基于 Word2vec 的克隆代码检测方法研究

贾 清,杨 抒
(新疆农业大学 计算机与信息工程学院,新疆 乌鲁木齐 830052)

摘 要:系统中的克隆代码会增加程序员理解代码、修改代码的时间,并且在代码中一处克隆代码出现错误可能会导致系统中多个相同代位置的代码出现错误,大大增大了程序员进行软件维护的成本。为了找到系统文件中的克隆代码,利用基于 Word2vec 的克隆代码检测方法,针对新疆马业电商平台中的代码进行克隆检测。通过对系统源代码进行数据清洗,去除不需要的字符;Word2vec 模型是一群浅并且双层的神经网络,选择 Word2vec 中的 skip-gram 模型进行训练并且构造词向量。训练完成后,模型可用来映射每个词到一个向量,用来表示词对词之间的关系。最后通过夹角余弦的方法来计算代码相似度,从而自动检测代码中的克隆代码。研究结果表明:基于 Word2vec 的克隆代码检测方法可以很好地检测出代码文件中的克隆代码,并且以指定的方式进行输出。

关键词:Word2vec;克隆代码;自动检测;相似度;软件维护
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2020)08-0124-05
doi:10.3969/j.issn.1673-629X.2020.08.021

Research on Clone Code Detection Method Based on Word2vec

JIA Qing, YANG Shu
(School of Computer and Information Engineering, Xinjiang Agricultural University, Urumqi 830052, China)

Abstract: The clone code in the system will increase the time for the programmer to understand the code and modify it, and the mistake of a clone code in the code may lead to the mistake of the code in the same generation position in the system, which greatly increases the cost of the programmer's software maintenance. In order to find the clone code in the system file, we use the clone code detection method based on Word2vec to clone the code in the Xinjiang Horse Industry e-commerce platform. The unnecessary characters are removed by data cleaning of the system source code. Word2vec model is a group of shallow and double-layer neural networks. Skip-gram model in Word2vec is selected to train and construct word vectors. After training, the model can be used to map each word to a vector to express the relationship between words. At last, the code similarity is calculated by the method of Angle cosine, so that the clone code in the code can be detected automatically. The results show that Word2vec-based clone code detection method can detect the clone code in the code file effectively and output it in the specified way.

Key words: Word2vec; clone code; automatic detection; similarity; software maintenance

0 引 言

近年来随着软件开发环境、操作系统及软件相关领域的不断发展,导致软件复用的规模越来越大,为了提高软件的开发效率,越来越多的软件开发成果被开发人员直接复用,这种被开发人员复用并且具有相似语法及语义特征的代码段称为克隆代码[1]。

目前学界对克隆代码的定义有很多。比如文献[1]中将克隆代码定义为:代码文件中多个相同或相似的代码片段[1]。文献[2]将克隆代码定义为:具有相似语法及语义特征的代码段[2]。文中统一将克隆代码定义为:具有相似语法及语义特征的代码段。

由于代码克隆被认为会降低软件的可维护性[1],有学者通过对两个开源软件系统的案例研究,发现 1% 到 3% 的克隆不一致的更改会导致软件缺陷,而在其他研究报告中的百分比要更高[2]。针对该问题,提出了几种代码克隆检测技术和工具,例如文献[3-11]。近年来,深度学习成为热门话题,有学者提出利用深度学习的相关技术对相似记录进行检测,例如文献[12-15]。文中利用 Word2vec 模型对新疆马业电子商务交易平台中的克隆代码进行检测。

收稿日期:2019-09-02 **修回日期**:2020-01-07
基金项目:新疆维吾尔自治区重大科技专项(2017A01002-5);新疆农业大学博士后科研流动站资助
作者简介:贾 清(1993-),女,硕士研究生,研究方向为农业信息化;通讯作者:杨 抒(1979-),男,博士,副教授,研究方向为软件工程、数据挖掘。

新疆维吾尔自治区重大科技专项“马产业升级技术创新工程”子课题“马产业科技创新平台建设”中的“新疆马业电子商务交易平台”,实现了包括在线马匹拍卖为主要功能的信息发布管理、信息展示管理、订单管理、资金结算管理的在线电商服务。在开发该系统的过程中,为了提高开发效率,开发人员针对已经实现的并具有相似功能的代码进行粘贴复制的操作,在开发者的思维中,这样的操作代码效率高并且错误率最低,可以节省开发系统的时间与成本,所以开发人员会复用大量代码,导致生成了大量的克隆代码,并且随着系统的不断更新,需要对系统进行增加功能或者修改功能,造成系统的体积越来越大,并且产生越来越多的克隆代码。为了解决该问题,利用基于 Word2vec 的克隆代码检测方法对新疆马业电子商务交易平台的代码进行克隆检测。

