第47卷第1期 中国科学技术大学学报 Vol. 47, No. 1
2017年1月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jan. 2017

文章编号:0253-2778(2017)1-0080-07

软件历史代码库词库自动构建技术及实现

孙伟松,孙小兵,李 斌,杨 辉

(扬州大学信息工程学院,江苏扬州 225127)

摘要:在对已有程序代码进行理解或者维护时,开发人员通常需要使用代码搜索技术搜寻感兴趣的代码,但有时候不知道该软件系统过去的开发者和维护者在软件开发和维护过程中定义了哪些元素以及这些元素之间存在着什么关系,因此就很难搜索到想要的代码来进行维护.针对以上问题,提出一个针对具体软件历史版本库的词库自动构建方法,基于该方法建立的词库可以有效地帮助开发人员进行系统的理解和维护.另外,给出了针对历史代码库进行词库建立的工具WB4HPR.WB4HPR可以为开发人员检索出他们想要了解的词语、词语之间的关系以及它们在历史库中的演化情况.基于WB4HPR,开发人员可以方便地理解软件系统在过去版本中使用的单词或词组以及使用单词之间存在的关系,能够有效地保证软件代码中词语前后使用的一致性.

关键词:源代码分析;词库;程序理解

中图分类号:TP 391 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2017.01.011

引用格式:孙伟松,孙小兵,李斌,等. 软件历史代码库词库自动构建技术及实现[J]. 中国科学技术大学学报,2017,47(1):80-86.

SUN Weisong, SUN Xiaobing, LI Bin, et al. On automatic construction of the word base for historical program repository[J]. Journal of University of Science and Technology of China, 2017,47(1):80-86.

On automatic construction of the word base for historical program repository

SUN Weisong, SUN Xiaobing, LI Bin, YANG Hui

(*School of Information Engineering, Yangzhou University, Yangzhou* 225127, *China*)

Abstract: Developers or maintainers are finging it harder to understand and maintain software. Given a system under maintenance, developers may use a code search technique to locate the code of their interests. However, it may be difficult for them to understand the elements and the relation among them for the system at hand. Thus, the returned code may not fit their needs. It is thus necessary to have a word base to recover the elements and their relations for a target system. A tool, WB4HPR(word base for historical program repository) was introduced which focuses on building the word base for a specific system. WB4HPR can retrieve the words, recover the relationship between them, and display the evolution of these words during the program evolution to help developers and maintainers comprehend the program,

收稿日期:2016-03-01;**修回日期**:2016-09-17

基金项目:国家自然科学资金(61402396),教育部博士后面向基金(2015M571489),江苏省普通高校研究生实践创新计划(SJLX15_0665),江苏省大学生创新创业训练计划项目资助.

作者简介:孙伟松,男,1994年生,本科生.研究方向:软件数据分析.E-mail: weisong_sun@126.com

通讯作者:孙小兵,博士/副教授. E-mail:xbsun@yzu.edu.cn

while effectively keeping the consistency of the use of words during the software maintenance process.
Key words: source code analysis; word base; program comprehension

0　引言

随着软件项目的不断维护，其复杂程度不断增加；在软件维护过程中，软件开发人员需要理解软件的过去版本系统. 一方面，对开发者来说，他们可能会搜索具有类似功能的程序代码，并且对类似功能的系统做一个全面的了解，以便于模仿类似系统的开发(软件复用)；但随着软件版本的不断更新，很难有大量的时间去理解所有版本系统. 另一方面，对已有代码进行理解或者维护时，工作人员经常使用代码搜索技术搜寻感兴趣的代码，但他们不知道该软件系统过去开发者和维护者在软件开发和维护过程中定义了哪些元素以及这些元素之间存在着什么关系，在搜索的时候使用哪些词进行搜索，因此就很难搜索到想要的代码. 考虑到软件历史库过于庞大，人工的方法费时费力. 此时，一个从具体软件历史版本库提炼出的词库(知识库)作为检索参考，对软件开发者或者维护者非常有帮助.

