GITG:面向Gitee平台的issue标题自动生成方法

杨 君[1],刘诗凡[1],陈 翔[2],崔展齐[1*]
(1. 北京信息科技大学计算机学院,北京 100192;2. 南通大学信息科学技术学院,江苏南通 226019)

摘 要: 在开源软件和开源平台中,开发人员可以通过提交issue来记录所发现的软件错误或提出新功能需求.由于缺乏经验、专业水平有限等原因,用户可能无法对issue内容进行准确有效地总结,导致issue标题质量较低,进而降低issue的解决效率. 此外,现有的issue标题自动生成方法主要面向GitHub等英文开源平台,当应用在Gitee等国产开源平台时表现不佳. 同时,现有方法主要使用issue主体描述作为输入,忽略了issue中的代码片段等重要信息. 为此,本文提出一种面向Gitee平台的issue标题自动生成方法GITG(Gitee Issue Title Generation),针对包含中文和英文文本的issue,使用构建的Gitee issue数据集对支持中文的预训练模型Chinese BART(Bidirectional and Auto-Regressive Transformers)进行微调,利用issue主体描述和代码片段的双模态信息来自动生成issue标题. 为验证GITG的有效性,构建了包含18 242个Gitee issue样本的数据集. 实验结果表明,GITG在ROUGE-1、ROUGE-2和ROUGE-L指标上相较于iTAPE和iTiger分别至少提升了13.09%、10.18%和12.84%,在BLEU和METEOR指标上同样取得了性能提升. 人工评价结果表明,GITG生成标题的平均得分在整体分数、流畅性、信息性和简洁性4个评价指标上相较iTAPE和iTiger分别至少提升了26.7%、20.8%、24.2%和20.0%.

关键词: Gitee;issue标题;主体描述;代码片段;预训练模型;软件维护
基金项目: 江苏省前沿引领技术基础研究专项(No.BK20202001);北京信息科技大学"勤信人才"培育计划项目(No.QXTCPB202406)
中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112(2025)05-1559-12
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20240434

GITG: Automatic Issue Title Generation Method for Gitee Platform

YANG Jun[1], LIU Shi-fan[1], CHEN Xiang[2], CUI Zhan-qi[1*]
(1. *School of Computer Science*, *Beijing Information Science and Technology University*, *Beijing* 100192, *China*;
2. *School of Information Science and Technology*, *Nantong University*, *Nantong*, *Jiangsu* 226019, *China*)

Abstract: In open-source software and platforms, developers can submit issues to report software bugs or suggest new feature requests. Due to the lack of experience and limited professional skills, users may struggle to summarize the content of issues accurately and effectively, resulting in low-quality issue titles, which in turn decreases the efficiency of addressing issues. Additionally, existing automatic issue title generation methods are primarily designed for English open-source platforms, such as GitHub, and the performance are degraded when applied to Chinese open-source platforms, like Gitee. Furthermore, existing methods mainly use the issue body description as inputs, ignoring the code snippets in the issue. In this paper, we propose a method called GITG (Gitee Issue Title Generation) specifically designed for Gitee, an open-source platform. GITG addresses the challenge of generating issue titles for both Chinese and English text by fine-tuning the Chinese BART (Bidirectional and Auto-Regressive Transformers) pre-trained model on a constructed Gitee issue dataset. It leverages the bi-modal information from the issue body description and code snippets to generate informative and accurate issue titles. A dataset consisting of 18 242 Gitee issue samples is constructed to validate the effectiveness of GITG. Experimental results demonstrate that GITG outperforms iTAPE and iTiger by at least 13.09%, 10.18%, and 12.84% on the ROUGE-1, ROUGE-2, and ROUGE-L metrics, respectively. GITG also achieves improvements in BLEU and METEOR metrics. Human evaluation results also indicate that the average scores of the titles generated by GITG are improved by at least 26.7%, 20.8%, 24.2%, and 20.0% in overall score, fluency, informativeness, and conciseness, respectively, compared to iTAPE and iTiger.

Key words: Gitee; issue title; body description; code snippet; pre-trained models; software maintenance

收稿日期:2024-05-10;修回日期:2025-02-21;责任编辑:李勇锋
*通讯作者:崔展齐

Foundation Item(s): The Leading-Edge Technology Program of Jiangsu Natural Science Foundation (No.BK20202001); Beijing Information Science and Technology University "Qin-Xin Talent" Cultivation Project (No.QXTCPB202406)

1 引言

在软件开发和维护过程中,开发人员通过缺陷跟踪系统中的缺陷报告来记录软件故障信息或提出新功能需求,帮助对软件进行修正性或完善性维护.然而,由于受到进度计划、专业知识等限制,缺陷报告的质量参差不齐,部分缺陷报告未能提供充分信息,甚至会提供误导性信息.研究表明,低质量的缺陷报告会使得软件维护人员难以准确获取有效信息[1],从而增加软件维护的时间开销,或导致缺陷报告得不到开发人员应有的关注.在开源平台中,用户和开发人员通过issue来记录软件缺陷、提出功能建议、报告文档缺失等,发挥着与缺陷报告相同的作用[2].为提高issue质量,常采用提供模板的方式,要求用户在提交issue时必须严格按照指定模板编写,例如Mozilla提供了issue指南供用户遵循(https://bugzilla.mozilla.org/page.cgi?id=bug-writing.html).在普遍采用的准则和模板中,标题都是必填的重要字段,应简短准确地概括issue的主要内容.

高质量标题可帮助软件开发和维护人员在阅读正文描述之前快速理解issue核心内容,从而提高issue解决效率,降低维护开销[3].在以列表视图的方式呈现issue信息时,标题质量尤为重要.然而,由于缺乏经验、专业水平有限等原因,用户可能难以对issue进行简洁准确地总结,从而导致标题质量偏低.

为解决上述问题,有研究尝试通过自动生成issue标题的方式来提高标题质量,待开发人员写完issue正文后再使用此类工具来生成高质量的issue标题.Chen等人[4]将issue标题生成转化为一句话摘要任务,并提出iTAPE方法,采用基于RNN(Rerrent Neural Network)的序列到序列框架,使用低频标记处理方法并结合复制机制来缓解集外词(Out Of Vocabulary, OOV)问题.Lin等人[5]基于对iTAPE生成标题质量的分析,提出了基于质量预测的过滤器TitleGen-FL,该方法会根据深度学习模块和信息检索模块的分析结果来判断iTAPE是否会生成高质量标题,并对iTAPE生成的低质量标题进行过滤,提高了iTAPE的实用性.Zhang等人[6]提出了基于预训练模型的issue标题生成方法iTiger,对BART(Bidirectional and Auto-Regressive Transformers)模型进行微调,结果表明该方法取得了目前最好的性能.