1　Word2vec 模型

Word2vec 本质为双层的神经网络,其作用是将词转换成向量,这些模型可以将词汇以固定维数的向量表示出来。较为常用的 one-hot 编码方式,可能每个词都是百万维的向量,占用大量内存,并且在判断意思相似的词语,相似的句子的时候效果不理想。而 Word2vec 会根据上下文,对上下文进行训练。每个词不再是稀疏向量(只有某一位为 1,其余位均为 0),而是一个稠密的拥有固定维数的向量,可以大大减少存储空间和计算时间。其次,Word2vec 经过训练后的词向量可以使用上下文信息来判断并且找到具有类似含义的词语。Word2vec 一共包含两种模型,分别是 CBOW 模型(continue bag of word),即用上下文预测中心词,以及 skip-gram 模型(跳字模型),即用中心词预测上下文。Word2vec 的本质是无监督学习,从词向量中可以看出该神经网络只有一层,因此必须要有输入和输出。而训练过程或者目标不是得到预测结果词,而是得到隐藏层的权重。

1.1　skip-gram 模型

数学描述为:skim-gram 模型需要最大给定任意中心词生成所有背景词的概率:

$$\prod_{t=1}^{T}\prod_{-m\leqslant j\leqslant m,j\neq 0}P(w^{(t+j)}\mid w^{(t)}) \tag{1}$$

上式的最大似然估计与以下最小化损失函数等价:

$$-\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leqslant j\leqslant m,j\neq 0}\log P(w^{(t+j)}\mid w^{(t)}) \tag{2}$$

$\boldsymbol{U}$ 和 $\boldsymbol{V}$ 分别用来表示背景词及中心词的词向量。词典中下标为 i 的词,中心词和背景词的词向量可以表示为 v_i 和 u_i,skip-grams 所要学习的模型参数正是词典中这两种词向量。损失函数为了放入该模型参数,需要使用模型参数表达损失函数中的给定中心词生成背景词的条件概率。假设中心词为 w_c,背景词为 w_o,所以输入中心词向量生成背景词向量的条件概率可以定义为:

$$P(w_o\mid w_c)=\frac{\exp(u_o^{\mathrm{T}}v_c)}{\sum_{i\in v}\exp(u_i^{\mathrm{T}}v_c)} \tag{3}$$

当序列长度 T 较大时,为了计算该子序列的损失,常用做法是在每次迭代时就随机采样一个相对短的子序列,会根据该损失来计算词向量的梯度及迭代词向量。随机采样的子序列的损失其本质是对子序列中确定中心词生成的背景词的条件概率的对数再求其平均数,然后再通过微分运算,可以获得上面公式中条件概率的对数关于中心词向量 v_c 的梯度。

$$\frac{\partial\log P(w_o\mid w_c)}{\partial v_c}=u_o-\sum_{j\in v}\frac{\exp(u_j^{\mathrm{T}}v_c)}{\sum_{i\in v}\exp(u_i^{\mathrm{T}}v_c)}u_j \tag{4}$$

经过转换后该公式可简化为:

$$\frac{\partial\log P(w_o\mid w_c)}{\partial v_c}=u_o-\sum_{j\in v}P(w_j\mid u_c)u_j \tag{5}$$

在训练模型的过程中,迭代运算其实是用梯度来迭代子序列中出现过的中心词和背景词的向量;在结束训练后,对于词典中任意索引为 i 的词,均得到该词作为中心词和背景词的两组词向量 v_i 和 u_i,其中 skip-gram 模型网络结构如图 1 所示。

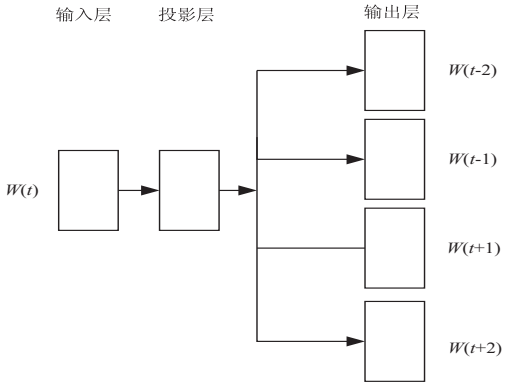

图 1　skip-gram 模型网络结构

1.2　CBOW 模型

CBOW 模型预测该中心词,需要用一个中心词在文本序列前后的背景词来预测。其数学描述为:CBOW 模型需要最大化给定任意背景词生成任意中心词的概率:

$$\prod_{t=1}^{T}P(w^{(t)}\mid w^{(t-m)},\cdots,w^{(t-1)},w^{(t+1)},\cdots,w^{(t+m)}) \tag{6}$$

