# 基于相关主题模型的程序网络自动构建与分析

孙小兵[1,2],刘湘月[1],李 斌[1,2],张伟佳[1]
(1.扬州大学信息工程学院,江苏扬州 225127;2.南京大学计算机软件新技术国家重点实验室,江苏南京 210023)

**摘 要:** 程序理解的目的在于获得足够的软件系统信息,以适用于人理解的形式展现出来,辅助开发人员对软件的理解.本文通过使用相关主题模型,为软件系统类层次的代码文件建立程序网络,并可视化展示整个软件系统的相关结构和功能,辅助开发者理解整个程序代码.该技术综合考虑了软件代码中的结构性信息和内容性信息,所建立的程序网络可帮助开发者更好的理解程序的语法依赖关系和语义功能相关关系.实验验证了建立的程序网络具有较好的准确性以及可以为指定的类推荐相关类.

**关键词:** 程序理解;相关主题模型;程序网络
**中图分类号:** TP311 **文献标识码:** A **文章编号:** 0372-2112 (2017)05-1052-05
**电子学报 URL:** http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.05.004

# On Automatic Construction and Analysis of Program Network via Relational Topic Model

SUN Xiao-bing[1,2], LIU Xiang-yue[1], LI Bin[1,2], ZHANG Wei-jia[1]
(1. *School of Information Engineering, Yangzhou University, Yangzhou, Jiangsu* 225127, *China*;
2. *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, Jiangsu* 210023, *China*)

**Abstract:** Program comprehension aims to obtain enough information in the software system to promote the comprehension of the target software. This paper proposes a novel technique, which uses relational topic model (RTM) to model code (class-level) documents in the software system into a program network. Then, the program network is visualized to help developers understand the whole software. The advantage of RTM is that it takes into account both the structural and textual information in the software system, which enables developers to fully understand the syntax dependence and semantic functional relationship in the program. The empirical results show that the program network is more accurate to model the relation among different classes, moreover, it is able to recommend relevant classes for a given class to understand a local part in the program.

**Key words:** program comprehension; relational topic model; program network

## 1 引言

软件维护是软件生存期中时间最长,消耗人力最多的一个阶段[1].软件维护通常需要对持久演化的大型遗留软件系统进行理解和再开发,这要求开发人员能够充分的理解软件项目.而软件理解对于开发者来说,需要耗费很多的时间和精力,特别是程序源码,从而导致维护任务变得相当困难,代价高并且很可能出现偏差[2,3].

软件系统经过长时间的运行和升级,使得遗留软件系统中包含了众多杂乱的信息,并且在此期间一个软件项目可能经历了很多不同的程序员,各种各样的编程风格,代码难以避免的变得复杂和难以理解.因此,能有效辅助软件理解的主要是依赖于当前系统所对应的程序代码.

程序代码的理解通常包括两方面的理解,一方面

收稿日期:2015-12-25;修回日期:2016-05-25;责任编辑:马兰英
基金项目:国家自然科学基金(No.61402396,No.61472344);江苏省教育厅自然科学基金(No.13KJB520027);南京大学计算机软件新技术国家重点实验室开放基金(No.KFKT2016B21);江苏青蓝工程

是程序语义的理解;另一方面是程序语法的理解. 程序语义主要关注程序中有哪些功能以及这些功能间的相关性;程序语法主要关注程序中的各个功能是怎么实现的. 本文使用相关主题模型[4]建立程序网络,该程序网络建立了程序元素(类)间的语法和语义关系,并以可视化的角度展示出软件程序中类的结构,为开发者理解程序提供帮助. 相关主题模型是一个关于文件属性和网络结构的模型,为分析与理解文档网络提供便利,该模型同时考虑到了文档的内容与文档间的联系. 因此,使用相关主题模型创建的网络中的边既考虑到类之间结构性(语法)的联系,也将类之间功能上的联系(语义)展示出来,为理解与维护软件系统提供新的条件.

## 2　相关主题模型

相关主题模型是一个关于文件属性和网络结构的模型,为分析与理解文档网络提供帮助. 此外,给出一个新节点及其联系,相关主题模型可以提供预测性的关于结点属性的分布情况.

