基于语法与语义分析的代码搜索结果优化

刘 石 李 合 王啸吟 张 路 谢 冰
(北京大学信息科学技术学院高可信软件技术教育部重点实验室 北京 100871)

摘 要 通过示例代码学习简单算法的实现和具体API的使用方式是程序开发人员在软件开发中进行软件复用的高效手段,也是使用代码搜索引擎的主要目的。代码搜索引擎从网页搜索技术发展而来,提供对网络上源代码资源的检索功能,能够有效定位与搜索内容相关的代码,为程序开发人员提供帮助。但现有的代码搜索引擎没有在搜索结果中区别API的实现代码与使用代码,搜索结果存在冗余,导致用户无法快速有效地找到提供有用信息的代码片段。为了使用户更好更快地找到代码搜索目标,阐述了应用语法与语义分析技术从区分API实现代码和使用代码、相似代码聚类、搜索结果摘要3个方面对代码搜索结果进行优化的方法,给出了一个代码搜索引擎的实现,并在实例研究中展示了该方法的有效性。

关键词 软件复用,代码搜索,语法与语义分析,API的实现与使用代码
中图法分类号 TP311.5 **文献标识码** A

Enhancement of Code Search Results Using Syntax and Semantic Analysis

LIU Shi LI He WANG Xiao-yin ZHANG Lu XIE Bing
(Key Laboratory of High Confidence Software Technologies, Ministry of Education, School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract Learning simple algorithm and specific API usage by code examples is an efficient way in software reuse and is the main purpose of using code search engine. Code search engine providing search service for source code on the internet was developed from Web search engine. It is able to locate source code related to the search input and brings great assistance to software development. However, the state-of-art code search engines do not make a distinction between API implementation and usage, the results are redundant and not easy to recognize. It is difficult for the user to obtain useful code segments from search result items. To address the problem, we proposed applying syntax and semantic analysis techniques to organizing the search results, clustering the similar code and acquiring better code digest. We implemented our method with a code search engine and evaluate its effectiveness in this paper, the experimental results demonstrate that our approach works efficiently.

Keywords Software reuse, Code search, Syntax and semantic analysis, API Implementation and Usage

1 引言

随着开源运动的发展,互联网上开源项目的数量在不断增长,高质量的开源软件为现实的软件开发提供了宝贵的资源;同时,在软件企业内部,多年的实践表明:软件复用有助于大大减少开发时间,改进质量,降低成本[1]。系统化的软件复用就包括从大量的遗产系统中提取有用的代码资源应用于软件开发过程,进行代码级的软件复用。海量的软件代码资源也带来了相关代码的检索以及代码是否可信等一系列问题,如何又好又快地找到开发人员感兴趣的高质量代码是需要解决的重要问题。

代码搜索引擎利用网页搜索引擎技术,对已有的代码资源进行整合,同时提供一个Web访问接口实现了基于文本的代码搜索功能。代码搜索引擎以文本检索技术为基础,帮助使用者找到与搜索输入文本相关的源代码片段。近年来,研究者通过建立不同代码间的使用关系,并应用链接分析进行代码实体的排序,那些与搜索关键词高度相关又获得广泛使用的代码更容易作为搜索结果返回给用户。

使用户在最短的搜索时间内,通过最小的努力找到其感兴趣的内容是研究人员不断努力的目标。与网页文件相比,代码文件有着自己的特点,现有的代码搜索引擎在移植网页搜索技术的基础上,针对代码特点做了很多改进,但与上述目标还有距离。我们认为还可以在以下几个方面对现有的代码搜索引擎进行改善。

首先,基于用户行为分析和一些非正式的调查,软件开发和维护人员进行代码搜索的主要目的是找寻API实现代码和已知API的使用实例代码。R Hoffmann等[2]通过分析通用搜索引擎的日志得到:在确信的与Java编程有关的搜索