然而,此前的issue标题自动生成方法主要面向GitHub平台,在生成标题时主要针对英文文本.近年来,开源环境的发展建设受到我国政府高度重视[7],国产开源软件及开源平台的普及有力地推动了我国软件产业的快速发展.作为国产开源平台的代表,截至2024年12月,Gitee平台用户数量超1 350万,开源项目数量超3 600万.我们从Gitee上随机选取了10 000个issue,发现超79%的issue中包含中文,且同样存在issue标题质量较低问题.此外,iTAPE、iTiger等方法只使用issue主体描述来生成标题,忽略了代码片段中的信息,而代码片段中包含着存在bug的代码等信息,有助于提升所生成标题的质量.参照Zhang等人[8]和Zhou等人[9]的工作,我们将issue主体描述和代码片段称为双模态信息.为此,我们提出了面向Gitee平台的issue标题生成方法GITG(Gitee Issue Title Generation),针对包含中文和英文文本的issue,使用双模态信息来生成标题.首先,进行数据处理,将issue主体描述和代码片段使用特殊的连接符进行拼接,对中文进行分词,在标识符前后插入额外标记,并输入模型.然后,使用迁移学习策略对支持中文的预训练模型Chinese BART[10]进行微调,以构建适用于本任务的模型.最后,将待生成标题样本中的issue主体描述和代码片段以同样方式拼接并输入到训练后的模型中,以生成标题.为验证GITG的有效性,我们构建了Gitee issue基准数据集,并与iTAPE和iTiger进行对比.实验结果表明,GITG在ROUGE指标上相较于iTAPE和iTiger至少提升了13.09%和10.18%,在BLEU和METEOR指标上同样取得了类似的性能提升.此外,我们还进行了人工评价.结果表明,GITG生成标题的平均得分在整体分数、流畅性、信息性和简洁性4个评价指标上相较iTAPE和iTiger至少分别提升了26.7%、20.8%、24.2%和20.0%,进一步验证了GITG的有效性.

本文研究的主要贡献如下:(1)提出了一种面向Gitee平台的issue标题自动生成方法GITG,针对包含中文和英文文本的issue,使用主体描述和代码片段来生成issue标题.(2)收集了Gitee平台中的issue样本并进行过滤,构建了一个包含18 242个样本的基准数据集,并在GitHub(https://github.com/YZS-web/Gitee_Issue_Title_Generation)和Gitee(https://gitee.com/YZS_yang/Gitee_Issue_Title_Generation)上进行发布,为后续研究提供帮助.(3)为验证GITG的有效性,在所构建的数据集上进行了实验,从ROUGE、BLEU等方面评估了GITG生成标题的质量,并进行了人工评价.

2 研究动机示例

现有研究主要针对GitHub平台的issue,通常仅在英文数据集上进行训练,而Gitee平台上的大部分issue

都包含中文,现有方法在生成 Gitee 平台的 issue 标题时可能表现不佳. 例如,表 1 为 Gitee 平台上 Mybatis-Plus 项目中的 issue(https://gitee.com/baomidou/mybatis-plus/issues/I28UC7). 其中,Issue Body 是 issue 的主体描述;Issue Code 是 issue 的代码片段;Reference 是开发人员人工编写的标题,将作为参考标题;iTAPE 和 iTiger 分别为使用 2 种方法直接生成的标题. 从图 1 可以看出,iTAPE 和 iTiger 均没有生成较为准确的标题.

Issue Body:
当前使用版本 3.4.1
这里执行的 sql 是 select * from user where name='jack'我想实现给表起别名,select * from user u where u.name='jack' 这个怎么实现
Issue Code:
LambdaQueryWrapper<User>lqw = new Wrappers.lambdaQuery(User.class);
lqw.eq(User::getName,"jack");
Reference: LambdaQueryWrapper 怎么给表起别名
iTAPE:sql 是 select from user where name = '
iTiger:jekyll 3.4.1 doesn't work as expected

图 1　issue 标题自动生成示例

以 iTAPE 方法为例,由于 iTAPE 仅在英文数据集上进行训练,缺乏对中文数据的学习,因此未能生成高质量标题. 在此示例中,iTAPE 选择 issue 主体描述中的部分信息作为标题,没有捕捉到"给表起别名"这一关键信息,而 iTiger 则是生成了毫不相关的标题.

此外,iTAPE 和 iTiger 只使用 issue 主体描述来生成标题,忽略了代码片段的重要性. 在该示例中,参考标题中的关键信息"LambdaQueryWrapper"仅存在于代码片段中,无法在 issue 主体描述中找到,因此 iTAPE 和 iTiger 均未能捕捉到这一关键信息. 虽然 issue 主体描述可以为 issue 标题生成任务提供有价值的信息,但所提供的信息有时候可能并不完整,而代码片段可以提供更多信息.

为此,本文提出了一种融合双模态信息的 issue 标题生成方法,通过结合 issue 主体描述和代码片段,可以为模型提供更充分的信息,提高生成标题的质量. 此外,我们使用构建的 Gitee issue 数据集对支持中文的预训练模型进行微调,与仅在英文数据集上训练的 iTAPE 和 iTiger 相比,可以为 Gitee 平台上的 issue 生成更准确的标题.

3　面向 Gitee 平台的 issue 标题自动生成方法 GITG

本节将介绍面向 Gitee 平台的 issue 标题自动生成方法 GITG. 图 2 为 GITG 的方法框架图,主要由 3 个阶段组成,分别是数据准备阶段、模型训练阶段和标题生成阶段. 在数据准备阶段,应用启发式规则构建 Gitee issue 数据集,并将样本中的 issue 主体描述和代码片段进行拼接,以表示双模态信息. 在模型训练阶段,应用预训练模型 Chinese BART,并使用 Gitee issue 数据集对预训练模型进行微调. 在标题生成阶段,对于待生成标题的 issue,将其主体描述和代码片段输入到微调后的模型中,以生成 issue 标题. 下面将分别对上述阶段进行详细介绍.

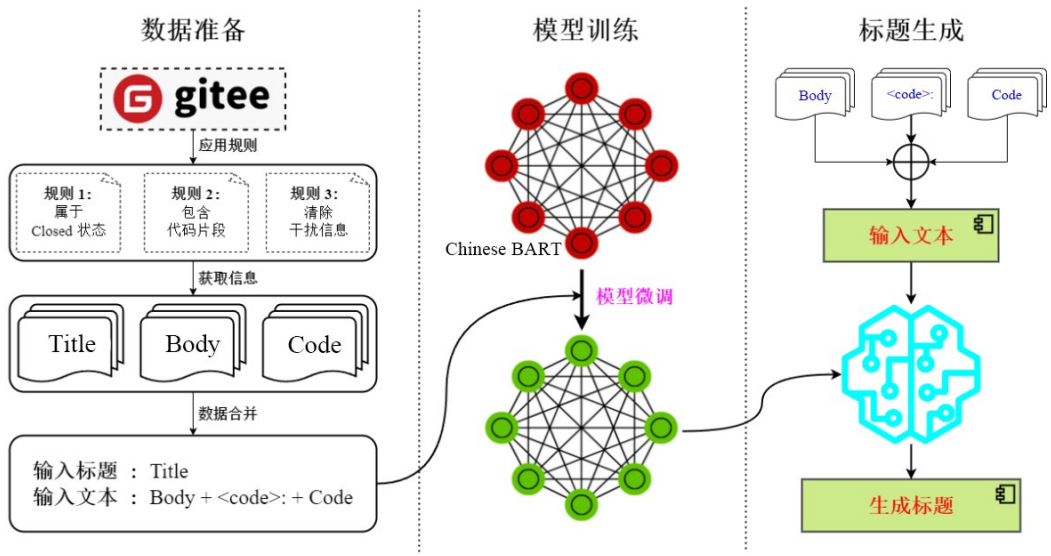

图 2　面向 Gitee 平台的 issue 标题生成方法 GITG 框架

3.1　数据准备阶段

数据准备阶段对收集的数据进行处理以构建模型的输入序列. 我们将 Gitee 平台中的 issue 样本以三元组"<Title, Body, Code>"的形式进行表示,其中 Title、Body、Code 分别表示参考标题、主体描述和代码片段. 在 GITG 中,使用 Body 和 Code 来自动生成 issue 标题,并

将Title用作参考标题. 为了更好地融合双模态信息,使用特殊的连接符“<code>:”将双模态信息进行拼接以构建issue描述序列Text,该序列的输入格式如式(1)所示:

$$\text{Text}=\text{Body}\oplus<\text{code}>:\oplus\text{Code} \tag{1}$$

其中,⊕表示将前后内容进行拼接.

