基于 Word Embedding 的软件工程领域语义相关词挖掘方法

胡望胜

(上海交通大学软件学院,上海 200240)

摘 要: 软件的开发及维护过程中经常要对代码进行搜索。基于关键字匹配的代码搜索面临与传统信息检索一样的问题，即用户查询关键字与代码文本用词不匹配。为提高代码搜索精度，需要挖掘软件中的语义相关词进行查询扩展。本文针对软件工程领域设计了一种基于 Word Embedding 的语义相关词挖掘方法，并且采用 IT 技术问答网站 Stack Overflow 的文档作为语料库训练得到了共包含 19332 个单词的语义相关词表。与前人工作的对比实验验证了本文方法挖掘的语义相关词能有效提高代码搜索精度。

关键词: 代码搜索；查询扩展；语义相关词

中图分类号: TP311 **文献标识码:** A **doi:**10.3969/j.issn.1006-2475.

Learning Semantically Related Words in Software through Word Embedding

HU Wang-sheng

(School of Software, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Searching for previously written code is important for software development and maintenance. The same as traditional information retrieval, the inherent difficulty of keyword based code search is vocabulary mismatch problem between user query and retrieved code. To improve the accuracy of code search, learning semantically related words in software for query expansion is needed. This paper designs a Word Embedding based method to learn semantically related words in software, and obtain semantically related words for 19332 words through training it on Stack Overflow documents. The experiment results show that the learned semantically related words can effectively improve code search accuracy.

Key words: code search; query expansion; semantically related words

0 引言

软件开发及维护过程中，开发者经常需要进行代码搜索来帮助完成代码学习和重用、代码重构以及 bug 定位等工作[1]。现有的代码搜索工具大部分是基于关键字文本匹配的搜索方法。与传统信息检索类似，这种搜索方法的一个关键问题在于用户查询关键字与代码文本用词不匹配[2]。因此需要对用户查询语句做语义相关词扩展以提高搜索精度。

由于软件工程领域的单词语义与自然语言存在很大差异，代码搜索无法使用自然语言的语义相关词做查询扩展，需要软件工程领域的语义相关词表。目前已有的软件工程领域语义相关词挖掘研究大多采用简单的文本相似度检测方法或基于词汇同现的统计方法[3-7]，具有较大的局限性。本文设计了一种基于 Word Embedding 的软件工程领域语义相关词挖掘方法，采用 IT 技术问答网站 Stack Overflow 的文档作为语料库训练得到了 19332 个单词的语义相关词表。与前人工作的对比实验表明采用本文语义相关词表进行查询扩展能有效提高代码搜索精度。

1 背景知识

1.1 代码搜索

现有的代码搜索工具如 Sando[8], Kugle[9]以及 Sourcerer[10]等都是基于关键字文本匹配的搜索方法，在实际使用中发现它们的搜索精度并不理想。这其中的一个关键问题在于用户查询关键字与代码文本用词不匹配。假设一个程序员想要搜索怎样执行一个线程的 java 代码，他的查询关键字是 execute thread，那么包含 Thread.run( )这个 java API 调用的正确结果就会被遗漏。这个时候就需要对用户查询做语义相关词扩展，将原始查询扩展为 execute thread OR run thread，其中的关键在于需要提供 execute 的语义相关词 run。因此需要对软件中的语义相关词进行挖掘。

目前自然语言的语义相关词挖掘工作已经比较成熟，但是代码搜索无法直接使用如英语词典[11]、WordNet[12]等自然语言的语义相关词表来提高搜索精度。这是因为软件工程领域的单词语义与自然语言有很大不同。如上文提到的 execute 与 run 在英语词典中并不是语义相关词。软件工程领域中

收稿日期：2006-08-20；

作者简介：胡望胜(1993－),男,湖南人,上海交通大学软件学院硕士研究生,研究方向：程序分析与测试。

还存在大量自然语言中并不存在的缩略词，如 interrupt 和 IRQ，其中 interrupt 常出现在用户查询中而 IRQ 则常出现在代码中。相对的，一些自然语言的语义相关词也并非软件工程领域的语义相关词，如 disable 和 torture 在英语中是同义词，而在软件工程领域中没有关系。Sridhara 等人的研究表明，采用自然语言的语义相关词进行查询扩展甚至会降低代码搜索的精度[13]。

