关系提取过程的自动化. 在给出 AutoADGC 的方法流程之前, 首先对 UML 活动图的相关概念以及 UML 活动图自动生成问题的数学建模予以介绍.

2.1 问题建模

本文研究的 UML 活动图遵循 UML 2.5.1 规范[23], 可以看作一种有向图[24]. 形式上, 有以下定义.

定义 1 (**UML 活动图节点**). UML 活动图的任意一个节点都可用四元组 $node=(id,ptt,atv,typ)$ 表示. 其中, id 表示当前节点的唯一标识, ptt 表示当前节点的责任所有者或动作发出者, atv 表示当前节点的动作描述, typ 表示当前节点的类型.

本文研究的 UML 活动图节点类型包括初始节点、动作节点、决策节点、合并节点、分叉节点、连接节点和结束节点共 7 种. 其中, 初始节点表示业务流程的开始, 动作节点显示多种业务活动, 决策节点表示条件分支中互斥发生的活动分支, 合并节点表示互斥活动分支的汇合, 分叉节点代表多个活动分支开始并行执行, 连接节点代表多个活动分支结束并行执行, 结束节点表示业务流程的结束. 非动作节点的动作描述为空. 表 1 展示了 UML 活动图各类节点对应的入度和出度.

表 1 UML 活动图各类节点对应的入度和出度

节点类型	入度	出度
初始节点	0	1
动作节点	1	1
决策节点	1	n ($n\geqslant 2$)
合并节点	n ($n\geqslant 2$)	1
分叉节点	1	n ($n\geqslant 2$)
连接节点	n ($n\geqslant 2$)	1
结束节点	1	0

定义 2 (**UML 活动图边**). UML 活动图的任意一条边都可用四元组 $edge=(id,node_{\text{org}},node_{\text{dst}},rlt)$ 表示. 其中, id 表示当前边的唯一标识, $node_{\text{org}}$ 和 $node_{\text{dst}}$ 分别表示当前活动转换的源节点和目标节点, rlt 表示当前活动转换对应的转换条件或其他补充说明.

定义 3 (**UML 活动图**). 一个 UML 活动图 G 是一个有序的二元组 (AN,AE). 其中, $AN=\{node_i|1\leqslant i\leqslant N\}$ 是一个由 UML 活动图节点组成的非空有穷集合, 称作 UML 活动图节点集合, N 表示节点总数. $AE=\{edge_j|1\leqslant j\leqslant E\}$ 是由 UML 活动图边组成的非空有穷集合, 称作 UML 活动图边集合, E 表示边总数.

以一项包裹快递业务为例[4], 其对应的 UML 活动图如图 3 所示. 图 3 右侧显示了该 UML 活动图的部分图示与其形式化语义表示的对应关系.

基于上述 UML 活动图表示法可知, 活动图自动生成的关键是从需求文本中识别并提取 UML 活动图节点四元组和边四元组中各个属性对应的文本向量. 由于很多词语在不同语境中含义不同, 而句子是意图明确的独立语义单元, 本文以句子为软件需求文本的最小单位. 因此, UML 活动图自动生成问题可以写成公式 (1).

$$F(x):S\rightarrow G \tag{1}$$

其中, $F(x)$ 为 UML 活动图图元素及其关系提取函数, $S=\{s_k|1\leqslant k\leqslant K\}$ 为由 K 个句子组成的需求文本句子集合, $G=(AN,AE)$ 为自动生成的 UML 活动图.

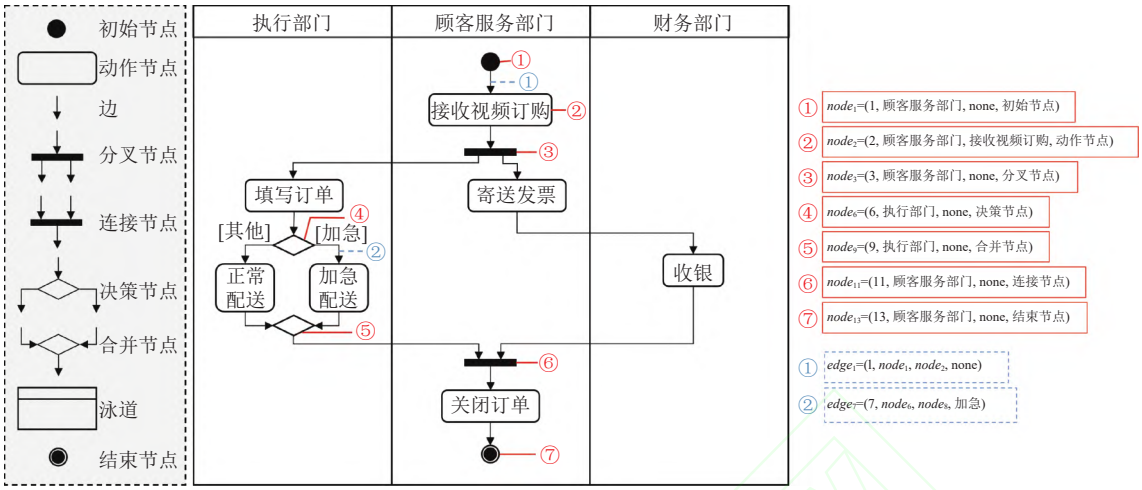


图 3 UML 活动图图示及其形式化语义表示示例

如前所述, UML 活动图图元素及其关系提取不仅要考虑每一条需求文本语句的语法特征提取, 还要考虑语义特征提取, 以及语法特征和语义特征的融合. 词性、句法等需求文本的语法特征是明确的, 而需求文本语句与活动图绘制的相关性、需求文本语句的时序性等语义特征是模糊难辨的. 本文将通过在活动图自动生成流程中引入多标签需求文本分类方法解决这一难题.

2.2 方法框架

本文提出的融合中文需求文本语义特征的 UML 活动图自动生成方法 AutoADGC 是结合软件需求文本的语义特征和语法特征提取 UML 活动图图元素及其关系的过程. 图 4 描述了 AutoADGC 方法的整体框架. 整个自动生成过程主要包括中文需求文本语义特征提取、UML 活动图图元素及其关系提取、UML 活动图绘制这 3 个主要步骤. 本文直接采用主流 UML 图绘制工具 PlantUML 作为图编辑引擎, 下文将重点对中文需求文本语义特征提取和 UML 活动图图元素及其关系提取的实现方法进行具体介绍.

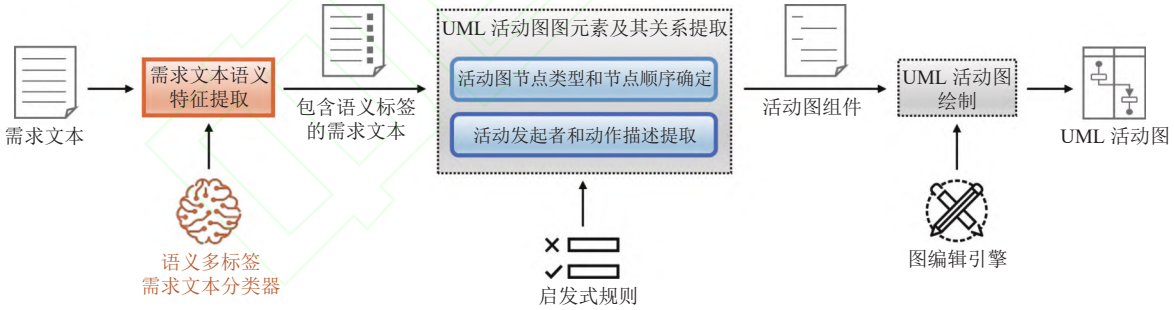


图 4 融合中文需求文本语义特征的 UML 活动图自动生成方法整体框架

2.2.1 中文需求文本语义特征提取

软件需求文本中可能同时存在包含业务活动描述的语句和不含业务活动描述的语句[25]. 以图 1 中的需求文本为例, 尽管一些不含业务活动描述的句子符合某种 UML 活动图图元素及其关系提取规则, 但它们却会为活动图提供错误的内容. 此外, 一些包含业务活动描述但不符合某种 UML 活动图图元素及其关系提取规则的句子可

能在图元素提取过程中被忽略, 从而造成最终生成的 UML 活动图成分残缺. 因此, 本文在进行活动图图元素及其关系提取之前先对需求文本进行筛选, 保留与业务活动相关的句子, 舍弃与业务活动无关的句子. 这种操作能够有效避免图元素及其关系提取过程中可能出现的错漏, 进而提高自动生成的活动图中正确图元素出现的概率. 在本文中, 包含业务活动描述的句子为与 UML 活动图自动生成相关的句子, 不包含业务活动描述的句子为与 UML 活动图自动生成无关的句子. 软件需求文本具备种类多样、有迭代变更诉求的特征[26]. 不同于完全依赖语法规则的方法[6−8], 基于神经网络的机器学习方法可以通过数据驱动的方式覆盖多样化的需求[27], 能够同时学习需求文本的显式特征和隐式特征, 弥补纯规则方法泛化性弱和可维护性差的不足. 因此, 本文采用文本分类器确定需求文本与 UML 活动图自动生成之间的语义相关性.

