Journal of Computer Applications
计算机应用,2018,38(10):2996-3001
ISSN 1001-9081
CODEN JYIIDU
2018-10-10
http://www.joca.cn

文章编号:1001-9081(2018)10-2996-06
DOI:10.11772/j.issn.1001-9081.2018020302

基于版本控制的中文文档到源代码的自动跟踪方法

沈　力[1],刘洪星[1,2],李勇华[1,2*]
(1.武汉理工大学 计算机科学与技术学院,武汉 430063;　2.武汉理工大学 交通物联网技术湖北省重点实验室,武汉 430070)
(*通信作者电子邮箱 liyonghua@whut.edu.cn)

摘　要:软件文档和源代码之间的可追踪性研究广泛使用了信息检索(IR)技术,但由于中文文档和源代码用不同的语言书写,使用传统IR技术进行自动跟踪时会导致精度不高。针对上述问题,提出一种基于版本控制的中文文档到源代码的自动跟踪方法。首先,结合文本到源代码的启发式规则,采用IR方法计算出文本和源代码之间的相似度得分;然后,使用软件开发和维护过程中提交到版本控制软件的更新信息来修正该分数;最后,根据设定的阈值确定中文文档与源代码之间的跟踪关系。实验结果表明,改进方法的精确度和召回率相比传统IR方法均有一定的提高,并且该方法能提取出传统IR方法中遗漏的跟踪关系。

关键词:可追踪性;版本控制;自动跟踪;信息检索;软件工程
中图分类号:TP311.5　　**文献标志码**:A

Automatic tracing method from Chinese document to source code based on version control

SHEN Li[1], LIU Hongxing[1,2], LI Yonghua[1,2*]
(1. *College of Computer Science and Technology, Wuhan University of Technology, Wuhan Hubei* 430063, *China*;
2. *Hubei Key Laboratory of Transportation Internet of Things, Wuhan University of Technology, Wuhan Hubei* 430070, *China*)

Abstract: Information Retrieval (IR) technology is widely used in automatic tracing from software documents to source codes, but Chinese document and source code are written in different languages, which leads to low accuracy of automatic tracing by using IR. In view of the above problems, an automatic tracing method of Chinese document to source code based on version control was proposed. Firstly, the similarity score between the documents and the source code was calculated by information retrieval method combined with text-to-source heuristic rules. Then the score was modified by the version update information which was submitted to the version control software during software development and maintenance. Finally, the tracing relationship between the Chinese document and source code was determined according to the set threshold. The experimental results show that the precision and recall of the proposed method have a certain improvement compared with the traditional IR method, and the tracing relationship missed in the traditional IR method can be extracted.

Key words: traceability; version control; automatic tracing; Information Retrieval (IR); software engineering

0　引言

软件通常被定义为文档和程序的集合,在软件的开发、使用以及维护阶段都会产生大量的文档,而这些文档含有丰富的信息,并且和源代码之间存在紧密的联系。需求跟踪(Requirement Traceability)是指在软件开发周期中向前或向后的描述和跟踪软件开发元素的能力[1]。其中,文档与源代码之间的跟踪关系显得尤为重要,它在程序理解、软件开发、软件维护、变更分析和软件复用等方面有重要的作用,所以建立文档和源代码之间的跟踪链成为软件工程中的重要内容。

文献[2]将信息检索(Information Retrieval, IR)技术与代码的调用关系及数据依赖关系相结合,提高了基于信息检索的自动跟踪能力;但代码依赖关系获取过程比较耗时,并针对了特定的程序语言。文献[3]研究了利用自然语言语义在获取自动化跟踪链时的潜在好处,比较了向量空间模型等几种信息检索及其改进方法,论证了明确的语义方法要好于潜在的语义方法;该方法的使用过程中需要大量语义分析,特殊领域需要特殊领域词典等工具支撑。文献[4]提出了一种基于IR和非文本技术自动结合的需求跟踪方法,首先确定直接的跟踪链,即通过IR方法计算文档与代码之间的跟踪链,通过改进的IR方法计算文档语句之间的跟踪链,通过加权的非文本方法计算代码类之间的跟踪链;然后确定非直接的跟踪链,即根据上一步跟踪链关系建立一个权重矩阵,通过矩阵乘法找出非直接的跟踪链;最后通过设定阈值获取文档与源代码之间的跟踪链。该方法相对于直接耦合的方法提升了精确度。文献[5]在标准用户反馈的方法上提出了一种自适应的反馈方法,该方法利用软件结构以及先前划分的正确链和错误链来决定是否以及如何使用反馈,相对标准的反馈方法表