1.2 语义相关词挖掘

目前国内外已经有一些针对软件工程领域语义相关词挖掘的相关研究。Shepherd 等人通过自然语言处理方法从软件代码及注释中提取相似 verb-DO 对来识别语义相关词[3]。一个 verb-DO(verb-Direct Object)对是指一个动词加上其直接作用名词。例如从软件 iReport 中找到了两个 verb-DO 对(add, element)和(find, element)，那么 add 和 find 就被识别为一对语义相关词。Hill 细化了 Shepherd 的研究，推出了语义相关词识别精度更高的 SWUM[4]。Yang 等人则在 Shepherd 研究的基础上进行一定扩展推出了 SWordNet，一个通过文本相似度比较挖掘语义相关词的工具[5-6]。例如 linux 内核代码中存在 disable all interrupt sources 和 disable all irq sources 这样两条语句，由于具有相同上下文，interrupt 和 irq 在 SWordNet 中被识别为一对语义相关词。上述两种方法均是从软件代码和注释中挖掘语义相关词。如果是从单个软件中挖掘，得到的语义相关词将不具备普适性；如果是从多个软件中挖掘，得到的语义相关词数量将大大受限。而且如果相似文本中包含自然语言单词，还会导致误报，如 SWordNet 由于一个软件的代码注释中同时存在 we have a match 和 we have a literal 这两条语句而错误地将 match 和 literal 识别成一对语义相关词。Tian 等人推出了 SEWordSim，基于词汇同现的统计方法对 Stackoverflow 的文档进行语义相关词挖掘[7]。由于简单的词汇同现无法体现单词的深层语义，SEWordSim 得到的语义相关词精确度也不够理想。

2 Word Embedding

单词不是简单的符号，它们具有语义。传统的自然语言处理技术使用简单的数字 ID 或 one-hot representation 来标识单词[14]，无法体现单词的真正语义，也无法衡量两个单词之间的语义距离。比如 one-hot representation 中，单词 execute 可能被分配到向量[1, 0, 0]而 run 被分配到向量[0, 1, 0]，从它们的向量表示无法判断这两个单词的语义是否相似。

Word Embedding 即 distributed representation，是一种自然语言技术，它将单词映射到固定维度的实数向量，使得两个语义越相似的单词在向量空间中距离也越短[15]。比如在一个十分简单的 Word Embedding 模型中单词 execute 可能被映射到[0.12, -0.32, 0.01]而 run 被映射到[0.12, -0.31, 0.02]，从二者的向量距离可以很容易地衡量它们的相似关系。

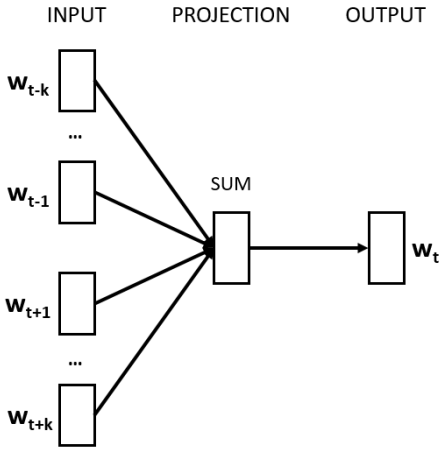

图 1 CBOW 模型

单词的向量表示可以从包含它们的语句中的上下文环境进行推断。一般来说，经常出现在相似上下文的语句中的两个单词它们的语义也倾向相似。比如，单词 image 和 img 经常出现在相似上下文中，如 save image as png 和 save img as png，据此可以推断 image 和 img 是一对语义相关词。为此需要建立语言模型，用来捕捉一个单词和其上下文单词之间的关系。本文采用了神经网络语言模型 CBOW(Continuous Bag-of-Words Model)[15]。CBOW 模型根据已知上下文对当前单词进行预测，它的学习目标是最大化对数似然函数 L：

$$L = \sum_{w \in C} \log p(w \mid Context(w))$$

其中 w 表示语料库 C 中任意单词，Context(w)表示单词 w 的上下文。p(w|Context(w))具体来说即 $P(w_t|w_{t-k},\ldots,w_{t-1}, w_{t+1},\ldots,w_{t+k})$，其中 $w_{t-k},\ldots,w_{t-1}, w_{t+1},\ldots,w_{t+k}$ 表示一条语句中单词 w_t 的前 k 个单词及后 k 个单词，k 是一个常量。在模型中每个单词都被分配一个实数向量。如图 1 所示，CBOW 模型的输入层是上下文单词的词向量，投影层对输入层进行求和，输出层输出单词 w_t 出现的概率。单词向量 w 是 CBOW 模型的参数，在训练开始时被赋予随机值，随着训练的进行不断被更新直到学习目标完成。