除了获取软件需求文本中包含业务活动信息的相关句, 还需要考虑如何从这些相关句中提取 UML 活动图节点和边四元组所需的文本向量. UML 活动图的边四元组包含两个活动图节点之间的时间次序关系和关系内容描述. 其中, 两个活动图节点之间的时序关系一般分为以下 3 种: (1) 逆序关系: 对于句子中的活动 A、B, 若自然语言陈述次序与活动实际发生的次序相反, 则称 A 与 B 之间存在逆序关系; (2) 顺序关系: 对于句子中的活动 A、B, 若自然语言陈述次序与活动实际发生的次序相同, 则称 A 与 B 之间存在顺序关系; (3) 同步关系: 对于句子中的活动 A、B, 若 A 与 B 可以同时发生, 则称 A 与 B 之间存在同步关系.

为了获得上述需求文本的相关性语义特征和时序性语义特征, 本文将中文需求文本的语义特征提取问题建模为一个多标签文本分类问题. 具体描述为: 给定语义多标签需求文本分类训练集 $T=\{(T_i,L_i)|1\leqslant i\leqslant M\}$, 其中 $T_i=\{w_i|1\leqslant i\leqslant D\}$ 表示由 D 个词构成的需求文本序列, $L_i=\{l_{i_1},l_{i_2}\}$ 为需求文本对应的标签序列, 其中 l_{i_1} 表示相关性语义标签, l_{i_2} 表示时序性语义标签, M 为训练集样本数. 语义多标签需求文本分类的目标是学习一种预测函数 $f(x)=T_i\rightarrow L_i$ 来预测输入需求文本序列 T_i 的对应标签集合 $L_i=\left\{(l_{i_1},l_{i_2})\middle|l_{i_1}\in\{0,1\},l_{i_2}\in\{0,1,2\}\right\}$, 其中 l_{i_1} 取 0、1 分别表示该需求文本与 UML 活动图自动生成无关、相关, l_{i_2} 取 0、1、2 分别表示该需求文本中的多个业务活动在开展次序上属于逆序、同步、顺序关系.

本文提出的语义多标签需求文本分类模型结构如图 5 所示. 该模型共分为两层, 底层为全局特征提取层, 用于提取需求文本和语义标签的全局特征; 顶层为局部特征提取层, 用于获得需求文本和语义标签进一步交互下的语义信息.

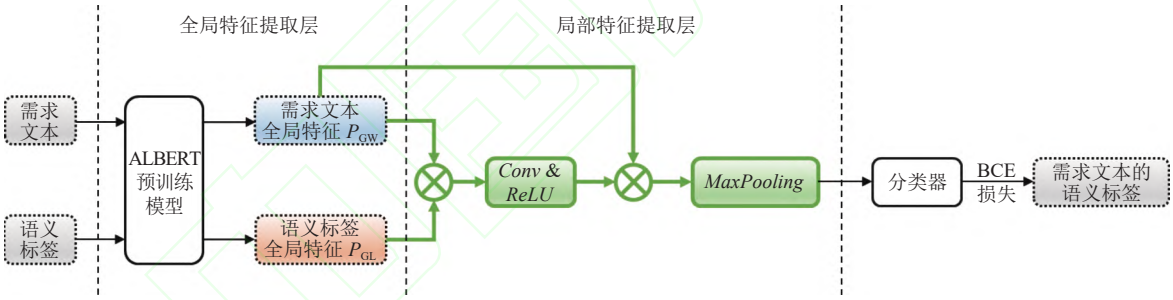


图 5 语义多标签需求文本分类模型

由于基于深度学习的特征提取方法通常需要大规模的专门数据进行训练, 并且对内存资源的需求较高, 故本文使用预先在 BookCorpus 和 Wikipedia 数据集上进行分类训练的 ALBERT 模型[28]提取需求文本的全局特征 P_{GW} 和语义标签的全局特征 P_{GL}. 该模型包含嵌入层和多个相同的编码器层, 主要通过多头注意力机制进行切分多层计算以获取深层次的语义信息, 且通过高度并行以充分利用有限的计算资源. 除了全局特征, 需求文本本身的局部特征也拥有丰富的语义信息. 由于在人工进行语义多标签需求文本分类时, 需求文本中某一关键词与语义标签之间的关联性通常决定对语义标签的判断. 根据文献 [29] 的建议, 局部特征提取层通过点积操作使需求文本词级信息与语义标签信息充分交互, 并使用卷积神经网络 $Conv(\cdot)$ 提取需求文本和语义标签的深层交互信息, 然后结合需求文本的全局特征 P_{GW} 经最大池化获得文本-标签局部特征表示 P_L:

$$P_L = MaxPooling(Conv(P_{GW} \cdot P_{GL}) \cdot P_{GW}) \tag{2}$$

最后, 本文选用二元交叉熵损失 (binary cross entropy loss)[30]作为损失函数, 其定义如下:

$$L_{bce} = -\sum_{i=1}^{I}\sum_{j=1}^{J} l_{ij}\log\left(\widehat{l_{ij}}\right) + (1-l_{ij})\log\left(1-\widehat{l_{ij}}\right) \tag{3}$$

其中, I 为需求文本句子总数, J 为语义标签总数, l_{ij} 和 $\widehat{l_{ij}}$ 分别为语义多标签需求文本分类中第 i 个句子对应标签集合 L_i 中第 j 个标签的真实值和预测值.

2.2.2 UML 活动图图元素及其关系提取

通过语义多标签需求文本分类, 需求文本中的每条句子都获得了对应的相关性语义标签和时序性语义标签. 为了创建 UML 活动图, 需要考虑如何从包含语义标签的需求文本句子中提取活动图节点四元组和活动图边四元组所需的向量信息.

UML 活动图边四元组包含两个相邻活动图节点之间的时间次序关系和关系内容描述. 需求文本的语义标签可以作为确定活动图节点类型和相邻节点连接次序的依据, 具体方式如算法 1 和算法 2 所示. 算法 1 利用需求文本的相关性语义标签保留与 UML 活动图自动生成相关的需求文本句子, 再借助相关句的时序性语义标签确定相关句之间的连接次序. 以图 1 中的需求文本 1 为例, 需求文本 1 中首句的相关性语义标签为 0, 表明该句与业务活动无关, 故该句被算法 1 舍弃, 不参与后续的 UML 活动图图元素及其关系提取. 对于相关性语义标签为 1 的相关句, 算法 1 将根据相邻相关句之间的时序性语义标签调整相关句的排列次序. 例如, 由于需求文本 1 中第 2 句和第 3 句描述的业务活动在执行时间上呈现顺序关系, 这两个句子将按照自然语言表述的次序进行连接.

算法 1. 需求文本相关句业务活动次序确定.

输入: 包含 n 个句子的需求文本句子序列 $T[s_1, s_2, \ldots, s_n]$ 及其对应的相关性语义标签序列; $Rel\_s[rel\_s_1, rel\_s_2, \ldots, rel\_s_n]$;
输出: 需求文本相关句序列 T''.

1. $T' \leftarrow [\,]$
2. **for** $i \leftarrow 1$ **to** n
3. **if** $rel\_s_i == 1$ **then** //根据相关性语义标签确定 s_i 与 UML 活动图自动生成具备相关性
4. 把 s_i 添加到 T' 末尾
5. $m \leftarrow T'$ 的长度
6. $T'' \leftarrow [s_1]$
7. **for** $i \leftarrow 2$ **to** m
8. $s_{temp} \leftarrow s_{last} + s_i$ //s_{last} 表示 T'' 中最后一个句子
9. $seq\_s \leftarrow s_{temp}$ 的时序性语义标签
10. **if** $seq\_s == 0$ **then** //s_{last} 与 s_i 为逆序关系
11. $temp \leftarrow s_{last}$
12. $s_{last} \leftarrow s_i$
13. 将 $temp$ 添加到 T'' 末尾
14. **else if** $seq\_s == 1$ **then** //s_{last} 与 s_i 为同步关系
15. $s_{last} \leftarrow s_{last} + s_i$
16. **else** //s_{last} 与 s_i 为顺序关系
17. 将 s_i 添加到 T'' 末尾

18. **end if**
19. **end for**
20. **return** T''

算法 2 借助相关句的时序性语义标签进一步确定该句所对应的活动图节点类型和相邻节点连接次序, 并提取构成当前活动图边四元组所需源节点、目标节点和活动转换内容描述的短语文本向量. 该算法将根据句中关联词、逗号和分号将句子划分为多个短语, 再按照时序性语义标签确定短语之间的连接次序. 如图 6 所示, 算法 2 将通过句中的关联词“如果”“那么”将需求文本 1 的第 3 句划分为两个短语, 分别对应决策节点和动作节点. 因为这两个短语中包含的业务活动在开展时间上属于顺序关系, 所以这两个短语对应的节点将按照与自然语言陈述次序一致的次序连接.

算法 2. UML 活动图节点类型和节点次序确定.

输入: 需求文本相关句序列 T'' 中的一个需求文本相关句 s;
输出: 一个包含节点次序的需求文本短语数组 P_h, 节点类型数组 P_t.