此外,根据Chen等人[4]的研究,在标识符和版本号前后插入额外标记可以辅助提升所生成标题的质量. 因此,GITG沿用了上述方法,在标识符(如方法名或变量名)前后插入额外的标记“<I>”和“</I>”. 此外,issue中的中文在输入到模型前会进行分词处理,帮助模型更好地理解中文文本.

为规范化文本,issue描述序列中的所有换行符将被删除,多个连续空格仅保留一个,再作为输入发送给模型. 对于参考标题,直接将其进行分词以构建参考标题序列,与经过上述处理后的issue描述序列一同输入到模型中. 如图3所示的示例中,前2行是原始的主体描述和代码片段,第3行是经过上述处理后的数据.

Issue Body Before Data Prepare:

当前使用版本 3.4.1

这里执行的sql是select * from user where name='jack'我想实现给表起别名,select * from user u where u.name='jack' 这个怎么实现

Issue Code Before Data Prepare:

```
LambdaQueryWrapper<User>lqw = new Wrappers.lambdaQuery(User.class);
lqw.eq(User::getName,"jack");
```

Input Text After Data Prepare:

当前 使用 版本 3.4.1 这里 执行 的 sql 是 select * from user where name = ' jack ' 我 想 实现 给表起 别名, select * from user u where u . name = ' jack ' 这个 怎么 实现<code:><I>LambdaQueryWrapper</I>

< User >lqw = new Wrappers . <I>lambdaQuery</I> (User . class) ;

lqw . eq (User : : <I>getName</I> , " jack ") ;

图3 数据准备示例

3.2 模型训练阶段

预训练模型在大量数据集上进行了训练,具备一定的先验知识和学习能力,在自然语言处理、计算机视觉等多个领域取得了极佳效果,可以通过迁移学习等策略来很好地解决下游任务,如代码注释生成[11]、视频问答[12]、行人重识别[13]. GITG选择基于Transformer架构的预训练模型Chinese BART作为基础模型,该模型使用了双向编码器和自回归解码器,是BART模型的中文版本. BART的原始版本仅在英文数据集上进行了训练,缺少对中文数据的先验知识. Chinese BART在大量中文数据集上进行了训练,可以更好地适应本文研究的场景.

进行模型训练时,GITG首先将issue描述序列和参考标题序列进行向量化表示,之后将向量化后的issue描述序列输入到编码器,将向量化后的参考标题序列作为参考答案输入到解码器. 编码器由若干层组成,每一层都包括多头自注意力机制和前馈神经网络(Feed Forward Network, FFN). 多头自注意力机制建立在自注意力机制的基础上,在计算时使用多个独立的注意力头,每个注意力头都可以专注于不同的子空间,然后对多个注意力头的计算结果进行整合,得到最终结果. 通过这种方式,模型能够有效地捕捉到输入序列中不同位置的依赖关系和重要信息. 自注意力通过查询(Query)、键(Key)和值(Value)进行计算,计算方法如式(2)所示:

$$\text{Attention}\left(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}\right)=\text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{2}$$

其中,$\boldsymbol{Q}$、$\boldsymbol{K}$、$\boldsymbol{V}$分别是Query向量、Key向量和Value向量,d_k是$\boldsymbol{Q}$和$\boldsymbol{K}$的向量维度. 多头自注意力的计算公式如式(3)所示,其中,$\boldsymbol{W}^O$是可学习的参数矩阵,head_i是第i个注意力头的注意力计算结果,n是注意力头的数量.

$$\begin{aligned}&\text{MultiHead}\left(\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}\right)\\&=\text{Concat}\left(\text{head}_1,\text{head}_2,\cdots,\text{head}_i,\cdots,\text{head}_n\right)\boldsymbol{W}^O\end{aligned} \tag{3}$$

FFN由2层线性变换组成,在2层之间使用了GELU激活函数,参数初始化服从$N(0,\ 0.02)$的正态分布. 通过这种方式,FFN对多头自注意力的输出结果进行非线性变换,将经过多头注意力机制得到的向量映射到新的特征空间中,以更好地捕捉向量特征. 此外,在每个多头自注意力层和FFN之间都有一个残差连接和层归一化操作,帮助模型更好地捕捉输入序列中的信息,缓解梯度消失问题并加速模型收敛.

解码器的结构与编码器类似,不同之处在于解码器中加入了带掩蔽的多头注意力子层. 解码过程一般为顺序操作过程,当前时刻的输出应取决于先前时刻的输出,因此在解码器中,自注意力机制需要进行掩码操作,以避免模型在生成序列时看到未来时刻的信息. 此外,解码器中还包含编码器-解码器注意力子层,用于建立编码器和解码器之间的交互,解码器需要同时考虑编码器的输出和当前时刻的解码器状态,帮助解码器在生成序列时更好地理解全局上下文信息.

GITG基于预训练的Chinese BART模型进行微调,在微调过程中模型参数被初始化为Chinese BART模型的参数,再使用构建的Gitee issue数据集对模型进行重新训练. 通过这种方式,可以利用预训练模型已经学习到的特征和知识,能够更快地获取适用于目标任务的模型,提高训练效率.

3.3 标题生成阶段

对于待生成标题的issue样本,将转化为issue描述

序列,并输入到微调后的模型中.GITG中的解码器会从左至右生成token,生成的token序列即构成issue标题.在标题生成过程中,GITG使用集束搜索方法,此方法在Stack Overflow问题标题重构[14]、提交消息生成[15]、代码注释更新[16]等序列生成任务中被广泛应用.集束搜索使用广度优先搜索来构建搜索树,树的每一层会生成当前层状态的所有后继状态,依据启发式规则对这些后继状态的条件概率进行降序排列,概率越大则表明当前状态更好,之后保留最优的前k个后继状态并舍弃其他后继,不断扩展直到序列生成结束.通过这种方式,可以减小算法的搜索空间,降低时空复杂度,但不能保证得到全局最优解.

图4展示了使用集束搜索方法为表1示例生成issue标题的过程,在构建搜索树时每一层会保留最优的k个候选状态.在树的每一层,集束搜索为最终的标题添加一个token,直到生成结束.

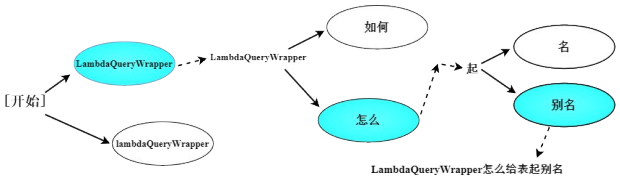

图4　集束搜索示例

4　实验设计

在本节中,将介绍GITG使用的数据集、评价指标、基线方法与实现细节.

4.1　数据集

此前的研究中缺乏Gitee平台的数据集供本文实验使用.因此,我们使用Gitee平台的项目构建了基准数据集.首先按照Gitee项目的stars进行排序,选择前400个代码仓库,从中收集了116 398个issue,并将开发人员人工编写的标题视为参考标题,将由"<code></code>"标记包装的内容视为代码片段,其余内容视为主体描述.为提升数据集中样本的质量,使用以下3条启发式规则对样本进行过滤,最终获得了18 242个issue样本.