目前，已有一些工具可以帮助软件开发人员建立软件词库，如 Yang[1] 提出了 SWordNet，基于软件上下文推断语义相关词汇，给用户推荐一类具有相似功能的软件系统中出现的相关词汇；Tian 等[2] 提出了 $\text{WordSin}_{\text{DB}}^{\text{SE}}$，利用 StackOverFlow 中的问答帖提炼出相关词组，向用户推荐相关度较高的词汇来扩展检索. 在构件化软件开发与维护中，钟林辉等[3-4] 对构件化软件演化信息建模和获取方法进行了研究，促进了构件化软件的知识库构建，便于软件开发者对构件化软件的理解[5]. 在软件代码结构理解方面，Lin 等[6-7] 提出一种基于图数据库的代码结构解析与搜索方法，对分析代码结构提供了很大的帮助. 这几种方法弥补了软件领域词库(知识库)的空缺，促进了软件领域知识库的建设. SWordNet 和 $\text{WordSin}_{\text{DB}}^{\text{SE}}$ 只是推荐一些词语给软件开发者和维护者，简单的推荐词语并不能很好地帮助软件开发者对类似软件系统进行了解，也很难帮助软件维护者对所维护目标系统进行一个全面的了解. 构件化软件的演化研究针对的是构件化软件，对其他软件系统理解帮助有限.

本文提出的词库构建方法是针对某一具体软件历史代码库的词库自动构建方法，基于该方法我们开发了一个构建具体软件系统词库(知识库)的工具 WB4HPR (word base for historical program repository). WB4HPR 是一个纵向解析一个软件系统的工具，可以让用户自己选择想要了解的具体的软件系统，针对性地检索内容. 同时，该工具还给出具体软件系统的每个词语随着软件系统版本演化的过程，给软件开发者和软件维护者提出词语的使用建议.

1　词库系统的设计与实现

WB4HPR 主要针对具体软件系统的代码库进行词库自动构建. 构建该词库的技术实现流程主要包括：①对软件系统历史代码库数据的处理，生成该软件系统的词库；②对用户输入词汇进行简单的处理，包括使用 WordNet 进行一些同义词和相似词推荐，然后使用这些词汇在词库中检索；③对词库的返回结果进行处理，显示给用户，包括对返回结果中的词组之间的关系可视化，流程如图 1 所示.

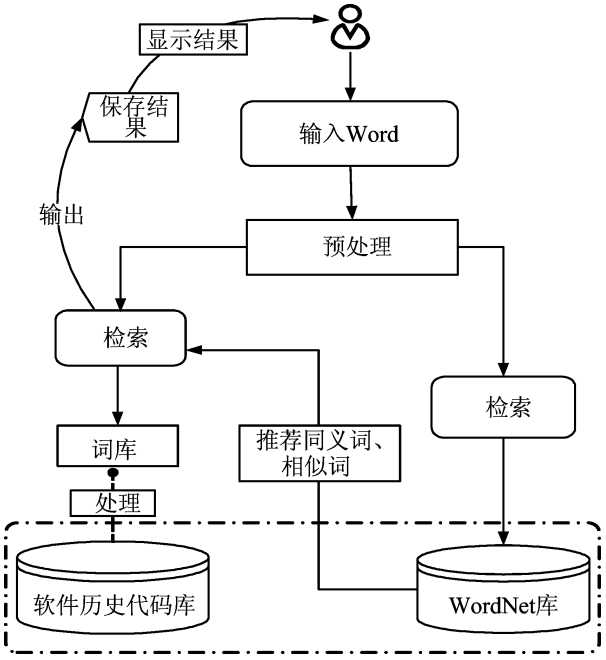

图 1　词库设计流程图

Fig. 1　Word base design flow

1.1 软件系统历史代码库数据处理

整个数据处理实现流程如图2所示. 主要包括以下5个步骤.