1. $P_h \leftarrow [\,]$
2. $P_t \leftarrow [\,]$
3. $W \leftarrow Tok(s)$ //分词
4. 根据关联词、逗号和分号将 s 划分为 n ($n \geqslant 1$) 个短语 $[sub_1, sub_2, \ldots, sub_n]$
5. $P_{sub} \leftarrow [sub_1, sub_2, \ldots, sub_n]$
6. $P_h \leftarrow [sub_1]$
7. **for** $i \leftarrow 2$ **to** n
8. $sub_{\text{temp}} \leftarrow sub_{\text{last}} + sub_i$ //sub_{last} 表示 P_{sub} 中最后一个句子
9. $seq\_s \leftarrow sub_{\text{temp}}$ 的时序性语义标签
10. **if** $seq\_s == 0$ **then** //sub_{temp} 与 sub_i 为逆序关系
11. $temp \leftarrow sub_{\text{last}}$
12. $sub_{\text{last}} \leftarrow sub_i$
13. 将 $temp$ 添加到 P_h 末尾
14. **else if** $seq\_s == 1$ **then** //sub_{temp} 与 sub_i 为同步关系
15. $sub_{\text{last}} \leftarrow sub_{\text{last}} + sub_i$
16. **else** //sub_{temp} 与 sub_i 为顺序关系
17. 将 sub_i 添加到 P_h 末尾
18. **end if**
19. **end for**
20. $P_t \leftarrow [typ_1, typ_2, \ldots, typ_n]$ //将 P_h 中各短语对应的节点类型初始化为“动作节点”
21. **for** $i \leftarrow 1$ **to** n
22. **if** sub_i 包含表示条件关系的关联词 **then**
23. $typ_1 \leftarrow$ “决策节点”
24. **end if**
25. **end for**
26. **return** P_h, P_t

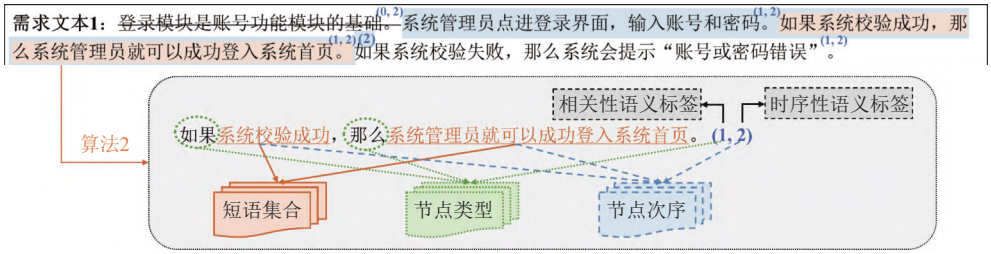


图 6　活动图节点类型和节点次序确定示例

UML 活动图节点四元组包含活动发起者、动作描述和节点类型信息. 在算法 1 和算法 2 中, 我们已经通过分析需求文本的语义特征确定了活动图包含的节点类型. 算法 3 将结合需求文本的语法特征和语义特征实现活动发起者和动作描述的自动化提取.

算法 3. UML 活动图活动发起者和动作描述提取.

输入: 通过算法 2 获取的需求文本短语数组 P_h 中的一个短语 *sub*;
输出: 活动图活动发起者 *as* 与动作描述 *ad*.

1. $W \leftarrow Tok(sub)$ //分词
2. $Pos \leftarrow PosTag(sub)$ //词性标注
3. $Dep \leftarrow DepPar(sub)$ //依存句法分析
4. $l \leftarrow Pos$ 的长度
//找出与动词或介词相邻的名词性主语作为活动发起者
5. **for** $start\_idx \leftarrow 0$ **to** $l-1$
6. **if** $Pos[start\_idx]$ 是名词&$Dep[start\_idx]$ 属于名词性主语 **then**
7. $end\_idx \leftarrow start\_idx$
8. **else if** $Pos[start\_idx]$ 是动词或者介词 **then**
9. **break**
10. **end if**
11. **end for**
//补全活动发起者描述
12. **for** $start\_idx \leftarrow 0$ **to** $l-1$
13. **if** $Pos[start\_idx]$ 是名词&$Dep[start\_idx]$ 属于名词性主语 **then**
14. **break**
15. **end if**
16. **end for**
17. $as \leftarrow W[start\_idx : end\_idx+1]$
//获取动作描述内容
18. **for** $start\_idx \leftarrow 0$ **to** $l-1$
19. **if** $Pos[start\_idx]$ 是动词或者介词 **then**
20. $end\_idx \leftarrow start\_idx$
21. **end if**
22. **end for**
23. $ad \leftarrow W[end\_idx :]$
24. **return** *as*,*ad*

活动发起者是发起业务活动的主体, 必然具备名词属性. 因此, 本文通过算法 3 提取主谓关系中的名词或名词短语作为活动发起者. 以图 6 中需求文本 1 第 3 句的第 2 个短语“系统管理员就可以成功登入系统首页”为例, 算法 3 先运用 NLP 技术对该短语进行分词、词性标注和依存句法分析以获取需求文本的语法特征, 然后在动词或介词前找到一个与之相邻的名词性主语“管理员”, 随后对该活动发起者描述进行补全, 最终得到完整的活动发起者描述“系统管理员”. 此外, 算法 3 以短语中的谓语动词或介词作为起始点, 将其与其后的短语成分均加入动作描述集合中, 作为活动图节点四元组动作描述属性的内容. 例如, 以图 6 中需求文本 1 第 3 句的第 2 个短语中动词“登入”为起始点, 其对应的活动图节点动作描述为“登入系统首页”.

鉴于部分需求文本描述中存在主谓关系缺失或主语为代词的现象, 算法 4 将按照算法 2 获取的活动图节点连接次序, 对算法 3 获取的活动图活动发起者序列进行遍历. 若当前活动图节点缺少活动发起者信息, 算法 4 将根据上下文之间的语义相关性, 推断并指定前一个节点的活动发起者为当前节点的活动发起者. 若当前节点的前一个节点也缺少活动发起者信息, 算法 4 将以其后一个节点的活动发起者作为当前节点的活动发起者. 若对齐所有节点的活动发起者与动作描述后, 活动发起者中仍存在人称代词, 则依据表达习惯, 使用“用户”替换第一人称代词.

算法 4. UML 活动图活动发起者对齐.

输入: 通过算法 3 获取的活动图活动发起者序列 $AS[as_1, as_2, \ldots, as_n]$;
输出: 对齐后的活动图活动发起者序列 AS.

1. **for** $i \leftarrow 2$ **to** n
2. **if** $as_i ==$ “” **then**
3. $as_i \leftarrow as_{i-1}$
4. **end if**
5. **end for**
6. **for** $i \leftarrow (n-1)$ **to** 1
7. **if** $as_i ==$ “” **then**
8. $as_i \leftarrow as_{i+1}$
9. **end if**
10. **end for**
11. **for** $i \leftarrow 1$ **to** n
12. **if** as_i 是人称代词 **then**
13. $as_i \leftarrow$ “用户”
14. **end if**
15. **end for**
16. **return** AS

在从需求文本获得绘制 UML 活动图节点和边所需的全部文本向量后, AutoADGC 方法采用名为 PlantUML 的开源工具将文本向量转换为可视化图形. PlantUML 将通过上述算法 4 提供的文本信息确定 UML 活动图的结构和内容, 在秒级时间内生成图形化的 UML 活动图结果.

3 实验验证

本节利用实际软件开发项目中的软件需求对本文设计的 UML 活动图自动生成方法进行实证研究, 以检验本文方法的有效性与实用性. 下面将详细介绍本文所使用的实验数据, 说明实验评价指标和对比方法, 阐述实验方法及模型参数设置, 并给出实验结果及其分析.

3.1 实验数据

本文的实验数据主要来源于合作企业近期结项或正在进行的真实软件开发项目. 当前公开可使用的需求文本数据集在规模上较为有限, 数据多样性不足, 无法全面支撑 UML 活动图自动生成任务的复杂需求, 难以训练出具有良好泛化能力的语义多标签需求文本分类模型. 因此, 研究人员在获得合作企业授权后, 从 100 个实际软件开发项目中收集了用中文撰写的自然语言需求文本, 其内容涵盖 20 余个不同的应用领域. 经过系统化整理, 所收集的需求文本共包含 1368 个业务活动实例, 平均每个实例包含 6 条需求描述语句. 一个业务活动实例可对应生成一个 UML 活动图. 由于需求文本可能包含敏感信息和隐私数据, 受到版权法或合同条款的限制, 这些需求文本需经过数据预处理才能够用于构建需求文本数据集. 具体的预处理规则及原因如下.

(1) 使用数据掩码和数据匿名化的方式对需求文本进行脱敏处理. 脱敏处理用于去除需求文本中的敏感信息, 保护企业隐私.

(2) 根据句号或换行符将所有需求文本划分为多个单句. 数据划分有助于后续对需求文本进行数据标注.

