计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200800193

基于多维度特征和混合神经网络的代码可读性评估方法

米 庆 郭黎敏 陈军成
北京工业大学信息学部 北京 100124
(miqing@bjut.edu.cn)

摘 要 对代码可读性进行定量、准确的评估是有效保障软件质量、降低沟通成本以及维护成本、提高软件开发和演化效率的重要途径。然而,现有的针对代码可读性评估的研究方案大多是基于特征工程的,受到源代码表征方式、技术手段等多方面因素影响,其评估准确率并不高。为此,文中采用深度学习作为主要技术手段,提出了一种基于多维度特征和混合神经网络的代码可读性评估方法,通过整合并运用各种单一神经网络的优势,从字符级、词条级等不同维度挖掘源代码中蕴含的结构信息和语义信息,最终实现对代码可读性的量化评估。实验表明,该方法能够获得高达84.6%的评估准确率,比单独使用卷积神经网络提升了9.2%,比单独使用循环神经网络模型提升了6.5%,并且其表现优于现有的5个可读性模型,验证了所提出的多维度特征和混合神经网络的有效性。

关键词:代码可读性;代码表征;深度学习;代码分析;软件质量保障
中图法分类号 TP311

Code Readability Assessment Method Based on Multidimensional Features and Hybrid Neural Networks

MI Qing,GUO Li-min and CHEN Jun-cheng
Faculty of Information Technology,Beijing University of Technology,Beijing 100124,China

Abstract Quantitative and accurate assessment of code readability is an important way to ensure software quality,reduce communication and maintenance costs,and improve the efficiency of software development and evolution. However,existing code readability studies depend mainly on the manual feature engineering method,which is likely to limit the model performance due to factors such as code representation strategies and technical means. Unlike prior studies,we propose a novel code readability assessment method based on multidimensional features and hybrid neural networks by using the technique of deep learning. Specifically,we first propose a representation strategy with different granularity levels to transform source codes into matrices and vectors as the input to deep neural networks. We then build a CNN-BiGRU hybrid neural network that can automatically learn structural and semantic features from the source code. The experimental results show that our method is able to achieve an accuracy of 84.6%,which is 9.2% higher than CNN alone and 6.5% higher than BiGRU alone. Moreover,our method can outperform five state-of-the-art code readability models,which confirms the feasibility and effectiveness of multidimensional features and hybrid neural networks proposed in this study.

Keywords Code readability,Code representation,Deep learning,Code analysis,Software quality assurance

1 引言

在软件产品的生存周期中,约有70%的成本耗费于维护阶段[1-2],而代码阅读在软件维护过程中耗时占比最高[3-4],开发者必须在进行任何更改比如缺陷修复或功能扩展之前,阅读并完全理解程序源代码。因此,对于一个软件系统而言,不仅需要重视其宏观上的架构决策,比如技术选型、模块划分、层次设计等,同时必须重视其微观上的代码细节,比如命名、注释、缩进等,以确保代码的高可读性,最小化代码阅读者需耗费的时间和精力[5]。

代码可读性(code readability or program readability)又称为易读性,通常是指源代码被阅读和理解的难易程度[2],主要关注源代码的写法,比如标识符命名、空间结构等对阅读者阅读速度和理解程度的影响。代码可读性是评价代码质量的一项关键指标[6-7],与代码的可维护性(maintainability)、可重用性(reusability)、可靠性(reliability)、可扩展性(portability)

到稿日期:2020-08-31 返修日期:2021-02-15 本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61702029);北京市自然科学基金(4192004);北京市教委项目(KM201810005023)
This work was supported by the National Natural Science Foundation of China(61702029),Natural Science Foundation of Beijing,China(4192004) and Project of Beijing Municipal Education Commission(KM201810005023).
通信作者:陈军成(juncheng@bjut.edu.cn)

有着紧密的关联[8-9]，而综合各种因素对代码可读性进行量化的过程就是代码可读性评估。对代码可读性进行定量、准确的评估是有效保障软件质量、降低沟通成本以及维护成本、提高软件开发和演化效率的重要途径[4,10]，然而学术界在相关领域的研究甚少，目前仍处于起步阶段。