在得到单词的向量表示之后，通过计算两个单词向量之间的余弦距离或欧几里得距离就可以推断它们的语义相似度。本文计算两个单词向量之间的余弦距离，值范围为 0 到 1，越接近 1 说明两个单词向量夹角越小，语义相似度也越高。

3 软件工程领域语义相关词的挖掘

3.1 语料库的选取

由于 CBOW 模型在训练时以语句作为上下文环境，选取的语料库需要同时满足两个条件：软件

工程领域相关而且文档以语句形式呈现。基于此，IT 技术问答网站 Stack Overflow 的文档正好满足要求。

图 2 是 Stack Overflow 网站上的一个典型页面。提问者在一个问题中会用一些语句来做详细阐述，同时还可能包含一些代码片段。代码属于结构化文本，本身不具备类似语句的特征，而且一般会包含大量诸如循环变量等无明确语义的单词，因此本文在选取语料库时仅考虑语句。一个问题下通常还会有一些回答，格式与问题类似，这里不再举例。

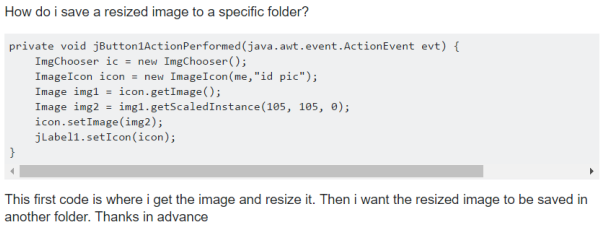

图 2 Stack Overflow 网页内容

考虑到程序语言特性，不同的程序语言一般具有不同的专有名词，它们的语义相关词也不同。如 jdbc 是 java 特有的数据库访问接口 API，仅在 java 语言中 jdbc 与 database 为一对语义相关词。本文主要针对 java 代码搜索，因此在选取语料库时仅选取带有 java 标签的 Stack Overflow 问题。最终本文收集了共 708473 个 Stack Overflow 问题。

3.2 预处理

在上一步拿到原始数据后还需做进一步处理。首先原始数据中包含许多很短的口语化语句，如 thank you 等，不具备训练价值。经观察这些语句大多单词数不超过 3，因此本文仅保留单词数大于等于 4 的语句。其次语句中可能包含函数名及 API 名，如 saveImage、save_image 以及 Thread.run，分别需要按驼峰式命名规则、下划线命名规则以及符号“.”进行分词。关于词干提取，本文在实际实验中发现，现有的词干提取工具不够精确，如 Porter stemmer 错误地提取 adding 的词干为 ad[16]，在代码搜索时会带来大量噪音；同时 CBOW 模型训练得到的结果能有效识别一个单词的词形变换。因此本文未对原始数据做词干提取。

在经过上述预处理后，语料库中包含共 5913765 条语句，共 73396324 次单词出现。

3.3 模型训练

CBOW 模型的实现采用了 word2vec[17]。CBOW 模型在训练前需要设置两个关键训练参数：单词向量维度和训练窗口大小（即第 2 节中提到的 k）。为选择最优的训练参数，本文从上一步收集的语料库中随机选取了 100000 条语句作为样本，单词向量维度以 100 为间隔从区间[100, 500]取值，训练窗口大小以 1 为间隔从区间[3, 8]取值，共得到 30 种不同参数组合进行训练，检查所得结果的精确度。根据实验结果最终选取的训练参数为：单词向量维度为 200，训练窗口大小为 5。

实际训练中，本文使用机器为 2.4GHz Intel Core i7 处理器，64G 内存，Ubuntu 14.04 LTS 64 位操作系统的配置。训练共耗时 5 小时 22 分钟。训练得到共 30992 个单词的向量表示。