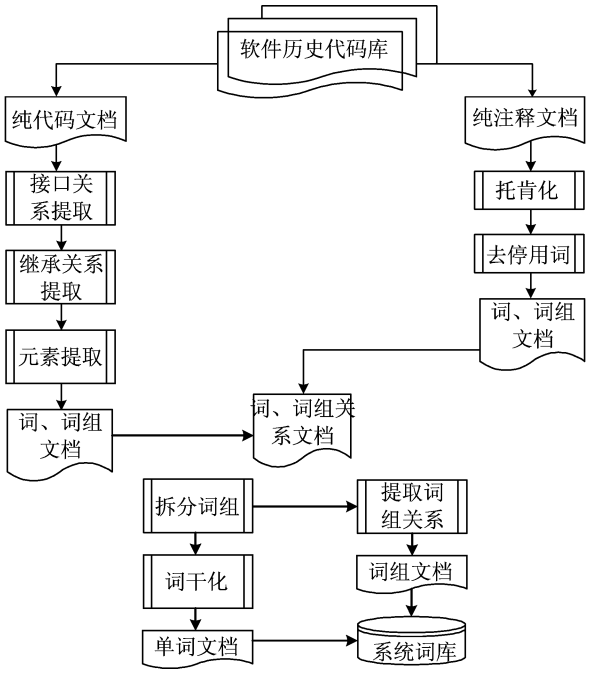

图2 数据处理流程图

Fig. 2 Data processing flow

步骤1 提取出软件系统(java语言开发的软件系统)的历史版本库中的代码和注释生成独立的文档语料库,并将该语料库分为纯代码文档库和纯注释文档库.

步骤2 对语料库中的纯代码文档进行预处理,包括托肯化、去停用词,提取元素(包括标识符、类名、方法名、变量名),得到单词和词组以及它们在代码中的支持度(Code-TF). 另外,在托肯化过程,利用java中的"<子类名>+extends+<父类名>"的语法,基于中间单词"extends"分析出类与类之间的的继承关系(kind-of). 利用java中的"<类名>+implements+<接口>"的语法,基于中间单词"implements"分析出类与接口的关系(realize-of),得到W\WG-Code(纯代码语料库中的词、词组关系库)库.

步骤3 对纯注释文档进行词性标注,然后进行预处理,使用自然语言处理(natural language processing)技术[8],包括托肯化、去停用词,提取出纯代码文档中的单词或词组以及它们在纯注释中的支持度(Comment-TF),利用注释匹配分析出缩略关系,得到W\WG-Comment(纯注释中的词、词组关系库)库.

步骤4 将纯代码语料库中词、词组关系库与纯注释语料库中词、词组关系库进行整合,去除相同或多余的单词和词组,得到整合后的W\WG(词、词组关系库)库.

步骤5 先将步骤4生成的词、词组关系库中的词组关系提取出来,得到词组关系库;其次对词、词关系进行拆分、词干化得到纯单词文档;再对得到的单词进行词性标注,分析出词组关系、同义关系、缩略关系,得到单词关系库;最后从词组关系库和单词关系库整理出本系统的完整系统词库.

1.2 WB4HPR中查询扩展

在软件开发者和使用WB4HPR工具的人员之间可能存在词语使用不一致问题,可能导致检索不到想要的结果,为了降低这种词语使用不一致带来的检索误差,我们利用WordNet推荐一些名词或动词的同义词和相似词. WordNet与传统词典不同,它不仅给出单词释义,而且给出了系统中单词之间的网络关系. WordNet是通过使用场景来定义词与词之间的同义关系,本工具在进行目标词语的同义词扩展时,使用了目标词语在WordNet中的所有动词和名词的使用场景. 由于WordNet是通过单词之间的语义关系构建的一个词库,具有广泛的使用范围,但这通常不包括软件领域的词汇,并且一些单词在WordNet中的含义与软件工程语境下的含义完全不同. WB4HPR并不是直接将WordNet推荐的同义词或者是相似词推荐给用户使用,而是使用其推荐的那些也存在于具体软件系统历史代码库构建的词库中的同义词. 使用WB4HPR进行检索,返回的结果不仅包含WordNet推荐出的一些单词在具体系统中的使用情况,还包含了与该单词有关的一些词组(比如类名、方法名)以及词组在具体软件系统中的详细使用细节. 另外,我们对这些推荐的词组也给出了词组与词组之间的关系图.

