计算机科学
COMPUTER SCIENCE

基于CodeBERT的设计模式语言模型

陈时非, 刘东, 江贺

引用本文

陈时非, 刘东, 江贺. 基于CodeBERT的设计模式语言模型[J]. 计算机科学, 2023, 50(12): 75-81.

CHEN Shifei, LIU Dong, JIANG He. CodeBERT-based Language Model for Design Patterns[J]. Computer Science, 2023, 50(12): 75-81.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.230100115

基于 CodeBERT 的设计模式语言模型

陈时非　刘　东　江　贺
大连理工大学软件学院　辽宁 大连 116620
(chenshifei@mail.dlut.edu.cn)

摘　要　设计模式是对实际软件设计方案的经验性总结,是软件开发中辅助软件设计的有效方案之一。现有设计模式挖掘研究的任务大多是在源代码中识别设计模式的实例,少有考虑用自然语言语料对设计模式建模。为了提升设计模式语言分类模型的推荐效果,将代码、类图或对象协作纳入考虑范围,提出了一种基于 CodeBERT 的设计模式分类挖掘模型 dpCodeBERT,以实现自然语言与代码语言的对照理解。首先,通过随机组合合成多分类算法数据和代码搜索数据作为模型输入,dpCodeBERT 模型能够获取 transformer 层中的模型为令牌生成的注意力权重;然后,分析令牌和语句注意力权重以发现更有效的模型输入类别,进一步改造训练输入;最后,dpCodeBERT 模型能够通过全连接层将分布式特征映射到样本空间并输出复数值的方式实现具体软件工程任务,如设计模式选择和设计模式代码搜索任务。在拥有 80 个软件设计问题的设计模式选择任务的数据集上的实验结果显示,相比同类基准模型,所提模型在设计模式检测准确率(RCDDP)和平均倒数排名(MRR)两个指标上平均提升了 10%～20%,设计模式选择更加准确。通过深度研究模型数据需求,dpCodeBERT 挖掘了 CodeBERT 对类级代码的理解,探索了 CodeBERT 在设计模式挖掘中的应用,具有预测准确、拓展性强等特点。

关键词:设计模式挖掘;自然语言处理;预训练语言模型;CodeBERT;模型精调;向量化
中图法分类号　TP311

CodeBERT-based Language Model for Design Patterns

CHEN Shifei, LIU Dong and JIANG He
School of Software Technology, Dalian University of Technology, Dalian, Liaoning 116620, China

Abstract　As summarizations of the experiences of practical software design, design patterns are regarded as an effective means for software design assistance. Most of the current researches on design patterns mining aim at recognition of design pattern instance in source codes, modelling design patterns with natural language corpus is largely unexplored. In order to enhance the performance of language model for recommending design patterns with codes, class diagram or object collaboration, a design pattern classification mining model based on CodeBERT, named dpCodeBERT, is proposed, achieving the contrast understanding of design patterns in natural language and programming language. Firstly, multi-classification dataset and code search dataset are generated using random combination and used as inputs of the model. Using dpCodeBERT to get attention weights of each layer of transformer of each token and statement from the inputs. Secondly, the input dataset is further improved by analyzing attention weights and discovering the most important category of inputs. Finally, dpCodeBERT is applied to specific software engineering downstream tasks such as design patterns selection and design patterns code search. The purposes of tasks are accomplished by mapping distributed features to sample space trough fully connected layers and outputting multi values. The result of the experiment on 80 software design problems in design pattern selection task shows that ratio of correct detection of design pattern(RCDDP) and mean reciprocal rank(MRR) of dpCodeBERT are improved by the average of 10%～20% compared with baseline models, and the design pattern selection is more accurate. Through in-depth study of the data demand of the model, dpCodeBERT improves the understanding of class code of CodeBERT and discovers the application of CodeBERT in design patterns mining. It has the characteristics of accurate prediction and great scalability.

Keywords　Design pattern mining, Natural language processing, Pre-trained language models, CodeBERT, Model fine-tuning, Vector quantization

到稿日期:2023-01-31　返修日期:2023-05-22
基金项目:国家自然科学基金(61722202)
This work was supported by the National Natural Science Foundation of China(61722202).
通信作者:江贺(jianghe@dlut.edu.cn)

1 引言

软件设计是整个软件开发生命周期中难度最大的任务之一[1]。为了克服这一困难，许多辅助软件设计的方法相继被提出，其中设计模式(Design Pattern，DP)是对实际软件设计方案的经验性总结，也是一种行之有效的节省软件设计成本的经验方法[2-3]。在已有的设计模式中，最广为人知的就是文献[4]中整理总结出的软件设计模式，它们在软件开发中被反复应用。