到稿日期:2009-01-16 返修日期:2009-03-09 本文受国家科技支撑计划(2006BAH02A02),国家高技术研究发展计划(863)(2006AA01Z189)资助。
刘 石 硕士研究生;李 合 硕士研究生;王啸吟 博士研究生;张 路 副教授;谢 冰 教授,博士生导师。

中，用户潜在的搜索目标主要与应用编程接口（APIs）相关，包括寻找一个适当的 API 解决某个具体的问题，查询某个 API 的详细信息，以及查询某个 API 的使用示例代码。根据 S. E. Sim 等[3]对软件开发与维护人员的调查，查找某些功能的实现以及找到使用某个功能的所有地方是他们对代码进行搜索的主要目标。如果能在搜索结果中有针对性地对 API 实现代码和对 API 的使用实例代码进行分离，将有助于用户更快地找到感兴趣的代码片断。

其次，应对搜索结果中的相似代码进行聚类。大量的不同工程的代码之间存在着很多相似的甚至完全相同的代码，这主要是由于完成相似功能的代码趋于同质，而这一现象被称为代码克隆[4,14]。针对同一搜索关键词的搜索结果由于其内在关联性，更趋于同质，造成搜索结果中会存在大量的相似的代码片段，这无疑是信息的冗余，更不利于展示实现方法和使用方法的多样性。应对搜索结果中的相似代码进行聚类，将多个相似搜索结果聚在一起返回给用户，这样才能更好地向用户展示 API 的不同实现方法和使用方法，而各个聚类数目也同时向用户展示了该种方法被应用的广泛程度。仅仅根据文本的相似（相同）来对搜索结果进行聚类并不能很好地实现代码聚类，应该充分利用代码的结构特点，将最新的代码克隆检测的成果应用于相似代码聚类。

再次，应该在搜索结果摘要中简明、集中地显示主体内容最具有代表性的信息。搜索结果摘要是显示给用户的结果网页中最重要的信息，是用户对各个搜索结果项相关与否进行决策的最直观最重要的信息来源。网页搜索结果的文本摘要集中体现了网页内容中与搜索项相关的内容，并将关键词高亮显示。代码摘要与之类似，但也具有自己的特点，如：以代码行为单位；同时包含与搜索相关的源代码和注释代码。现有的代码搜索引擎大都做到了以上两点，而我们认为代码摘要还应反映所代表代码实体的结构，突出代码实体的基调，如类或函数的声明行；且应注意与搜索关键词文本无关但语义相关的内容，目的是使代码摘要作为一个整体完整地表达主体代码语义。

以上 3 个方面都可以对代码应用语法和语义分析技术加以改善。可以通过代码编译等技术，获取源代码的语法结构及语义，并将这些信息最终反映到搜索结果中，从而达到优化代码搜索结果、改进代码搜索质量的目的。为了区分 API 的实现代码和使用代码，可以应用函数调用分析等技术，构建类似 Web 链接图的代码使用链接图，那些与搜索相关并更多被使用的代码实现便更可能是 API 的实现代码；为了更好地完成代码聚类，可以使用语法结构分析方法，利用基于语法树的代码克隆检测工具找到更多更准确的相似代码；为了实现更好的动态摘要方法，可以对代码进行语法和语义分析，根据各代码行的语法结构和语义信息更好地完成摘要信息的取舍。

本文基于以上思想提出了应用语法与语义分析优化代码搜索结果的实现方法，并应用上述 3 个方面的改进技术实现了一个代码搜索引擎。本文第 2 节将对这部分工作进行详细介绍；第 3 节是实例研究，展示了代码搜索引擎并验证了改进的有效性；最后是相关工作和总结。