收稿日期:2018-02-01;**修回日期**:2018-03-29;**录用日期**:2018-03-31。
基金项目:中央高校基本科研业务费专项资金资助项目(2016III028)。
作者简介:沈力(1993—),男,湖北钟祥人,硕士研究生,主要研究方向:需求工程、数据库系统;　刘洪星(1963—),男,湖北洪湖人,教授,博士,主要研究方向:数据库系统、信息系统集成;　李勇华(1977—),男,湖北武汉人,副教授,博士,主要研究方向:需求工程。

现更好;但目前仅测试于规模较小的项目,还需进一步验证。文献[6]分析了在利用文本和代码结构相结合的方法来提取跟踪链过程中的优缺点,提出利用软件工程师在对候选链进行分类时提供的反馈来规范使用结构化信息的方法,结果表明该方法优于纯粹的基于 IR 的方法和结合文本与结构信息的方法,但过程中需要大量的人工干预。文献[7]通过定义 24 个特定的变化场景,通过上一个版本的需求文档、代码和跟踪链与当前版本的需求文档、代码进行比较,得出当前版本的跟踪链;但是该方法局限性比较强,需要上一版本的跟踪链,可操作性比较低。

上述研究大多数是利用基于文本词汇信息的信息检索模型来建立源文件和目标文件之间的跟踪链,而软件文档通常由自然语言书写,源代码由代码语言书写,因此使用传统的信息检索模型建立软件文档和源代码间的跟踪链时,存在精度过低的问题。虽然有的研究注意到代码的结构信息,但忽略了中文文档语义和代码版本更新记录等有效信息,并且当前关于中文文档与源代码间可追踪关系的研究较少。

针对以上问题,本文提出一种基于版本控制的中文文档到源代码的自动跟踪方法。该方法在利用传统的信息检索技术建立中文文档与源代码跟踪链的基础上,加入对源代码版本信息和中文软件文档语义信息的分析,以便提高可追踪性跟踪链的精度。

1　本文方法

基于版本控制的中文文档到源代码的自动跟踪方法结构如图 1 所示。其中包含 4 个主要的处理阶段:数据获取与处理、文本信息检索、版本控制信息处理及跟踪关系推荐,而启发式规则用来辅助进行文本信息检索。该方法在进行文档到代码跟踪关系推荐时,首先需要获取数据并进行预处理,然后结合文档到代码的启发式规则,使用信息检索模型计算文档和源代码间的相似度,同时从版本更新信息中获取描述语句与源代码映射关系文档,计算软件文档与映射关系文档间的相似度,最后结合这两者的相似度得分得出文档到代码的跟踪链列表。

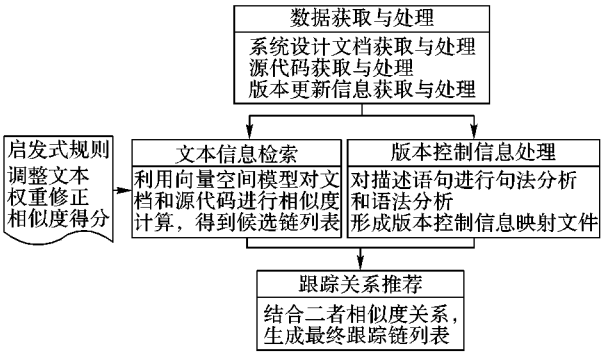


图 1　基于版本控制的中文文档到源代码的自动跟踪方法结构

Fig. 1　Architecture of automatic tracking method from Chinese document to source code based on version control

1.1　相关定义及启发式规则

1.1.1　句法分析与语义分析

中文文档、版本控制信息中的描述语句以及源代码中的注释信息通常是一些简单句句型,而简单句的基本句型由命令式的祈使句和表示条件、位置和修饰动词的状语构成,因此提出如下定义。

定义 1　句法分析。软件工程文档中的简单句 S_{ip} 由 0 ~ N 个介词短语 S_{pp}、0 ~ N 个名词短语 S_{np} 和 1 个动词短语 S_{vp} 构成。句法结构分析规则如下:

$$S_{ip} = S_{vp} \mid S_1 + S_{vp} \mid S_2 + S_{np}$$

$$S_1 = S_{pp} \mid S_1 + S_{pp}$$

$$S_2 = S_{np} \mid S_2 + S_{np}$$

定义 2　语义分析。简单句 S_{ip} 可由四种短语结构成分组成,可以表示成一个多元式的语义结构形式,即 $S_v = (Act, Pla, Dep, Con)$,其中:

Act 为“动作”,由中心谓语动词和中心宾语(即不包含修饰词的部分)组成。

Pla 为“位置”,是指 *Act* 执行的位置,通常是单个名词短语或介词“在”的中心宾语。

Dep 为“凭借”,是指 *Act* 执行的凭借,用来修饰动词,通常是介词“根据”“用”的中心宾语。

Con 为“条件”,是指 *Act* 执行的条件,通常是介词“当”“只有”之后的条件子句。

根据定义 1、定义 2,推导出下列语义分析规则:

规则 1　*Act* 推导规则。

当简单句 S_{ip} 中直接存在动词短语 S_{vp},且 S_{vp} 直接由动词短语 S_{vpl} 或者由其他成分 S_o 和动词短语 S_{vpl} 构成,S_{vpl} 可以是动词 W_v 或动词及名词短语 S_{np} 的形式,那么该动词短语 S_{vpl} 即为 *Act*:

$$\exists S_{vp} \in S_{ip} \wedge (S_{vp} = S_{vpl} \vee S_{vp} = S_o + S_{vpl}) \wedge (S_{vpl} = W_v \vee S_{vpl} = W_v + S_{np}) \Rightarrow S_{vpl} \in Act$$

规则 2　*Pla* 推导规则。

A) 当简单句 S_{ip} 中直接存在名词短语 S_{np} 时,认为 S_{np} 为 *Pla*:

$$\exists S_{np} \in S_{ip} \Rightarrow S_{np} \in Pla$$

B) 当简单句 S_{ip} 中直接存在由介词 W_p 引导的介词短语 S_{pp},其中介词 W_p 属于引导 *Pla* 的介词集合 C_{pp},S_{pp} 由 W_p 和名词短语 S_{np} 构成,有些 S_{pp} 中还存在表示 *Pla* 的名词 W_n,W_n 属于表示 *Pla* 的名词集合,则认为名词短语 S_{np} 属于 *Pla*:

$$\exists S_{pp} \in S_{ip} \wedge (S_{pp} = \{W_p + S_{np} \mid W_p \in C_{pp}\} \vee S_{pp} = \{W_p + S_{np} + W_n \mid W_p \in C_{pp} \wedge W_n \in C_{np}\}) \Rightarrow S_{np} \in Pla$$

规则 3　*Dep* 推导规则。

当简单句 S_{ip} 中存在动词短语 S_{vp},S_{vp} 由介词短语 S_{pp} 和动词短语 S_{vpl} 构成,S_{pp} 由介词 W_p 和名词短语 S_{np} 构成,其中 W_p 处于表示 *Dep* 的介词集合中,则名词短语 S_{np} 属于 *Dep*:

$$\exists S_{vp} \in S_{ip} \wedge S_{vp} = S_{pp} + S_{vpl} \wedge S_{pp} = \{W_p + S_{np} \mid W_p \in C_{pd}\} \Rightarrow S_{np} \in Dep$$

规则 4　*Con* 推导规则。

A) 当简单句 S_{ip} 中直接存在由介词 W_p 引导的介词短语 S_{pp},其中介词 W_p 属于引导 *Con* 的介词集合 C_{pc},S_{pp} 由 W_p 和名

词短语 S_{np} 构成，而 S_{np} 由简单句 S_{ip2}、标记词 W_{dec} 和名词 W_n 构成，其中 W_{dec} 属于标引词集合 C_{dec}，W_n 属于表示 Con 的名词集合中，则简单句 S_{ip2} 属于 Con：

$$\exists S_{pp} \in S_{ip} \wedge S_{pp} = \{W_p + S_{np}, S_{np} = S_{ip}2 + W_{dec} + W_n \mid W_p \in C_{pc} \wedge W_n \in C_{nc} \wedge W_{dec} \in C_{dec}\} \Rightarrow S_{ip2} \in Con$$

B）当简单句 S_{ip} 中直接存在由介词 W_p 引导的介词短语 S_{pp}，其中介词 W_p 属于引导 Con 的介词集合 C_{pc}，S_{pp} 由介词 W_p 和方位词短语 S_{lcp} 构成，S_{lcp} 由简单句 S_{ip2} 和方位词构成，其中 W_{lc} 属于方位词集合，则简单句 S_{ip2} 属于 Con：

$$\exists S_{pp} \in S_{ip} \wedge S_{pp} = \{W_p + S_{lcp}, S_{lcp} = S_{ip2} + W_{lc} \mid W_p \in C_{pc} \wedge W_{lc} \in C_{lc}\} \Rightarrow S_{ip2} \in Con$$