1.3 词语之间的关系建立及可视化

确立词语之间的主要关系包括:类与类或者接口与接口之间的继承关系、类与接口之间的实现关系、词与词的组合关系(group-of)、同义关系(synonym-of)、缩略关系(abbreviation-of)(如cwd为current working directory的首字母缩写). 同义关系是利用WordNet推荐词汇与本软件系统推荐词汇相结合确定;组合关系是推荐出来的一些标识符,这些标识符中包含用户输入的单词;继承关系、

接口的实现关系是由 WB4HPR 推荐的“相关词汇”中存在一些类名和一些接口名确定的. 这些关系的确立有利于用户更清晰地了解对象软件系统的构架.

2　实例应用

目前,词库系统 WB4HPR 的设计开发和运行环境是:Windows 7 及以上版本的 Windows 操作系统, Eclipse SDK 4.2 及以上版本, JDK 1.7 及以上版本, 系统的界面设计主要应 Java Swing 框架搭建. 本节主要给出工具在 Eclipse 的 Debug 插件实例系统上的应用说明. 其系统功能从如下几个方面进行说明:实例代码库介绍、词语的检索、词与词之间关系的可视化、源码查看、词语版本的演化过程.

2.1　软件历史代码库

我们使用 Eclipse 中的 Debug 插件历史版本代码库作为我们的实际案例对 WB4HPR 进行介绍. 实验过程中,过滤掉非 java 文件,对非 java 文件不进行处理. 主要是将代码库中的文件提取出两个语料库,一个是纯代码(即不包含注释的)语料库和一个纯注释(不包含代码)的语料库;然后对这两个语料库分别进行自然语言处理,构建这个软件系统的一个词库.

2.2　词语检索

在使用 WB4HPR 工具时,软件工作人员可以通过选择“选择历史库”按钮,添加历史代码库所存储的路径. 待检索的词汇输入到该工具的的 Word 输入栏中,然后点击 search 按钮即可检索系统内容,如图 3 所示.