相关主题模型在各个领域已有成功的应用,包括识别社交网络中潜在的好友,得出一个科学性论文的所有引用,定位一个网页的相关网页[4],以及在软件工程中也有一定的应用,如 Bavota 等人提出了利用相关主题模型进行重构识别的方法,该方法从语法和语义的角度为方法之间建立联系从而辅助方法的重构,识别模块化的类[5]. Gethers 等人将类用相关主题模型建模后的联系结果作为新型的耦合衡量标准,然后基于该耦合结果支持修改影响分析[6]. Liu 等人利用相关主题模型为软件系统建立基于语义的理解模型[7]. 而本文将基于语法和语义利用相关主题模型建立类之间的关系程序网络,辅助程序理解.

## 3　使用相关主题模型建立程序网络

相关主题模型的输入有两个,一是文档的语义信息;另一个是文档的结构信息. 本文以程序中的每个类作为文档. 本文建立程序网络分为以下几步(如图 1 所示):(1)预处理:从软件系统的类中提取需要的信息;(2)相关主题模型建模;(3)生成程序网络;(4)推荐相关类.

### 3.1　预处理

为得到软件系统中准确的语义信息,要对源代码进行预处理,去除一些冗余的信息和噪音,通常使用自然语言处理技术进行预处理[8]. 预处理主要对非结构化的源代码数据中提取的数据包括注释和标识符,按照预处理的流程——词条化、去除停用词、分词、词形归一化、词干还原并将源代码内容将提取出来的数据进行处理,这样得到预处理之后的数据.

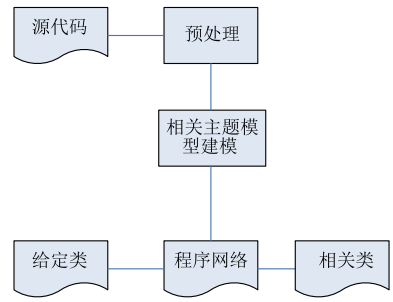

图1　使用相关主题模型建立程序网络

### 3.2　相关主题模型建模

使用相关主题模型建模前,需提取出代表类内容的语义信息和代表类关系的语法结构信息,其中语义信息的形式为词频矩阵,结构信息是代码元素之间的依赖关系矩阵;最后,基于这些关系使用相关主题模型生成生成联系概率矩阵.

#### 3.2.1　语义信息

语义信息代表了文档的内容,是从文档中提炼出的可以代表该文档的词汇. 上面预处理之后的信息被存储在一个 $M \times N$ 的矩阵中,这个矩阵称作词频矩阵;其中,$M$ 是出现在类中的单词的数量,$N$ 是软件系统中类的数量. 在本文中使用该矩阵作为文档的语义信息.

相关主题模型会计算词频矩阵中单词的权重,以求得能代表类文档的主题集. 相关主题模型使用的衡量算法称作 tf-idf 算法[4]. tf-idf 用以评估一单词对于一个文件集或一个语料库中的其中一份文件的重要程度. *tf* 描述的是文档中词出现的频率;而 idf 是和词出现文档数相关的权重. 使用 tf-idf 算法处理过的词频矩阵将作为代表软件系统的语义信息的模型进行处理.

#### 3.2.2　结构信息

本文参照 Bavota 等人[5,6]考虑了类与类之间两种语法结构关系:(1)类之间结构性依赖关系(例如一个类使用了另一个类)(2)初始的系统设计(例如两个类在同一个包里[9]). 后者是一个简单的布尔型 $N\ x\ N$ 矩阵,如果类 $C_i$ 与类 $C_j$ 在同一个包中,那么矩阵中该元素 $O_{i,j}$ 等于 1,否则 $O_{i,j}$ 等于 0.

$$O_{i,j} = \begin{cases} 1, & C_i.\,package = C_j.\,package \\ 0, & C_i.\,package \neq C_j.\,package \end{cases} \tag{1}$$

我们用使用依赖度度量类之间的结构性依赖关系[5],使用依赖度考虑的是类之间的使用;然后根据该关系生成使用矩阵.

$$Di->j = \begin{cases} \dfrac{\text{uses}(C_i, C_j)}{\text{used}(C_j)}, & \text{used}(C_j) \neq 0 \\ 0, & \text{used}(C_j) = 0 \end{cases} \tag{2}$$

$\text{uses}(C_i, C_j)$ 是类 $C_i$ 使用 $C_j$ 的次数,$\text{used}(C_j)$ 是类

$C_j$ 使用其他类的总次数. $D_{i->j}$的范围介于 0 到 1 之间,使用依赖度 $D$ 越接近 1,类 $C_i$ 与类 $C_j$ 间的结构性依赖越紧密.