设计模式挖掘是近年来软件技术领域的热点问题。Mayvan 等总结了预训练语言模型出现之前的主流 DP 挖掘工具与方法[5]，包括 DP 表达式的代数形式[6]，基于 SVM、KNN 和 C4.5 等机器学习算法的 DP 识别[7-8]，基于微结构信息的 DP 识别[2]和基于语义图的 DP 识别[9]等。Dwivedi 等先后通过神经网络和逻辑回归[10]、假阳性与假阴性结果问题[11]改进 DP 识别技术；Petterson 等[12]提出了设计模式重叠问题的解决方案；Yu 等[13]研究了 DP 变体检测；文献[14]则关注了评估基准、评估策略等影响 DP 挖掘正确率的有效性威胁。然而，这些研究面向的任务是在源代码中识别设计模式的实例，由于这些文献大多依赖于从设计模式的理论描述中得出的特征，目前用自然语言语料对设计模式建模研究的工作识别准确率不高且并不多见。

预训练语言模型的出现极大地改善了对特征选取的依赖，其优势在于利用海量的无标注语料学习得到通用的语言词嵌入(Embedding)，来表征单词序列和语句序列之间更深层次的关联。第一代模型如 Word2Vec 和 Glove 学习得到与上下文无关的词嵌入，用于 DP 建模时 DPWord2Vec 已在通用指标上超过代表性的 DP 分类方法[15] 24.2%～120.9%[16-17]。但这类方法在构建语料库的过程中通常会舍弃所有代码片段，未将 DP 代码、类图或对象协作纳入考虑的范围，也不能实现自然语言与代码语言的对照，难以“理解”场景与设计模式的语义关系[17]。第二代模型如 BERT，GPT 和 CodeBERT 得到的是与上下文相关的词嵌入，能够区分一词多义的语境，理论上更有希望克服 DP 建模的困难。然而，这些二代模型需要的是文本序列，且效果会受到文本长度和文本长依赖的制约。其次，用于 DP 建模的样本数量少，例如 DPWord2Vec 的 Douglass 数据集中 DP 描述用词少，有接近一半的文字描述不足 150 个英文词；有的 DP 问题如“A large monolithic design does not scale well as new graphing or monitoring requirements are levied”，描述太过简洁，即使是开发人员也很难理解。因此，作为下游任务的 DP 建模存在小数据上的训练不足和过拟合的风险。

本研究针对现有 DP 识别方法的不足和任务数据的风险，提出了一种基于 CodeBERT 的设计模式建模方法，对设计模式和自然语言的一般关系进行建模，通过词嵌入技术和可扩展的精调数据实现进行推理和学习，更好地利用自然语言信息进行 DP 相关任务的辅助。本文的主要贡献如下：1)提出了一种以 CodeBERT 词嵌入和可扩展的数据合成为特点的 DP 建模方法；2)通过分析 GoF 基准数据集的注意力输出结果，发现单词、注意力、协作关系等特征表达对 DP 分类挖掘性能的重要影响；3)将所提方法应用于两个 DP 相关的任务，即设计模式选择和设计模式代码推荐，实验评估结果表明，其效果优于一些专门处理这些任务的主流方法。这表明 dpCodeBERT 具有基于语义特征和上下文关系的表征学习能力，极大地提升了模型理解设计模式的性能，为相关工作优化提供了借鉴。

2 基于 CodeBERT 的设计模式挖掘

2.1 模型整体框架

CodeBERT 通过在文本序列之前使用分类令牌[CLS]，以及用分隔符令牌[SEP]连接句子来实现特征表达。将与[CLS]令牌相对应的最终隐状态用作聚合序列表征，从中预测分类任务的标签。对预训练的 CodeBERT 模型进行精调，需要针对 DP 分类挖掘任务设计恰当的句子序列，甚至是扩展网络结构。据此提出 dpCodeBERT，其整体框架如图 1 所示。