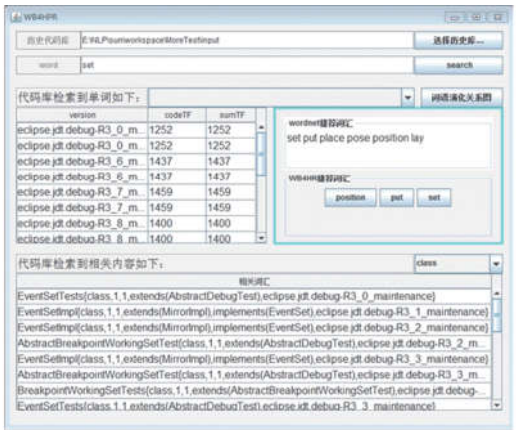

图 3　WB4HPR 的个性化词汇检索界面

Fig. 3　Personalized retrieval interface of WB4HPR

用户选择的目标软件系统历史代码库是以文件的形式存放于计算机的某一路径下,该路径用户可以自行选择,WB4HPR 首先对用户选择的目标历史代码库进行数据处理,将构建的词库存放在文件中. 当用户在 Word 输入框中输入待检索的词语点击了 search 之后,系统会自动根据用户输入的 word 存放词库文件中通过索引的方式检索该词语. 检索到的内容包括:源代码库中该单词的存在情况及显示在“相关词汇”表格中与该单词有关的推荐词汇. WB4HPR 利用 WordNet 词典对用户输入的词进行同义词、相似词扩展,以丰富检索中的关键词,从而降低用户和原软件开发者使用词语不一致性产生的误差,其扩展结果显示在 WordNet 栏目中. WB4HPR 会利用 WordNet 中推荐的词汇再回到软件代码库中检索,检索到的结果也会显示在屏幕 WordNet 栏右侧. 用户只需要点击 WordNet 与系统推荐的词汇,就可以查看“相关词汇”表中与推荐词的相关词汇. 在“相关词汇表”中可以选择查看具体元素类型(class、method、attribute、interface、package)的相关词汇.

2.3　词与词之间关系可视化

通过词与词之间关系的可视化,可使开发者很清楚地看到这些词之间的关系,例如,可由一个类扩展联想到另一个类有关的类文件.

WB4HPR 对词与词之间的关系进行了可视化. 在“相关词汇”表中点击具体的某一行,即可弹出具体词组的详细信息以及与其他词组之间的关系,如图 4 所示. 表格中的内容为推荐词组的详细信息,包括元素分类、代码中的支持度、注释中的支持度、总支持度、版本支持度、继承关系、接口的实现关系. 该实例中展示了词组(类名、接口)与词组(类名、接口)之间的继承关系、词组(类名)与词组(接口)之间的接口实现关系. 这些关系的确立有利于用户更清晰地了解软件系统的设计和实现.

2.4　源代码查看

一个类可能继承另一个类、实现多个接口. 对软件开发者来说,只看类名、接口名无法详细了解这个系统,需要对类的接口进行更详细的了解,这时就需要快速查看类文件和与之相关的类文件. 另外,对于类似软件系统的开发者来说,他们可能希望检索出感兴趣的具有相同功能的代码.

WB4HPR 提供了快速查看代码文件和注释文件的功能. 不论是类或接口继承关系还是类与接口之间的实现关系,这些类名构成了一个列表,列表是可点击的,开发者只需简单地点击某一个类名,系统

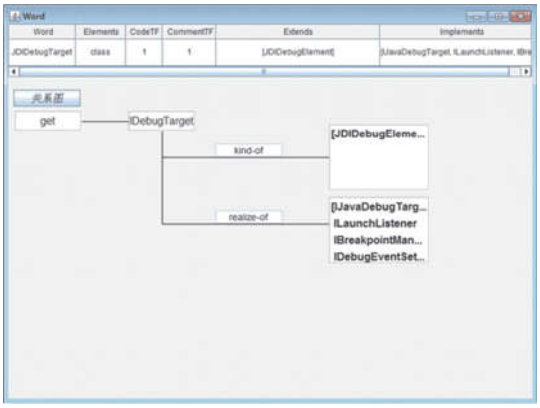

图 4 词组的详细信息与关系可视化图

Fig. 4 Detailed information and relationship visualization

就会提示用户是否查看具体的类文件，有“代码”和“注释”两种选择，开发者自行选择想要查看的类文件. 此外，WB4HPR 提供了直接拷贝功能，开发者可以直接使用他们检索到的感兴趣代码.

2.5 词语的版本演化过程

一个软件系统中可能用到很多词语，这些词语随着软件的不断演化，既有使用频率越来越高的词语，也有使用频率越来越低的词语. 使用频率逐渐增加的词语，可能在软件系统中扮演着越来越重要的角色；而对于使用频率越来越低的词语，也许在未来某一个版本中可能就会消失了. 我们给出单词的演化过程，旨在通过其演化过程告知软件开发者和软件维护者慎用一些词语，给他们一个有益的提示，使得软件系统保持词语使用的一致性；因此，WB4HPR 另一个主要功能是分析出用户所搜索的词语及其相关词随着软件版本的演化而变化的过程，该过程是关于各单词的一个折线图，告知软件开发者搜索词语及其相关词的版本演化特征. 在 WB4HPR 的主界面上，本文提供了“词语演化关系图”按钮，点击该按钮即可查看输入词汇随着软件系统版本的演化而演化的过程，如图 5 所示.

