计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(10):262−266 [doi: 10.15888/j.cnki.csa.007523]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 TextRank 的软件变更任务搜索术语识别①

王　杉

(云南大学滇池学院 理工学院, 昆明 650228)
通讯作者: 王　杉, E-mail: 52298256@qq.com

摘　要: 在软件工程的演进或维护阶段, 有很多软件变更要求需要软件开发人员处理, 这些变更要求通常都使用自然语言文本进行编制, 而且通常涉及一个或多个相关问题域. 软件开发人员要将这些概念准确映射到软件项目中的相应源码位置, 已进行所要求的变更. 完成这样的映射需建立若干搜索术语项, 并在项目中进行搜索. 而研究表明, 开发人员在为任务变更提出准确而合适的搜索条件时具有一些困难. 因此本文提出了一种基于 TextRank 的软件变更任务搜索术语的识别方法, 通过分析自然语言描述的任务来识别和提出软件变更的搜索术语项, 以提高搜索的准确性、平均精度和召回率.

关键词: TextRank; 变更任务位置; 搜索术语; 软件工程

引用格式: 王杉.基于 TextRank 的软件变更任务搜索术语识别.计算机系统应用,2020,29(10):262–266. http://www.c-s-a.org.cn/1003-3254/7523.html

Search Term Recognition for Software Change Tasks Based on TextRank

WANG Shan

(Institute of Technology, Dianchi College of Yunnan University, Kunming 650228, China)

Abstract: At the evolution or maintenance stage of software engineering, software developers are required to handle many Software Change Orders (SCOs), which are generally prepared in natural language texts and involve one or more problem domains. The developers will accurately map these orders to corresponding source codes in the software programme and make ordered changes. This mapping requires creation of several search terms and search of them in the programme. Studies show that developers have difficulties in creating accurate and suitable search conditions for changes. In this study, the author proposes a TextRank-based search term recognition method for software change tasks. It enables identification and creation of search terms for software changes by analyzing tasks described in natural languages, to improve the accuracy, average precision, and recall rate of searching.

Key words: TextRank; change tasks location; search term; software engineering

软件工程研究表明, 在软件开发过程中, 用于软件维护和演进的投入占总投入的大约 70%−80%[1]. 在软件维护期, 开发者通常会处理很多软件变更的请求, 这需要确定变更任务在软件项目中的确切位置 (更改的类或方法). 变更请求通常是由用户提出的, 因此一般都用自然语言编写, 并且软件用户虽然熟悉软件产品的应用领域, 但他们却很难具备在源代码中实现功能的能力. 另一方面, 参与维护的人员也不了解项目的底层架构, 他们在识别需要更改的源码位置也有困难. 因此需要进行一种从软件变更处到源码位置 (类、方法

① 基金项目: 云南省高校科技创新团队支持计划 (2017)
Foundation item: Science and Technology Innovation Team Supporting program of Higher Educations of Yunnan Province (2017)
收稿时间: 2019-12-20; 修改时间: 2020-01-22; 采用时间: 2020-02-11; csa 在线出版时间: 2020-09-30

等) 的映射, 该映射任务通过一个或多个搜索项, 从项目内进行搜索, 最终得到映射位置, 以快速而准确地完成软件变更任务.

在以往的研究中, 有一些尝试通过搜索查询以支持开发者进行功能定位任务的方法, 例如轻量级启发式方法[2]、查询重构或扩展策略[3]、查询质量分析法[4]、数据字典及挖掘方法[5] 等. 这些方法都要求开发者能提供可改进的初始搜索查询, 而对于开发者来说, 这也是一项繁重的任务. 其中, Kevic 和 Fritz 提出了一种用于自动识别软件变更任务的初始搜索项的启发式模型[2], 模型考虑了与搜索频率、位置、词性和任务描述术语符号相关的启发法. 该模型在一定程度上解决了上述问题, 但也存在两个不足: 首先, 模型使用有限数量的变更任务进行训练, 缺乏其他项目的变更任务进行交叉验证, 成熟度和可靠性有限; 其次, 模型把 tf-idf 作为主要度量参数, 而 idf 计算受测试数据集大小的影响, 对于不同大小的测试数据集, 会造成相同模型呈现不同的表现, 并且模型可能需要频繁的重新训练以保持其可用性.