例句 1“在装卸作业预确报界面，当选中船舶单选框时，根据船舶名称和航次查询船舶预确报”。

对例句 1 使用 Stanford-parser 工具[8]进行句法分析，结果呈树形结构，如图 2 所示。其中根节点 IP 表示整个句子；孩子节点表示句子的成分，如 PP 为介词短语，PU 为符号，VP 为动词短语等；叶子节点呈现句子中每个词的词性，如 P 为介词，NN 为名词，VV 为动词等。

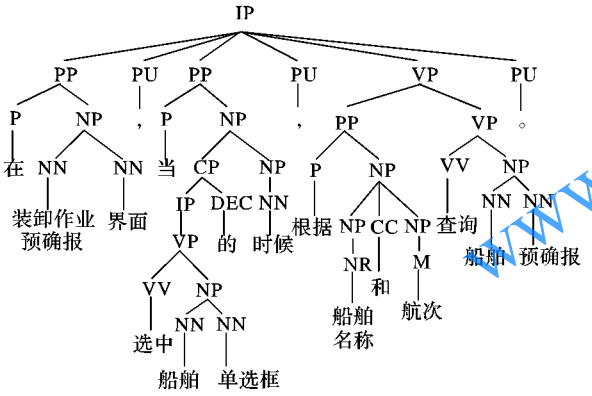


图 2 例句 1 句法结构分析图

Fig. 2 Syntactic analysis of sentence 1

根据语义分析规则，例句 1 转换成对应的短语结构如表 1所示。

表 1 例句 1 语义结构

Tab. 1 Semantic structure of sentence 1

语义成分	语句	推导规则	语义成分	语句	推导规则
Act	查询船舶预确报	规则 1	*Dep*	船舶名称和航次	规则 3
Pla	装卸作业预确报	规则 2B	*Con*	选中船舶单选框	规则 4A

1.1.2 文档到代码启发式规则

当文档或描述语句中存在表“凭借”的介词短语时，该介词短语中名词个数与方法中参数个数相等；当文档或描述语句中存在表“条件”的介词短语时，源代码方法中存在分支结构或循环结构。规则如下：

规则 5 介词短语与代码结构的映射规则。

A）表“凭借”的介词短语：

$$\exists S_{np} \in Dep \wedge (S_{np} = W_n \vee S_{np} = W_n + S_{np}) \Rightarrow num(W_n \mid W_n \in S_{np}) = num(mp)$$

其中：mp 表示方法参数；$num(x)$ 表示 x 的个数。

B）表“条件”的介词短语：

$$\exists S_{ip} \in Con \Rightarrow \exists S_{bra} \vee \exists S_{cir} \in method$$

其中：S_{bra} 表示分支结构；S_{cir} 表示循环结构；$method$ 表示方法。

根据源代码特点，常见动名词和源代码之间存在相应的启发式规则：

规则 6 常见动名词启发式规则如表 2 所示。

表 2 常见动名词启发式规则

Tab. 2 Heuristic rules of common verbs and nouns

关键词	代码启发式规则
查询、增加、新增、添加、插入、保存、…	BLL/DAL 调用、SQL 操作
导入、导出、…	API 调用（EXCEL、FILE 等）、ProgressBar 调用
上传、…	API 调用（OpenFileDialog、FILe 等）、Byte[]
点击、单击、…	函数名（*_Click）
加载、进入、…	函数名（*_Load）
退出、关闭、…	函数名（*_Close、*_Exit）
报表	API 调用（Rdlc、DataSource）
自动刷新、刷新、…	API 调用（Thread）
进入/跳转到…界面、打印、…	API 调用（ShowDialog）

1.2 版本控制信息处理

在软件开发和维护的过程中，通常会使用版本控制软件来辅助开发人员进行代码的管理；软件版本迭代时，将会存在大量的版本更新信息。版本更新信息通常包含版本号（Revision）、作者（Author）、日期（Date）、描述语句（Message）、改动文件的索引（Index），本文方法主要提取版本控制信息中的版本号、描述语句和改动文件的索引。

版本描述语句通常会包含本次代码修改的主要内容，并且基本都是由介词短语、名词短语和动词短语构成的简单句。根据版本控制信息的特点，结合定义 1 和定义 2，可以得出如下假设：

假设 1 在版本描述语句中：

如果存在 Pla，则该 Pla 与类名存在关联关系，即

$$\exists S_{np} \in Pla \Rightarrow \text{conj}(S_{np}, CN)$$

其中 CN 表示类名。

如果存在 Act，则该 Act 与方法名存在关联关系，即

$$\exists S_{vp} \in Act \Rightarrow \text{conj}(S_{vp}, MN)$$

其中 MN 表示方法名。

还存在一些对检索帮助不大的工作性动词。

例如，在例句 1 中，表示 Pla 的名词短语“装卸作业预确报”对应本次版本提交中的类，而表示 Act 的动词“查询船舶预确报”对应本次版本提交中的类中修改的方法。