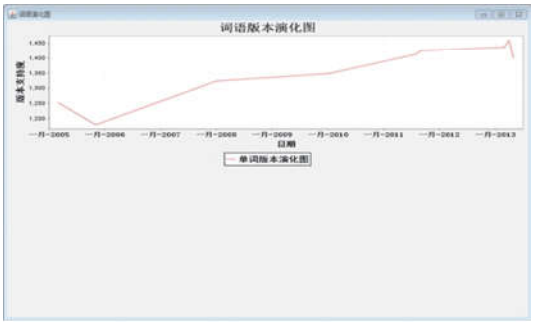

图 5 词语版本演化图

Fig. 5 Version evolution

3 实验案例分析

3.1 WB4HPR 扩展查询的有效性

WB4HPR 构建的词库是基于用户选择的历史代码库，我们使用 eclipse. jdt. debug-R3_0_maintenance 到 eclipse. jdt. debug-R3_8_maintenance 这 9 个版本进行了实验，9 个版本 java 文件代码的总行数达到 2 480 065 行，其中包含 12 273个 java 文件、1 649 427 行的代码和 578 493 行注释，如表 1 所示. WB4HPR 工具从这 2 480 065 行代码中共提取出 71 413 个不重复元素，包括包名、类名、接口名、属性名和方法名；然后 WB4HPR 对这些元素进行词组拆分，共拆分出单词 24 865 个不重复单词.

表 1 各版本代码数目详细情况

Tab. 1 Details of the number of different versions of the code

版本号	代码行数	注释行数	代码总行数	java 文件数
0	157 305	52 189	234 068	1 156
1	160 142	48 391	233 188	1 127
2	177 753	58 433	263 403	1 309
3	185 077	65 991	279 482	1 403
4	188 592	68 083	285 666	1 436
5	191 087	69 463	289 998	1 446
6	191 865	69 383	290 647	1 449
7	195 139	70 917	295 946	1 469
8	202 512	75 643	307 667	1 478
所有版本	1 649 427	578 493	2 480 065	12 273

在人工检索的条件下，从 12 273 个 java 文件中检索目标词汇所花费的时间和精力是巨大的. 在使用 WB4HPR 的条件下，构建一个目标系统的词库花费时间会多一些，但是在词库的基础上检索目标词语的效率会很高. 例如，构建 9 个版本的 Debug 插件历史版本代码库的词库共花费 28 分钟，在构建完词库以后检索一个目标词语的时间小于 3 秒.

在检索过程中，用户输入的词语在目标代码库中可能不存在，使用 WordNet 进行同义词推荐，有时推荐的结果会很多，用户可能需要对推荐的结果一一进行检索，所花费的时间和精力也是巨大的. 一方面，WB4HPR 在 WordNet 推荐的词汇基础上进行了筛选，过滤掉目标历史代码库中不存在的词，明显减少了检索次数，如表 2 中的列“减少检索次数”. 另一方面，WB4HPR 并不完全依赖 WordNet 推荐

的词汇，比如表 2 中的“definition”，WordNet 没有推荐任何同义词，但是该单词在目标历史代码库中存在，所以 WB4HPR 推荐了该词.