现有针对代码可读性评估的研究大多是基于机器学习框架的[2,10-11]，即利用人工抽取的特征在训练集上构建分类器，在这个方向上的研究主要关注如何设计更加有效的特征来获得更好的分类效果。不同于传统的基于特征工程的方法，本文引入深度学习技术，提出了一种基于多维度特征和混合神经网络的代码可读性评估方法，通过综合利用卷积神经网络和循环神经网络的优势直接提取并学习隐藏在源代码中的结构特征和语义特征。实验表明，该方法能够有效挖掘源代码中蕴含的可读性信息，获得高达 84.6% 的分类准确率，高于现有的所有可读性模型。

本文的主要贡献在于：

(1)提出了一种基于多维度特征的源代码表征方式，能够最大限度地保留源代码中的可读性信息以供算法和模型使用。

(2)整合并运用各种单一神经网络的优势，从结构、语义等不同角度挖掘源代码中蕴含的可读性信息，实现对代码可读性的量化评估。

2　相关工作

尽管代码可读性这一概念很早就出现在软件工程领域[12-13]，但是早期的研究多集中于探索并优化影响代码可读性的因素。例如，Binkley 等[14] 和 Sharif 等[15] 通过多项实证研究分析了不同的标识符命名风格(即 camelCase 和 under_score)对代码可读性的影响。而随着现代软件规模的日益增长，一线软件开发工程师所面临的开发与维护负担越来越重，工业界开始逐步重视代码可读性的评估问题。在一项对微软 110 名开发与管理人员的调查中，90% 的参与者表示希望拥有代码可读性的评估指标[16]，这一需求激发了学术界对代码可读性系统化、标准化的研究。根据分析思路和关键技术的不同，现有研究大致可以分为公式法与分类法两种。

2.1　基于公式法的代码可读性评估

在早期的代码可读性研究中，研究者们偏向于代码可读性公式的研发，通过选取部分影响代码阅读的、可量化的特征作为变量，来构建线性方程。代码可读性公式假定因变量与自变量线性相关，我们将其研究范式归纳为：

$$\boldsymbol{y}=\boldsymbol{\beta}_0+\boldsymbol{\beta}_1\boldsymbol{x}_1+\boldsymbol{\beta}_2\boldsymbol{x}_2+\cdots+\boldsymbol{\beta}_n\boldsymbol{x}_n$$

其中，$\boldsymbol{x}_i$ 表示代码特征，$\boldsymbol{\beta}_i$ 表示系数，$\boldsymbol{y}$ 表示可读性分数。例如，Aggarwal 等[17] 认为代码可读性与注释的详尽程度密切相关，据此提出了一个简单的评估代码可读性的指标 CR(Comments Ratio)，其公式为 $CR=LOC/LOM$，其中 LOC 表示代码的总行数，LOM 表示注释的总行数。CR 值越低，代码的可读性越高。不同于 Aggarwal 等的研究，Posnett 等[9] 将 3 个不同维度的代码特征纳入可读性公式 $z=8.87-0.033V+0.40Lines-1.5Entropy$，其中 V 表示 Halstead 复杂度分析中的程序容量，Lines 表示代码长度，Entropy 表示代码中字符或者词条的熵。z 值越高，代码的可读性越高。参照 Flesch Reading Ease Score(一个广泛使用的英文文本可读性公式)，Börstler 等[18] 提出了 SRES(Software Readability Ease Score)来评估代码的可读性，$SRES=ASL-0.1AWL$，其中 AWL 表示词条(即关键字、标识符和符号)的平均长度，ASL 表示语句或者代码块所包含的平均词条数(由花括号和分号分隔)。SRES 值越低，代码的可读性越高。

使用公式法可以快速获得对代码可读性的评估结果，具有简便、经济、客观等优点。但是，影响代码可读性的因素众多，公式法往往只能考虑几个有限可量化的代码特征，并不能真实、充分地反映一个代码片段是否可读，因此其效度一直颇受争议。