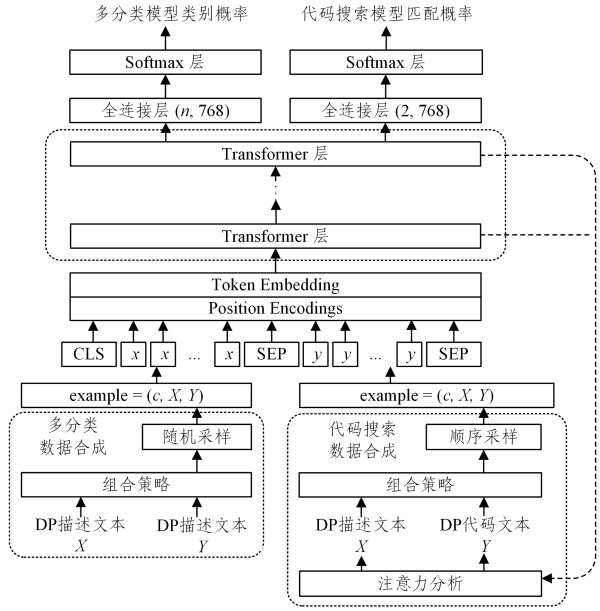

图 1 dpCodeBERT 整体框架图

Fig. 1 Overall framework of dpCodeBERT

dpCodeBERT 将 CodeBERT 模型扩展为两个分支：一个支路如图 1 左侧所示，由多分类数据合成模块、嵌入层、Transformer 层、全连接层(n,768)和 Softmax 层构成；另一个支路如图 1 右侧所示，由代码搜索数据合成模块、嵌入层、Transformer 层、全连接层(2,768)和 Softmax 层构成。虽然两个支路共享同一个 Transformer 骨干网络，但它们是为两个下游任务专门设计的。多分类数据合成模块中，模型通过 DP 描述文本按一定策略组合的方式生成设计模式选择数据集，然后通过随机采样输入模型；而在代码搜索数据集合成模块中，模型则是按一定策略组合 DP 描述文本和模式源代码生成代码搜索数据集，且可通过输出每层 Transformer 的注意力分数，对数据集进行改良以增强语句、函数、类级的高层特征表达。由于该任务数据集中文本与代码需要一一匹配，因此需要通过顺序采样输入模型。在流程上，数据集会通过 CodeBERT 嵌入层、Transformer 层、全连接层以及 Softmax 层，最终得到输出概率，并根据此概率得到具体的预期实验数据。

值得注意的是，我们对两段文本之间、英文描述和代码之间的隐态关联更感兴趣，因此将两段文本组合起来以增强 CodeBERT。形式上，样本模型记为(c,X,Y)，其中 $X=\{x_1,x_2,\cdots,x_{|X|}\}$表示第一段分词后的序列，$Y=\{y_1,y_2,\cdots,y_{|Y|}\}$表示第二段分词后的序列，整数 $c\in Z$ 表示与[CLS]令牌相对应的任务语境下的类别。全连接层的输出可以根据具体的任务进行调整，例如代码搜索任务的输出张量大小为(2,768)，在设计模式选择任务中可以根据 DP 个数指定分类数 n。

2.2 dpCodeBERT 多分类模型

多分类模型主要用于 DP 选择任务，训练数据中的训练样本和评估样本都是 DP 的文字描述。针对相当比例的文字描述太短的问题，算法 1 以模式名称集 PN、模式描述集 LD、比例 ρ、单词数下限 β_{low} 和上限 β_{up} 为输入，按比例合成满足单词数限定的用于训练 D_{train} 和评估 D_{eval} 的多分类序列数据(包含标签、模式名和短描述)。模型通过学习文本句子之间在隐状态空间中的关联关系，对设计模式问题进行 n-标签分类。

算法 1 多分类数据合成算法

输入：(PN，LD，ρ，β_{low}，β_{up})

输出：(D_{train}，D_{eval})

1. $D_{train}\leftarrow\emptyset$；$D_{eval}\leftarrow\emptyset$；TD←∅；VD←∅；
2. for k←1 to |PN| do /* 止于第 10 行 */
3. s←1/* 行号 */；SD[k]←∅ /* 短描述 */；
4. for l←1 to |LD[k]| do /* 止于第 10 行 */
5. /* 对于第 k 个 DP 的第 l 行描述 */
6. if words(LD[k]$_s^l$) $<\beta_{low}$ then continue
7. if words(LD[k]$_s^l$) $<\beta_{up}$
8. then SD←LD[k]$_s^l$；s←l+1；
9. else SD←LD[k]$_s^{l-1}$；s←l；
10. 更新 SD[k]←SD[k]∪SD
11. for k←1 to |PN| do /* 止于第 16 行 */
12. C←k；(X,Y)←TD[k]的任意一对描述；
13. if 概率 p >ρ then
14. $D_{train}\leftarrow D_{train}\cup(C,X,Y)\cup(C,Y,X)$
15. else $D_{eval}\leftarrow D_{eval}\cup(C,X,Y)\cup(C,Y,X)$
16. return(randomize(D_{train})，randomize(D_{eval}))

算法 1 将 PN 和 LD 视为有序集合，第 3－10 行表示第 k 个设计模式拼接满足单词数目限制的若干短描述；第 13－15 行表示合成对称性的样本，按比例分配到训练集和验证集；第 16 行随机化样本次序后返回结果。

2.3 dpCodeBERT 代码搜索模型

代码搜索模型主要用于 DP 代码搜索任务，其输入是由标签、英文描述以及源代码组成的序列样本。模型通过学习文本嵌入和代码嵌入在隐状态空间中的关联强度，从而识别输入文本的匹配代码。

样本模型(c,X,Y)中，句子 X 为文本，句子 Y 是文本或是 X 的代码，注意 $c=\{0,1\}$用于指示 X 和 Y 在源设计文档中是单独的还是前后依赖的。文本情况下的标签 1 表示 Y 依赖于 X。对于文本和代码，一个典型的样本例子为：

[CLS] Provide an interface for creating families of related or dependent objects without specifying their concrete classes. … [SEP]class A{\n private static A obj;\n private A(){} \n public static A getA(){…}\n}\n [SEP]. 这种情况的 $c=1$，表示 Y 是 X 的实现代码。

总体而言，我们有 3 种与不同输入数据模式相对应的训练制度：仅描述(des-only)、仅代码(code-only)和描述代码(des-code)。对于 des-code，使用标准的 CodeBERT 双语言目标用于训练模型，对于 des-only 和 code-only，进一步使用同一语言对齐目标，总体训练目标是使 3 个单目标的加权总和最小。继续训练时，des-only 目标迫使 dpCodeBERT 在理解 DP 描述和参与者协作方面做得很好；code-only 目标迫使它在程序源码上学习 DP 方法调用次序；des-code 目标迫使 dpCodeBERT 学习自然语言和程序语言之间的对应关系。因此，合成训练数据的组合策略 π 与 3 个训练目标一一对应。具体地，算法 2 以结构化的模式定义 PD、组合策略 π 和模式 $patname$ 的程序源码 SRC 为输入，满足样本模型定义的句子序列。