2 我们的方法

本节首先介绍代码搜索引擎的结构，然后针对上述 3 个改进点分别给出解决方案。

2.1 代码搜索引擎的结构

图 1 显示了代码搜索引擎的结构，包括代码资源的获取、代码信息的抽取和组织、文本搜索与相关度排序、查询结果及显示等部分[3]。

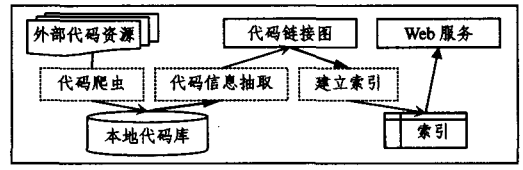


图 1 代码搜索引擎结构图

在代码资源的获取部分，一般使用代码爬虫从开源代码库中下载开源工程，以构造本地代码库。在代码信息的抽取和组织部分，抽取 Java 类和方法作为检索的代码实体，然后抽取代码实体之间的函数调用、读取、写入等关系，构造跨越 Java 工程的代码链接图，并将该图结构以及结点和边所代表的信息保存在数据库中。在后两个部分，首先将代码实体看作结构化文本，根据关键词所处的不同的语法位置，抽取具有不同权重的关键词并使用 Lucene[12] 建立倒排文件。在用户搜索时，结合 Lucene 检索返回的搜索结果的文本相似度和通过链接分析得到的 Rank 值，找到可能的 API 实现代码；同时利用已有的代码链接图，找到对每个 API 实现代码的使用代码，作为搜索结果。利用代码克隆检测工具的检测结果，对搜索结果中存在的相似代码进行聚类。最后，选取聚类中排序最高的代码单元，应用动态摘要算法得到代码摘要，生成网页，以 Web 页面的方式显示给用户。

2.2 分析代码使用关系区分 API 的实现代码与使用代码

在搜索结果中区分 API 的实现代码与使用代码是一个重要目标。一般来说，好的 API 的实现应该是被广泛使用的类或函数接口，而用户在搜索 API 使用的过程中针对的是已知的或特定的 API 的实现。HITS 算法[13]提出权威（authority）网页和中心（hub）网页两个概念，分别表示一个被多次引用或被重要网页引用的网页以及指向多个重要网页的网页。两类网页相互指向，因而形成了一个相互增强的结构。代码实体相互之间的使用关系与网页之间的引用关系非常相似，应用该算法的思想用于寻找“权威的”API 实现代码。

首先，使用 Eclipse JDT[12] 构造信息抽取器，通过对本地代码构建 Java 代码模型并对模型树（语法树）进行周游的方法，抽取 Java 类和类中的方法作为检索的代码实体，然后对模型树进行二次周游，找到 Java 方法之间的函数调用关系（即方法体中出现对其他方法调用的语句），Java 方法与 Java 类之间的函数返回、属性读取和属性修改关系，以及 Java 类（包括接口）之间的扩展与实现关系。上述关系可以统一定义为使用关系，这样便建立了一个基于本地代码库的跨越不同 Java 工程的代码链接图。

然后，对划分好的代码实体进行编译，区分方法（类）名、内部代码、Javadoc 注释，对各部分进行关键词切词，作为不同的域（field）建立的索引，其中局部变量、Java 保留字等不作为关键词索引，上述各域的重要程度依次递减，其权重也因此依次降低。

最后，当用户进行搜索时，首先将查询关键词交由 Lucene，并将检索得到的文本相关度最高的前 200 个代码实体

作为根集合(Root Set)；对根集合中的每个代码实体，通过链接关系进行扩展，将那些使用该实体和被该实体使用的代码实体加入，形成基础集合(Base Set)；然后根据基础集合内部的新的链接图，构造邻接矩阵，初始化 Authority 和 Hub 数组，进行迭代(Power Iteration)，待达到预定精度时停止，得到各代码实体的 Authority 和 Hub 值，按 Authority 值的高低将搜索结果返回给用户，作为与搜索相关的 API 的实现代码。对于每个 API 实现，从代码链接图中找到对该 API 的使用代码，返回给用户。这样，通过两栏显示的方式，实现了对 API 实现和使用代码的区别。