2.2　基于分类法的代码可读性评估

近年来的研究工作倾向于将代码可读性评估形式化为一个分类问题，以一个代码片段作为输入，以一个可读性级别的预测作为输出[2,10-11]。现有的研究方案大多是基于机器学习框架的，首先邀请多位领域专家标注给定代码片段；然后对影响可读性的代码特征进行归纳、分析和提取；最后构建机器学习模型预测代码的可读性类别。

例如，Buse 等[2,4] 于 2008 年构建了第一个正式的代码可读性分类模型。从特征构造角度来看，该模型包含了 25 个较为简单的代码特征，比如标识符的数量、平均行长度等；从模型选择角度来看，Buse 等对比了 Bayesian Classifier，Voting Feature Interval，Multilayer Perceptron 等分类模型，研究表明其分类准确率可以达到 75%～80%，与人工分类的准确率相当。基于 Buse 等的研究，Dorn[11] 从视觉、空间、语言 3 个维度进一步扩充了特征集，相比 Buse 等而言，Dorn 的研究更加重视源代码呈现在集成开发环境中的视觉效果对于可读性的影响。2016 年，Scalabrino 等[10] 从文本可读性评估领域引入了一组针对标识符和注释的文本特征，比如文本连贯性(Textual Coherence)、词汇熟悉度(Identifier Terms in Dictionary)等，并选用 Logistic Regression 作为分类器，研究表明加入所有特征的模型预测效度最好，分类准确率可以达到 81.8%。

分类法能够纳入更多的代码特征，从而尽可能多地从原始数据中提取可读性信息以供模型使用，因此其评估效果与公式法相比具有显著优势。但是这种广泛使用的特征工程方法存在一定的局限性：首先，特征的设计与选取不仅耗时耗力，而且需要大量的领域知识；其次，人工选取的特征之间难免存在冗余与重叠，一方面可移植性不强，另一方面限制了模型的表现。针对这些问题，Mi 等[19] 于 2018 年首次将深度学习技术引入代码可读性评估领域，所提出的模型 DeepCRM 由 3 个结构相同的卷积神经网络组合而成，偏重于从不同粒度进行结构特征的提取，为代码可读性评估提供了一种新的研究路径。

本文研究的方法属于分类法的一种。与传统的基于机器学习的方法[2,10-11] 相比，本文模型不再依赖于特征工程，而是能够从源代码中自动提取并学习可读性相关特征；与基于深度学习的方法[19] 相比，本文模型能够综合利用卷积神经网络和循环神经网络的优势，从结构、语义等不同维度挖掘源代码中蕴

含的可读性信息，进而获得更加准确的代码可读性评估效果。

3 基于多维度特征和混合神经网络的代码可读性评估方法

本文的整体技术方案如图 1 所示。首先，将源代码转化为统一的、结构化的向量形式作为深度学习模型的输入；其次，构建混合神经网络，挖掘源代码中的结构特征和语义特征，以实现准确的代码可读性分类预测。

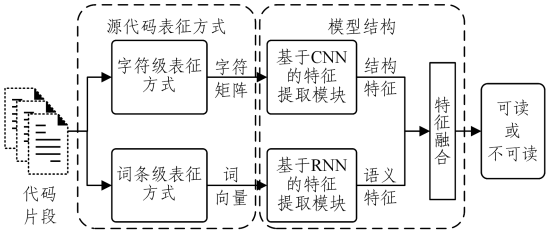

图 1 整体技术方案

Fig. 1 General overview of proposed approach

3.1 源代码的表征方式

源代码的表征方式决定了信息抽取的程度和粒度，进而影响深度学习模型的精度和效率，是代码可读性评估的基石。考虑到经由不同表征方式处理所得到的信息通常具有互补性，因此本文将从字符级、词条级等不同维度构建合适的编码方案。

3.1.1 字符级表征方式

为了有效保留源代码中蕴含的结构信息，本文将源代码中的每个字符都视为独立的符号并进行编码。具体地，采用 Mi 等[19]所提出的方法，将每个代码片段都转化为一个字符矩阵。