参照文献 [31,32], 我们从国内软件开发行业龙头企业邀请了 3 名软件工程师对需求文本进行人工标注. 这些软件工程师深耕于需求调研与分析领域, 具备 5 年以上的实战经验, 均具有高级软件工程师资质. 针对相关性语义标签, 标注者需要将每个与 UML 活动图相关的句子标注为 1, 将其他所有句子标注为 0. 针对时序性语义标签, 标注者需要将属于逆序关系的句子标注为 0, 将属于顺序关系的句子标注为 1, 将属于同步关系的句子标注为 2. 考虑到个别句子与 UML 活动图的相关性不明确, 标注者需要在相关性难以判定时将句子标注为相关句, 如此便可以尽量避免遗漏相关句以保证活动图图元素的完整性与正确性. 研究引入弗莱斯卡帕 (Fleiss' kappa) 系数 κ [33]评估 3 位标注者对相同需求文本句子进行标注时的一致性. 依据文献 [34] 所述方法, 计算得出的 κ 值为 0.89. 该结果表明 3 位标注者的标注结果一致性较高, 据此可推断标注结果可靠性较强.

此外, 本文还采用 PROMISE 数据集 (https://github.com/opensciences/opensciences.github.io) 作为对比实验的测试数据. 该数据集是一个广泛用于软件工程领域的公开数据集, 涵盖多种不同类型的开源软件项目[35]. 其下的 requirement-other 子集包含少量可用于构建 UML 活动图的需求用例描述. 由于 PROMISE 数据集中的用例描述均为英文, 而 AutoADGC 目前仅支持中文分析, 本文借助谷歌翻译工具将该数据集中的英文用例描述翻译成中文以供使用. 采用谷歌翻译工具而非人工翻译旨在降低翻译过程的主观性, 并确保该过程易于复现.

3.2 评价指标及对比方法

在本文中, 参照文献 [10], 从完整性、正确性和冗余性这 3 个不同但互补的评价指标来评估自动生成活动图的质量. 表 2 中的数据来源于自动生成的 UML 活动图和软件工程师创建的参考活动图.

表 2 用于推导 UML 活动图质量指标的变量

#	变量名称	变量释义	#	变量名称	变量释义
1	N_{ca}	一个自动生成的活动图中正确的节点数量	5	N_{ta}	一个自动生成的活动图中的全部节点数量
2	N_{ra}	一个参考活动图的全部节点数量	6	N_{tr}	一个自动生成的活动图中的全部边数量
3	N_{cr}	一个自动生成的活动图中正确的边数量	7	N_{ia}	一个自动生成的活动图中冗余的节点数量
4	N_{rr}	一个参考活动图的全部边数量	8	N_{ir}	一个自动生成的活动图中冗余的边数量

自动生成所得 UML 活动图的完整性是指图中正确图元素与参考活动图中全部图元素之比. 自动生成所得 UML 活动图的正确性是指图中正确图元素与图元素总数的比例. 自动生成所得 UML 活动图的冗余性是指图中冗余图元素与图元素总数的比例. 表 3 详细描述了通过自动化手段生成的 UML 活动图在完整性、正确性和冗余性指标上的具体计算方法.

由于目前所知的 UML 活动图自动生成方法均未提供开源代码, 且未能与这些方法的研究人员取得联系, 所以本文主要与部分已公开应用实例和实现细节的方法进行比较. 具体而言, 本文将 AutoADGC 与 GKP4ACTIVITY[5]、SBVR2ACTIVITY[9]、EADGD[13]和 NLPHR[15]方法进行比较. 这些方法都采用自然语言编写的软件需求文本作为输入. 除与上述方法进行对比外, 本文还使用大模型 ChatGPT4 生成与中文需求文本对应的 PlantUML 脚本, 利用

PlantUML 工具绘制 UML 活动图, 以比较基于 ChatGPT4 的方法和 AutoADGC 在 UML 活动图自动生成方面的表现.

表 3　UML 活动图质量指标的计算方法

指标名称	中间变量	计算公式
完整性 (CM_{ad})	节点完整性 $CM_{as}=\frac{N_{ca}}{N_{ra}}$ 边完整性 $CM_{ae}=\frac{N_{cr}}{N_{rr}}$	$CM_{ad}=\frac{CM_{as}+CM_{ae}}{2}$
正确性 (CR_{ad})	节点正确性 $CR_{as}=\frac{N_{ca}}{N_{ta}}$ 边正确性 $CR_{ae}=\frac{N_{cr}}{N_{tr}}$	$CR_{ad}=\frac{CR_{as}+CR_{ae}}{2}$
冗余性 (RD_{ad})	节点冗余性 $RD_{as}=\frac{N_{ia}}{N_{ta}}$ 边冗余性 $RD_{ae}=\frac{N_{ir}}{N_{tr}}$	$RD_{ad}=\frac{RD_{as}+RD_{ae}}{2}$

3.3　实验方法

为了探究需求文本的语义特征对 AutoADGC 性能的影响, 本文设计了变体方法 AutoADGC−SF. 该方法移除了语义多标签需求文本分类模型, 仅利用需求文本的语法特征从中文需求文本中提取 UML 活动图图元素及其关系. 在方法名称中, “SF”是语义特征 (semantic feature) 的首字母缩写. 在相同的软硬件环境下, 本文分别使用 AutoADGC 和 AutoADGC−SF 方法根据相同的软件需求文本自动生成 UML 活动图. 之后, 根据第 3.2 节中所述的评价指标, 对这两种方法所得 UML 活动图的完整性、正确性和冗余性进行比较. 在实验过程中, 受邀的 3 位软件工程师根据收集的 100 份中文需求文本独立绘制 UML 活动图, 并就所有绘制结果进行讨论以达成共识, 最终形成一致的绘制结果. 这份绘制结果将被作为实验中的参考活动图.

为了验证 AutoADGC 方法的有效性, 本文将 AutoADGC 方法与其他 UML 活动图自动生成方法进行比较. 其中, 从文献 [5,9,13] 中收集文中所提方法的应用实例, 并将这些实例中的输入文本作为本文方法的输入, 比较通过本文方法获取的 UML 活动图与通过文献方法获取的 UML 活动图. 文献 [15] 没有给出应用实例, 但提供了方法的实现细节, 因此本文根据文献 [15] 的描述复现了文中方法. 上述对比方法均仅适用于英文需求文本, 而本文提出的方法目前仅支持对中文需求文本进行分析. 为解决语言不统一的问题, 本文先采用谷歌翻译将英文需求文本翻译成中文, 再使用 AutoADGC 方法对其进行解析和提取. 此外, 本文还利用大模型 ChatGPT4 生成与中文需求文本匹配的 PlantUML 脚本, 并由 PlantUML 工具根据该脚本绘制 UML 活动图, 以供比较分析.

3.4　参数设置

为了训练语义多标签需求文本分类模型, 本文将已标注的需求文本句子分为训练集、验证集和测试集. 三者的内容互不重叠. 根据文献 [36] 中的最佳实践, 将训练集、验证集和测试集之间的数据分割比例设定为 8:1:1, 旨在进一步降低信息泄漏风险, 同时更准确地反映模型效能. 本文采用 12 层的 ALBERT-Base-Chinese 作为预训练语言模型, 模型的向量维度为 768, 总参数数量为 24M. 该模型采用 Adam[37]优化器进行训练, 学习率为 5E−5, 批量大小为 16, 最大输入序列长度为 64.

本文实验的硬件环境为 Intel(R) Core(TM) i9-10900K CPU、3.70 GHz 主频、128 GB 内存, 软件环境为 Python 3.8.10. 本文选用 HanLP 工具包 (https://github.com/hankcs/HanLP/) 作为中文需求文本处理工具, 并借助 Transformers 3.4.0[38]、Keras 2.3.1[39]和 TensorFlow 2.2.0[40]框架实现语义多标签需求文本分类模型.

3.5　实验结果与分析

为评估融合中文需求文本语义特征的 UML 活动图自动生成方法 AutoADGC 的有效性, 本文将具体回答以

下 3 个问题.

- RQ1: 需求文本的语义特征是否有助于提高自动生成 UML 活动图的质量?
- RQ2: 与其他 UML 活动图自动生成方法相比, AutoADGC 表现如何?
- RQ3: 在实际应用中, AutoADGC 的效率如何?

3.5.1 消融实验结果与分析

为了回答 RQ1, 本文将包含语义多标签需求文本分类的 AutoADGC 方法与移除语义多标签需求文本分类的 AutoADGC−SF 方法对比. 根据表 2 从上述两种方法自动生成的 UML 活动图以及人工绘制的参考活动图收集数据, 依照表 3 计算自动生成 UML 活动图的完整性、正确性和冗余性, 以评估自动生成活动图的质量. 实验结果如图 7 所示.

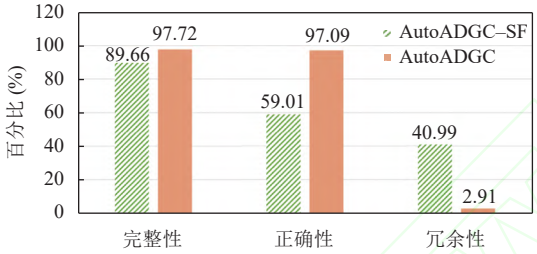


图 7 AutoADGC 与 AutoADGC−SF 方法所生成 UML 活动图的完整性、正确性和冗余性统计结果

图 7 中统计结果显示, AutoADGC 在 3 项评价指标上均优于 AutoADGC−SF 方法. 就完整性而言, AutoADGC 的表现较 AutoADGC−SF 有明显提升. 这说明 AutoADGC 能够更有效地从需求文本中提取出绘制 UML 活动图所需的信息. 这种能力的增强主要归功于对需求文本语义特征的准确把握. AutoADGC 所生成活动图的正确性高达 97.09%, 显著高于 AutoADGC−SF 所生成活动图的正确性. 这是因为语义多标签需求文本分类具备过滤软件需求文本中与业务活动无关句子的功能, 并能够为活动图图元素及其关系的正确提取提供有效信息. 为了更加直观地展示语义多标签需求文本分类在 UML 活动图自动生成任务中起到的重要作用, 我们从实验数据中选取了一个典型案例. 图 8 展示了该案例的输入以及在 AutoADGC−SF 和 AutoADGC 方法上的输出结果.