规则1:数据集的issue应属于closed状态.

根据之前关于Stack Overflow社区研究的建议[17],带有公认答案的社区帖子通常具有更高的质量,更加受到用户的欢迎.因此,我们认为被项目贡献者所接受的,即处于closed状态的issue具有更高的质量.

规则2:数据集的issue应同时包含主体描述和代码片段.

在此前研究中,只使用issue主体描述来生成标题,这可能会导致生成的标题中缺乏部分重要信息,这些信息可能存在于代码片段中.与先前研究不同,GITG使用双模态信息,即同时使用主体描述和代码片段,这可以为issue标题自动生成任务提供更充分的信息.

规则3:issue中的干扰标记都应被清除.

收集的issue样本以统一的HTML格式进行组织,部分样本中可能包含如"<a href='xxx'/>"的无效标记.同时,issue样本中可能包含图像或视频等多媒体信息,虽然这些信息在实际中发挥着重要作用,可以提供更多的重要信息,但它们为模型的处理带来了新的挑战,本文未考虑此类数据.

然后,随机打乱数据集,按照8:1:1的比例将数据集划分为训练集、验证集和测试集.最终,训练集、验证集、测试集分别由14 595、1 825、1 822个样本构成.

4.2　评价指标

为评估生成标题的质量,我们选取了ROUGE[18]、BLEU[19]和METEOR[20]等自动评价指标,这些指标在之前研究中被广泛应用[21].

ROUGE用于衡量机器生成的摘要质量,本文选择使用ROUGE-N(N=1,2)和ROUGE-L指标.其中,ROUGE-N以n元短语(N-gram)为单元,计算生成标题和参考标题之间N-gram的重合度.与ROUGE-N不同,ROUGE-L考虑句子层面的结构相似性,通过计算生成标题和参考标题之间的最长公共子序列长度来评估生成标题的质量.

BLEU用于评估机器翻译方法的性能,本文选择使用BLEU-N(N=1,2,3,4)指标.BLEU指标更偏重于精确率,通过比较生成标题和参考标题之间n元短语的重合程度来评估生成标题的质量.此外,BLEU在计算时引入了惩罚因子的概念,以缓解BLEU分数的偏向性.

METEOR用于评估机器翻译结果与参考答案之间的相似性,使用单精度的加权调和平均数和单字召回率来评估生成标题与参考标题之间的相似性,并设立了基于词序变化的惩罚机制,还额外考虑了同义词匹配、词形变化、词干匹配等因素,与BLEU相比能够更准确地评估生成标题的质量.

实验使用Rouge库(https://github.com/pltrdy/rouge)计算ROUGE,并使用NLTK库(https://github.com/nltk/nltk)计算BLEU和METEOR.此外,由于生成标题和参考标题可能同时包含中文和英文,因此在计算时采用单词级粒度,即先将中文进行分词处理,将每个单词视为一个单元.

4.3　基线方法

为验证GITG的有效性,本文选择iTAPE[4]和iTiger[6]作为比较对象.其中,iTAPE是第一个研究issue标题自动生成的工作,其基于序列到序列框架,结合复制机制和低频标记处理方法来提高生成标题的质量.iTiger是基于预训练模型的issue标题生成方法,对已使

用英文数据集预训练的BART模型进行微调,以缓解缺少大规模高质量训练数据的问题,实验结果表明iTiger在多项评价指标上超过iTAPE,取得目前的最佳表现.

为了进行公平比较,我们使用构建的Gitee issue基准数据集在iTAPE和iTiger上进行了重新训练. 在训练过程中,我们使用iTAPE和iTiger提供的开源代码并沿用其论文中的超参数设置. 对于双模态信息,我们以相同方式构建issue描述序列,并输入上述模型.

4.4 实现细节

本文使用深度学习框架PyTorch(https://pytorch.org/)和Transformers(https://github.com/huggingface/transformers)来实现GITG. 其中,我们微调了预训练语言模型Chinese BART,该模型有6层编码器和6层解码器,每层包含768个隐藏单元,该模型可以从HuggingFace(https://huggingface.co/fnlp/bart-base-chinese)中获取,并使用Adam优化器来优化模型参数,初始学习率设置为3×10^{-5}. 在训练期间,批量大小设置为4. 所有实验都在配备32 GB RAM、AMD Ryzen 5700X 8-Core Processor和NVIDIA GeForce 3060 GPU的计算机上完成.

5 实验结果与分析

为评估所提出issue标题自动生成方法GITG的有效性,我们提出了以下四个研究问题.

研究问题1:与其他issue标题生成方法相比,GITG生成的标题质量如何?

动机:为分析GITG与其他基准方法相比表现如何,即GITG在issue标题生成的效果上是否具有优势,在这个研究问题中,我们选择iTAPE和iTiger作为对比对象,并使用ROUGE等8项自动评价指标来评估这些方法生成标题的质量.

研究问题2:使用双模态信息时,即同时使用issue主体描述和代码片段是否有助于提升GITG生成issue标题的效果?

动机:GITG使用issue主体描述和代码片段的双模态信息来生成标题,在这个研究问题中,我们进行了消融实验,以研究主体描述和代码片段是否有助于提升issue标题生成效果.

研究问题3:使用标记插入方法和微调策略是否会提升GITG的性能?

动机:GITG使用标记插入方法和微调策略以提升issue标题生成效果,在这个研究问题中,我们进行了消融实验,以研究标记插入方法和微调策略是否有助于提升GITG的表现.

研究问题4:与其他issue标题生成方法相比,GITG生成的标题在人工评价方面表现如何?

动机:自动评价指标着重于评价生成标题和参考标题的文本相似度,忽略了语义层面的相似性,仅使用自动评价指标难以准确地评估标题质量. 在这个研究问题中,我们对GITG和基线方法生成的标题质量进行了人工评价,以对比它们的性能.

5.1 研究问题1

为回答这个问题,我们在构建的Gitee issue数据集上与iTAPE和iTiger进行了对比实验. 在实验过程中,我们将数据集的每个样本进行分词处理,将分词后的issue文本序列输入到iTAPE和iTiger中进行训练,实验结果如表1所示.

从表1中可以看出,与基准方法iTAPE和iTiger相比,GITG在ROUGE和METEOR指标上取得了最佳的效果,与iTAPE相比,ROUGE-1、ROUGE-2、ROUGE-L和METEOR分别提升了13.09%、29.56%、13.60%和19.52%,与iTiger相比分别提升了13.91%、10.18%、12.84%和19.10%. 在BLEU指标上,GITG的BLEU-1指标相较于iTiger提升了17.54%,相较于iTAPE虽然有所下降,但下降幅度较低,仅下降了1.38%. 分析发现,虽然BLEU-1和ROUGE-1都考虑了生成标题和参考标题在1-gram上的重叠程度,但BLEU-1在计算时引入的长度惩罚因子BP(Brevity Penalty)会降低短标题的分数. 经过统计,iTAPE生成标题的平均长度为12.26个单词,而GITG生成标题的平均长度只有8.06个单词,这导致GITG在BLEU-1上的分数较低,相较iTAPE有所下降. 在BLEU-2、BLEU-3和BLEU-4这3个指标上,GITG较iTAPE分别提升了5.41%、10.14%和14.40%,较iTiger分别提升了20.71%、20.56%和19.07%.