通过算法 1 得到版本控制信息与源代码之间的映射文件。

算法 1 版本控制信息中描述语句与源代码关联算法。

输入 版本控制信息；

输出 版本控制信息映射文档。

1）提取版本控制信息，对版本更新描述语句进行分词、

去停用词等,提取本次更新相关的代码的类和方法。

2)根据定义 1,并用 Stanford-Parser 工具对其进行句法分析。

3)根据定义 2 对版本描述语句进行语义分析,得出相应的多元式表达形式 (Act, Pla, Dep, Con)。

4)对处理后的版本描述语句运用假设 1 得出和相关类存在关联关系的名词或名词短语,以及和类中相关方法存在关联关系的动词或动词短语,并写入版本控制信息映射文档中。

5)重复步骤 1)~4),如果在写入过程中发现同一类或方法的关键词中已存在相同词语,则将该词语频次加 1,直到所有版本控制信息处理完毕。

6)输出版本控制信息映射文档。

算法 1 的思想是提取版本控制软件中的版本更新关键信息,将版本描述语句中的关键词与本次提交代码的类和方法关联起来,形成版本控制信息映射文件。

例如,使用 svnkit 工具[9] 从版本控制软件中提取版本更新关键信息,并获取本次更新修改的类中的方法,以 XML 文件的形式保存如图 3 所示,包括了版本号和版本更新信息等。

```
<svnlog>
  <id>3800</id>
  <logmes>王诗宇：提运单办理->当件数和重量均
          为0的时候不能保存</logmes>
  <changepath>
    /CTOS/BusinessSystem/
    CargoBill_ADD.cs#buttonAddTYD_Click;
  </changepath>
</svnlog>
```

图 3　版本更新关键信息

Fig. 3　Key information of version updating

对版本更新关键信息进行处理。首先,对 logmes 中版本更新描述语句进行预处理(分词、去停用词等),并运用定义 2 和相关语义分析规则得出版本描述语句的多元组表达形式 $S_v = (Act, Pla, Dep, Con)$,其中,$Act$ = "保存"、Pla = "提运单办理"、Dep = Null、Con = "件数和重量均为 0";然后,提取本次更新关键信息中源代码的相关信息:

类名:CTOS. BusinessSystem. CargoBill_ADD。

方法名:buttonAddTYD_Click。

最后,根据假设 1 得出版本控制信息中描述语句与源代码之间的映射关系如表 3 所示。

表 3　版本更新信息中映射关系

Tab. 3　Mapping of version updating information

描述语句	源代码
提运单办理	CTOS. BusinessSystem. CargoBill_ADD
保存	buttonAddTYD_Click

1.3　文本信息检索及启发式规则运用

文本信息检索主要包括对预处理后的数据构建词频-逆文本频率(Term Frequency-Inverse Document Frequency, TF-IDF)倒排索引表、文本相似度计算两个过程。

构建目标文档索引表的结果是生成一个 $m \times n$ 的标引词-文档矩阵 $\boldsymbol{A}$,m 是所有软件文档中出现的标引词个数,n 是文档的数量,其中第 j 行文档向量 $\boldsymbol{d}_j = (w_{1,j}, w_{2,j}, \cdots, w_{n,j})$,$w_{i,j}$ 表示第 i 个标引词在第 j 个文档中的权重,权重计算采用 TF-IDF 模式与启发式规则结合的方式。

采用向量空间模型对文档和源代码进行相似度计算,将每一条文档语句当作查询向量,设查询向量 $\boldsymbol{q} = (w_1, w_2, \cdots, w_n)$,将所有源代码文档当作目标查询文档,对查询向量和源代码文档向量 $\boldsymbol{d}_j$ 计算余弦相似度,如式(1)所示,其中 w_i 为 $\boldsymbol{q}$ 中标引词的权重,$w_{i,j}$ 为 $\boldsymbol{d}_j$ 中标引词的权重。

$$sim(\boldsymbol{q}, \boldsymbol{d}_j) = \frac{\boldsymbol{q} \times \boldsymbol{d}_j}{\|\boldsymbol{q}\| \times \|\boldsymbol{d}_j\|} = \frac{\sum_{i=1}^{N} w_i w_{i,j}}{\sqrt{\sum_{i=1}^{N} w_i^2} \sqrt{\sum_{i=1}^{N} w_{i,j}^2}} \quad (1)$$