定义 1 模式集合 PD 中的任意一个模式定义为 $PD[k]=(name,I,D,P,C,S)$，其中 $PD[k].name$ 为模式 k 的名字，$PD[k].I$ 为其意图，$PD[k].D$ 为其文字描述，$PD[k].P$ 为其参与者，$PD[k].C$ 表示参与者间的协作步骤，$PD[k].S$ 为其程序源码。

定义 2 任意模式参与者 p 包含类名、接口、属性和方法等信息，形式上定义为 $p=Class\times Interface\times Attribute\times Method$。

算法 2 代码搜索数据合成算法

输入：(PD，π，ρ，SRC，patname)

输出：(D_{train}^{π}，D_{eval}^{π})

1. 初始化：$D_{train}^{\pi}\leftarrow\emptyset$；$D_{eval}^{\pi}\leftarrow\emptyset$；
2. for k←1 to |PD| do /* 止于第 25 行 */
3. if π=des_only then /* 止于第 12 行 */
4. j←不同于 k 的新序号；
5. X←PD[k].I；Y←PD[j].I；
6. update(ρ，D_{train}^{π}，D_{eval}^{π}，(0,X,Y))
7. X←PD[k].I；Y←clause(PD[j].P)；
8. update(ρ，D_{train}^{π}，D_{eval}^{π}，(1,X,Y))
9. for i←1 to |PD[k].C|－1 do /* 止于第 11 行 */
10. X←PD[k].C[i]；Y←PD[k].C[i+1]；
11. update(ρ，D_{train}^{π}，D_{eval}^{π}，(1,X,Y))
12. if PD[k].name=patname then
13. SRCP←participants(SRC) /* 抽取参与者 */
14. if π=code_only then
15. for i←1 to |PD[k].S|－1 do /* 止于第 17 行 */
16. X←PD[k].S[i]；Y←PD[k].S[i+1]；
17. update(ρ，D_{train}^{π}，D_{eval}^{π}，(1,X,Y))
18. (c,X,Y) ←match(PD[k],i,SRCP)
19. update(ρ，D_{train}^{π}，D_{eval}^{π}，(c,X,Y))
20. if π=des_code then
21. for i←1 to |PD[k].C| do /* 止于第 11 行 */
22. X←PD[k].C[i]；Y←PD[k].S[i]；
23. update(ρ，D_{train}^{π}，D_{eval}^{π}，(1,X,Y))
24. (c,_,Y) ←match(PD[k],i,SRCP)
25. update(ρ，D_{train}^{π}，D_{eval}^{π}，(c,X,Y))
26. return(D_{train}^{π}，D_{eval}^{π})

算法 2 中，结构化的模式定义 PD 是手工创建的，update(,,,)函数按比例 ρ 扩增训练样本 D^{π}_{train} 和评估样本 D^{π}_{eval}，与算法 1 第 13－15 行的区别是样本是非对称的。第 12－13 行表示从额外的模式源码 SRC 中为 patname 分别抽取更多的参与者信息 SRCP。调用 match(,,)函数，返回 SRCP 中与当前模式定义参与者匹配的描述和代码，进一步扩增训练评估数据。具体地，假如考虑设计模式 $PD[k].name=$"*factory-method*"，一是根据定义 2，该设计模式协作者分别为 $PD[k].P=\{\{$"*Product*","*ConcreteProduct*"$\}$,$\{$"*Creator*","*ConcreteCreator*"$\}$,$\{$"*Client*"$\}\}$，因为接口实现关系 *isInterface*("*Concrete-Product*","*Product*")，*match*(,,)要返回的第 1 步协作是 $PD[k].C[1]=$"*Product defines the interface of objects the factory method creates.*"。二是算法第 13 行，可简单地使用正则表达式匹配，或者复杂地通过程序解析在抽象语法树上使用访问者模式，实现从源代码中抽取协作者信息 *SRCP*(定义 2)。例如，通过某协作者：*SRCP.Class*=$\{$"*aPr*"$\}$，*SRCP.Interface*=$\{$"*Pr*"$\}$，*SRCP.Method*=$\{\}$，*SRCP.Attribute*=$\{\}$可知 *isInterface*("*aPr*","*Pr*")。根据接口以及类和类间依赖关系，可知"*aPr*"与"*Concrete-Product*"匹配程度较高，*match*(,,)返回的第 1 步协作者代码是 *SRC* 中的"*aPr*"接口实现代码，这时令 $c=0$。如果失配或者匹配度低于设定阈值时，则 *match*(,,)函数返回一对失配的描述和代码，这时令 $c=0$。