3.4 语义相关词表的生成

在得到训练结果后还需做进一步处理。首先训练结果中包含大量无意义的数字符号，如 1.0f、1e-4 等；其次还包含一些停词，如 a、the 这些本身不具备具体含义的单词；除此之外，还有一些单词由于在语料库中出现频率过低，训练得到的向量不够精确（这些单词通常是一些冷门单词或拼写错误）。这些单词在训练前为保证语句上下文的完整性没有去掉，在训练完成后需要过滤。本文首先移除训练结果中所有数字符号，然后根据 SWordNet 使用的停词表移除停词，最后根据采样实验的结果确定以词频 30 为阈值移除词频过低的单词。经过处理后得到共 19332 个单词的向量表示。

接下来本文采用余弦距离计算任意两个单词之间的语义相似度，并且根据观察实验初步选取距离每个单词最近的前 40 个单词作为它的语义相关词（数量太多不利于查询扩展，数量太少则可能遗漏）。由于不同单词的语义相关词数量也不同，对某些单词来说距离最近的前 40 个单词并非都是语义相关词。根据采样实验，本文最终确定以相似度 0.48 为阈值对所得语义相关词表进行过滤。最终得到 19332 个单词的语义相关词表，共 342106 对语义相关词。

表 1 语义相关词表

单词	语义相关词（按相似度由高到低排序）
save	saves, saving, saved, store…
delete	deletes, remove, deleting, insert…
directory	folder, directories, dir, subdir…
image	images, img, picture, bitmap, png…
db	database, databases, jdbc, sql…
excel	xls, xlsx, csv, spreadsheet…

表 1 截取了最终得到的语义相关词表中的几个例子。从表 1 可以看出，本文方法可以有效挖掘不同形式的语义相关词。本文方法能识别一个单词的词形变换，如 save 和 saves, saving, saved；能识别同义词，如 delete 和 remove；能识别反义词，如 delete 和 insert；能识别缩略词，如 image 和 img；能识别语义相关词，如 excel 和 xls，xlsx, csv（这些均为 excel 的文档格式）以及 db 和 sql 等等。注意到 3.1 节中提到的 jdbc 也被成功识别为 db 的语义相关词。

4 实验评估

本文实验主要从以下两个方面验证本文方法挖掘的语义相关词表的有效性：

1)精确度。即所得语义相关词表中正确的语义相关词对所占比重大小。这直接影响到该表的实际应用价值。

2)对代码搜索精度的提升。挖掘语义相关词的最终目的是用于提升代码搜索精度，因此本文设计实验与前人工作进行对比，验证所得语义相关词表是否能有效提升代码搜索精度。

4.1 精确度

4.1.1 实验设定

本实验对本文所得语义相关词表的精确度进行验证。笔者从所得语义相关词表的 19332 个单词中随机选取了 100 个单词，共 1740 个语义相关词对作为样本，人工验证它们是否正确。为减少主观性的影响，验证由 3 个软件工程方向研究生完成，且采取较严格的评判标准，即三个验证者必须同时标记某个语义相关词对正确，才将该词对标记为正确。然后笔者检查这些正确的语义相关词对是否出现在英语词典或 WordNet 的语义相关词库中。

4.1.2 实验结果

表 2 验证结果

用例数量	1740
正确语义相关词对	1352
精确度（%）	77.7
不在英语词典或 WordNet 中	1011
不在英语词典或 WordNet 中比例（%）	74.8

实验结果如表 2 所示。可以看出样本的精确度为 77.7%，说明本文所得语义相关词词表的精确度平均在 77.7%左右，根据 Mikolov 等人在自然语言文档上训练 CBOW 模型的实验结果可知这是一个比较理想的数值[15]。同时，在所有正确语义相关词对中，不在英语词典或 WordNet 中的占 74.8%，进一步验证了 1.1 节中提到的单词语义在软件工程领域与自然语言之间的差异性。可以看出这种差异是相当大的。