对查询文档向量和源代码文档向量分别进行相似度计算,得到相似度列表。对每一条查询语句及存在相似度关系的源代码采用启发式规则,对相似度得分进行修正。

1.4　跟踪关系推荐

根据以上分析,本文提出如下关联关系提取算法。

算法 2　基于版本控制信息的文档到源代码关联关系提取算法。

输入　项目文档、源代码、版本控制信息映射文档;

输出　文档与源代码文件的相似度列表。

1)对文档进行文本提取,以句子为最小粒度进行切分,并进行预处理(分词、去停用词)后,存入 XML 文件中。

2)提取项目源代码,以方法为最小粒度,以索引 + 方法文件的形式保存。

3)提取源代码中类名、方法名以及注释内容,对类名和方法名进行词型规范化,并映射成文字,对注释内容进行预处理。结合处理后的类名、方法名和注释内容形成表示方法的关键信息。

4)使用向量空间模型结合代码检索规则对文档条目和源代码关键信息进行相似度计算,得出文档和源代码间的相似度 $Score1$。

5)计算文档与版本控制信息映射文档之间的相似度 $Score2$。

6)分别赋予步骤 4)和步骤 5)计算出的相似度不同的权重值 (a, b),不同权重下两者之和即为文档与源代码文件间的相似度 $Score = aScore1 + bScore2$。

7)以文件的形式输出相似度列表。

算法 2 的思想是利用向量空间模型计算出中文文档和源代码间的相似度得分,然后根据启发式规则进行修正,再结合源文档和版本控制信息映射文件之间的相似关系,得出文档与源代码文件的相似度列表。由于版本控制信息中的描述语句与本次提交的源代码之间具有较高的相关性,故文档与版本控制信息映射文档间的相似度具有较大的可信度,从而 a 的取值应小于 b,故选取 a 的取值范围为 $[0, 0.5]$,b 的取值范围为 $[0.5, 1]$。由于不同文档的书写风格以及涉及到的软件实现方式不同,a 和 b 的取值均不是固定值,需要通过训练集训练得到,步骤如下:

1)从测试数据集中随机选取若干数据作为训练集,一般为数据集的前 5% 或不少于 5 条数据;

2)对训练集开始训练,取初始值 $a = 0, b = 1$,步长 $\lambda = 0.05$ 开始计算所有 a、b 取值下的精确度和召回率;

3)选取训练结果中精确度和召回率最高的 a、b 取值作为实验权重值。

例如,从中文文档中提取语句"在装卸作业预确报界面,当选中船舶单选框时,根据船舶名称和航次查询船舶预确报",对其进行预处理,得出多元组表达形式如表 1 所示。提取项目相关源代码关键信息,以索引(GY. SDD. 4) + 文本(4. txt)的形式保存,提取关键信息:

类名:CTOS. BusinessSystem. YQ_Main。

类注释:装卸作业预确报界面及相关功能。

方法名:queShipYQB。

方法注释:船舶预确报查询。

提取由算法 1 得出的版本控制映射文档中相关信息:

类名:CTOS. BusinessSystem. YQ_Main。

关键词:装卸作业预确报#2、预确报#5。

方法名:queShipYQB。

关键词:查询船舶预确报#1。

计算出文档与源代码之间的相似度得分 $Score1 = 0.7599$,文档与版本控制信息映射文档之间的相似度得分 $Score2 = 1$,通过训练集训练得出权重最优值 $a = 0.35, b = 0.65$,综合得分 $Score = 0.9160$。

2 实验

2.1 实验数据集

由于目前公开的相关实验数据集大多针对英文文档,并且无法获取其完整的版本更新记录,故实验采用作者参与开发的、现已正式运行的重庆果园港生产业务管理系统作为实验对象,选取软件设计文档作为源素材,软件源代码作为目标素材,并提取了软件开发及维护过程中版本控制软件中的版本更新记录,作为跟踪方法的测试集。

经过粒度划分等处理,共得到具有实际含义的设计文档语句 857 条,提取了项目类文件 365 个,这些类中包含的方法 2872 个,从版本控制软件中提取了系统开发和维护过程中版本更新信息 2806 条。由于软件文档和代码数量较多,本实验从系统众多功能模块中选取了商务模块的源素材,通过人工整理出了 38 条具有关联关系的跟踪链作为实验结果集。

2.2 实验过程

1)源素材处理。

对素材中提取出的中文语句进行预处理,包括分词、去停用词等。本实验采用 ICTCLAS(Institute of Computing Technology, Chinese Lexical Analysis System)词法分析系统[10],提取系统中的数据库表名和字段名等作为用户词典,对中文素材进行分词。采用哈尔滨工业大学发布的中文停用词表,去掉一些对文本内容识别作用不大的词。