表1 不同issue标题生成方法性能对比

方法名	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
iTAPE	27.51	15.46	26.54	**39.75**	29.97	25.34	21.67	22.54
iTiger	27.31	18.18	26.72	33.35	26.17	23.15	20.82	22.62
GITG	**31.11**	**20.03**	**30.15**	39.20	**31.59**	**27.91**	**24.79**	**26.94**

注:加粗数据为最优结果.

此外,ROUGE指标在计算时重点关注文本的局部相似度,如短语的连贯性、内容的一致性等;而BLEU指标在计算时更关注文本的全局相似度,如句子结构、用词准确性等. 在大多情况下,GITG生成标题的ROUGE和BLEU指标高于iTAPE和iTiger,因此无论是文本的局部连贯性还是全局准确性,都取得了超过基准方法

的性能.

图5为使用不同方法为图1示例生成issue标题的结果. 表中Reference为参考标题,其他行则对应不同方法的生成结果. 与图1不同,其中iTAPE和iTiger对应行是使用Gitee issue数据集重新训练模型后生成的issue标题. 从图5中可以看出,iTAPE捕捉到了关键信息“LambdaQueryWrapper”和“别名”,但生成的标题中包含“进行select查询”等额外的无价值信息,没有反映issue的核心意图,且生成的语句不完整. 另外,iTiger仅捕捉到了关键信息“LambdaQueryWrapper”,错误地将主体描述中的部分语句“这里执行的sql”作为标题,且生成的标题不通顺. 相比之下,GITG准确捕捉到了issue的核心意图,生成的标题与参考标题完全一致,表明了GITG的有效性.

Issue Body:
当前使用版本 3.4.1
这里执行的sql是select * from user where name='jack'我想实现给表起别名,select * from user u where u.name='jack' 这个怎么实现

Issue Code:
LambdaQueryWrapper<User>lqw = new Wrappers.lambdaQuery(User.class);
lqw.eq(User::getName,"jack");

Reference: LambdaQueryWrapper怎么给表起别名

iTAPE: LambdaQueryWrapper别名不起作用,但是在进行select查询

iTiger: LambdaQueryWrapper这里执行的sql

GITG: LambdaQueryWrapper怎么给表起别名

图5　不同方法为表1示例生成的issue标题结果

对研究问题1的回答:在ROUGE和METEOR指标上,GITG的性能优于iTAPE和iTiger,相比基线方法至少提升了10.18%和19.10%. 在BLEU指标上,GITG除BLEU-1略低于iTiger外,其余指标均表现更优. 总体来说,GITG生成issue标题的表现优于其他对比方法.

5.2　研究问题2

为验证使用双模态信息生成issue标题的有效性,实验分析了另外2种情况来量化不同输入模式对GITG的贡献,即仅使用issue主体描述的输入模式和仅使用代码片段的输入模式,并采用不同的下标来区分上述输入模式. 其中,$GITG_{body}$表示仅使用主体描述作为输入,$GITG_{code}$表示仅使用代码片段作为输入. 表2显示了GITG在不同输入模式下的性能.

从表2中可以看出,当同时使用主体描述和代码片段作为输入时,GITG取得了最好的效果. 具体而言,当使用双模态信息作为输入时,GITG在ROUGE指标上的分数分别为31.11、20.03和30.15. 与只使用主体描述作为输入相比,GITG的ROUGE分数分别提升了3.94%、4.38%和3.82%;与只使用代码片段作为输入相比,GITG的ROUGE分数分别提升了33.75%、41.16%和33.35%. 在BLEU指标上,GITG与只使用主体描述的变体$GITG_{body}$相比分别提升了0.87%、2.53%、3.18%和3.64%,与只使用代码片段的变体$GITG_{code}$相比分别提升了12.13%、23.01%、29.33%和33.42%. 在METEOR指标上,GITG相较$GITG_{body}$和$GITG_{code}$这两个变体分别提升了3.10%和34.58%. 实验结果表明,issue主体描述和代码片段中的信息是互补的,在生成issue标题时应同时使用这两种信息. 此外,GITG在只使用主体描述作为输入时的性能要高于只使用代码片段作为输入时的性能,这表明在issue标题自动生成任务中,issue主体描述相较于代码片段具有更充分、更有价值的信息.

对研究问题2的回答:当使用双模态信息作为输入时,与只使用主体描述作为输入相比,GITG在8项指标上分别提升了0.87%~4.38%;与只使用代码片段作为输入相比,GITG在8项指标上分别提升了12.13%~41.16%. 此外,issue主体描述可以为issue标题生成任务提供比代码片段更有价值的信息.

表2　不同输入模式对GITG的性能影响

方法名	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
$GITG_{body}$	29.93	19.19	29.04	38.86	30.81	27.05	23.92	26.16
$GITG_{code}$	23.26	14.19	22.61	34.96	25.68	21.58	18.58	20.04
GITG	**31.11**	**20.03**	**30.15**	**39.20**	**31.59**	**27.91**	**24.79**	**26.97**

注:加粗数据为最优结果.

5.3　研究问题3

为回答这个问题,我们将GITG与GITG w/o insert和GITG w/o fine-tune这2个变体进行了一项对比实验. 其中,GITG w/o insert是指在数据中不插入额外标记,其余与GITG完全一致;GITG w/o fine-tune是指不使用微调策略而是直接使用预训练模型Chinese BART来生成标题,其余与GITG完全一致. 实验结果如表3所示.

从表3中可以看出,相较于GITG w/o fine-tune变体,GITG的性能大幅上升,这表明了微调策略对GITG的有效性. 具体来说,与未使用微调策略的变体相比,GITG在3项ROUGE指标上分别提升了111.92%、222.54%和117.53%,在4项BLEU指标上分别提升了

19.15%、53.50%、75.53%和97.22%，在METEOR指标上提升了111.13%. 这是因为，GITG w/o fine-tune变体没有在本任务的数据集上进行训练，而是直接使用原有的预训练模型Chinese BART生成标题. 虽然Chinese BART已经在大量数据集上进行了预训练，但未针对issue标题生成任务的数据进行学习，不具备该领域的先验知识，难以生成更高质量的标题.

相较于GITG w/o insert变体，在全部8项评价指标上，GITG有5项指标分数更高，这表明标记插入方法可以在一定程度上提升GITG的性能. 对于ROUGE-2和BLEU-4指标，GITG略低于GITG w/o insert变体. 分析发现，GITG w/o insert变体相较于GITG更有可能生成与参考标题具有较长连续词汇重叠的标题，这使得ROUGE-2和BLEU-4的值偏高. 对于METEOR指标，GITG也略低于GITG w/o insert变体. 分析发现，相较于GITG生成的标题，GITG w/o insert变体生成的标题更有可能与参考标题之间具有更多空间上能够对齐的单词，这使得GITG w/o insert变体的METEOR指标偏高. 总体上看，GITG的性能要优于GITG w/o insert变体.

对研究问题3的回答：预训练模型微调策略可以显著提升GITG的性能，相较于不使用微调策略的GITG w/o fine-tune变体，GITG在8项指标上分别提升了19.15%~222.54%. 当使用标记插入方法后，GITG的总体性能略有提升.

表3 标记插入方法和微调策略对GITG的性能影响

方法名	ROUGE-1	ROUGE-2	ROUGE-L	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR
GITG	**31.11**	20.03	**30.15**	**39.20**	**31.59**	**27.91**	24.79	26.94
-w/o fine-tune	14.68	6.21	13.86	32.90	20.58	15.90	12.57	12.76
-w/o insert	30.98	**20.17**	30.09	38.84	31.37	27.75	**24.95**	**27.02**

注：加粗数据为最优结果.