如图 2 所示，m 表示代码行数，n 表示最大行长，使用 ASCII 值对字母(即 a－z 和 A－Z)、数字(即 0－9)和符号(比如“＝”和“＋”)进行编码，并保留在以往的代码分析中经常被忽略的空白符(即空格、水平制表符和行结束符)，最终的输出采用二维矩阵的形式，对于矩阵中无符号的位置，使用 NULL(即 0)进行填充。

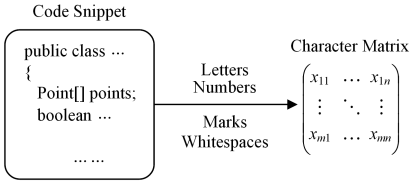

图 2 字符级表征方式

Fig. 2 Character-lever representation

字符级表征方式可以完整保留源代码中隐藏的数据特征，比如代码空间结构(缩进、行长度等)以及标识符书写格式(camelCase、under_score 等)，对于提升代码可读性评估的准确率具有重要的作用。值得注意的是，由于字符级表征方式和语法、语义之间的关系较为松散，如果完全依赖这种表征方式将难以保留足够的可读性信息，因此本文针对代码可读性评估任务额外引入词条级表征方式。

3.1.2 词条级表征方式

为了有效保留源代码中蕴含的语法、语义信息，本文将源代码中的每个词条都视为独立的符号并进行编码。具体地，参照在 NLP 领域中较为成功的词嵌入方法，将每个代码片段都转化为一个词向量序列。

通常而言，源代码中包含 3 类词条：1)表示数据类型(比如 int 和 long)、控制逻辑(比如 if 和 for)等的关键字；2)自定义标识符；3)各种符号(比如“＜＝”和“＋＋”)。不同于传统的面向自然语言的词嵌入方法，本文在词条选取方面做了 3 点改进：1)额外保留对代码可读性有着显著影响的空白符，即将空格、水平制表符和行结束符同样视为词条并进行编码；2)将自定义标识符合并处理[1)]，进一步归纳为 TypeName，FunctionName 和 VariableName 3 类，用于解决由自定义标识符的无限性引起的数据稀疏与维度爆炸问题；3)将代码中的符号也视为词条并进行编码，这是因为代码中的符号对于语义有着非常重要的影响，比如“＋”表示字符串拼接、“.”表示成员访问等。具体地，本文采用 One-Hot 编码方法将源代码映射到向量空间中[2)]。

3.2 模型结构

本文所采用的混合神经网络的整体结构如图 1 所示，其主要包括基于卷积神经网络(Convolutional Neural Network，CNN)的结构特征提取模块、基于循环神经网络(Recurrent Neural Network，RNN)的语义特征提取模块以及特征融合模块。

3.2.1 结构特征提取模块

结构特征提取模块使用 CNN 像捕捉图片中的边角信息一样捕捉并提取源代码中的结构特征。具体地，本模块以字符级表征方式处理所得的大小为 50×305 的字符矩阵，并将其作为输入，通过结合两组传统卷积层与最大池化层，实现结构特征的提取并缩小特征图。其中，卷积核数量分别设置为 32 和 64，卷积核尺寸均设置为 3×3，池化核尺寸均设置为 2×2。中间加入 Dropout 层，以 0.2 的概率随机将部分隐藏层节点权重归零，防止网络过拟合。每一层卷积后都使用 ReLU[20]作为激活函数，同时使用批量标准化减少网络对学习率的要求，加快网络收敛。

3.2.2 语义特征提取模块

传统 RNN 按顺序处理输入序列，且信息的传播过程往往是单向的，很可能会忽略掉下文信息，这与代码的阅读和理解方式略有不同。因此针对代码可读性评估任务，本模块采用双向门控循环神经网络(Bidirectional Gated Recurrent Unit，BiGRU)对源代码中的语义特征进行提取。BiGRU 的优势在于参数少、结构简单、收敛性强，并且其双向神经网络结构能够充分考虑到上下文信息和语序关系，使得当前时刻的输出与前后时刻的状态都产生联系，进而捕捉到更加重要的语义特征。