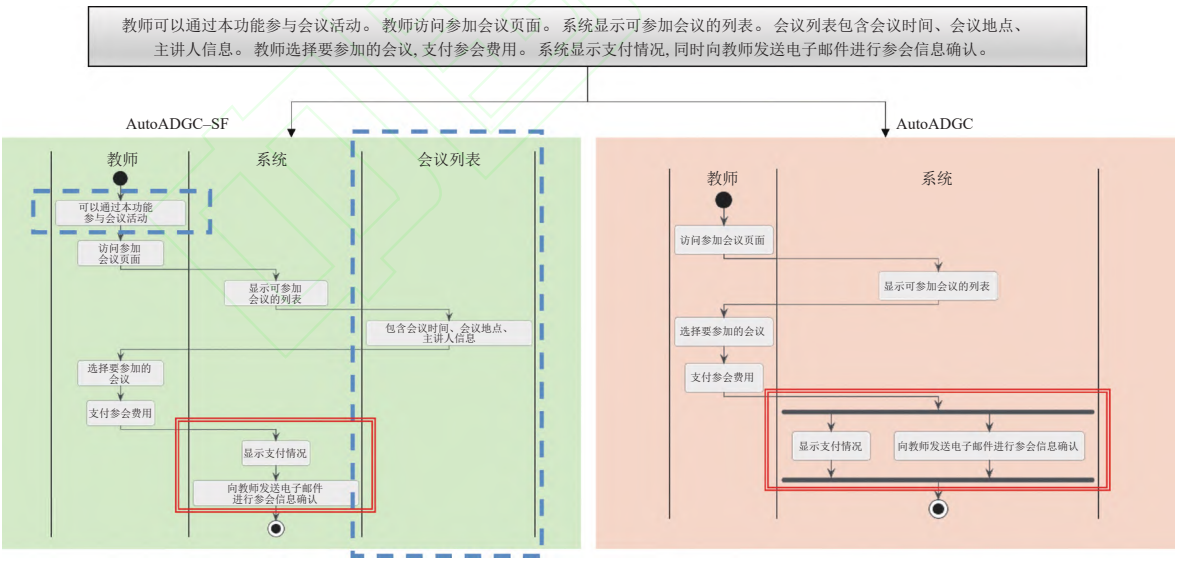


图 8 AutoADGC−SF 与 AutoADGC 方法针对同一应用实例自动生成的 UML 活动图结果

如图 8 中虚线方框部分结果所示, 在根据软件需求文本自动生成 UML 活动图的过程中, 如果去除语义多标签需求文本分类, 虽然仍能够提取到正确的活动图图元素及其关系, 但是也很可能提取到与业务活动无关的信息, 从而导致最终生成的 UML 活动图正确性较低. 对比图 8 中双线方框部分结果可知, 相较 AutoADGC−SF 方法, 包含语义多标签需求文本分类的 AutoADGC 方法正确识别并表示了示例需求文本中的同步时序关系. 此外, 在软硬件条件相同的情况下, 相较于 AutoADGC−SF 的结果, AutoADGC 的结果在冗余性上降低了 38.08%. 这个结果与正确性指标的结果相符, 进一步验证了需求文本的语义特征有助于保留与业务活动相关的句子, 去除与业务活动无关的句子, 降低最终生成活动图的冗余性, 提高最终生成活动图的正确性.

3.5.2 对比实验结果与分析

为了回答 RQ2, 本文对比了 AutoADGC 方法与现有 UML 活动图自动生成方法的性能. 图 9 展示了 AutoADGC 与 GKP4ACTIVITY[5]、SBVR2ACTIVITY[9]和 EADGD[13]这 3 种方法在应用实例上的结果对比. 图 9(a) 中结果表明, AutoADGC 与 EADGD 方法针对同一应用实例生成的结果基本一致. 这可能缘于图 9(a) 应用实例中的每一句需求文本都与业务活动相关, 且需求文本语句中仅包含单一的顺序关系, 需求文本语义特征所发挥的作用难以体现. 这也说明 AutoADGC 方法能够识别软件需求文本中与业务活动相关的内容, 并将其转换为 UML 活动图中的图元素及其关系. 此外, 图 9(b) 与 (c) 中结果表明, 在有多个活动发起者的情况下, AutoADGC 方法自动生成的 UML 活动图包含泳道图元素, 能够清晰展现每个业务活动与活动发起者的对应关系, 更符合 UML 2.5.1 规范[23]. 相比 SBVR2ACTIVITY, AutoADGC 中活动图节点的内容描述和边的动作描述更完整.

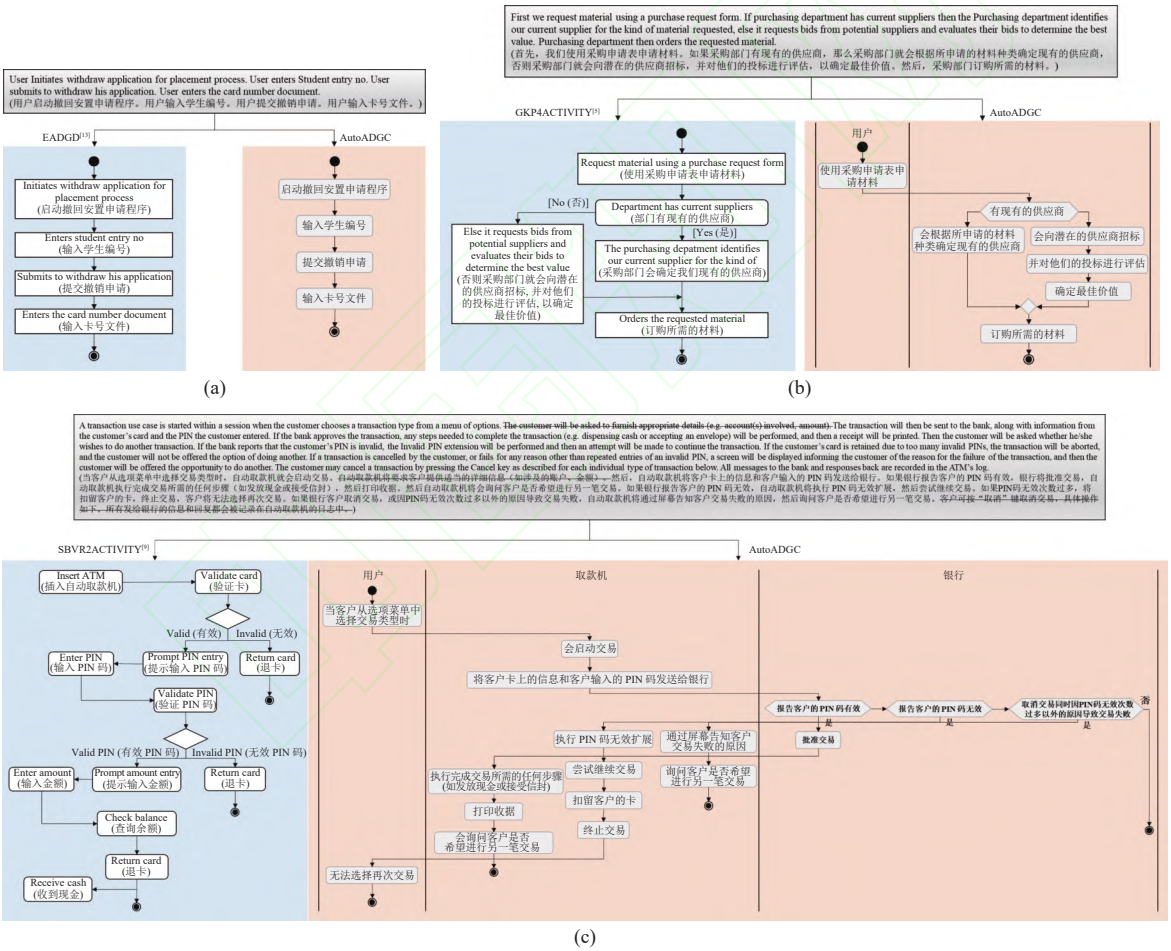


图 9 不同的 UML 活动图自动生成方法针对同一应用实例自动生成的 UML 活动图结果 1

本文还依据文献 [15] 中对 NLPHR 方法的具体描述复现了该方法, 将其作为对比方法之一. 同时, 本文将 ChatGPT4 与 PlantUML 工具结合的 UML 活动图半自动生成方法也作为一种对比方法, 简称 ChatGPT4. 表 4 展示了 AutoADGC、NLPHR 与 ChatGPT4 针对企业提供的真实需求文本以及 PROMISE 数据集中的用例描述所生成 UML 活动图在完整性、正确性和冗余性评价指标上的统计结果. 图 10 是一个 AutoADGC、NLPHR 与 ChatGPT4 针对同一需求文本生成 UML 活动图结果的实例.