上式的最大似然估计与最小化损失函数等价:

$$-\sum_{t=1}^{T}\log P(w^{(t)}\mid w^{(t-m)},\cdots,w^{(t-1)},w^{(t+1)},\cdots,w^{(t+m)}) \tag{7}$$

用 $\boldsymbol{V}$ 和 $\boldsymbol{U}$ 来表示中心词和背景词的向量。对于词典中某个位置为 i 的词,它在作为中心词和背景词时的向量可以分别表示为 v_i 和 u_i。而词典中的这两种向量正是 CBOW 所要学习的模型参数。损失函数需要将模型参数植入,需要使用模型参数表达损失函数中的给定背景词生成中心词的概率。设中心词 w_c 在词典中索引为 c,背景词 $w_{o1},\cdots,w_{o2m}$,损失函数中的背景词生成中心词的概率可以使用 softmax 函数定义为:

$$P(w_c \mid w_{o1},\cdots,w_{o2m}) = \frac{\exp\left(\frac{u_c^{\mathrm{T}}(v_{o1}+\cdots+v_{o2m})}{2m}\right)}{\sum_{i\in v}\exp\left(\frac{u_c^{\mathrm{T}}(v_{o1}+\cdots+v_{o2m})}{2m}\right)} \tag{8}$$

与 skip-gram 模型一样,当序列 T 较大时,通常在每次迭代时随机采样一个较短的子序列来计算有关该子序列的损失。根据损失计算词向量的梯度并迭代词向量。通过微分,可以计算出上式中条件概率的对数有关任一背景词向量 $V_{oi}(i=1,2,\cdots,2m)$ 的梯度为:

$$\frac{\partial \log P(w_c \mid w_{o1},\cdots,w_{o2m})}{\partial v_{oi}} = \frac{1}{2m}\left[u_c - \sum_{j\in v}\frac{\exp(u_j^{\mathrm{T}} v_c)}{\sum_{i\in v}\exp(u_i^{\mathrm{T}} v_c)}u_j\right] \tag{9}$$

该式子也可写为:

$$\frac{\partial \log P(w_c \mid w_{o1},\cdots,w_{o2m})}{\partial v_{oi}} = \frac{1}{2m}\left[u_c - \sum_{i\in v}P(w_j \mid w_c)\,u_j\right] \tag{10}$$

对于词典中索引为 i 的词,会在训练结束后得到该词作为背景词和中心词的两组词向量 v_i 和 u_i,其 CBOW 模型的网络结构如图 2 所示。

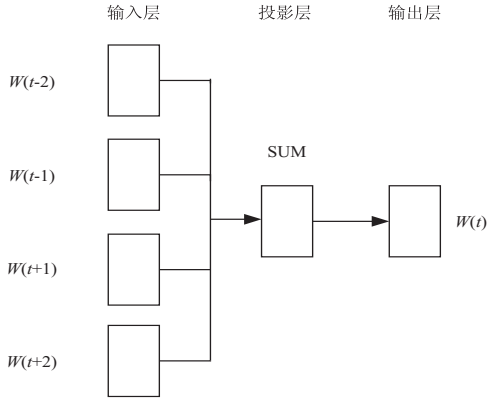

图 2 CBOW 模型网络结构

2 克隆代码检测

新疆马业电子商务交易平台是一款基于 Django 框架的马匹在线竞拍系统,服务于马匹竞拍的日常管理,为了满足新疆马业的发展,科学管理,提高效率而开发的一款系统。而开发过程中为了提高开发效率,降低开发的时间成本,在实现相同或者相似的功能时,开发人员大都使用了粘贴复制的方式进行开发,所以在该系统中存在克隆代码。本研究中的数据即为新疆马产业科技创新平台中的所有代码,该平台共有五个子平台,分别为马业生产过程数据采集平台、马业电子商务交易平台、赛马赛事组织管理信息平台、马产业大数据决策支持平台、马业科技信息服务管理平台,实验训练集选取该平台中的 16 914 行代码,验证集 1 000 行以及测试集 1 251 行。

本研究的目的在于能够利用 Word2vec 检测出新疆马业电商平台中存在的克隆代码。找到系统中具有高相似度的克隆函数,为后期代码重构、系统维护提供数据支持,其流程为:

①获取源代码:马产业科技创新平台建设中的部分代码。

②数据预处理:将数据分为训练集、验证集、测试集,并且删除注释、停用词、空格、标点符号。

②获得语料库:经过数据预处理后的数据。

④训练模型:选择 skim-gram 模型训练。

⑤获取词向量:训练结束后会获得 vocab 字典,包含了所有语料库中的词(去除了出现频数较少的词汇)。

⑥计算相似度:利用夹角余弦的方式分别计算两个列表的距离。

⑦结果输出:根据相似度计算的结果,根据序号找到对应代码,将结果输出至 txt 文件中。

基于 Word2vec 克隆代码检测流程如图 3 所示。

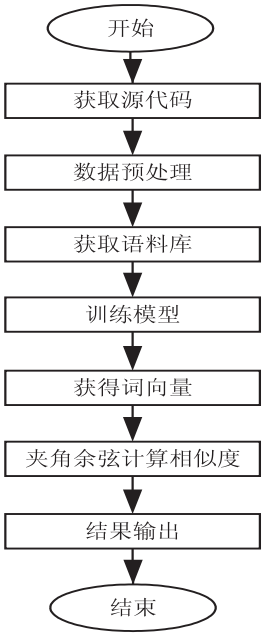

图 3 克隆代码检测流程

2.1　数据预处理

将新疆马产业创新平台中的所有 view 作为训练集的代码提取到一个名为 test. csv 的文件中。该文件中的代码不能直接用于 Word2vec 的训练,需要对其进行一些预处理。

以字符串的格式读取 train. csv 文件,按照换行符分割字符串,此分割方式可以将每个方法单独分割成一个字符串;循环遍历方法字符串,删除字符串中的换行符(/n)、回车符(/r);利用 jieba 分词器对代码进行分词。结巴分词器会根据已经写好的停用词文件将代码中的停用词进行删除操作。

将数据加载到 Python Pandas DataFrame 中,pandas 库将 DataFrame 定义为具有行和列的二维数据,大小可变的数据结构,每行代表一个数据样本。

将经过处理后的 test 数据写入一个名为 test_after_process_text_dir. csv 的文件中。

经过数据预处理的代码,保留了关键词、逻辑符等。

2.2　模型训练

在本次实验中用到的模型为 skip-grams,即用中心词来预测文本序列背景词的概率。

在通过 Word2vec 训练模型时,会将训练集中的所有词存入 vocab 中,如果有新词出现,则存入,如果已经有该词汇,不再重复操作。同时,每次存储一个新词汇,都要检测 vocab 的 size 是否达到上限。若达到上限,会将出现次数不达标的词汇对应的 vocab 空间释放。在新疆马产业科技创新平台中,选取部分函数,共计有 118 572 个词汇用来训练模型,训练完成后会自动生成一个扩展名为. model 的模型。

2.3　相似度计算

计算相似度的方法有欧氏距离(Euclidean distance)、曼哈顿距离(Manhattan distance)、标准化欧氏距离(standardized Euclidean distance)、夹角余弦(cosine)。在此次实验中用到计算相似度的方法为夹角余弦,用该方法来衡量样本向量之间的差异,其公式为:

$$\cos\theta = \frac{\sum_{k=1}^{n} x_{1k}\, x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2}\sqrt{\sum_{k=1}^{n} x_{2k}^2}} \tag{11}$$

在本次实验中,将马匹竞拍系统中的代码作为测试集,共有 92 个函数。将 test 中的数据与 train 中的代码进行相似度计算,但是在计算相似度之前,程序会将每个函数中的词汇与训练模型 vocab 中的词汇进行对比,删除不在 vocab 中的词汇,否则在计算相似度时,会因为某些没有转换成词向量的词汇导致程序错误。然后将每一个函数分别与马匹竞拍系统中的所有代码一一进行相似度计算。计算完成后可以将相似度的值输出至 pycharm 的 run 窗口中,如图 4 所示。

```
test_line_list: 64
sim_score [('64', 0.99999994), ('65', 0.99974304), ('79', 0.9958919), ('26', 0.9957624),
 ('57', 0.9957209), ('63', 0.9954981), ('62', 0.9952945), ('198', 0.9912509), ('137',
 0.98446447), ('114', 0.98257124), ('24', 0.9809129), ('197', 0.9807432), ('275',
 0.9804123), ('101', 0.97521704), ('25', 0.9608364), ('128', 0.94309443), ('140',
 0.94204813), ('199', 0.9136751), ('271', 0.9136751), ('1131', 0.9136751), ('1176',
 0.9136751), ('1211', 0.9136751), ('797', 0.9075165), ('418', 0.75425255), ('591',
 0.75425255), ('627', 0.75425255), ('671', 0.75425255), ('204', 0.70065516), ('211',
 0.6992111), ('200', 0.6874208), ('68', 0.68462163), ('1062', 0.67190313), ('70',
 0.62483156), ('71', 0.5858681), ('59', 0.5809409), ('58', 0.5423875), ('105', 0.52786916),
 ('134', 0.5217474), ('250', 0.5208318), ('327', 0.5208318), ('812', 0.5208318), ('831',
 0.5208318), ('847', 0.5208318), ('124', 0.5198862), ('89', 0.5112056), ('170', 0.5112056),
 ('180', 0.5112056), ('1111', 0.5112056), ('1119', 0.5112056), ('844', 0.5051188)]
```