如图 3 所示，本模块以词条级表征方式处理所得的词向

1) Wang 等在一项有关缺陷预测的研究中使用了类似的方法[24]

2) 考虑到源代码与自然语言的差异性，本文无法采用在 NLP 领域中广泛使用的预训练词向量

量序列作为输入，其中词向量维度设置为 64，最大输入长度限制为 300 个词条，输出由两个 GRU 的状态共同决定，其计算过程为：

$$\vec{\boldsymbol{h}}_t = \boldsymbol{f}(\vec{\boldsymbol{W}}\boldsymbol{x}_t + \vec{\boldsymbol{V}}\boldsymbol{h}_{t-1} + \vec{\boldsymbol{b}})$$

$$\overleftarrow{\boldsymbol{h}}_t = \boldsymbol{f}(\overleftarrow{\boldsymbol{W}}\boldsymbol{x}_t + \overleftarrow{\boldsymbol{V}}\boldsymbol{h}_{t+1} + \overleftarrow{\boldsymbol{b}})$$

$$\boldsymbol{y}_t = \boldsymbol{g}(\boldsymbol{U}[\vec{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t] + \boldsymbol{c})$$

其中，$\boldsymbol{x}_t$ 为词向量序列，$\boldsymbol{h}_t$ 为待激活状态，$\boldsymbol{W}$，$\boldsymbol{V}$ 和 $\boldsymbol{U}$ 为权重矩阵，$\boldsymbol{b}$ 和 $\boldsymbol{c}$ 为偏置矩阵。

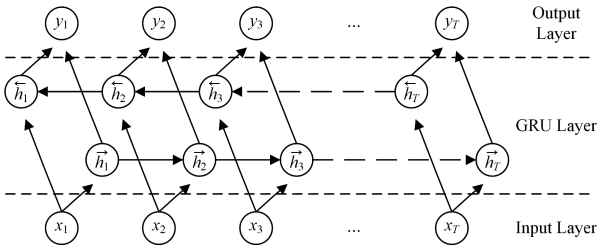

图 3　语义特征提取模块

Fig. 3　RNN-based feature extraction module

3.2.3　特征融合和参数设置

在特征融合模块中，首先将 CNN 所提取的 128 维结构特征以及 BiGRU 所提取的 128 维语义特征拼接为 256 维的输出向量，然后依次送入全连接层以及 sigmoid 层，最终实现对代码可读性的分类预测。

本文模型采用 Adam[21] 作为优化器，将学习率设置为 0.01。模型的训练过程以最小化网络损失为目的，采用二元交叉熵函数作为损失函数：

$$J = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} p_j^{(i)} \log q_j^{(i)}$$

其中，$N=16$ 为批尺寸，$M=2$ 为类别数，p_j 表示样本 i 是否属于类别 j，q_j 表示模型所预测的概率。

4　实验设置

4.1　数据集

本文实验所使用的数据集来自现有研究提供的已标定数据集，即 D_{Buse}[2]，D_{Dorn}[11] 和 $D_{Scalabrino}$[10]，具体信息如表 1 所列。

表 1　实验数据集的详细信息

Table 1　Statistical summary of dataset

Dataset	Source	Total # of Code Snippets	Programming Language	Total # of Annotators	Lines of Code	
					Mean	SD
D_{Buse}	Provided by [2]	100	Java	120	7.61	2.59
D_{Dorn}	Provided by [11]	360	Java, Python, CUDA	5468	29.74	16.36
$D_{Scalabrino}$	Provided by [10]	200	Java	9	26.56	10.29

在 D_{Buse}，D_{Dorn} 和 $D_{Scalabrino}$ 中，由多位标定者使用李克特量表[22]对给定代码片段进行 1 分（Very Unreadable）至 5 分（Very Readable）的难易度评价，并将得分均值作为其可读性标签，最终得到可读与不可读两组代码片段。值得注意的是，D_{Dorn} 中包含使用 Java，Python 和 CUDA 语言编写的代码片段，为了与其他数据集保持一致，本文仅保留 Java 样本。