2.3 使用代码克隆检测工具对相似代码进行聚类

使用基于语法结构的代码克隆检测工具 Deckard [4] 进行相似代码检测。相比基于文本相似度的检查方法，Deckard 能更好地利用代码的语法结构的相似性进行检测，避免某些有修改的克隆行为造成的影响。在建立索引之前使用 Deckard 对本地代码库中的所有代码进行检测，然后对检测结果进行分析，如果 Deckard 得到的代码克隆片段与划分的代码实体片段相匹配，便将该聚类 ID 作为代码实体的属性保存。在得到搜索结果后，将具有相同聚类 ID 的代码实体聚在一起，选取其中 Rank 值最高的代码实体作为代表显示，其余搜索结果则可通过点击查看。

2.4 应用语法与语义分析获取更好的摘要

代码摘要以代码行为单位，将摘要固定为 5 行，并显示每行在文件中的行号，动态摘要算法就是要完成对代码搜索结果主体每个代码行的取舍。基本做法是对代码搜索结果主体的每一行根据赋值规则赋予权值，选取权值最高的 5 行作为摘要，具体做法如下：

1)为突出代码实体的基调，如类或函数的声明行会赋予较高权值，以保证其出现在摘要中；而一些代码标记，如无用的/*、{、}等权值会很低，以保证不会作为一行在摘要中出现。这部分通过语法分析技术，通过对结果代码编译，进而获取各行主体内容的语法成分。

2)使用 Lucene 检索返回的结果中包括命中的关键词在正文中的位置，计算得到关键词所在行，并对该行赋予一定的权值。同时，令代码行的权重高于注释行。

3)为了摘要出与搜索关键词文本无关但语义相关的内容，在索引建立阶段对某些关键词进行换名，令其对应语义上更接近的关键词。我们认为局部变量名本身对于代码片段的语义贡献不大，但其类型名对于代码语义有着重要影响，故将局部变量名替换成其类型名作为关键词被索引。也就是说，用户在搜索类型名时，对应的局部变量名也会命中，这样对该变量的使用便有可能进入摘要内容。举例说明如下：

```
648  public Object clone()
650      DateFormat other = (DateFormat) super.clone();
651      other.calendar = (Calendar) calendar.clone();
652      other.numberFormat = (NumberFormat)
             numberFormat.clone();
653      return other;
```

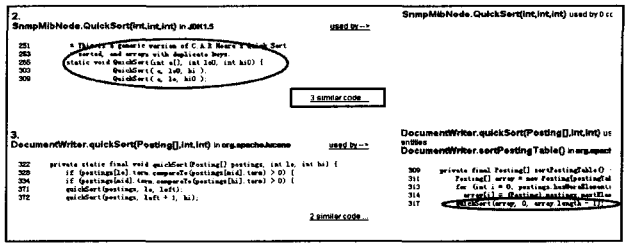
以上是用户在搜索 dateFormat 时，对搜索结果 DateFormat.clone()函数的摘要内容。other 作为 DateFormat 的对象实例被高亮显示，与 other 相关的语句都被算入摘要，这些内容虽然与 dateFormat 关键词文本无关，但与类 DateFormat 语义相关，而且对于理解该函数的执行过程以及与类 DateFormat 的关系十分重要。

3 实例研究

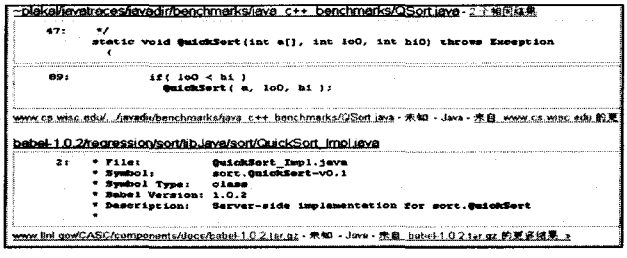
为了验证方法的有效性，现通过手工下载的方式将 56 个开源项目，共计 19183 个 Java 文件构建为本地代码库，并按照上节所述方法实现了一个代码搜索引擎。本节用一个实例展示搜索引擎的功能；同时与已有代码搜索引擎进行比较，并设计一个实验以验证该方法在减少用户的检索代价、提高代码搜索质量方面的有效性。