4.2 对代码搜索精度的提升

4.2.1 实验设定

本实验对本文所得语义相关词表能否有效提升代码搜索精度进行验证。为与前人工作进行比较，本文采用与 SWordNet 同样的实验数据及实验方法。具体描述如下：

1)代码搜索的对象为表 3 中的 4 个 java 软件。测试数据集为 Shepherd 提供的与这 4 个 java 软件相关的 8 个关注定位（concern location）任务，以及关注定位结果[18]。关注定位是指用户对软件中某一个功能或模块感兴趣，需要通过用户提供的关键字查找到软件中所有与该功能或模块相关的代码。本测试数据集中搜索对象为软件中所有函数签名。图 3 是测试数据的一个样例，可以看出针对软件 iReport 的关注定位任务 Add Textfield 共有 5 个相关结果。

表 3 测试软件（LOC:代码行数，#M:函数个数）

软件	描述	LOC	#M
iReport	报表可视化设计器	74506	7587
javaHMO	多媒体服务器	25988	1787
jBidWatcher	eBay 拍卖监控软件	23502	1918
jajuk	音乐管理播放软件	20679	2132

```
Project: iReport   Concern: Add Textfield

Gold Set:
jReportFrame.dropNewTextField(Point, String, String, String)
jReportFrame.dropNewTextField(Point, String, String)
jReportPanel.drop(DropTargetDropEvent)
TextFieldReportElement.TextFieldReportElement(int, int, int, int)
TextReportElement.TextReportElement(int, int, int, int)
```

图 3 测试数据

2)与 SWordNet 的测试方法一样，在进行代码搜索时采取正则表达式匹配的搜索算法。具体来说，假设用户查询语句是 gather file，那么搜索的正则表达式是.*gather.*file.*以及.*file.*gather.*。在进行查询扩展时，假设 collect 是 gather 的一个语义相关词，那么查询语句扩展为.*gather.*file.*，。*collect.*file.*，.*file.*gather.*以及.*file.*collect.*。

4.2.2 实验结果

表 4 关注定位实验结果（SWN:SWordNet）

查询关键字	精确度（%）		覆盖率（%）	
	SWN	本文	SWN	本文
add textfield	14.30	21.14	40	40
compile report	4	6.52	25	37.50
gather music files	28.60	28.57	50	50
load movie listing	0	0	0	0
add auction	3.70	9.41	100	72.74
save auction	1.61	7.41	33.30	66.67
set snipe price	0	2.60	0	16.67
play song	0	11.54	0	75
平均	6.53	**10.90**	31.04	**44.82**

对比实验结果如表 2 所示，其中 SWordNet 的实验结果取自 Yang 的论文[6]。注意在同样的实验上 SWordNet 与 SWUM 进行了比较并且表现更优，限

于篇幅这里不再列举 SWUM 的实验结果。可以看出在 8 个搜索任务中有 5 个本文方法的表现要优于 SWordNet，有 2 个与 SWordNet 的表现相当。总体来看，平均精确度及覆盖率均优于 SWordNet，精确度提升了 67%，覆盖率提升了 44%。

从表 4 可以看出，本文方法可以识别 SWordNet 不能识别的语义相关词，如 play song 查询中，本文方法成功将 song 和 playlist 识别成一对语义相关词而 SWordNet 没有。这是由于 SWordNet 采用简单的文本相似度比较挖掘语义相关词，无法真正理解单词的深层语义。而且 SWordNet 挖掘对象仅限于软件代码及注释，无法挖掘软件文档中的语义相关词。本文方法还能有效提升代码搜索的精确度，8 个搜索任务中有 6 个本文方法精确度要优于 SWordNet，说明本文方法挖掘的语义相关词精确度更高。同时本文方法仍存在缺陷，在 load movie listing 查询中，本文方法与 SWordNet 表现一样，未能找到任何相关代码。这是因为该查询中相关代码包含关键字为 reload 和 container，本文方法成功识别了 load 和 reload 为语义相关，未能识别 listing 和 container。这是因为本文方法使用的训练数据仅来自 Stack Overflow，语料库不够充分，需要从不同渠道收集更多更广泛的训练数据。