考虑到 D_{Buse}，D_{Dorn} 和 $D_{Scalabrino}$ 的规模偏小，本文实验将合并使用这 3 个数据集，一方面可以增加数据量，以维持深度学习模型的训练过程；另一方面可以增加数据的多样性。具体地，本文实验所使用的数据集与文献[19]一致，共包含 210 个代码片段（其中可读组与不可读组各包含 105 个代码片段），并将训练集与测试集的比例分别设置为 70% 和 30%。

4.2　评价指标

本文实验是一个二分类问题，为评估模型的性能，采用准确率（accuracy）作为评价指标。准确率是在代码可读性评估领域中广泛使用的一个性能指标[2,9,19]，是指被正确预测的样本数在总样本中所占的比例，该度量值越高，代表模型的性能越好。

4.3　比较模型

为验证本文提出的方法在代码可读性评估领域中的有效性，将本文模型与现有的可读性模型进行比较，分别为基于机器学习的 M_{Buse}[2]，M_{Dorn}[11]，$M_{Scalabrino}$[10] 和 M_{All}[10]，以及基于深度学习的 DeepCRM[19]。

（1）M_{Buse}：基于 25 个较为简单的代码特征，Buse 等提出的第一个正式的代码可读性分类模型。

（2）M_{Dorn}：基于 Buse 等的研究，Dorn 从视觉、空间、语言 3 个维度进一步扩充了特征集。

（3）$M_{Scalabrino}$：考虑到标识符和注释中可能涉及到的语义信息，Scalabrino 等额外引入了一组文本特征。

（4）M_{All}：由以上提到的所有代码特征构成的一个更加全面、通用的可读性分类模型。

（5）DeepCRM：不同于传统的机器学习方法，Mi 等运用卷积神经网络构建的第一个基于深度学习的代码可读性分类模型。

为了实现对比实验的公平，本文使用在代码可读性评估任务中表现最好的 Logistic Regression 作为 M_{Buse}，M_{Dorn}，$M_{Scalabrino}$ 和 M_{All} 的底层分类器。

4.4　实验流程和实验环境

本文模型由 Keras[1)] 实现，在给定数据集上进行训练，并采用准确率作为评价指标，通过与现有模型进行比较，来验证本文提出的方法在代码可读性评估领域中的有效性。实验所用的处理器为 Intel Core i7（4 核），主频为 2.3GHz，内存为 32GB。

5　实验结果与分析

本文实验包括两个部分，第一部分（内部实验）的目的是验证本文提出的混合神经网络的有效性；第二部分（外部实验）的目的是验证本文提出的方法在代码可读性评估领域中的有效性。

1) https://keras.io/

5.1 内部实验

为验证本文提出的混合神经网络的有效性，将其与单独的 CNN 以及单独的 BiGRU 进行对比，实验结果如图 4 所示。

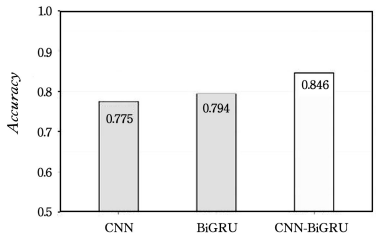

图 4 混合神经网络与单独模型的比较

Fig. 4 Comparison of hybrid neural networks and separate models

由图 4 可以看出，混合神经网络的表现优于单独的 CNN 以及单独的 BiGRU，在准确率上分别有 9.2%和 6.5%的提升。为了进一步验证混合神经网络与单独的 CNN 以及单独的 BiGRU 在分类准确率上是否存在显著差异，使用 Brunner-Munzel Test(显著性水平 $\alpha=0.05$)[23]进行检验，所得 p 值分别为 0.00(小于 0.05)和 0.00(小于 0.05)，证实了本文提出的混合神经网络的有效性。换言之，本文模型能够综合利用不同表征方式的优点，有效建立代码特征与可读性之间的关系。