5.4 研究问题4

由于人工评价所有样本耗费成本较高，因此我们沿用了Yang等人[22]的方法，采用一种广泛应用的抽样方法来随机选择一定数量的样本，样本数量的计算方法如式(4)所示：

$$\mathrm{MIN}=\frac{n_0}{1+\frac{n_0-1}{\mathrm{size}}} \tag{4}$$

其中，n_0取决于所选的置信水平和期望的误差范围，size是测试集中的样本数量. n_0的计算方法如式(5)所示. 其中，Z是置信水平的z值，e是误差幅度.

$$n_0=\frac{Z^2\times 0.25}{e^2} \tag{5}$$

在人工评价中，我们选择在0.05的误差幅度、99%的置信水平下进行随机抽样. 经过计算，需抽取488个样本进行评价. 我们邀请了6名硕士研究生来评估不同方法所生成issue标题的质量，他们从事软件工程方向相关研究，均具有3年以上的软件开发经验.

从测试集中随机选取的488个样本对应着问卷中的488道问题，每道问题包括issue主体描述、代码片段、参考标题、GITG生成的标题和2个对比方法生成的标题. 我们为每位参与者制作了1份问卷，在问题中会对3种方法生成的标题进行随机排序，问卷样本如图6所示.

在人工评价开始前，我们对所有的参与者进行了培训，以帮助参与者更准确地对标题进行评价. 对于每个问题，参与者需要在阅读全部信息后进行整体评分，评分标准如下：

Issue Body:

您好:FileUtil的loopFiles,如果FileFilter传空的话会取不到file的.都会被这个条件屏蔽掉

Issue Code:

```
if (fileFilter != null &&fileFilter.accept(file)) {fileList.add(file);}
```

Reference: FileUtil的bug

Title1: FileUtil的loopFiles,如果FileFilter	
整体分数:	○0 ○1 ○2 ○3
流畅性:	○0 ○1 ○2 ○3
信息性:	○0 ○1 ○2 ○3
简洁性:	○0 ○1 ○2 ○3
Title2: FileUtil的loopFiles,如果FileFilter传空	
整体分数:	○0 ○1 ○2 ○3
流畅性:	○0 ○1 ○2 ○3
信息性:	○0 ○1 ○2 ○3
简洁性:	○0 ○1 ○2 ○3
Title3: FileUtil中loopFiles的loopFiles使用报错的问题	
整体分数:	○0 ○1 ○2 ○3
流畅性:	○0 ○1 ○2 ○3
信息性:	○0 ○1 ○2 ○3
简洁性:	○0 ○1 ○2 ○3

图6 人工评价调查问卷样本

(1) 优秀(3分)：此标题与参考标题表达的意思完全相同，无需修改.

(2) 良好(2分)：此标题可以体现issue的主要思想，但存在轻微问题(例如不完整、重复或语法问题)，只需稍作修改即可.

(3) 一般(1分)：此标题可以反映issue的部分思

想,但缺乏必要细节或表达有误,需多处修改.

(4) 糟糕(0分):此标题不能反映甚至曲解了issue的思想,需要重新书写标题.

此外,参与者还需要从流畅性、信息性和简洁性3个角度来衡量不同方法生成标题的质量,每个指标单独评价,分数的取值范围同样在0~3之间,这3个评价指标的含义如下:

(5) 流畅性:衡量生成的标题是否通顺流畅、是否符合语法规范.

(6) 信息性:衡量生成的标题是否包含issue中的重要信息或核心关键词.

(7) 简洁性:衡量生成的标题是否包含冗余、重复或不相关的信息,内容是否简练.

图7直观地展示了GITG和2种对比方法在人工评价上的统计结果. 其中,图7(*a*)~图7(*d*)分别对应着整体分数、流畅性、信息性和简洁性,每幅图的左侧为GITG结果、中间为iTAPE结果、右侧为iTiger结果. 从图7中可以看出,对于GITG生成的标题,有75.8%的人工评价整体得分在2分以上,即生成的标题质量较好,无需进行重大修改即可使用. 这项数值比iTAPE高20.2个百分点,比iTiger高49.2个百分点. 此外,在GITG生成的标题中仅有2.0%的人工评价整体得分为0分,与iTAPE相比这项数值降低了7.8个百分点,与iTiger相比降低了16.2个百分点.

在流畅性方面,有97.7%的人工评价分数在2分以上,这项数值比iTAPE高12.5个百分点,比iTiger高19.3个百分点;有0.1%的人工评价得分为0分,与iTAPE相比这项数值降低了0.3个百分点,与iTiger相比降低了0.2个百分点. 在信息性方面,有76.3%的人工评价分数在2分以上,这项数值比iTAPE高18.4%,比iTiger高48.1个百分点;仅有1.6%的人工评价得分为0分,与iTAPE相比这项数值降低了8.1个百分点,与iTiger相比降低了15.9个百分点. 在简洁性方面,有95.8%的人工评价分数在2分以上,这项数值比iTAPE高16.3个百分点,比iTiger高13.5个百分点;仅有0.5%的人工评价得分为0分,与iTAPE相比这项数值降低了1.3个百分点,与iTiger相比降低了2.1个百分点.

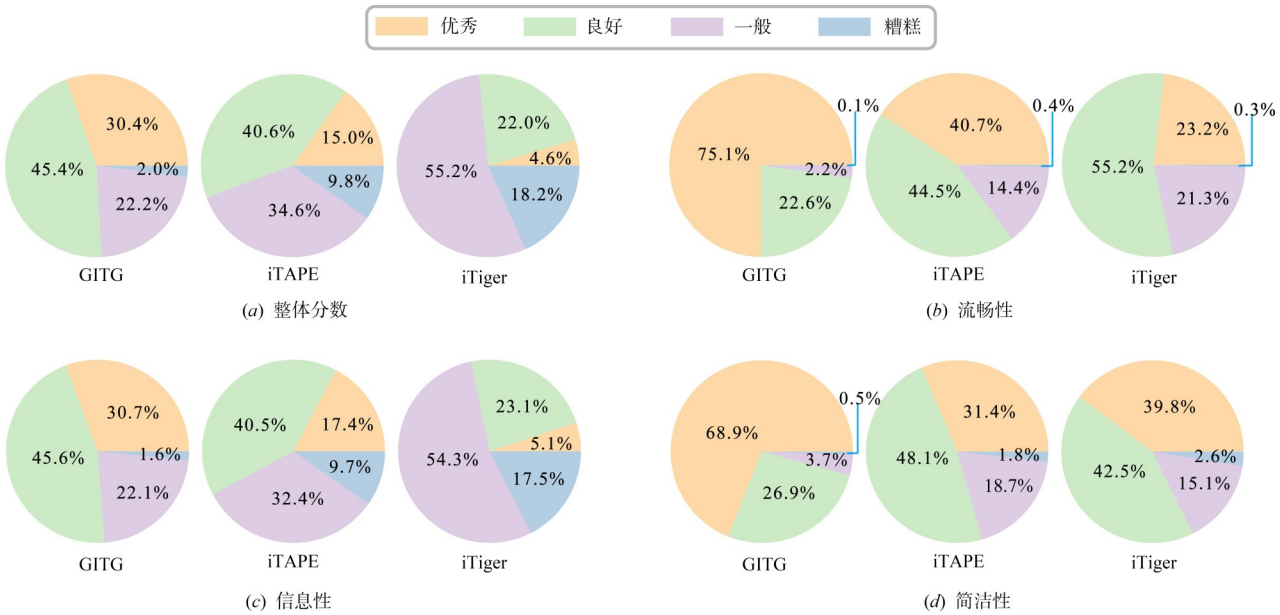

图7　不同方法生成issue标题的人工评价得分情况

图8为3种方法的人工评价平均得分统计结果. 从图8中可以看出,GITG在4项人工评价指标上的平均得分均取得了最佳表现. GITG的平均分数分别为2.04、2.73、2.05和2.64分,相较于iTAPE分别提高了0.43、0.47、0.40和0.55分,提升幅度分别为26.7%、20.8%、24.2%和26.3%;相较于iTiger分别提高了0.91、0.72、0.89和0.44分,提升幅度分别为80.5%、35.8%、76.7%和20.0%. 从人工评价的结果可以看出,与iTAPE和iTiger相比,GITG生成的标题质量更高.