表 4　AutoADGC 与两种对比方法生成的 UML 活动图在完整性、正确性和冗余性指标上的统计结果 (%)

数据集名称	完整性			正确性			冗余性		
	NLPHR[15]	ChatGPT4	AutoADGC	NLPHR[15]	ChatGPT4	AutoADGC	NLPHR[15]	ChatGPT4	AutoADGC
企业数据集	79.04	96.23	**95.99**	54.13	75.09	**96.09**	45.87	24.91	**3.91**
PROMISE	92.72	99.77	**98.65**	81.60	91.26	**99.53**	18.40	8.74	**0.47**

从表 4 可以看出, AutoADGC 与 ChatGPT4 自动生成所得 UML 活动图的完整性均超过 95%, 这表明上述两种方法都能够从需求文本中有效提取正确的活动图图元素及其关系. 在正确性和冗余性指标上, AutoADGC 的性能明显优于两种对比方法. 结合图 10 的实验结果进行分析, 这可能是因为 AutoADGC 方法借助语义多标签需求文本分类剔除了需求文本中与业务活动无关的内容, 并准确获取了正确的节点间关系, 进而减少了错误活动图节点和边的产生. 对比图 10(a) 与 (b) 中虚线方框部分结果发现, AutoADGC 方法自动生成的结果不包含“线上购物商场是一种便捷的购物方式”这一与业务活动非直接相关的需求文本句子. 对比图 10(a)、(b) 与 (c) 中双线方框结果发现, 只有 AutoADGC 方法正确表示了需求文本中的同步关系. 除此以外, 前文图 2(a) 中结果显示, 在没有受到充分的约束或指导时, 大模型可能会自主添加 UML 活动图的节点和边, 以满足其对活动图结构的理解, 而忽视实际业务需求. 这导致 ChatGPT4 生成的 UML 活动图中经常出现多余或错误的节点和边, 使其正确性较低, 冗余性较高. 虽然这种由大模型生成的 UML 活动图对业务活动的考虑可能比原始需求文本更全面, 但是过于复杂的 UML 活动图会增加软件开发人员理解、分析和维护的难度, 在实际项目开发中并不受欢迎. 同时, 在实验过程中, 由于大模型首次生成的 PlantUML 编码有误, ChatGPT4 在 51 份需求文本上未能成功生成 UML 活动图, 而 AutoADGC 在所有需求文本上均能成功生成 UML 活动图. 这凸显了 AutoADGC 在可靠性和稳定性方面的优势.

综上所述, 融合中文需求文本语义特征的 UML 活动图自动生成方法 AutoADGC 能够有效提升活动图自动生成的完整性和正确性, 降低活动图冗余性. AutoADGC 相对于其他 UML 活动图自动生成方法有较大优势, 具备较高的实用性.

3.5.3　运行时间结果与分析

为了回答 RQ3, 本节从 UML 活动图自动生成工具用户的角度对 AutoADGC 方法自动生成 UML 活动图的时间进行了评估. 鉴于训练语义多标签需求文本分类模型是一个一次性的过程, 且不涉及最终用户, 本实验主要关注 AutoADGC 方法在已训练好的模型上分析用户需求以自动生成 UML 活动图的过程. 参照 Wang 等人[41]的实验设计, 本文统计了第 3.1 节测试集中每份需求文本所包含的句子数量, 并按照句子数量对所有需求文本进行升序排序. 图 11 显示了 AutoADGC、NLPHR 与 ChatGPT4 根据测试集中的需求描述自动生成 UML 活动图的时间消耗. 鉴于 GKP4ACTIVITY、SBVR2ACTIVITY 与 EADGD 方法未公开实现代码且缺乏详细的算法描述, 本文无法对 AutoADGC 与这些方法的运行时间进行对比分析. 所有的运行时间均在配备 3.70 GHz Intel(R) Core(TM) i9-10900K CPU 和 128 GB 内存的云服务器上进行测量.

从图 11 可以看出, AutoADGC 方法能够根据测试集中的需求文本在 30 s 以内自动生成 UML 活动图, 运行时间远少于软件工程师人工绘制 UML 活动图的时间. 而图 7 的统计结果显示, AutoADGC 方法所生成的 UML 活动图在完整性、正确性和冗余性指标上同人工绘制的参考图相差较小. 此外, 在绝大多数情况下, AutoADGC 的性能明显优于 NLPHR 方法. 这是因为 AutoADGC 利用语义多标签需求文本分类的结果, 排除了需求文本中与 UML 活动图自动生成无关的句子, 提高了算法中自然语言处理与图元素及其关系提取的效率. 当需求文本的句子数量为 8 时, AutoADGC 方法的运行速度略慢于 NLPHR 方法. 通过查看对应需求文本发现, 这些需求文本几乎不包含

与 UML 活动图自动生成无关的句子, AutoADGC 中的分类器并未发挥筛除无关句的作用, 因此 AutoADGC 在这一情境下难以体现出优势. 得益于大模型强大的解析能力和计算能力, ChatGPT4 整体耗时较少, 在运行时间上更具优势. 然而, 表 4 的统计结果表明, 与 ChatGPT4 相比, 结合需求文本语义特征的 AutoADGC 方法在提高 UML 活动图的完整性和正确性、降低 UML 活动图的冗余性方面表现出更为优越的性能. 上述事实说明 AutoADGC 方法能够在秒级时间内为软件开发人员提供高质量的 UML 活动图草稿, 具备实用价值.