事实上，根据 Buse 等的研究[2,4]，最有效的 5 个区分代码是否可读的特征分别是标识符的数量、行长度、“{”和“(”的数量、“.”的数量以及缩进的数量。行长度作为一种空间结构信息可以体现在字符级表征方式中，由基于 CNN 的结构特征提取模块进行提取；而对于标识符的数量、“{”和“(”的数量、“.”的数量以及缩进的数量这些统计学特征则可以体现在词条级表征方式中，由基于 RNN 的语义特征提取模块进行提取。这也从领域知识的角度支持了本文提出的多维度特征的有效性和必要性。

5.2 外部实验

为验证本文提出的方法在代码可读性评估领域中的有效性，将本文模型与 M_{Buse}，M_{Dorn}，$M_{Scalabrino}$，M_{All} 和 DeepCRM 进行比较，实验结果如图 5 所示。

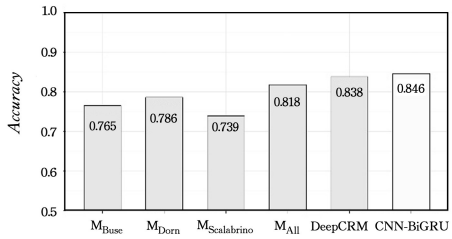

图 5 可读性模型的分类准确率

Fig. 5 Accuracy achieved by code readability models

由图 5 可以看出，M_{Buse}，M_{Dorn} 和 $M_{Scalabrino}$ 的表现一般，很可能是因为 M_{Buse}，M_{Dorn} 和 $M_{Scalabrino}$ 的设计是针对于特定的数据集，所以其普适性与扩展性不佳。例如，根据 Buse 等的研究[2]，M_{Buse} 在 D_{Buse} 上可以取得 80%左右的分类准确率，而在本文实验所用的数据集上(包括 D_{Buse}，D_{Dorn} 和 $D_{Scalabrino}$)仅取得了 76.5%的准确率，这也进一步说明了基于机器学习框架的研究方法的局限性。相对而言，加入所有特征的 M_{All} 以及基于深度学习的 DeepCRM 的预测效果较好，分类准确率可以达到 81.8%和 83.8%。而本文提出的方法在代码可读性评估任务中表现最优，其分类准确率高于现有的 5 个可读性模型。具体地，相较于传统的机器学习分类模型 M_{Buse}，M_{Dorn}，$M_{Scalabrino}$ 和 M_{All}，本文模型在分类准确率上得到了明显的提升；而与 DeepCRM 相比，本文模型仅表现出了小幅度的提升，可能是因为 DeepCRM 针对神经网络层数、神经元个数等超参数进行了细致的调优，而本文模型并未包含手动调参过程。综上所述，多维度特征和混合神经网络的引入对于代码可读性评估任务有着重要的意义。

结束语 代码可读性的概念早已出现在学术界，直到近几年，才正式引起软件工程领域研究者的关注。但是，现有的基于特征工程以及机器学习框架的研究方案，无论是在评估准确率还是泛化能力上都差强人意。为此，本文提出了一种基于多维度特征和混合神经网络的代码可读性评估方法，通过整合并运用各种单一神经网络的优势，从结构、语义等不同角度挖掘源代码中蕴含的可读性信息，在代码可读性的定量评估技术上实现了突破。为了进一步提高模型的分类准确率，未来的研究工作可以考虑从如下两个方面展开：

(1)借鉴 NLP 领域的最新成果和技术，并结合代码的结构特性以及领域知识，进一步优化源代码表征方式。

(2)通过优化超参数、改进损失函数和不同特征向量之间的融合方式，进一步提升深度学习模型的表现。

参 考 文 献

[1] HOOIMEIJER P, WEIMER W. Modeling bug report quality [C]// Proc. Twenty-Second IEEE/ACM Int. Conf. Autom. Softw. Eng. (ASE '07). 2007: 34.

[2] BUSE R P L, WEIMER W R. Learning a Metric for Code Readability[J]. IEEE Trans. Softw. Eng., 2010, 3(4): 546-558.

[3] SIVAPRAKASAM P. Improving Software Quality Through the Development of Code Readability[J]. International Journal of Advanced Research in Computer and Communication Engineering, 2012, 1(6): 472-477.