2)源素材和目标素材文本相似度计算。

对中文语句采用 Stanford-Parser 进行句法分析,根据定义 2及相关规则对句法分析后的语句进行语义分析,并采用 TF-IDF 模型和代码检索规则对源素材(设计文档语句)和目标素材(源代码关键信息)进行权重分配,利用向量空间模型计算其相似度得分,根据规则 5 和规则 6 判断源素材和目标素材中源代码结构之间的关系,从而修正源素材和目标素材之间的相似度得分 $Score1$。

3)版本控制信息处理及源素材和版本控制信息映射文档间相似度计算。

根据假设 1 得出版本控制信息映射文档,计算源素材中 *Pla* 和 *Act* 与版本控制信息映射文档中类与方法相关的关键词之间的得分,综合得出源素材和版本控制信息映射文档之间的得分 $Score2$。

4)基于版本控制信息的中文文档和源代码间的相似度计算。

根据算法 2 中权重值 a、b 的确定规则,从实验数据中选取 8 条中文文档语句和所有源代码素材,并给出正确的跟踪链关系,作为训练集;然后,取初始值 $a = 0, b = 1$,步长 $\lambda = 0.05$,采用 a 递增、b 递减的方式进行训练;最后,得出最优取值 $a = 0.35, b = 0.65$。最终得分 $Score = aScore1 + bScore2$,根据分数排序得出相似度列表。

2.3 结果分析

本实验与传统信息检索方法中的向量空间模型(Vector Space Model, VSM)方法[11]及文献[4]方法进行对比,采用相似度阈值法过滤实验结果,采用信息检索中最通用的查准率[11]、查全率[11]以及 F_2measure[12]来度量不同方法所得出的跟踪关系的质量。相似度阈值法即为设定一个相似度分数阈值,当实验结果分数大于该阈值时才选取该结果,小于该阈值时则舍弃,而查准率和查全率均在选取的结果中计算,因此实验结果中的查全率会随着阈值的增大而降低,查准率会随着阈值的增大而升高。在本实验中,当阈值大于 0.2 时,三种方法查准率虽然会有所提高,但是查全率显著降低,F_2measure 的值也会显著降低,对比意义不大,因此本文选取 0.2 作为最大阈值,实验结果的平均值如表 4 所示。

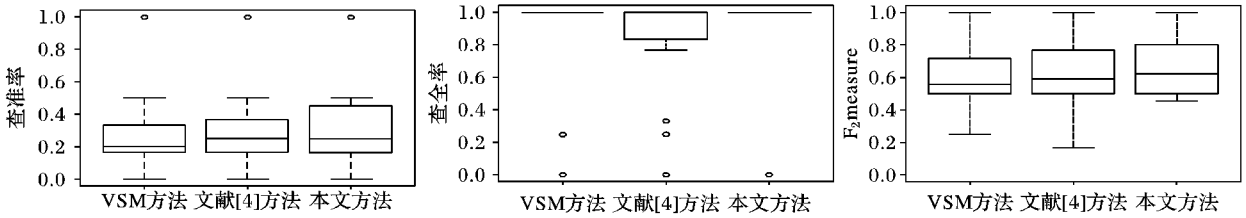
表 4 实验结果平均值

Tab. 4 Average value of experimental result

阈值	评价指标	VSM 方法	文献[4]方法	本文方法
0.1	查准率	0.1583	0.1557	0.1745
	查全率	1.0000	1.0000	1.0000
	F_2measure	0.4560	0.4486	0.4691
0.2	查准率	0.2930	0.3143	0.3165
	查全率	0.9500	0.8740	0.9714
	F_2measure	0.5948	0.5980	0.6340

由于实验方法不同,三种实验在相同阈值下的横向对比意义不大。根据实验结果平均值可以看出,三种方法的综合评判指标 F_2measure 值均在阈值为 0.2 达到最大,为了更好地说明实验效果,选取阈值为 0.2 时的实验数据,使用 R 工具[13]自动生成箱线图,从图 4 所示的箱线图可以清晰地显示数据的离散度、中间值等,图中离散的圆点表示数据的异常值,上下两条横线分别为数据的上下边缘,表示数据的整体范