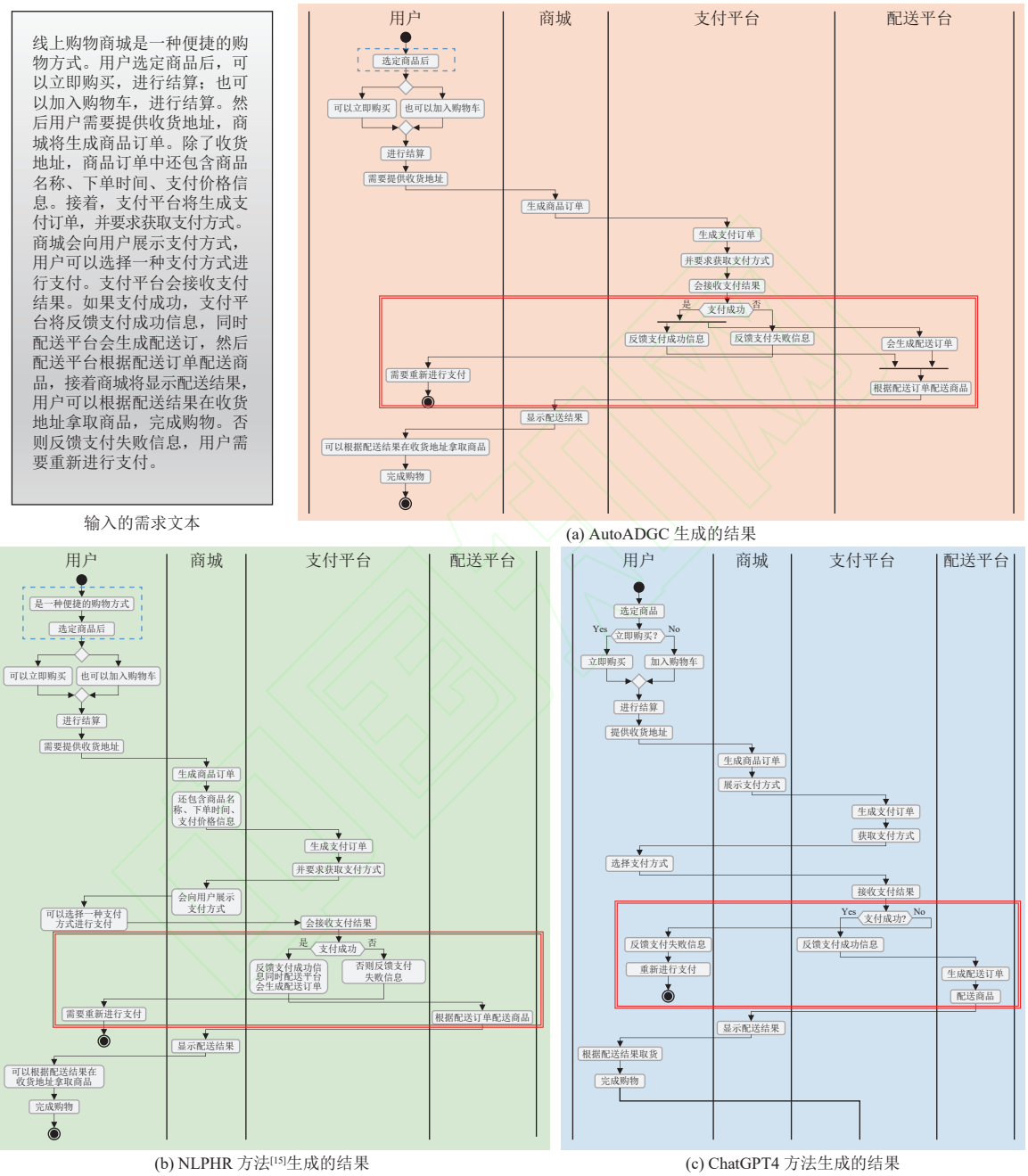


图 10　不同的 UML 活动图自动生成方法针对同一应用实例自动生成的 UML 活动图结果 2

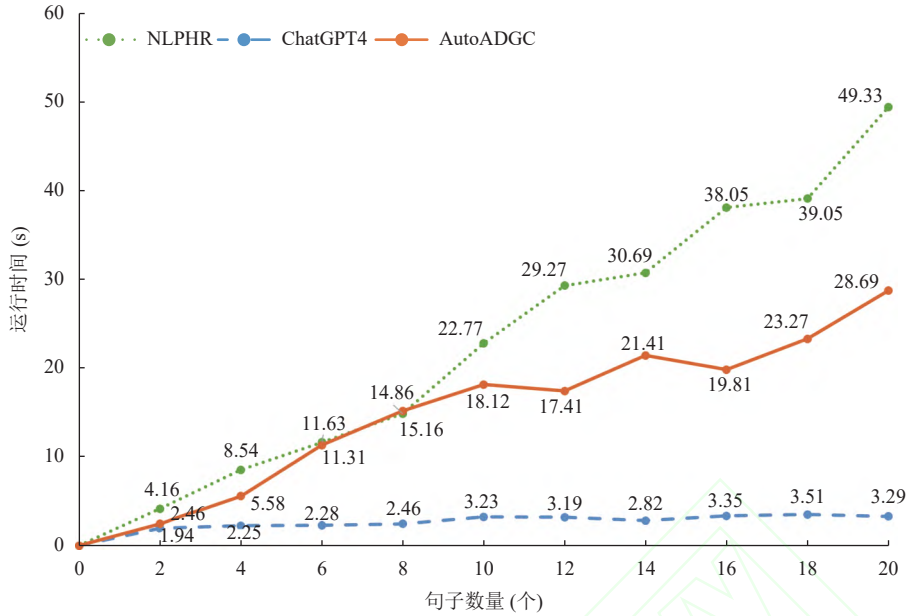


图 11　不同的 UML 活动图自动生成方法在测试集上的运行时间

4　有效性威胁

本节将简要讨论本文方法可能面临的内部有效性威胁、外部有效性威胁和结构有效性威胁. 针对这些有效性威胁, 我们提出了一系列应对措施以确保本文方法的严谨性与普遍性.

内部有效性主要关注研究过程中是否存在系统性错误. 本文方法的内部有效性威胁主要来源于需求文本数据集构建过程中可能产生的偏差. 为了降低这一风险, 我们聘请了 3 位未曾接触过本文技术方案的高级软件工程师, 要求他们独立对需求文本数据进行标注. 数据标注的具体过程已在第 3.1 节中详细说明. 其中, 采用弗莱斯卡帕系数[33]评估标注者之间的标注一致性, 从而减少标注过程中的主观偏差. 为了降低机器学习模型随机性带来的影响, 本文还采用十折交叉验证法优化模型参数. 此外, 由于在描述决策活动和并行活动时, 用户倾向于使用单个句子来描述分支中的一系列业务活动, 并以逗号分隔, 而非采用多个句子, 本文方法通过时序性语义标签确定句子间的节点次序, 利用关联词识别决策节点和分叉节点, 并以句号、分号或换行符作为分支活动的结束标志. 上述方法能够较好处理单句内包含多个节点或多句内包含多个非分支节点的情况, 但难以解决多句内包含分支节点时可能产生的分支终止节点确定问题. 因此, 拟在后续研究中引入多语句业务流分析模块, 建立跨句分支终止判定规则, 以应对在多语句描述情况下分支活动结束时的复杂情形.

外部有效性威胁通常与研究结果的可推广性和可扩展性相关. 在可推广性方面, 本文从多家合作企业搜集了覆盖 20 余个不同应用领域的需求文本数据以训练语义多标签需求文本分类模型. 鉴于数据来源的广泛性和第 3.5 节中实验结果的优越性, 相信本文方法在面对更广泛领域的多样化需求文本时仍能保持其有效性. 在可扩展性方面, 由于本文方法的算法架构设计采用与语言无关的语义表示技术, 且核心的图元素及其关系提取流程本质上独立于特定的语言结构, 因而可以兼容非中文语料处理. 在将其拓展至其他语言的应用过程中, 将涉及收集大规模的非中文需求文本以构建训练集, 并根据不同语言的语法特性对图元素及其关系提取的规则库进行相应的调整.

结构有效性涉及研究结论的合理性与可靠性. 为了最小化这种有效性威胁, 本文将 UML 活动图自动生成问题分解为需求文本语义特征提取、活动图图元素及其关系提取、活动图绘制这 3 个子问题. 这有助于针对性地突破各个子问题的难点. 此外, 本文方法从句子和短语两种粒度对需求文本进行解析, 从而更加精确地获取需求文本的语义特征和语法特征.

5 总 结

根据自然语言描述的软件需求文本自动生成 UML 活动图对软件开发人员正确把握客户需求、提高需求分析效率至关重要. 为进一步提高自动生成 UML 活动图的质量, 本文提出了一种融合中文需求文本语义特征的方法 AutoADGC. 该方法通过语义多标签需求文本分类将需求文本的相关性语义特征和时序性语义特征融入 UML 活动图的自动生成过程, 并结合需求文本的语义特征和语法特征提取 UML 活动图图元素及其关系. 在 100 个企业实际应用案例上的消融实验结果表明, 结合需求文本的语义特征设计 UML 活动图自动生成方法有助于提高自动生成活动图的完整性和正确性, 降低活动图的冗余性. 此外, 与多种 UML 活动图自动生成方法的比较验证了本文所提方法不仅可以在秒级时间内从需求文本中快速提取构建 UML 活动图所需的基本信息, 还可以有效避免需求文本中无关信息的干扰, 准确识别需求文本包含的时序关系, 使得自动生成 UML 活动图的正确性最高提升 41.96%, 显著提高了 UML 活动图自动生成方法的性能和效率.

虽然目前本文方法仅适用于中文需求文本, 但只要有充足的需求文本数据和全面的语法知识, 本文方法可以被扩展应用到任何语言的需求文本. 因此, 我们计划后续将 AutoADGC 方法用于分析英文软件需求文本. 此外, 本文仅对 UML 活动图自动生成问题进行了初步探讨, 未来可着眼于将该方法推广至自动生成与需求分析和软件设计相关的其他图表, 如 UML 用例图、数据库实体-关系图和用户界面等.

References

[1] Ahmad K, Abdelrazek M, Arora C, Bano M, Grundy J. Requirements engineering for artificial intelligence systems: A systematic mapping study. Information and Software Technology, 2023, 158: 107176. [doi: 10.1016/j.infsof.2023.107176]

[2] Li BY, Smidts C. A zone-based model for analysis of dependent failures in requirements inspection. IEEE Trans. on Software Engineering, 2023, 49(6): 3581–3598. [doi: 10.1109/TSE.2023.3266157]

[3] Ordoñez-Briceño K, Hilera JR, de-Marcos L, Otón-Tortosa S, Cueva-Carrión S. UML profile to model accessible Web pages. IEEE Access, 2024, 12: 77181–77213. [doi: 10.1109/ACCESS.2024.3406688]

[4] Larman C. Applying UML and Patterns: An Introduction to Object-oriented Analysis and Design and Iterative Development. Pearson, 2004.

[5] Sharma R, Gulia S, Biswas KK. Automated generation of activity and sequence diagrams from natural language requirements. In: Proc. of the 9th Int'l Conf. on Evaluation of Novel Approaches to Software Engineering. Lisbon: IEEE, 2014. 1–9.

[6] Nassif M, Robillard MP. Identifying concepts in software projects. IEEE Trans. on Software Engineering, 2023, 49(7): 3660–3674. [doi: 10.1109/TSE.2023.3265855]

[7] Gutiérrez JJ, Nebut C, Escalona MJ, Mejías M, Ramos IM. Visualization of use cases through automatically generated activity diagrams. In: Proc. of the 11th Int'l Conf. on Model Driven Engineering Languages and Systems. Toulouse: Springer, 2008. 83–96. [doi: 10.1007/978-3-540-87875-9_6]

[8] Yue T, Briand LC, Labiche Y. An automated approach to transform use cases into activity diagrams. In: Proc. of the 6th European Conf. on Modelling Foundations and Applications. Paris: Springer, 2010. 337–353. [doi: 10.1007/978-3-642-13595-8_26]

[9] Iqbal U, Bajwa IS. Generating UML activity diagram from SBVR rules. In: Proc. of the 6th Int'l Conf. on Innovative Computing Technology. Dublin: IEEE, 2016. 216–219. [doi: 10.1109/INTECH.2016.7845094]

[10] Yue T, Briand LC, Labiche Y. aToucan: An automated framework to derive uml analysis models from use case models. ACM Trans. on Software Engineering and Methodology (TOSEM), 2015, 24(3): 13. [doi: 10.1145/2699697]

[11] Franch X, Palomares C, Quer C, Chatzipetrou P, Gorschek T. The state-of-practice in requirements specification: An extended interview study at 12 companies. Requirements Engineering, 2023, 28(3): 377–409. [doi: 10.1007/s00766-023-00399-7]

[12] Kamarudin NJ, Sani NFM, Atan R. Automated transformation approach from user requirement to behavior design. Journal of Theoretical and Applied Information Technology, 2015, 81(1): 73–83.