这些结构性分析方法得到的类间的联系将作为相关主题模型所需要输入的边. 相关主题模型使用词频矩阵来描述类中的主题分布,其中每一个主题被一个单词的概率分布描述,并使用与主题最相关的单词集代表该主题.

### 3.3 生成程序网络

本方法以类为粒度为系统建立网络,类对应于网络中的结点,相关主题模型预测出的类之间的联系对应于网络中的边. 根据上述预处理得出的三个矩阵作为参数,用于建立相关主题模型,生成程序网络.

**定义 1**(程序网络):我们定义程序网络 $PN = <N, L>$,$N$ 是网络中的结点,$L$ 是结点之间的边,结点和边定义如下:

**结点($N$)**:每个类将作为程序网络的结点. 而预处理之后的类的内容作为结点的属性. 所以结点 $N$ 包含两个部分,即 $N = <classname, content>$,*classname* 即为类的名称,*content* 即为预处理之后的类的内容.

**边($L$)**:程序网络中的边表示两个类之间的联系. 相关主题模型会根据输入的词频矩阵和使用矩阵得到联系概率矩阵 *Prob*. 程序网络中类之间的联系对应于图中的边,该边由概率矩阵 *Prob* 确定,其值是两个类间有联系的概率. 给定一个概率阈值,若 $Prob_{i,j}$大于该值,就认为该联系存在,即类 $C_i$ 和类 $C_j$ 间有边,否则认为类 $C_i$ 和类 $Cj$ 之间不存在边. 因此,$L = <Node1, Node2, value>$,当结点 *Node*1 和结点 *Node*2 之间有边,则 *value* 的值为 1,反之为 0.

### 3.4 推荐相关类

相关主题模型是一个综合了文档文本信息和结构信息的网络模型,使用相关主题模型构建的程序网络具有以下功能:

(1)程序网络中潜藏着大量结构信息和内容信息的网络,可帮助开发者从软件系统整体布局和框架上理解软件系统.

(2)如同为社交网络里的用户推荐好友,程序网络可以推荐出与指定类相关的类(即网络中与指定类有边的类),帮助开发者从其感兴趣的局部理解程序.

## 4 实验评估

本节通过实验来检验我们方法的有效性,实验问题如下:

**问题 1** 基于程序网络推荐的类是否有效?

**问题 2** 与已有程序依赖图[10]相比,本文程序网络是否更能辅助程序员对程序的理解?

### 4.1 实验设置

#### 4.1.1 实验对象

JHotDraw①:一个用于支持用 Java 开发的图形编辑器,共 299 个类.

JFreeChart②:一个 Java 平台上开放的图表绘制类库,共 990 个类.

#### 4.1.2 实验验证过程

在对相关主题模型进行参数设置时参考 Gethers 和 Poshyvanyk 的工作[5],而建立程序网络中的边时,取概率阈值为 0.7.

我们选取 10 位参与者以人工的方式挑选出与指定类有联系的类. 在事先不了解实验对象的情况下,将类文件分配给他们,他们需要找出这些类的相关类. 每个参与者都会给出所指定类文件的一个结果. 但不同的参与者可能给出不同的结果,他们进行讨论并一致决定最终的相关类结果. 我们将这个最终的相关类结果作为权威结果来衡量我们的方法以及基于依赖图的结果. 对于基于传统的依赖图技术进行相关类推荐时,采用程序切片技术计算相关类[10]. 由于本文中程序网络是建立的类层次的程序网络,因此使用程序依赖图时,也使用类层次的程序依赖图,主要根据类与类之间的继承、使用等关系,从程序语法的角度建立类层次依赖图,然后使用切片技术得到关于某个类的切片结果. 为了定量比较这两种方法,采用查准率和查全率来衡量[11]. 查准率主要衡量某个推荐技术所得出的相关类结果中的类文件是否真的相关(基准为权威结果),查全率衡量的是通过我们的方法所得的相关结果在权威聚类中的类文件的比重.

### 4.2 实验结果

**问题 1** 问题 1 考察的是根据相关主题模型生成的程序网络所推荐的类的有效性. 从 JHotDraw 和 JFreeChart 中各抽取 5 个类,选取人工的方式从类中挑选出指定类的有联系的类作为权威结果,并与相关主题模型推荐的相关类与权威结果进行比较,并查看权威结果在推荐类的排名,表 1 和表 2 是其中两个类的推荐比较结果,表 3 为 10 个类的平均排名.