表 2　实验测试详细结果

Tab. 2　Detailed experimental results

实验输入	WordNet 推荐	WB4HPR 推荐	推荐数量比	减少检索次数
initialize initialise	initialize initialise format initialise initialize format	initialize format	3/2	1
initialization initialisation	initialization initialisation initialisation initialization	initialization	2/1	1
postion	position put set place pose lay	position put set	6/3	3
watch	watch observe follow view see catch determine checkascertain learn ticker vigil lookout sentinel sentry spotter scout picket	watch catch check determine view	18/5	13
handle	handle manage deal care treat cover plow address palm wield grip handgrip hold	handle address care hold manage	13/5	8
define	define specify delineate delimit delimitate	define specify	5/2	3
defininiition		defininiition	0/1	0
presentation	presentation presentment demonstration display introduction intro	presentation display	6/2	4
change	change alter modify vary switch shift exchange commute convert interchange transfer deepen alterationmodification variety	change convert modification modify shift switch	15/6	9

从表 2 可发现，动词“initialize”与“initialise”以及其名词形式“initialization”与“initialisation”仅仅是字母“s”与“z”不同，词义是相同的，也就是用户在文件中输入“initialise”或者“initialisation”可能检索不到任何结果，而 WB4HPR 推荐的结过中都推荐了“initialize”以及它们的名词“initialization”，显然推荐的效果是令人满意的.

由于程序代码中的标识符很多都不是自然语言中词语，即使是字面相同，词义也可能完全不同，所以 WB4HPR 结合了 WordNet 进行了同义词扩展，WordNet 是通过使用场景来定义词与词之间的同义关系，本工具在进行目标词语的同义词扩展时，使用了目标词语在 WordNet 中的所有动词和名词的使用场景，而且最终 WB4HPR 推荐的词语来自目标历史代码库，所以 WB4HPR 尽可能地降低了使用场景不同导致的检索效率低的问题.

该软件历史代码库是与 debug 有关的一个插件系统，我们在 WB4HPR 输入 point 单词进行检索，检索出相关的元素有 8 个（这 8 个相关元素都为方法名），这 8 个词组的使用情况如表 3 所示，其中 V_0，V_1，…，V_8 分别为版本 0 至版本 8；“0—8”是在 9 个版本中都出现过该元素，“2—8”是只有第二至第八版本中出现过该元素，即 0 版本与 1 版本中没出现过该元素，“4—8”是只有第四至第八版本中出现过该元素，即在 0、1、2、3 版本中没出现过该元素. 由此可以看出，runWatchPointTest 元素使用频率最高，即在该历史代码库中该词组与 point 最相关；testWatchPoint 元素频率最低，即在该历史代码库中该词组与 point 相关度最低.

表 3　与 potint 相关元素使用情况

Tab. 3　The use of point related elements

相关元素（方法）	出现的版本	V0	V1	V2	V3	V4	V5	V6	V7	V8	平均次数
createNestedTypeWatchPoint	2—8	0	0	7	5	8	3	2	2	5	3.6
Point	0—8	3	3	1	2	1	1	1	2	1	1.7
runWatchPointTest	2—8	0	0	12	12	12	15	15	15	12	10.3
testClassLoadBreakPoint	2—8	0	0	2	1	3	1	1	1	8	1.9
testLineBreakPoint	2—8	0	0	5	4	3	5	5	5	3	3.3
testMethodBreakPoint	2—8	0	0	5	4	5	3	3	3	5	3.1
testWatchPoint	4—8	0	0	0	0	1	1	1	1	1	0.6
testWatchPointBreakPoint	2—8	0	0	2	1	2	1	1	1	2	1.1

3.2 WB4HPR 词语演化过程图的有效性

图 6 是词 position 的演化示意，图 7 是词 set 的演化示意. 从图 6 可以看出，position 这个单词随着 debug 插件系统版本的演化，使用频率有明显下降的趋势，可能在未来某一版本中该词语就不再被使用，因此我们会建议软件开发者或软件维护者尽可能地用 position 的其他同义词来替代使用 position. 相反，从图 7 我们可以看出，set 这个单词随着软件系统版本的演化，使用频率有着明显的上升的趋势，我们可以大胆地推测，在最近一段时间或在更长的时间内该词语不会消失，会推荐使用该词汇.