[13] Gulia S, Choudhury T. An efficient automated design to generate UML diagram from natural language specifications. In: Proc. of the 6th Int'l Conf. Cloud System and Big Data Engineering. Noida: IEEE, 2016. 641–648. [doi: 10.1109/CONFLUENCE.2016.7508197]

[14] Alami N, Arman N, Khamyseh F. A semi-automated approach for generating sequence diagrams from Arabic user requirements using a natural language processing tool. In: Proc. of the 8th Int'l Conf. on Information Technology. Amman: IEEE, 2017. 309–314. [doi: 10.1109/ICITECH.2017.8080018]

[15] Maatuk AM, Abdelnabi EA. Generating UML use case and activity diagrams using NLP techniques and heuristics rules. In: Proc. of the 2021 Int'l Conf. on Data Science, E-learning and Information Systems. Ma'an: ACM, 2021. 271–277. [doi: 10.1145/3460620.3460768]

[16] Kumar DD, Sanyal R. Static UML model generator from analysis of requirements (SUGAR). In: Proc. of the 2008 Advanced Software Engineering and its Applications. Hainan: IEEE, 2008. 77–84. [doi: 10.1109/ASEA.2008.25]

[17] Li G, Peng X, Wang QX, Xie T, Jin Z, Wang J, Ma XX, Li XD. Challenges from LLMs as a natural language based human-machine collaborative tool for software development and evolution. Ruan Jian Xue Bao/Journal of Software, 2023, 34(10): 4601–4606 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/7008.htm [doi: 10.13328/j.cnki.jos.007008]

[18] SBVR Team. Semantics of business vocabulary and rules (SBVR). Technical Report dtc/06-03-02, Needham: Object Management Group, 2006.

[19] Manning C, Surdeanu M, Bauer J, Finkel J, Bethard S, McClosky D. The stanford CoreNLP natural language processing toolkit. In: Proc. of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations. Baltimore: ACL, 2014. 55–60.

[20] Habash N, Rambow O, Roth R. MADA+TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In: Proc. of the 2nd Int'l Conf. on Arabic Language Resources and Tools. Egypt, 2009. 242–245.

[21] Abdelnabi EA, Maatuk AM, Abdelaziz TM, Elakeili SM. Generating UML class diagram using NLP techniques and heuristic rules. In: Proc. of the 20th Int'l Conf. on Sciences and Techniques of Automatic Control and Computer Engineering. Monastir: IEEE, 2020. 277–282. [doi: 10.1109/STA50679.2020.9329301]

[22] Atadoga A, Umoga UJ, Lottu OA, Sodiy EO. Tools, techniques, and trends in sustainable software engineering: A critical review of current practices and future directions. World Journal of Advanced Engineering Technology and Sciences, 2024, 11(1): 231–239. [doi: 10.30574/wjaets.2024.11.1.0051]

[23] Bendraou R, Gervais MP, Blanc X. UML4SPM: A UML2.0-based metamodel for software process modelling. In: Proc. of the 8th Int'l Conf. on Model Driven Engineering Languages and Systems. Montego Bay: Springer, 2005. 17–38. [doi: 10.1007/11557432_3]

[24] Haga SW, Ma WM, Chao WS. Using the structure-behavior coalescence method to formalize the action flow semantics of UML 2.0 activity diagrams. Journal of Computing Science and Engineering, 2023, 17(2): 60–70. [doi: 10.5626/JCSE.2023.17.2.60]

[25] Arif M, Mohammad CW, Sadiq M. UML and NFR-framework based method for the analysis of the requirements of an information system. Int'l Journal of Information Technology, 2023, 15(1): 411–422. [doi: 10.1007/s41870-022-01112-7]

[26] Madampe K, Hoda R, Grundy J. A framework for emotion-oriented requirements change handling in agile software engineering. IEEE Trans. on Software Engineering, 2023, 49(5): 3325–3343. [doi: 10.1109/TSE.2023.3253145]

[27] Liang YH, Huang H, Cai ZQ, Hao ZF, Tan KC. Deep infrared pedestrian classification based on automatic image matting. Applied Soft Computing, 2019, 77: 484–496. [doi: 10.1016/j.asoc.2019.01.024]

[28] Lan ZZ, Chen MD, Goodman S, Gimpel K, Sharma P, Soricut R. ALBERT: A lite BERT for self-supervised learning of language representations. In: Proc. of the 8th Int'l Conf. on Learning Representations. Addis Ababa: OpenReview.net, 2020.

[29] Li FF, Su PZ, Duan JW, Zhang SC, Mao XL. Multi-label text classification with enhancing multi-granularity information relations. Ruan Jian Xue Bao/Journal of Software, 2023, 34(12): 5686–5703 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6802.htm [doi: 10.13328/j.cnki.jos.006802]

[30] Tan W, Nguyen ND, Du L, Buntine W. Harnessing the power of beta scoring in deep active learning for multi-label text classification. In: Proc. of the 38th AAAI Conf. on Artificial Intelligence. Vancouver: AAAI Press, 2024. 15240–15248. [doi: 10.1609/aaai.v38i14.29447]

[31] Ezzini S, Abualhaija S, Arora C, Sabetzadeh M. Automated handling of anaphoric ambiguity in requirements: A multi-solution study. In: Proc. of the 44th Int'l Conf. on Software Engineering. Pittsburgh: IEEE, 2022. 187–199. [doi: 10.1145/3510003.3510157]

[32] Cai L, Wang ST, Liu JH, Zhu YY. Survey of data annotation. Ruan Jian Xue Bao/Journal of Software, 2020, 31(2): 302–320 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5977.htm [doi: 10.13328/j.cnki.jos.005977]

[33] Fleiss JL. Measuring nominal scale agreement among many raters. Psychological Bulletin, 1971, 76(5): 378–382. [doi: 10.1037/h0031619]

[34] Landis JR, Koch GG. An application of hierarchical kappa-type statistics in the assessment of majority agreement among multiple observers. Biometrics, 1977, 33(2): 363–374. [doi: 10.2307/2529786]

[35] Pan C, Lu MY, Xu B, Gao HL. An improved CNN model for within-project software defect prediction. Applied Sciences, 2019, 9(10): 2138. [doi: 10.3390/app9102138]

[36] Xu QQ, Peng JJ, Zheng CZ, Tan SH, Yi F, Cheng F. Short text classification of Chinese with label information assisting. ACM Trans. on Asian and Low-resource Language Information Processing, 2023, 22(4): 119. [doi: 10.1145/3582301]

[37] Kingma DP, Ba J. Adam: A method for stochastic optimization. In: Proc. of the 3rd Int'l Conf. on Learning Representations. San Diego: OpenReview.net, 2015.

[38] Wolf T, Debut L, Sanh V, *et al*. Transformers: State-of-the-art natural language processing. In: Proc. of the 2020 Conf. on Empirical Methods in Natural Language Processing: System Demonstrations. ACL, 2020. 38–45. [doi: 10.18653/v1/2020.emnlp-demos.6]

[39] Chollet F. Building autoencoders in Keras. The Keras Blog, 2016. https://blog.keras.io/building-autoencoders-in-keras.html

[40] Abadi M, Agarwal A, Barham P, *et al*. TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv:1603.04467, 2016.

[41] Wang CH, Jin Z, Zhang W, Zowghi D, Zhao HY, Jiao WP. Activity diagram synthesis using labelled graphs and the genetic algorithm. Journal of Computer Science and Technology, 2021, 36(6): 1388–1406. [doi: 10.1007/s11390-020-0293-9]

附中文参考文献

[17] 李戈, 彭鑫, 王千祥, 谢涛, 金芝, 王戟, 马晓星, 李宣东. 大模型: 基于自然交互的人机协同软件开发与演化工具带来的挑战. 软件学报, 2023, 34(10): 4601–4606. http://www.jos.org.cn/1000-9825/7008.htm [doi: 10.13328/j.cnki.jos.007008]

[29] 李芳芳, 苏朴真, 段俊文, 张师超, 毛星亮. 多粒度信息关系增强的多标签文本分类. 软件学报, 2023, 34(12): 5686–5703. http://www.jos.org.cn/1000-9825/6802.htm [doi: 10.13328/j.cnki.jos.006802]

[32] 蔡莉, 王淑婷, 刘俊晖, 朱扬勇. 数据标注研究综述. 软件学报, 2020, 31(2): 302–320. http://www.jos.org.cn/1000-9825/5977.htm [doi: 10.13328/j.cnki.jos.005977]

作者简介

袁中锦, 博士生, 主要研究领域为智能化软件工程.

黄翰, 博士, 教授, CCF 杰出会员, 主要研究领域为微计算理论与方法, 智能化软件工程, 数据智能工程, 生化反应计算机.

向毅, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为智能化软件工程, 演化计算, 软件产品线.

刘方青, 博士, 助理研究员, CCF 专业会员, 主要研究领域为智能化软件工程, 路径覆盖测试, 演化计算.

郝志峰, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为代数学及其应用, 数据科学理论, 人工智能, 数学建模.