3 实验与结果分析

下面将 dpCodeBERT 应用于设计模式选择和设计模式代码推荐两个 DP 相关的任务，对实验结果进行评估。

3.1 设计模式选择

3.1.1 实验数据

通过算法 1 合成的设计模式选择数据集的统计结果如表 1 所列。和文献[15]一样，本实验使用的训练数据集是设计模式的文字描述信息，包括设计模式的定义、举例说明、使用场景等，具体来自 3 本英文版的设计模式书籍《GoF》[4]、《Douglass》[18]和《Security》[19]。为使用方便，对照书名将训练数据集分别称为 GoF，Douglass 和 Security，其中 GoF 训练数据集由 23 种软件设计模式的文字描述信息组成；Douglass 训练数据包含 34 个与嵌入式软件开发有关的设计模式的描述信息；Security 训练数据描述的是 46 个有关软件安全行的设计模式。与文献[15]一致，本实验测试数据一共有 80 个软件设计问题，按 30:30:20 的比例分别覆盖了 GoF，Security 和 Douglass 这 3 本设计模式书籍。对于测试数据集中的每个软件设计问题，正确答案唯一。遵照文献[1,15]的设定，实验分别在 GoF，Douglass 和 Security 设计模式数据集上进行训练和预测。

表 1 设计模式选择任务数据集统计

Table 1 Statistics of datasets in design patterns selection task

数据集	训练数据	验证数据
GoF	5508	3300
Douglass	3228	2770
Security	13024	11114

实现上，测试数据集以 jsonl 文件格式存储，目的是方便模型存取测试集数据。在测试数据集中，模型需求的有效信息为标签、文本 A 以及文本 B，对应到 jsonl 文件中分别放在"label""maincode"以及"shortdes"关键词下，分词方式与 CodeBERT 模型一致。值得注意的是，由于本节实验进行的是自然语言文本多标签分类任务，以测试 dpCodeBERT 模型对自然语言文本的理解能力，在本次实验的测试以及训练数据集中，关键词"maincode"下存储的也是自然语言文本。为了能够让模型正常学习不同设计模式的区别，在训练数据集中，任意两段完整连续的描述同一设计模式的自然语言设计模式描述文本会被组合在一起，作为一个样本的文本 A 以及文本 B 形成一条测试用例，分别放在 jsonl 文件中同一行的"maincode"和"shortdes"下。在测试数据集中，为了让模型能够继承训练中理解两段文本关系从而进行分类任务的工作方式，软件设计问题同样会被放在"shortdes"下，而"maincode"为空。

3.1.2 评估指标

设计模式选择任务需要对模型预测的设计模式名称和真实答案进行比对，因此通常使用设计模式检测正确率(RCDDP)对模型进行评估[1,15]，具体定义为：

$$RCDDP=\frac{1}{N}\sum_{i=1}^{N}\frac{|SDP_i\bigcap CDP_i|}{|SDP_i|}$$

其中，N 为测试数据集中软件设计问题的个数，CDP_i 表示第 i 个软件设计问题的真实答案(在本节使用的数据集中只有一个设计模式是正确的)，SDP_i 表示模型为第 i 个软件设计问题所选择/预测的设计模式是什么。

根据上述定义，RCDDP 指标的值会受到阈值 $\sigma1$ 和 $\sigma2$ 的影响，不同算法得到最优的 RCDDP 所需要的阈值可能是不同的，这会使得对比算法优劣的标准变得更复杂[2]。为避免阈值对实验结果的影响，本实验还使用平均倒数排名(MRR)对不同算法的性能进行评估。

MRR 是国际上通用的对搜索算法进行评价的机制，常用于软件工程相关任务[20]。具体定义为：

$$MRR=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_c^i}$$

其中，N 为测试数据集中软件设计问题的个数，$rank_c^i$ 为第 i 个软件设计问题的真实答案在模型给出的结果中按 logits 值大小排序的位置序号。若真实答案对应的 logits 值是所有备选答案中最大的，则该排序值为 1；若真实答案对应的 logits 值是所有值中第二大的，则排序值为 2，以此类推。根据 MRR 的公式定义，模型在进行分类时若不认为软件设计问题的真实答案有较大的可能为预测结果，它将不会获得好的 MRR 评估值。反之若模型认为该可能性很大，则其可以取得理想的 MRR 值。同时，这项指标对模型的评估将不会受到阈值的影响，同时可以得到公正准确的结果。

为对比分析 dpCodeBERT 的性能表现，本实验选用的基线算法包括 DPWord2Vec[16]以及文献[1,15]提出的设计模式选择方法 DPS-SVM 和 DPS-FCM。其中 DPS-SVM 为使用支持向量机(SVM)的分类算法模型，DPS-FCM 为使用模糊 C-Means[21]聚类方法的模型。虽然这两个模型并非完整

模型，而是方法框架，本节实验仍然可以选用文献中表现良好的参数设置方式，使其可以作为软件设计问题分类算法模型使用。同时本节还使用了特征选择方法[22]约减文本向量的维数。为了找到较合适的特征个数，获取最好的 MRR 指标所需的设置，本节实验从 50 步长开始，逐渐增加步长，每次增加 50 步长，以此探索最佳的参数设置。对于设计模式选择任务，两个阈值取值范围分别为{0，0.1，0.2，…，1.0}和{0，0.01，0.02，…，0.1}。实验会记录所有阈值下两种方法取得的 RCDDP 值，并展示最好的结果。