**表 1** JHotDraw 中的 Colors 类

| Colors 类 | 推荐中是否包含 | 联系概率值 | 概率值排名 |
|---|---|---|---|
| ColorChooserAction | 是 | 0.796290574469405 | 4 |
| ColorAttributeSet | 是 | 0.881806858908308 | 1 |
| ColorIcon | 是 | 0.779065774378949 | 5 |

① http://sourceforge.net/projects/jhotdraw

② http://sourceforge.net/projects/jfreechart

表 2　JFreeChart 中的 ChartEntity 类

| ChartEntity 类 | 推荐类中是否包含 | 联系概率值 | 概率值排名 |
|---|---|---|---|
| ChartEditor | 是 | 0.910358381725662 | 2 |
| ChartTheme | 否 | 0.660073530443734 | 无 |
| ChartDeleter | 是 | 0.800349542714698 | 5 |

表 3　权威结果在推荐类中平均排名

| JH1 | JH2 | JH3 | JH4 | JH5 | JF6 | JF7 | JF8 | JF9 | JF10 |
|---|---|---|---|---|---|---|---|---|---|
| 3.3 | 3 | 7 | 3.5 | 4 | 3.5 | 2 | 5 | 4.2 | 9 |

表 1 中类 Colors 是关于颜色的一个模型类,定义了关于颜色的属性、方法等,人工选出的 3 个相关类 ColorChooserAction、ColorAttributeSet 和 ColorIcon 均出现在了程序网络推荐的相关类中,其中联系概率值最高的是 ColorAttributeSet 类,原因是该类是对颜色属性进行设置操作的类,类中会多次出现 Colors 中属性或者调用其方法,从而在结构和内容上都产生关联. 表 2 中的类 ChartEntity 是关于图表实体的类,是对图表的特征进行描述的类,我们挑选出的相关类 ChartEditor 和 ChartDeleter 出现在了程序网络推荐的相关类中,但是 ChartTheme 却没有被推荐. 在分析了 ChartTheme 的代码后,发现其功能是给图表设置主题,但是和 ChartEntity 类不在同一个包中,导致了其概率值偏低. 表 3 中,从所有的 10 个类的推荐结果的平均排名来看,每个类的权威结果在各个类的推荐类中排名平均值在前 10 的推荐类中.

**问题 2**　问题 2 主要进行了实验比较研究. 根据问题 1 中 JHotDraw 和 JFreeChart 中各抽取 5 个类进行实验,利用程序网络技术和程序依赖图技术分别得到这 10 个类所推荐的类,并计算它们的查准率和查全率,在计算查准率和查全率时,都是选择各个技术中前 10 个类作为推荐结果,实验结果如表 4 所示.

表 4　程序网络技术和程序依赖图技术推荐类的查准率和查全率

| 实验对象 | 程序网络技术 | | 程序依赖图技术 | |
|---|---|---|---|---|
| | P | R | P | R |
| JHotDraw | 0.34 | 0.58 | 0.22 | 0.32 |
| JFreeChart | 0.28 | 0.62 | 0.26 | 0.30 |

从表 4 结果可以看到,两个技术的查准率基本差不多,但程序网络技术所推荐类的查全率比程序依赖图技术更高. 而查准率评估的是所推荐的类的准确性,从实验结果中可看到,程序网络和程序依赖图技术都能推荐一些相关的在权威结果中的类. 而查全率是所推荐的类能覆盖权威结果中类的百分比,从实验结果中可看到程序网络技术能推荐更多的覆盖权威结果的类. 从程序理解的角度出发,如能推荐更多准确的相关类,有助于开发人员理解程序[9].

## 5　相关工作

当前程序理解方面的研究较多,特别是对代码中的语法依赖关系进行分析,将其转化为结构化的依赖图进行分析,包括构建程序依赖图技术[12]、切片技术[10]、程序聚类技术[13]等. 这些技术主要从程序语法的角度理解软件如何实现一些功能. 但对软件语法的理解并不能有效辅助理解软件语义功能方面的特性.

软件程序中除了有语法方面的依赖关系,还有一些标识符变量的定义和注释,这些标识符和注释具有一些语义方面的依赖关系. 当前已有一些研究对软件程序的语义进行了分析. Thomas 等人利用文本挖掘技术中的主题模型技术对程序代码中的标识符和注释进行分析,提取它们的主题,通过分析主题理解程序代码的语义功能[14]. Liu 等人通过使用层次主题模型技术挖掘程序,将程序分解为从粗粒度到细粒度的层次化语义树[15].