在本文中, 提出了一种基于 TextRank 进行自动识别并得到软件变更任务搜索术语的方法. TextRank 方法是 PageRank 方法的改进[6], 对于自然语言文本, 其中的文本可被视为词汇的词汇或语义网络. TextRank 技术被广泛应用于各类信息检索中, 例如关键字提取、摘要提取、词义消歧、词法分析和其他基于图的属于加权任务. 因此该算法适合在特征定位任务的上下文中进行搜索项提取, 这是因为软件变更请求通常由开发团队以外的人员进行, 他们通过相关问题域的概念和自然语言文本来表达需求, 并可以使用基于图的任务描述来揭示不同术语之间的重要语义关系. 其次, TextRank 算法利用图中术语的连通性, 用于确定该术语的权重 (即重要性), 然后递归执行该过程, 从而确定出搜索术语.

例如, 在表 1 中显示的软件变更任务可以图的方式表示为图 1 中的文本图. 本文提出的方法分析图中术语的连通性, 计算每个术语的权重 (即重要性), 然后提取最高权重和最适合的五个术语: Mac, selection, installs, improvement 和 JREs, 作为搜索任务的初始搜索项进行查询.

表 1 来自 eclipse.jdt.debug 的变更请求示例

Issue ID: 401 358
Product: JDT, **Component**: Debug
Summary:
Description: When you search for a JDK/JRE on Mac, we use information from the plist file to compute a name. This works fine most of the time, but if you happen to have more than one of the same version of VM installed thevare added with the same name. To make matters a bit worse, if you edit one of the JRES the wizard starts out with an error complaining that the name is already in use. The attached screen shot shows the duplicated names for the Java 7 JRES.

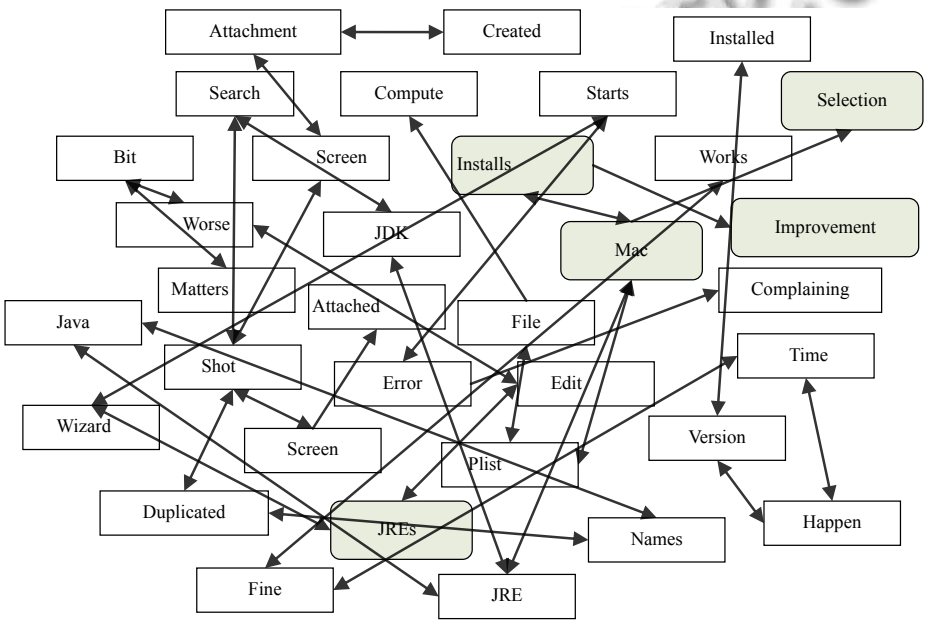

图 1 表 1 中的变更请求文本图

并且选取了两个主题系统 Apache Log4j(日志分析) 和 Eclipse JDT Debug 中的 336 个变更任务进行实验, 实验表明了提出的方法可以返回 49.65% 的变更任务相关结果 (即 Java 类), 平均精度为 58.00%, 召回率达 63.48%. 与 Kevic 和 Fritz 的方法[2] 比较, 本文提出的方法在各项评价指标上有更好的表现. 因此, 本文做了以下工作:

① 展示了 TextRank 算法在识别和获取软件变更任务的相关搜索术语中的新用途.