3.1.3 结果分析

表 2 列出了不同模型/算法在 3 个测试数据集上的性能表现，包括评估指标及其参数设置，用粗体表示性能最好的结果。从表中可以看出，在所有测试数据集上，dpCodeBERT 的表现最佳：对于 GoF 测试数据集，在 RCDDP 和 MRR 两个指标上分别超出 DPS-SVM 37%和 26%，超出 DPS-FCM 方法 61%和 60%，超出 DPWord2Vec 方法 33%和 22%；对于 Douglass 测试数据集，在 RCDDP 和 MRR 两个指标上分别超出 DPS-SVM 方法 25%和 20%，超出 DPS-FCM 方法 43%和 36%，超出 DPWord2Vec 方法 22%和 18%；对于 Security 测试数据集，在 RCDDP 和 MRR 两个指标上分别超出 DPS-SVM 方法 11%和 12%，超出 DPS-FCM 方法 52%和 54%，超出 DPWord2Vec 方法 4%和 3%。综上，dpCodeBERT 在 RCDDP 和 MRR 这两个指标上平均超出 DPS-SVM 方法 25%和 19%，超出 DPS-FCM 方法 52%和 50%，超出 DPWord2Vec 方法 20%和 14%。

表 2 设计模式选择任务的实验结果和参数设置

Table 2 Experiment results and parameter settings of design pattern selection task

数据集	算法	*RCDDP*	*MRR*	($\sigma1$,$\sigma2$)
GoF	DPS-SVM	0.5333	0.6368	(0,0)
	DPS-FCM	0.3333	0.3417	(0,0)
	DPWord2Vec	0.5667	0.6808	(0,0)
	dpCodeBERT	**0.8478**	**0.8636**	(0,0.09)
Douglass	DPS-SVM	0.6000	0.6917	(0,0)
	DPS-FCM	0.4583	0.5542	(0,0.08)
	DPWord2Vec	0.6225	0.7058	(0,0.1)
	dpCodeBERT	**0.8000**	**0.8623**	(0,0)
Security	DPS-SVM	0.8000	0.8278	(0,0)
	DPS-FCM	0.4333	0.4500	(0,0)
	DPWord2Vec	0.8667	0.9111	(0,0)
	dpCodeBERT	**0.9000**	**0.9400**	(0,0)

总体来说，DPS-FCM 的效果较差，一个可能的原因是它在确定设计模式候选集时遗漏了很多正确的设计模式。dpCodeBERT 在 GoF 和 Security 设计模式集合上取得的效果是最好的，这可能是因为 GoF 和 Security 数据集一般通过一整个章节描述设计模式，根据设计模式的参与者分工、代码架构等信息对每个设计模式进行了充足的定义，使得 dpCodeBERT 能够更准确地筛选出有效信息，并据此选择最合理的设计模式。

3.2 设计模式代码搜索

本节实验主要包含两个部分，分别是注意力分析和设计模式代码搜索。注意力分析实验对比设计模式数据集和主要包含函数代码片段的 CodeSearchNet 数据集中的 java 数据集各自从模型中获取的注意力权重的不同，分析其原因并对设计模式数据集进行改进。设计模式代码搜索是本节实验的主体，通过学习改进后的设计模式数据集，分析代码与文本的匹配情况。

3.2.1 实验数据

为了训练模型学习不同设计模式的描述信息，实验数据集选择从 3 本英文设计模式书中获取，并根据设计模式书的名字将数据集分别命名为 douglass，gof 以及 security。如本文模型构建中所述，这些设计模式书通常使用文字与代码穿插的方式，将设计模式进行形象化描述，以此表示设计模式的代码架构、主要功能、应用场景等信息。这部分数据也将用作注意力分析以改进数据集的构建方式，分析过程参照文献[23]，同时使用另一部分数据集，从 CodeSearchNet 数据集中获取[24]，该数据集包含从开源软件项目获取的代码以及代码对应的注释。由于设计模式数据集中代码文本为 Java 代码，因此本节实验同样选取 CodeSearchNet 数据集中 Java 代码的部分，模型可以通过这部分数据学习得到基本的 Java 代码中的语义关系和结构。