此外,在人工评价过程中,我们使用组内相关系数(Intraclass Correlation Coefficient,ICC)[23]来衡量这6位参与者评价结果的一致性. 经过计算,6位参与者人工评价结果的ICC值为0.903,这表明了6位参与者之间的评分较为一致.

对研究问题4的回答:在人工评价中,GITG生成的标题有75.8%的整体得分在2分以上,与基线方法相比至少提高了20.2个百分点;在流畅性、信息性和简洁性方面,GITG也表现更优. 对于4项人工评价指标的平均

分数，GITG 与基线相比至少提升了 20.0 个百分点．人工评价的结果进一步验证了 GITG 的有效性．

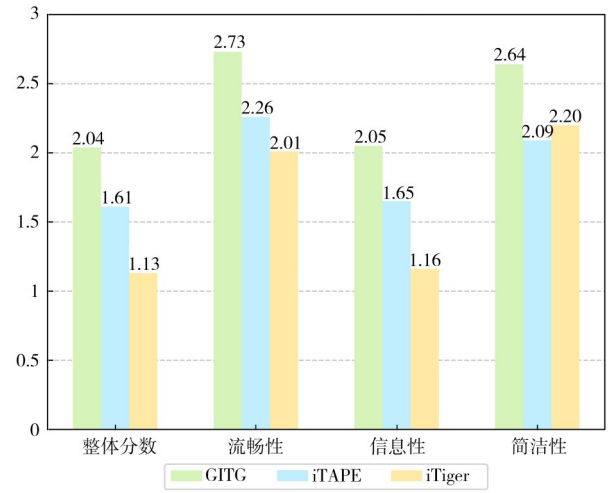

图 8 不同方法生成 issue 标题的人工评价平均得分对比

6 讨论

在本节中，将讨论 GITG 表现不佳的情况，以及有效性威胁．

6.1 GITG 表现不佳的情况分析

虽然 GITG 取得良好结果，但仍然生成了一些质量较低的标题．我们检查了 GITG 生成标题中 3 项 ROUGE 分数均低于 10 的 issue 样本，经过统计一共有 513 个，并分析了其质量不高的原因，具体可以总结为以下 2 种情况．

第 1 种情况是参考标题质量较高但有较多信息无法在 issue 描述中找到时，GITG 通常表现不佳．抽样样本中有 133 条属于这类情况．如图 9 的示例中，表格第 1 行和第 2 行是 issue 的主体描述和代码片段；第 3 行是参考标题；第 4~6 行是 GITG 和其他方法生成的标题．从图 9 中可以看出，参考标题中的主要信息“类重定向设置”无法在 issue 主体描述和代码片段中找到，因此 iTAPE 等方法和 GITG 都没有生成正确的标题．分析发现，iTAPE 生成的标题与 issue 相关性较弱，iTiger 生成的标题很不完整．相比之下，虽然 GITG 生成的标题也并不完整，但相较于 iTiger 有较大提升，且捕获到了“HttpRequest”这个关键信息．尽管 GITG 的表现不佳，但仍然优于其他方法．

第 2 种情况是参考标题本身质量不高时，导致 GITG 生成的标题与参考标题差异较大．抽样样本中有 62 条属于这类情况．如图 10 的示例中，参考标题只包含“xpath”一个单词，无法从标题中看出任何其他的有价值信息，无法推测出开发人员想要表达的核心思想，因此参考标题的质量较差．对于这种样本，GITG 和其他方法生成的标题都无法取得较高的评价指标得分．分析发现，iTAPE 生成的标题同样与 issue 相关性较弱，iTiger 生成的标题虽然可以在 issue 中找到类似的文本但并无实际意义．相比之下，虽然 GITG 生成的标题也存在较大问题，但优势是捕获到了“xpath”这个信息．

对于以上 2 种情况，GITG 和其他方法均未能生成高质量标题，但与 iTAPE 和 iTiger 相比，GITG 具有一定优势．导致第 1 种情况的原因主要是我们将 issue 标题生成任务转化为一句话摘要任务，而在实际情况中，issue 标题可能是 issue 描述的补充内容而非简明摘要．导致第 2 种情况的原因主要是由于数据集的问题，虽然已经采用启发式规则对数据集进行过滤，提升了数据集的质量，但仍然存在部分质量较低的样本．

Issue Body:此方法参数 isFollowRedirects 有什么意义?

Issue Code:

public HttpRequest setFollowRedirects(Boolean isFollowRedirects) {return setMaxRedirectCount(2);}

Reference: HttpRequest 类重定向设置

iTAPE:[Bug][DOC]增加调用调用方法参数描述

iTiger:此方法参数 isFollowRedirects 有

GITG: HttpRequest 参数 isFollowRedirects 有什么意

图 9 GITG 表现不佳的示例 1

Issue Body:这么改下就可以了

Issue Code:

```
public static void main(String[] args) {String source = "<div></div><
div class=\"content_1YWBm\"><a><div>要获取的</div><div></div>
</a></div>";
String xpath = StringFunctionExtension.xpath(source, "//*[@class=
\"content_1YWBm\"]/a/div/text()");
String xpath1 = StringFunctionExtension.xpath(source, "//div[2]/a/div
[1]/text()");
System.out.println(xpath);
System.out.println(xpath1);// 应该也能获取到值,然而没有}
```

Reference:xpath

iTAPE:关于一个小模块:->使用时不显示

iTiger:获取改下就可以

GITG:使用 StringFunctionExtension.xpath(source,"/

图 10 GITG 表现不佳的示例 2

6.2 有效性分析

本方法的内部有效性威胁主要涉及实现 GITG 和复现基准方法产生的缺陷．为缓解内部有效性威胁，我们在实现 GITG 时使用了 PyTorch、Transformers 等被广泛应用的第三方库来保证代码实施的正确性，使用

iTAPE、iTiger提供的开源代码,按照其论文中的参数设置进行了训练,以确保达到相似性能.

本方法的外部有效性威胁主要涉及所构建的数据集. 为缓解外部有效性威胁,我们在收集issue样本时按照stars数量选取了Gitee上排名前400的代码仓库,这些顶级代码仓库通常维护良好,具有一定的代表性. 此外,我们应用三条启发式规则对数据集样本进行了过滤和清洗,提升了数据集的质量.

本方法的构造有效性威胁主要涉及自动评价指标的有效性. 为缓解构造有效性威胁,本文选取了在标题生成任务中广泛应用的评价指标ROUGE、BLEU和METEOR来评价方法的性能,并使用开源库进行计算. 此外,为缓解人工评价偏见所带来的有效性威胁,我们邀请了6名有3年以上项目开发经验的软件工程方向硕士研究生进行人工评价,在评价前进行了培训,并将不同方法生成的标题打乱顺序展示.