而本文同时考虑了程序中的语法和语义关系,提出了一种使用相关主题模型建立程序网络的方法来辅助程序的理解. 使用相关主题模型创建的程序网络中的边既考虑到类之间结构性(语法)的联系,也考虑了类之间功能上的联系(语义).

## 6　结论与展望

本文提出了一种使用相关主题模型建立程序网络的方法来辅助程序的理解,该程序网络同时考虑了软件程序中的结构性和内容性信息. 通过两个实验对象进行了实验分析,实验结果表明程序网络能够有效地推荐给定类的相关类供开发人员去理解感兴趣的程序.

接下来,我们将考虑从多个角度建立程序网络,例如基于程序的文档,这些文档对于开发者理解程序同样重要. 其次,在利用相关主题模型进行程序网络的构建时,会有一系列的参数需要设置,后期我们将研究如何基于不同的实验对象自动设置相应的更加合适的参数进行程序网络的构建.

### 参考文献

[1] Rajlich V. Software evolution and maintenance[A], Proceedings of the on Future of Software Engineering[C]. USA:ACM,2014. 133 - 144.

[2] Obrien M P. Software comprehension: A review and research direction[R]. Tech. rep. (2003).

[3] Rajlich V, Wilde N. The role of concepts in program comprehension[A]. Proceedings of the 10th International Workshop on Program Comprehension[C]. USA: IEEE,

2002. 271 – 278.
[4] Chang J, Blei D. Relational topic models for document networks[A]. Proceedings of the Artificial Intelligence and Statistics[C]. USA: MIT Press, 2009. 81 – 88.
[5] Bavota G, Oliveto R, Gethers M, Poshyvanyk D, De Lucia A. Methodbook: Recommending move method refactorings via relational topic models[J]. IEEE Transactions on Software Engineering, July 2014, 40(7): 671 – 694.
[6] Gethers M, Poshyvanyk D. Using relational topic models to capture coupling among classes in object-oriented software systems[A]. Proceedings of 26th IEEE International Conference on Software Maintenance[C]. USA: IEEE, 2010. 1 – 10
[7] Liu X, Sun X, Li B. PFN: A novel program feature network for program comprehension[A]. Proceedings of the 13th IEEE/ACIS International Conference on Computer and Information Science[C]. USA: IEEE, 2014. 349 – 354.
[8] Sun X, Liu X, Hu J, Zhu J. Empirical studies on the nlp techniques for source code data preprocessing[A]. Proceedings of the 2014 International Workshop on Evidential Assessment of Software Technologies[C]. USA: ACM, 2014. 32 – 39.
[9] Abdeen H, Sahraoui H A, Shata O, Anquetil N, Ducasse S. Towards automatically improving package structure while respecting original design decisions[A]. Proceedings of the Working Conference on Reverse Engineering[C]. USA: IEEE, 2013. 212 – 221.
[10] Sun X, Li B, Tao C, Zhang S. HSM-based change impact analysis of object-oriented java programs[J]. Chinese Journal of Electronics, Apr. 2011, 20(2): 247 – 251.
[11] Van Rijsbergen C J. Information Retrieval[M]. Butterworth, London, 1979.
[12] Sun X, Li B, Tao C, Wen W, Zhang S. Analyzing impact rules of different change types to support change impact analysis[J]. International Journal of Software Engineering and Knowledge Engineering, 2013, 23(3): 259 – 288.
[13] Harman M, Binkley D, Gallagher K, Gold N, Krinke J. Dependence clusters in source code[J]. ACM Transactions on Programming Languages and Systems. 2009, 31(1): 1 – 33.
[14] Thomas S W, Adams B, Hassan A E, Blostein D. Modeling the evolution of topics in source code histories[A]. Proceedings of the Working Conference on Mining Software Repositories[C]. USA: ACM, 2011. 173 – 182.
[15] Liu X, Sun X, Li B. Top-down program comprehension with multilayer clustering based on LDA[A]. Proceedings of the International Workshop on Evidential Assessment of Software Technologies[C]. USA: IEEE, 2014. 56 – 59.

## 作者简介

**孙小兵(通信作者)** 男,1985 年出生,江苏姜堰人,博士、副教授、硕士生导师,主要研究领域为软件分析、维护与演化.
E-mail: 18252740912@163.com

**刘湘月** 女,1990 年出生,江苏盐城人,硕士研究生,研究方向为程序理解.
E-mail: 495296335@qq.com