3.2.2 注意力分析

通过突出放大重点令牌的方式，形成了设计模式数据集的词云，如图 2 所示。该数据集包含 15627 条设计模式代码。

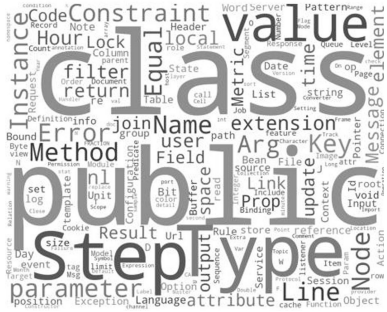

图 2 设计模式数据集词云

Fig. 2 Design patterns dataset word cloud

图 3 给出了一个设计模式示例代码，并通过令牌底色深浅形成了语句以及令牌注意力权重的热图。

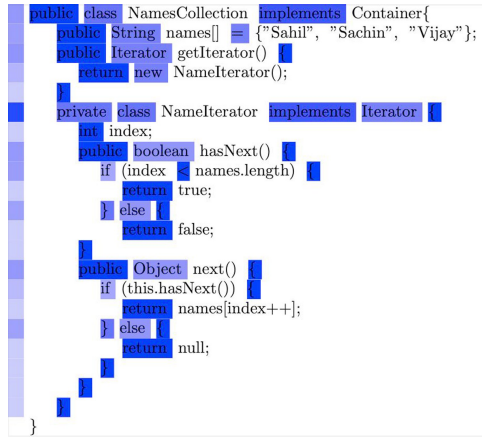

图 3 设计模式代码注意力权重热图

Fig. 3 Heatmap of attention weights on source code of a design pattern

与上述的总结类似，方法签名是模型分析中最重要的代

码，它能够说明当前代码语句在整体代码结构中的位置以及作用。分支语句的注意力权重是最小的，这可能是由于代码层次被扩充到类级别后，模型将更多的注意力放到了类级别语句，使得原本注意力权重偏低的分支、循环等语句注意力被进一步分散。同时，在类级代码中，这类语句对类的理解帮助很小。

图 4 给出了 dpCodeBERT 为设计模式代码文本中不同类型的语句分配的具体注意力权重值大小。未经设计模式代码训练模型的注意力分析结果和文献[23]相同，设计模式数据集虽然没有让模型对代码的理解方式发生根本性的改变，但由于 CodeSearchNet 数据集中均为函数级代码，设计模式数据集引入了新的类级代码，也产生了新的 class 语句，同时 class 语句也获得了较高的注意力分数，与方法签名语句相近。这很有可能是模型理解类的定义的标志。

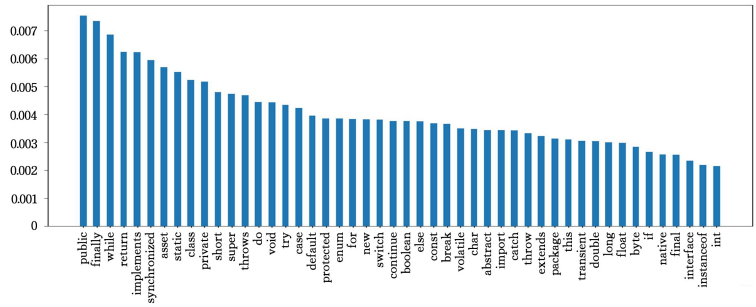

图 4 dpCodeBERT 获取语句注意力权重柱状图

Fig. 4 Attention weights of statements learned by dpCodeBERT

由图 2—图 4 可以得出结论：虽然模型没有经过大批量数据集再训练，其仍然注意到了设计模式数据集中类语句对代码整体含义的影响。这说明模型正确地理解了设计模式代码与一般函数代码的区别，并学习了设计模式代码与设计模式描述之间的匹配关系。

3.2.3 结果分析

表 3 列出了 dpCodeBERT 模型在阶段性训练机制下的性能表现。评估指标包括 Accuracy(准确率)和 F1 值，用粗体表示最佳结果。注意，dpCodeBERT 训练始于 CodeSearchNet 数据集，然后按 des_only，code_only 和 des_code 的策略完成第二阶段的模型精调，相应得到 3 个版本的 dpCodeBERT 模型。若训练数据都是纯文本，即 GoF(des_only)只使用 DP 文字描述，此时 dpCodeBERT 在准确率和 F1 值评估指标上远低于 DPWord2Vec。但是，由于 DPWord2Vec 不具备处理代码文本的能力[17]，在接下来的代码搜索任务上其性能表现会显著退化。虽然 dpCodeBERT 的起点不高，但通过 code_only 和 des_code 阶段的模型调整，从表 2 可以看出 dpCodeBERT 模型的 3 个版本的准确率和 F1 值逐步提高，这说明连续训练 dpCodeBERT 学习到了自然语言描述的设计模式和源代码之间的关联关系。相比文献[23]中的 DietCode，dpCodeBERT 捕捉到的是对象、类级别的程序语言概念。

表 3 设计模式代码搜索任务的实验结果

Table 3 Results on design pattern code search

算法	数据集	*Accuracy*	*F1*
DPWord2Vec	GoF(des_only)	0.8000	0.8899
dpCodeBERT_v1	GoF(des_only)	0.3333	0.3417
dpCodeBERT_v2	GoF(code_only)	0.6000	0.7499
dpCodeBERT_v3	GoF(des_code)	**0.8260**	**0.8260**

尽管 dpCodeBERT 在处理纯文本任务上性能不佳，但是纯代码训练模型的表现仍优于纯文本模型，这很可能是因为其基础模型 CodeBERT 是针对代码文本优化的预训练模型，在处理代码文本任务上的性能优于处理自然语言文本。表 2 也说明了注意力分析过程中对数据集的改造能够突出代码中的重要信息，使其能被模型接受，从而提高模型精调的效率。

结束语 本文提出了一种基于 CodeBERT 的设计模式挖掘模型，与传统基于机器学习的设计模式模型不同，该模型实现了对设计模式和自然语言的一般关系进行建模。在设计模式选择任务和代码搜索任务中，dpCodeBERT 通过词嵌入技术和模型精调均取得了优于 dpWord2Vec 的效果，不仅证明了其在代码文本处理上的优越性，也证明了其在传统自然语言处理任务上同样优于传统模型。未来将致力于设计模式变体与设计模式共享实例挖掘方法的优化、设计模式参与者间附加关系的检测等研究工作。

参 考 文 献

[1] HASHEMINEJAD S M H, JALILI S. Design patterns selection: An automatic two-phase method [J]. The Journal of Systems and Software, 2012, 85(2): 408-424.