7　总结与展望

本文提出了一种面向Gitee平台的issue标题自动生成方法GITG,将issue标题自动生成任务转化为一句话摘要任务,使用issue主体描述和代码片段的双模态信息,为包含中文和英文描述的issue自动生成标题. GITG使用构建的Gitee issue数据集对支持中文的预训练模型Chinese BART进行微调以提高所生成标题的质量. 为评估GITG的有效性,从Gitee平台上收集大量issue构建了基准数据集. 实验结果表明,在ROUGE、BLEU等指标上,GITG取得了比iTAPE和iTiger方法更好的表现. 此外,我们还进行了人工评价,结果同样表明GITG更加有效.

本文是对面向Gitee平台的issue标题自动生成任务的初步探索. 在未来研究中,我们计划用代码片段的结构化特征或使用图神经网络来表征代码片段,以进一步提升GITG的性能. 此外,我们还将进一步改进所构建数据集的质量,以优化模型性能.

参考文献

[1] ERFANI JOORABCHI M, MIRZAAGHAEI M, MESBAH A. Works for me! Characterizing non-reproducible bug reports[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. New York: ACM, 2014: 62-71.

[2] SOLTANI M, HERMANS F, BÄCK T. The significance of bug report elements[J]. Empirical Software Engineering, 2020, 25(6): 5255-5294.

[3] MA X X, KEUNG J W, YU X, et al. AttSum: A deep attention-based summarization model for bug report title generation[J]. IEEE Transactions on Reliability, 2023, 72(4): 1663-1677.

[4] CHEN S Q, XIE X Y, YIN B G, et al. Stay professional and efficient: Automatically generate titles for your bug reports[C]//Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering. New York: ACM, 2020: 385-397.

[5] LIN H, CHEN X, CHEN X J, et al. TitleGen-FL: Quality prediction-based filter for automated issue title generation[J]. Journal of Systems and Software, 2023, 195: 111513.

[6] ZHANG T, IRSAN I C, THUNG F, et al. iTiger: An automatic issue title generation tool[C]//Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. New York: ACM, 2022: 1637-1641.

[7] 中华人民共和国工业和信息化部. “十四五”软件和信息技术服务业发展规划[EB/OL]. (2021-11-15)[2024-05-10]. https://www. miit. gov. cn/jgsj/ghs/zlygh/art/2022/art_f43c068acfb14f15b8daf4238945deb0.html.

[8] ZHANG F J, YU X, KEUNG J, et al. Improving stack overflow question title generation with copying enhanced CodeBERT model and bi-modal information[J]. Information and Software Technology, 2022, 148: 106922.

[9] ZHOU Y L, YANG S Y, CHEN X, et al. QTC4SO: Automatic question title completion for stack overflow[C]//Proceedings of the 31st IEEE/ACM International Conference on Program Comprehension. Piscataway: IEEE, 2023: 1-12.

[10] SHAO Y F, GENG Z C, LIU Y T, et al. CPT: A pre-trained unbalanced transformer for both Chinese language understanding and generation[J]. Science China (Information Sciences), 2024, 67(5): 152102.

[11] 陈翔, 于池, 杨光, 等. 基于双重信息检索的Bash代码注释生成方法[J]. 软件学报, 2023, 34(3): 1310-1329.
CHEN X, YU C, YANG G, et al. Bash code comment generation method based on dual information retrieval[J]. Journal of Software, 2023, 34(3): 1310-1329. (in Chinese)

[12] 郭丹, 姚沈涛, 王辉, 等. 嵌入局部聚类描述符的视频问答Transformer模型[J]. 计算机学报, 2023, 46(4): 671-689.
GUO D, YAO S T, WANG H, et al. Embedding VLAD in transformer for video question answering[J]. Chinese Journal of Computers, 2023, 46(4): 671-689. (in Chinese)

[13] 孙锐, 谢瑞瑞, 张磊, 等. 基于灾难性遗忘及组合叠加擦除的跨模态行人重识别预训练方法[J]. 电子学报, 2023, 51(10): 2925-2935.

SUN R, XIE R R, ZHANG L, et al. Cross-modal pedestrian re-identification pre-training method based on catastrophic forgetting and combination superimposed eras-ure[J]. Acta Electronica Sinica, 2023, 51(10): 2925-2935. (in Chinese)

[14] LIU K, CHEN X, CHEN C Y, et al. Automated question title reformulation by mining modification logs from stack overflow[J]. IEEE Transactions on Software Engineering, 2023, 49(9): 4390-4410.

[15] LIU S Q, GAO C Y, CHEN S, et al. ATOM: Commit message generation based on abstract syntax tree and hybrid ranking[J]. IEEE Transactions on Software Engineering, 2022, 48(5): 1800-1817.

[16] 刘诗凡, 崔展齐, 陈翔, 等. MMCUP: 融合多模态信息的代码注释自动更新方法[J]. 计算机学报, 2024, 47(1): 172-189.
LIU S F, CUI Z Q, CHEN X, et al. MMCUP: Updating code comments based on multi-modal information[J]. Chinese Journal of Computers, 2024, 47(1): 172-189. (in Chinese)

[17] BAJAJ K, PATTABIRAMAN K, MESBAH A. Mining questions asked by web developers[C]//Proceedings of the 11th Working Conference on Mining Software Repositories. New York: ACM 2014: 112-121.

[18] LIN C Y. ROUGE: A package for automatic evaluation of summaries[J]. Text Summarization Branches Out, 2004: 74-81.

[19] PAPINENI K, ROUKOS S, WARD T, et al. BLEU: A method for automatic evaluation of machine translation[C]//Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. Stroudsburg: ACL, 2002: 311-318.

[20] BANERJEE S, LAVIE A. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments[C]//Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. Stroudsburg: ACL, 2005: 65-72.

[21] 陈翔, 杨光, 崔展齐, 等. 代码注释自动生成方法综述[J]. 软件学报, 2021, 32(7): 2118-2141.
CHEN X, YANG G, CUI Z Q, et al. Survey of state-of-the-art automatic code comment generation[J]. Journal of Software, 2021, 32(7): 2118-2141. (in Chinese)

[22] YANG G, CHEN X, ZHOU Y L, et al. DualSC: Automatic generation and summarization of shellcode via transformer and dual learning[C]//Proceedings of the 29th IEEE International Conference on Software Analysis, Evolution and Reengineering. Piscataway: IEEE, 2022: 361-372.

[23] CARO T M, ROPER R, YOUNG M, et al. Inter-observer reliability[J]. Behaviour, 1979, 69(3): 303-315.

作者简介

杨 君 男,2000年10月生,北京人. 现为北京信息科技大学硕士研究生. 主要研究方向为issue标题自动生成.
E-mail: yangjun1026@bistu.edu.cn

陈 翔 男,1980年3月生,江苏南通人. 现为南通大学副教授,硕士生导师. 主要研究方向为软件缺陷预测、软件缺陷定位、回归测试和组合测试.
E-mail: xchencs@ntu.edu.cn

刘诗凡 男,1997年9月生,陕西汉中人. 已于北京信息科技大学获得硕士学位,现为北京科技大学博士研究生. 主要研究方向为代码注释生成和软件测试.
E-mail: pawn2017@bistu.edu.cn

崔展齐 男,1984年2月生,贵州金沙人. 现为北京信息科技大学教授,博士生导师. 主要研究方向为软件分析与软件测试技术.
E-mail: czq@bistu.edu.cn