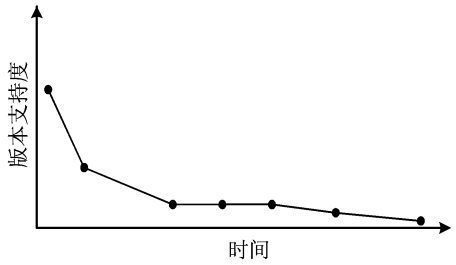

图 6 Position 词语演化图

Fig. 6 Position word evolution

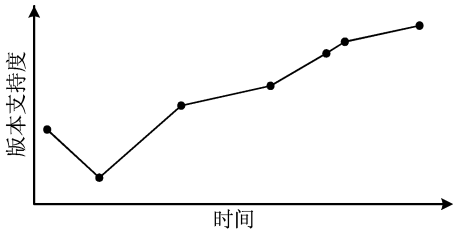

图 7 Set 词语演化图

Fig. 7 Set word evolution

图 6、图 7 是给出单个单词在随着软件历史版本库演化而变化的过程. 另外，我们也给出了与单词有关的相关词组的演化图，如图 8 所示. 它是与单词 point 相关的 8 个词组随着版本的演化.

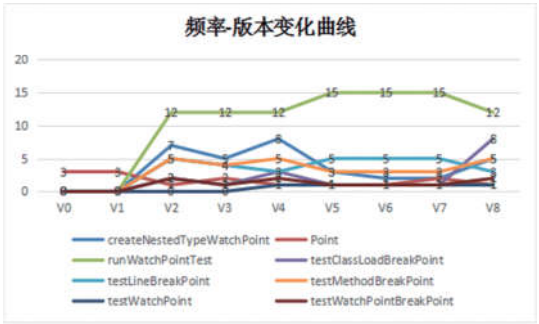

图 8 相关词组的使用演化图

Fig. 8 Evolution of the use of related phrases

4 结论

WB4HPR 致力于构建具体历史代码库词库，在构建出词库的同时，希望能给软件开发者推荐良好的用词建议. 不仅推荐使用词语而且推荐相关词组以及词组之间的关系，从而提供一个相对清晰的关系图供软件开发者参考. 在后期的研究工作中，我们会进一步丰富词语之间的关系图以及具体软件历史代码库中词语的演化过程.

参考文献(References)

[1] YANG J, TAN L. Swordnet: Inferring semantically related words from software context[J]. Empirical Software Engineering, 2014, 19(6): 161-170.

[2] TIAN Y, LO D, LAWALL J. Automated construction of a software-specific word similarity database[C]// IEEE Conference on Software Maintenance, Reengineering and Reverse Engineering. Antwerp, Belgium: IEEE Press, 2014: 44-53.

[3] 钟林辉，宗洪雁. 基于本体的构件化软件演化信息获取及度量研究[J]. 计算机科学，2015，42(1)：196-200.

[4] 钟林辉，谢冰. 构件化软件演化信息建模和获取方法研究[J]. 计算机应用研究，2014，31(2)：401-403.

[5] 施心悦，鲁扬扬，李戈，等. 按需动态组织的知件库系统[J]. 计算机科学与探索，2015，9(6)：660-668.

[6] 林泽琦，赵俊峰，谢冰. 一种基于图数据库的代码结构解析与搜索方法[J]. 计算机研究与发展，2016，533(3)：531-540.

[7] LIN Z, ZHAO J, XIE B. A graph database based crowdsourcing infrastructure for modelling and search [C]// Proceedings of the 6th Asia-Pacific Symposium on Internetware. Hongkong, China: ACM Press, 2014: 15-24.

[8] ABEBE S L, TONELLA P. Natural language parsing of program element names for concept extraction[C]// Proceedings of the 18th International Conference on Program Comprehension. Braga, Portugal: IEEE Press, 2010: 156-159.