[2] FONTANA F A, MAGGIONI S, RAIBULET C. Design patterns: a survey on their micro-structures [J]. Journal of Software: Evolution and Process, 2013, 25(1): 27-52.

[3] ZHANG C, BUDGEN D. What do we know about the effectiveness of software design patterns? [J]. IEEE Transactions on Software Engineering, 2012, 38(5): 1213-1231.

[4] GAMMA, E, HELM R, JOHNSON R, et al. Design Patterns: Elements of Reusable Object-Oriented Software [M] // Reading. MA: Addison-Wesley, 1995.

[5] MAYVAN B B, RASOOLZADEGAN A, YAZDI Z G. The state of the art on design patterns: a systematic mapping of the literature [J]. Journal of Systems and Software, 2017, 125(3): 93-118.

[6] ZHU H, BAYLEY I. An algebra of design patterns [J]. ACM Transactions on Software Engineering and Methodology, 2013, 22(3): 23-61.

[7] ZANONI M, FONTANA F A, STELLA F . On applying machine learning techniques for design pattern detection [J]. Journal of Systems and Software, 2015, 88(5): 102-117.

[8] CHIHADA A, JALILI S, HASHEMINEJAD S M H, et al. Source code and design conformance, design pattern detection from source code by classification approach [J]. Applied Soft Computing, 2015, 26(1): 357-367.

[9] MAYVAN B B, RASOOLZADEGAN A. Design pattern detection based on the graph theory [J]. Knowledge-Based Systems, 2017, 120(1): 211-225.

[10] DWIVEDI A K, TIRKEY A, RATH S K. Applying learning-based methods for recognizing design patterns[J]. Innovations in Systems and Software Engineering, 2019, 15(2): 87-100.

[11] DWIVEDI A K, TIRKEY A, RATH S K. Software design pattern mining using classification-based techniques [J]. Frontiers of Computer Science, 2018, 12(5): 908-922.

[12] PETTERSON N, LÖWE W, NIVRE J. Evaluation of accuracy in design pattern occurrence detection[J]. IEEE Transactions on Software Engineering, 2010, 36(4): 575-590.

[13] YU D, ZHANG P, YANG J, et al. Efficiently detecting structural design pattern instances based on ordered sequences [J]. Journal of Systems and Software, 2018, 91(5): 35-56.

[14] XIAO Z Y, HUANG H, HE P, et al. Evaluation strategy of efficiency in design pattern detection tools [J]. Journal of Frontiers of Computer Science and Technology, 2018, 12(3): 380-392.

[15] HUSSAIN S, KEUNG J, KHAN A A. Software design patterns classification and selection using text categorization approach [J]. Applied Soft Computing, 2017, 58: 225-244.

[16] LIU D, JIANG H, LI X, et al. DPWord2Vec: better representation of design patterns in semantics [J]. IEEE Transactions on Software Engineering, 2020, 48(4): 1228-1248.

[17] LIU D. Data-Driven Software Design Pattern Analysis and Application[D]. Dalian: Dalian University of Technology, 2022.

[18] DOUGLASS B P. Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems [M]. Boston MA: Addison-Wesley/Longman Publishing, 2002.

[19] SCHUMACHER M, FERNANDEZ-BUGLIONI E, HYBERTSON D, et al. Security patterns: Integrating security and systems engineering [M]. Hoboken: John Wiley & Sons, 2006.

[20] BAO L, XING Z, XIA X, et al. Psc2code: Denoising code extraction from programming screencasts [J]. ACM Transactions on Software Engineering Methodology, 2020, 29(3): 1-21, 48.

[21] BEZDEK J C. Pattern recognition with fuzzy objective function algorithms [M]. New York: Springer Science & Business Media, 2013.

[22] UYSAL A K. An improved global feature selection scheme for text classification [J]. Expert Systems with Applications, 2016, 43: 82-92.

[23] ZHANG Z, ZHANG H, SHEN B, et al. Diet code is healthy: Simplifying programs for pre-trained models of code[C]// Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2022: 1073-1084.

[24] HUSAIN H, WU H H, GAZIT T, et al. Codesearchnet challenge: Evaluating the state of semantic code search[J]. arXiv: 1909.09436, 2019.

CHEN Shifei, born in 1998, master. His main research interest is natural language processing.

JIANG He, born in 1980, Ph.D, professor, is a distinguished member of China Computer Federation. His main research interests include system software and software engineering.

（责任编辑：杨雪敏）