[4] BUSE R P L, WEIMER W R. A metric for software readability [C]// Proceedings of the 2008 International Symposium on Software Testing and Analysis (ISSTA '08). 2008: 121.

[5] BOSWELL D, FOUCHER T. The Art of Readable Code: Simple and Practical Techniques for Writing Better Code[C]// O'Reilly Media, Inc.. 2011.

[6] FAKHOURY S, ROY D, HASSAN A, et al. Improving source code readability: theory and practice [C] // 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC). 2019: 2-12.

[7] SANTOS R M D, GEROSA M A. Impacts of coding practices on readability[C]// Proc. Int. Conf. Softw. Eng.. 2018: 277-285.

[8] TASHTOUSH Y, ODAT Z, ALSMADI I, et al. Impact of Programming Features on Code Readability[J]. Int. J. Softw. Eng. Its Appl., 2013, 7(6): 441-458.

[9] POSNETT D, HINDLE A, DEVANBU P. A simpler model of software readability[C]// Proceeding of the 8th Working Conference on Mining Software Repositories (MSR '11). 2011: 73.

[10] SCALABRINO S,LINARES-VASQUEZ M,POSHYVANYK D,et al. Improving code readability models with textual features [C]// 2016 IEEE 24th International Conference on Program Comprehension (ICPC). 2016:1-10.

[11] DORN J. A General Software Readability Model[D]. Virginia: Univ. Virginia,Charlottesville,2012.

[12] CROOKES D. Generating readable software[J]. Softw. Eng. J. , 1987,2(3):64-70.

[13] BAECKER R. Enhancing program readability and comprehensibility with tools for program visualization[OL]. https://dl.acm.org/doi/10.5555/55823.55858.

[14] BINKLEY D,DAVIS M,LAWRIE D,et al. To camelcase or under_score [C]// 2009 IEEE 17th International Conference on Program Comprehension. 2009:158-167.

[15] SHARIF B,MALETIC J I. An Eye Tracking Study on Camelcase and Under_score Identifier Styles[C]// 2010 IEEE 18th International Conference on Program Comprehension. 2010:196-205.

[16] BUSE R P L,ZIMMERMANN T. Information needs for software development analytics[C]// 2012 34th International Conference on Software Engineering (ICSE). 2012:987-996.

[17] AGGARWAL K K,SINGH Y,CHHABRA J K. An integrated measure of software maintainability[C]// Annual Reliability and Maintainability Symposium. 2002:235-241.

[18] BÖRSTLER J,CASPERSEN M E,NORDSTRÖM M. Beauty and the beast:on the readability of object-oriented example programs[J]. Softw. Qual. J. ,2016,24(2):231-246.

[19] MI Q,KEUNG J,XIAO Y,et al. Improving code readability classification using convolutional neural networks [J]. Inf. Softw. Technol. ,2018,104:60-71.

[20] MAAS A L,HANNUN A Y,NG A Y. Rectifier Nonlinearities Improve Neural Network Acoustic Models[C]// Proc. 30th Int. Conf. Mach. Learn. . 2013.

[21] KINGMA D P,BA J. Adam:A Method for Stochastic Optimization[J]. arXiv:1412.6980v5.

[22] LIKERT R. A technique for the measurement of attitudes[OL]. https://psycnet.apa.org/record/1933-01885-001.

[23] NEUBERT K,BRUNNER E. A studentized permutation test for the non-parametric Behrens-Fisher problem[J]. Comput. Stat. Data Anal. ,2007,51(10):5192-5204.

[24] WANG S,LIU T,TAN L. Automatically Learning Semantic Features for Defect Prediction[C]// 2016 IEEE/ACM 38th International Confernce on Software Engineering. 2016:297-308.

MI Qing, born in 1987, Ph.D, lecturer, is a member of China Computer Federation. Her main research interests include code readability assessment, deep learning and empirical experiments.

CHEN Jun-cheng, born in 1980, Ph.D, lecturer, is a member of China Computer Federation. His main research interests include software testing, compiler optimization, machine learning and deep learning.