② 通过一个涉及两个主题系统的软件变更任务案例研究, 并与 Kevic 和 Fritz 的现有方法进行比较, 展示了所提出方法的有效性.

1 方法

图 2 展示了提出方法的搜索识别技术示意图和变更任务的基本步骤. 接下来将讨论该方法涉及的步骤.

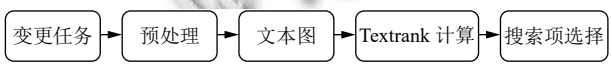

图 2 所提出方法的示意图

1.1 变更任务的数据采集

通常, 同户提供的软件变更任务是使用自然语言描述的 (如表 1 所示), 通过收集了 336 个开发实际任务作为数据集进行分析. 这些任务用半结构化的方式提交, 包含了问题 ID、产品、组件、摘要、描述等几个字段, 由于任务变更的具体问题基本都只在摘要和描述中提及, 因此分析仅选取最后两个字段的内容. 同时为了保持算法的简单性和轻量级, 也暂时不考虑其他附加到更改任务中的额外信息.

1.2 文本预处理

在这一步骤中, 将分析软件变更请求中的摘要和描述, 并在将文本转化为文本图之前进行多个预处理. 预处理将每个句子视为一个逻辑文本单元, 这样有助于整体任务描述, 接下来从每个句子中提取能传达有用语义或表达软件问题域的术语, 然后从句子里删除副词、介词等非重要词, 并将包含分隔符号 (例如“.”) 的词分解为组成它们的更简单的词. 这样的分解有助于单独分析每个概念. 例如: ”org.eclipse.ui.part.PageBookView.createPartControl”, 这个句子包含了一个包名 (org.eclipse.ui.part)、一个类名 (PageBookView) 和一个方法名 (createPartControl). 需要指出的是, 在这里不采用软件代码标识符命名规则里常用的“驼峰大小写”表示法进行拆分, 因为变更请求可能包含不同的技术工件, 例如库名、类名、方法名等, 这些名称往往已经使用了“驼峰大小写”, 那么进一步拆分会破坏工件的完整性.

1.3 文本图开发

在文本预处理之后, 将会得到一个句子列表, 每个句子都包含有在语义上很重要的术语以及与问题域相关的术语, 然后使用这些术语来描述任务的开发术语图. 可以将每一个术语表示为图中的不同节点, 并考虑句子中这些术语的“同现”, 作为它们之间语义关系 (依赖关系) 的指征[6]. 例如考虑句子“This works fine most of the time, but if you happen to have more than one of the same version of VM installed they are added with the same name”(表 1), 预处理会形成一个有序的术语短句: “works fine time happen version installed”. 转换后的句子会包含“works fine”、“version installed”等短语, 这些短语中的术语在语义上依赖, 当“窗口大小”以 2 作为单词的语义单位时[6], 可获得以下关系: works←→fine, fine←→time, time←→happen, happen←→version, version←→installed. 然后将这种关系表示为文本图中相应节点之间的邻接边.