注意到虽然本文方法在精确度及覆盖率上有提升，但平均精确度及覆盖率仍不高。这是由于以下两个原因：

1)测试数据为关注定位任务，关注定位要求给定查询关键字，返回项目中所有与查询内容相关的代码。然而由于代码搜索的固有局限性，代码搜索能够返回的仅仅是与该关键字在文本层面上直接相关的部分代码，更深层次的相关代码（比如在直接相关函数中调用的辅助函数）需要用户以搜索的返回结果作为起点使用代码浏览工具进行检索[1]。如图 3，函数 jReportPanel.drop(DropTargetDropEvent) 与查询 add textfield 在文本层面没有直接关联，但是在软件 iReport 中是实现 add textfield 功能的一个辅助函数。因此代码搜索的覆盖率难以达到 100%。

2)搜索算法采用的是正则表达式匹配，而且考虑了所有可能的语义相关词的组合。这是基于保守的思想，因为本文的实验设计中覆盖率比精确度更重要。对用户来说，检查搜索结果的正确性比起猜测软件中可能使用的语义相关词要简单得多[6]。

5 结束语

本文研究现有的软件工程领域语义相关词挖掘方法，针对它们的不足提出了一种基于 Word Embedding 的语义相关词挖掘方法，并且进行了与前人工作的对比实验。实验结果表明本文方法挖掘的语义相关词精确度较高，能有效提升代码搜索精度。

参考文献:

[1] SHEPHERD D, DAMEVSKI K, ROPSKI B, et al. Sando: an extensible local code search framework[C]. // Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering, 2012:1-2.

[2] Furnas G W, LANDAUER T K, GOMEZ L M, et al. The vocabulary problem in human-system communication[J]. Communications of the ACM, 1987,30(11):964-971.

[3] SHEPHERD D, FRY Z P, HILL E, et al. Using natural language program analysis to locate and understand action-oriented concerns[C]. // Proceedings of the 6th International Conference on Aspect-oriented Software Development, 2007:212-224.

[4] Hill E. Integrating natural language and program structure information to improve software search and exploration[D]. University of Delaware, 2010.

[5] YANG Jinqiu, LIN Tan. Inferring Semantically Related Words from Software Context[C]. // Proceedings of the 9th IEEE Working Conference on Mining Software Repositories, 2012:161-170.

[6] YANG Jinqiu, LIN Tan. Swordnet: Inferring semantically related words from software context[J]. Empirical Software Engineering, 2014,19(6):1856-1886.

[7] TIAN Yuan, LO D, LAWALL J. Sewordsim: software-specific word similarity database[C]. // Proceedings of the 36th International Conference on Software Engineering, 2014:568-571.

[8] Sando. Sando code search tool[EB/OL]. [2016-10-05]. http://sandosearch.weebly.com/.

[9] Aragon Consulting Group, Inc. Krugle code search[EB/OL]. [2016-10-05]. http://www.krugle.com/.

[10] LINSTEAD E, BAJRACHARYA S, NGO T, et al. Sourcerer: mining and searching internet-scale software repositories[J]. Data Mining and Knowledge Discovery, 2009,18(2):300-336.

[11] Merriam-Webster, Incorporated. Merriam-webster english dictionary and thesaurus[EB/OL]. [2016-10-05]. http://www.merriam-webster.com/.

[12] Princeton University. WordNet[EB/OL]. [2016-10-05]. http://wordnet.princeton.edu/.

[13] SRIDHARA G, HILL E, POLLOCK L, et al. Identifying word relations in software: A comparative study of semantic similarity tools. // Proceedings of the 16th IEEE International Conference on Program Comprehension, 2008:123-132.

[14] TURIAN J, RATINOV L, BENGIO Y. Word representations: a simple and general method for semi-supervised learning[C]. // Proceedings of the 48th annual meeting of the association for computational linguistics, 2010:384-394.

[15] MIKOLOV T, CHEN Kai, CORRADO G, et al. Efficient estimation of word representations in vector space[C]. // Proceedings of the 1st International Conference on Learning Representations, 2013.

[16] Martin Porter. The Porter Stemming Algorithm[EB/OL]. [2016-10-05]. https://tartarus.org/martin/PorterStemmer/.

[17] word2vec. Tool for computing continuous distributed representations of words[EB/OL]. [2016-10-05]. https://code.google.com/archive/p/word2vec/.

[18] SHEPHERD D. Action-oriented concerns[EB/OL]. [2016-10-05]. https://www.eecis.udel.edu/~gibson/context/action_oriented_concerns.txt.