3.1 实例

用户搜索关键词 QuickSort，潜在的搜索目标有两个：一是希望得到快速排序算法的具体实现，二是寻找实现快速排序算法的 API 及其使用实例以便模仿使用。使用我们的搜索引擎和 Google 代码搜索[10]分别将“QuickSort”作为关键词进行搜索，搜索结果对比如下(见图 2)。



(a)



(b)

图 2 我们的搜索结果与 Google 代码搜索的比较(图(a)为我们的搜索结果)

我们的搜索引擎首页显示的前 8 个结果中，有 5 个快速排序函数实现、3 个快速排序相关类，用户通过摘要能够看到实现函数的函数定义和递归调用过程，如：

```
265  static void QuickSort(int a[], int lo0, int hi0) {
303              QuickSort( a, lo0, hi );
309              QuickSort( a, lo, hi0 );
```

在图 2(b)的搜索结果栏中是针对图 2(a)搜索结果的使用实例。对于很多搜索结果，用户仅通过对应使用代码摘要完全可以了解相关 API 的使用方式，参数的设置(特别是参数与数组长度的关系)，如：

```
317          quickSort(array, 0, array.length -1);
```

图 2(a)中矩形框标识的是相似代码聚类的数目，用户可以点击查看具体信息，在我们的 8 个搜索结果中的 4 个搜索结果都检测到了相似代码，有效减少了同质代码对搜索结果的影响。

Google 代码搜索首页显示的前 10 个结果均为算法实现类，9 个结果是名为 QuickSort 的 Java 类，所有结果都没有在摘要中显示算法实现的相关内容，只给出若干关键词出现的语句，其中 2 个结果摘要中包含 API 使用内容，但没有在摘

要中同时显示函数定义和函数调用信息，API 使用方式难以通过摘要获得。

可以看出，我们的搜索引擎在搜索结果摘要包含的信息量、有效分别 API 的实现和使用代码以及搜索结果多样性方面存在着优势。

3.2 实验

在这个实验中，选用 3 个算法名称和 5 个 JDK 的 API 接口名称作为搜索输入，设定用户搜索目标分别为搜索算法实现代码和具体 API 的使用代码。比较用户使用搜索引擎和 Google 代码搜索，为获得一个完整实现或使用信息，所需要阅读的搜索结果项的个数。我们分"用户通过阅读代码摘要确定搜索结果是否相关"和"用户通过逐个阅读搜索结果代码内容来确认"两种情况来进行实验。实验结果如表 1 所列。

表 1 用户阅读搜索结果个数的比较

搜索输入	通过阅读搜索结果摘要		通过阅读搜索结果内容	
	我们的方法	Google 代码搜索	我们的方法	Google 代码搜索
Quick Sort	1	1	1	1
Bubble Sort	1	1	1	1
depth first search	1	1	1	1
substring(java. lang. String)	7	18	2	10
replaceWith(java. lang. String)	2	5	2	5
Matcher find(java. util. regex)	2	3	2	3
BufferedReader(java. io)	2	5	2	4
Logger(java. util. logging)	2	19	2	19

由于无法获知 Google 代码搜索的内部实现，加之基础设施和代码基础集合差距很大，对于两个搜索引擎的实际搜索时间的比较便没有意义，这里仅通过比较用户阅读搜索结果个数以代表用户的检索代价。从实验结果可以看出在搜索 API 的实现代码方面，两个代码搜索引擎都能在搜索的第一个结果给出正确的实现代码；而在搜索 API 的使用代码方面，用户使用 Google 代码搜索要花费更多的检索代价以寻找一段适当的使用代码。同时，通过试验结果也可看到，代码摘要的质量也显著影响着用户的搜索代价。