围;中间矩形的上下两边为上下四分位数,表示数据集中在这一范围中;矩形中的横线表示数据的中位数,表明数据总体的中间值。从图 4 可以看出,在查准率方面,三种方法的整体范围和下四分位数均相同,本文方法的上四分位数高于其他两种方法,中位数与文献[4]方法相同,略高于 VSM 方法;在查全率方面,三种方法中位数相同,文献[4]方法浮动范围稍大;在 F_2measure 方面,三种方法下四分位数相同,本文方法中位数高于其他两种方法,并且本文方法的数据集中区域略高于其他两种方法。结合实验数据和箱线图可以看出在查全率方面本文方法与其他两种方法相差不大,在查准率和综合评价指标方面,本文方法在中间值和整体范围方面均优于其他两种方法。



(a) 查准率　(b) 查全率　(c) F_2measure

图 4　实验结果箱线图

Fig. 4　Box-plot of experimental result

3　结语

本文提出了一种基于版本控制信息的中文文档到源代码的自动跟踪方法。该方法首先提取以句子为最小粒度的中文文档及以方法为最小粒度的源代码关键信息,结合自定义的文档到代码的启发式规则,利用向量空间模型计算出中文文档与源代码之间的相似度得分;然后结合中文文档与版本更新信息之间的相似度关系,调整中文文档与源代码之间的相似度得分,确定中文文档与源代码之间的跟踪关系。对于相同的测试对象,本文方法相对于传统的基于信息检索的方法提高了精确度和召回率;但是当中文文档中存在大量的结构化图形描述时,本文方法效果不太理想,这是今后需要研究的内容。

参考文献(References)

[1] GOTEL O C Z, FINKELSTEIN A C W. An analysis of the requirements traceability problem [C]// Proceedings of the 1994 International Conference on Requirements Engineering. Piscataway, NJ: IEEE, 1994: 94 – 101.

[2] KUANG H, NIE J, HU H, et al. Analyzing closeness of code dependencies for improving IR-based traceability recovery [C]// Proceedings of the 2017 International Conference on Software Analysis, Evolution and Reengineering. Piscataway, NJ: IEEE, 2017: 68 – 78.

[3] MAHMOUD A, NIU N. On the role of semantics in automated requirements tracing [J]. Requirements Engineering, 2015, 20(3): 281 – 300.

[4] JYOTI, CHHABRA J K. Requirements traceability through information retrieval using dynamic integration of structural and co-change coupling [C]// Proceedings of the 2017 International Conference on Advanced Informatics for Computing Research. Berlin: Springer, 2017: 107 – 118.

[5] PANICHELLA A, LUCIA A D, ZAIDMAN A. Adaptive user feedback for IR-based traceability recovery [C]// Proceedings of the 2015 International Symposium on Software and Systems Traceability. Piscataway, NJ: IEEE, 2015: 15 – 21.

[6] PANICHELLA A, McMILLAN C, MORITZ E, et al. When and how using structural information to improve IR-based traceability recovery [C]// Proceedings of the 2013 European Conference on Software Maintenance and Reengineering. Piscataway, NJ: IEEE, 2013: 199 – 208.

[7] RAHIMI M, GOSS W, CLELANDHUANG J. Evolving requirements-to-code trace links across versions of a software system [C]// Proceedings of the 2016 International Conference on Software Maintenance and Evolution. Washington, DC: IEEE Computer Society, 2016: 99 – 109.

[8] The stanford NLP group. The stanford parser: a statistical parser [EB/OL]. [2018-01-15]. https://nlp.stanford.edu/software/lex-parser.shtml.

[9] SVNKit [EB/OL]. [2018-01-15]. https://svnkit.com/.

[10] 中国科学院计算技术研究所. NLPIR 汉语分词系统[EB/OL]. [2018-01-15]. http://ictclas.nlpir.org/. (Institute of Computing Technology Chinese Academy of Sciences. Chinese lexical analysis system[EB/OL]. [2018-01-15]. http://ictclas.nlpir.org/.)

[11] MANNING C D, RAGHAVAN P. Introduction to Information Retrieval [M]. Cambridge: Cambridge University Press, 2010: 76 – 92.

[12] LI Y, CLELANDHUANG J. Ontology-based trace retrieval[C]// Proceedings of the 2013 International Workshop on Traceability in Emerging Forms of Software Engineering. Piscataway, NJ: IEEE, 2013: 30 – 36.

[13] R core team. The R project for statistical computing [EB/OL]. [2018-01-15]. https://www.R-project.org/.

This work is partially supported by the Fundamental Research Funds for the Central Universities (2016III028).

SHEN Li, born in 1993, M. S. candidate. His research interests include requirement engineering, database system.

LIU Hongxing, born in 1963, Ph. D., professor. His research interests include database system, information system integration.

LI Yonghua, born in 1977, Ph. D., associate professor. His research interests include requirement engineering.