1.4 TextRank 计算

为了计算文本图中每个术语的权重 (重要性), 方法采用了一种基于 Web 链接分析的 PageRank 改进的 TextRank 算法[6]. 该算法递归分析文本图中每个术语的传入链接和传出链接等细节, 并计算其权重 $W(V_i)$, 如下式所示:

$$W(v_i)=(1-\varphi)+\varphi\sum_{j\in In(v_i)}\frac{W(v_j)}{\left|Out(v_j)\right|}\quad(0\le\varphi\le 1)$$

这里, $In(v_i)$ 表示通过输入链路指向 v_i 的节点集合, $Out(v_j)$ 表示由 v_j 发出的输出链路指向的节点集合, φ 表示阻尼系数. 在文本图中, 每条边都是无向边 (即术语彼此依赖), 因此节点入度等于出度. 对于 φ 的取值, 通常按一般取值为 0.85, 然后采用 0.25 的默认值初始化图中每个术语, 并开始迭代计算, 直到项的分数收敛到 0.0001 的极限值或达到最大迭代限制值 100. 基于 TextRank 的推荐机制, 如果术语 B 以任何方式补充术语 A 的语义, 则术语 A 就推荐术语 B[6], 那么这个算法将根据文本图中个的传入链接获得对术语的推荐,

并确定术语的重要性. 一旦计算结束, 图中的每个术语都会得到一个分值, 该分值可被认为是这个术语在文本中的权重 (重要性).

1.5 搜索术语选择

计算 TextRank 后, 会根据权重对术语进行排序, 并以启发方式选择变更任务的搜索术语, 其中任务“摘要”和“描述”中的术语最适合作为搜索术语. 但是这些字段中的术语存在重叠而不足以形成搜索查询, 因此需对其进行调整. 首先在变更任务的摘要中查找权重最高的术语, 如果摘要内容太少而无法提供所有条件, 那么就从任务描述中收集其余内容. 然后基于它们的权重或等级来选择术语, 这些术语是通过递归分析文本图中该术语周围的术语计算出的. 例如表 1 所示的变更任务, 得到搜索项: Mac (权重 0.64)、selection (权重 0.27)、installs (权重 0.27)、improvement (权重 0.25) 和 JREs (权重 1.0, 来源于描述).

2 实验

为了验证本文所提出的方法, 将使用了两个开源的主题系统的变更任务进行实验, 分别是 eclipse 中的调试模型插件 eclipse.jdt.debug, 以及 Apache 的日志操作包 Log4j. 并与一种现有的方法进行了比较.

2.1 数据集和文件搜索

实验中, 选择了两个开源主题系统的变更请求任务. 首先从缺陷追踪数据库 BugZilla 中收集变更任务, 分析其 BugID(变更任务的标识), 然后根据 BugID 从来源代码数据库 GitHub 中找到相应的项目 (GitHub 中, 每个软件错误解决和变更请求的提交操作通常会在其提交消息中提到相应 BugID). 基于此, 在 Log4j 中发现了 223 次提交, 在 eclipse.jdt.debug 中发现了 113 次提交, 共计 336 个变更任务.

然后, 为每个变更任务收集变更集 (即已更改文件列表), 并开发解决方案集. 为了搜索变更任务的文件, 采用了基于向量空间模型的搜索引擎 Apache Lucene[7]. 由于项目中的源文件会包含常规文本外的内容 (例如代码段), 那么将对每个源文件应用有限的预处理, 删除所有标点符号, 这就会将源代码转换为纯文本, 有助于搜索引擎更有效地执行. 启动搜索后, 搜索引擎使用布尔搜索模型过滤语料库中的无关文件, 并应用基于 tf-idf 的评分技术来返回相关文档的排序列表, 最后选择返回的排名前 10 以内的结果项进行操作.

2.2 评价指标

为了清晰地评价提出方法的效果和性能, 会采用 K 值平均精确率 (MAPK) 和平均召回率 (MR) 两个主要指标, 对搜索结果进行评价. K 值平均精确率用于表示所有搜索结果的平均相关精度, 即查准率; 平均召回率用于表示搜索到的结果集的返回率, 即查全率.

2.3 实验结果

对两个开源的主题系统的 336 个变更任务进行了实验, 并应用解决任务数、解决任务百分比、平均精确率和平均召回率 4 个不同的指标进行总结. 结果如表 2 所示.