下一步，将搭建更大规模的、趋于实用的代码搜索基础平台，并对 3 个改进方面的有效性分别进行验证。

4 相关工作

在代码搜索及相关领域已有一些很好的研究和实践，Codase[5]，CSourceSearch[6]对查询进行了精细的分类，查询者可以定制查询关键字在源代码中扮演的角色（出现的位置），分为函数定义、函数调用、类名、包名等等，通过这种方式，丰富了查询语义，也在一定程度上实现了 API 使用和实现的分离，但使得查询本身变得严格而复杂，不符合简单通用的查询原则；同时，单纯在函数定义或函数调用语句上进行相关词的搜索，遗失了很多其他信息，会使得搜索的命中率降低。Assieme[2]是一个 Web 搜索接口，它将 Web 可访问的 JAR 包、API 文档以及含有解释性文字和示例代码的网页相结合作为搜索结果返回给用户，这样的结果能够很好地解释用户对 API 的诸多疑问，但具有示例代码的网页数量有限、存在着 API 文档遗失或不可访问的情况，该方法不具有可扩展性，也不是严格意义上的代码搜索。Sourcerer[3]详细介绍了一个代码搜索引擎各个部分的设计与实现，这对后续代码搜索引擎工程实践的开展有重要的借鉴意义；Koders[8]和 Krugle[9]是已经商业化的代码搜索企业，他们不仅提供对于开源代码的搜索，而且可以为企业定制内部代码的搜索服务；Google Code Search[10]是 Google 提供的源代码搜索引擎，针对代码克隆的问题，Google Code Search 在查询结果显示时将"完全一样的文件"聚在一起返回给用户，但忽略了代码特有的语法与语义信息，难以检测某些有修改的克隆行为；这些搜索引擎是目前这一领域的代表者，它们在搜索准确度和结果的展示方面都比较出色，但对于我们在引言中提到的 3 个问题都没有给出很好的解决方案，除非用户对查询的 API 已经有了相应的理解，否则用户需要花费很多时间甚至多次查询得到结果。

结束语 本文针对程序开发人员对于代码搜索的需求，分析了现有的代码搜索引擎和相关技术存在的问题，提出了在 3 个方面应用语法和语义分析技术优化代码搜索结果的方法，以达到提高用户的搜索效率和搜索结果质量的目标。我们应用上述方法实现了一个代码搜索引擎，并通过实例研究展示了该引擎的效果。

参 考 文 献

[1] Jacobson I，等. 软件复用：结构、过程和组织[M]. 韩柯，译. 北京：机械工业出版社

[2] Hoffmann R，Fogarty J. DS Weld Assieme：finding and leveraging implicit references in a web search interface for programmers[C]//Proceedings of UIST'2007

[3] Sim S E，Clarke C L A，Holt R C. Archetypal source code searches：A survey of software developers and maintainers[C]//IWPC. 1998

[4] Jiang Lingxiao，Misherghi G，Su Zhendong，et al. DECKARD：Scalable and Accurate Tree-based Detection of Code Clones[C]//ICSE 2007

[5] Codase[OL]. http://www. codase. com/

[6] CSourceSearch[OL]. http://csourcesearch. net/

[7] Sourcerer：A Search Engine for Open Source Code[OL]. http://sourcerer. ics. uci. edu/

[8] Koders[OL]. http://www. koders. com/

[9] Krugle[OL]. http://www. krugle. com

[10] Google Code Search[OL]. http://www. google. com/codesearch

[11] Eclipse JDT[OL]. http://www. eclipse. org/jdt/

[12] Lucene[OL]. http://lucene. apache. org/

[13] Kleinberg J M. Authoritative Sources in a Hyperlinked Environment[C]//Proceedings of the ACM-SIAM Symposium on Discrete Algorithms，1998. IBM Research Report RJ 10076. May 1997

[14] 曹羽中，金茂忠，刘超. 克隆代码检测技术综述[C]//NASAC 2006