从实验结果表中可以看到, 当使用 5 个搜索词时, 查询效果最佳. 例如, 它们返回了来自 Log4j 的 223 个更改任务中的 107 个 (47.98%) 的相关结果, 以及来自 eclipse.jdt.debug 的 113 个更改任务中的 58 个 (51.33%), 这是预期良好的. 因此, 平均而言, 该查询从数据集中检索 63.48% 的解决方案, 平均精度为 58.00%.

表 2 实验结果指标

指标	Log4j(223)			eclipse.jdt.debug(113)			平均(T_5)
	T_3	T_4	T_5	T_3	T_4	T_5	
解决任务数(NTS)	82	93	107	46	51	58	—
解决任务百分比(PTS)(%)	36.77	41.70	47.98	40.71	45.13	51.33	49.65
平均精确率(MAP)(%)	65.84	63.12	62.76	55.99	52.65	53.24	58.00
平均召回率(MR)(%)	52.93	55.42	55.11	72.98	73.31	71.86	63.48

注: T_3表示3个搜索词的结果, T_4表示4个搜索词的结果, T_5表示5个搜索词的结果

为了从其他方面评估提出方法的性能, 还从以下几个方面进行了实验:

① 使用 6 个搜索词进行查询, 但是查询结果在这两个主题系统中表现相对较差.

② 在没有预处理的情况下, 基于现有语料库重新进行了实验, 没有出现较明显的性能下降, 证明了提出

的搜索术语对于变更任务的稳健性.

③ 在变更任务中, 不采用 TextRank 计算, 而是随机选择了五个搜索词进行查询, 结果非常糟糕, 从 log4j 只返回最多 52 个任务的相关结果, 从 eclipse.jdt.debug 只返回 37 个任务 (本文的方法返回了 107 个和 58 个结果).

④ 将变更任务调整为中文描述, 按中文文法进行预处理后, 该方法仍能以较好的召回率和精度得到搜索结果, 说明该方法有良好的语言通用性.

因此, 实验结果表明, 本文提出的搜索词方法和技术能完成任务, 并且有不错的可靠性、稳定性和通用性.

3 结论和未来工作

总而言之, 本文提出了一种新的基于 TextRank 的技术方法, 可以自动识别并得到软件更改任务的搜索项. 通过来自两个主题系统的 336 个更改任务的实验表明, 该方法平均可以返回 49.65% 的变更任务的相关结果 (即 Java 类), 平均精度为 58.00%, 召回率为 63.48%. 方法能完成预期目标, 并具有良好的性能.

虽然这些初步研究结果证明了该方法的潜力, 但需要验证更多不同类型和变化任务的主题系统, 例如更大体量的中文数据集等, 以期进一步改进该方法的性能, 获得更好的应用前景.

参考文献

1 张海藩. 软件工程导论. 5 版. 北京: 清华大学出版社, 2003. 190.

2 Kevic K, Fritz T. Automatic search term identification for change tasks. Proceedings of the 36th International Conference on Software Engineering. Hyderabad, India. 2014. 468–471.

3 李纲, 胡蓉. 信息搜寻中用户查询重构研究综述. 图书情报工作, 2014, 58(11): 123–129.

4 Haiduc S. Automatically detecting the quality of the query and its implications in IR-based concept location. Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011). Lawrence, KS, USA. 2011. 637–640.

5 Howard MJ, Gupta S, Pollock L, *et al*. Automatically mining software-based, semantically-similar words from comment-code mappings. Proceedings of the 2013 10th Working Conference on Mining Software Repositories (MSR). San Francisco, CA, USA. 2013. 377–386.

6 Mihalcea R, Tarau P. TextRank: Bringing order into text. Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing. Barcelona, Spain. 2004. 404–411.

7 Ponzanelli L, Bacchelli A, Lanza M. Seahawk: Stack overflow in the IDE. Proceedings of the 2013 35th International Conference on Software Engineering (ICSE). San Francisco, CA, USA. 2013. 1295–1298.