图 4　相似度计算结果

2.4　结果输出

根据 sim_result 的计算结果,将 sim_result. csv 中序号对应的代码一一找出:

```
64,end _ dt datetime datetime strptime str _ enddt % Y % m % d % H % M % S
***最相似的前20个***
64,end _ dt datetime datetime strptime str _ enddt % Y % m % d % H % M % S
65,now _ dt datetime datetime strptime str _ nowdt % Y % m % d % H % M % S
79,bidding _ time datetime datetime now strftime % Y % m % d % H % M % S
26,current _ dt datetime datetime now strftime % Y % m % d % H % M % S
57,current _ datetime datetime datetime now strftime % Y % m % d % H % M % S
63,str _ nowdt datetime datetime now strftime % Y % m % d % H % M % S # 系统 当前 时间
62,str _ enddt obj end _ dateTime astimezone local _ tz strftime % Y % m % d % H % M % S
198,end _ time auction end _ dateTime astimezone local _ tz strftime % Y % m % d % H % M % S
137,% Y % m % d % H % M % S
114,str _ time item bidding _ time astimezone local _ tz strftime % Y % m % d % H % M % S
24,end _ dt auction _ info auction end _ dateTime astimezone local _ tz strftime % Y % m % d % H % M % S
```

图 5　部分检测结果输出

①构建文档字典:遍历文件,此处的文件分别为 test_after_process_text_dir. csv 及 train_after_process_text_dir. csv。将每行结果去除/r 及/n,以“,”作为分割符,判断列表长度是否为 2,若不做此操作,可能会报数字越界的错误。以列表的序号为 key 值,以内容为 value 值。

②相似结果输出:打开 sim_result. csv 文件,以“,”作为分割符,在对应 id 与 test_after_process_text_dir. csv 及 train_after_process_text_dir. csv 分别进行匹配,找出相应的内容,将结果输出至 result. txt 文件中,部分结果如图 5 所示。

3 结束语

基于 Word2vec 的克隆代码研究将新疆马业电商平台中的代码作为语料库,利用 Word2vec 训练词向量模型,使用的模型为 skip-grams,开发工具为 pycharm,开发语言为 python,分词器选用 jieba 分词器。

将新疆马产业创新平台中的 view 中部分代码作为训练集,包含 400 个方法,验证集包含 59 个方法,测试集包含 92 个方法,共计 551 个方法,用到 118 572 个词汇。

在基于 Word2vec 的克隆代码研究中,①将新疆马产业创新平台中的 view 中部分代码进行预处理,去除空白符、注释、标识符、回车符号;②将处理后的数据作为 Word2vec 的语料库,利用 Word2vec 训练向量模型,其中模型选择 skip-grams,sentences 是要训练模型时的语料库,该研究中,sentences 是遍历文件所得。size 是构造的词向量的维数,默认值是 100,该维数的大小取决于 sentences 的大小,在该实验中 size 为 150。Window 是词向量的上下文最大距离,若 Window 越大,和某一次较远的词也会产生上下文关系,默认值为 5。sg 是 Word2vec 关于选择两个模型,如果 sg 的值为 0,选择的是 CBOW 模型,若取值为 1,则选择的是 skip-grams 模型,若未写 sg,则该值默认选择的是 CBOW 模型。min_count 是计算词向量出现的最小频数。这个值可以自动过滤很多出现次数小于该值的词汇,默认是 5。如果语料库较小,其值可以更改得更低一些。Iter 为随机梯度下降法中迭代的最大次数,默认为 5,该值的大小取决于语料库的大小。alpha 是随机梯度下降法中迭代的初始步长,默认是 0. 025。min_alpha 给出了最小迭代步长值,这是由于算法支持在迭代的过程中逐渐减小步长;③将验证集的代码进行预处理,去除空白符、注释、标识符、回车符号;④将处理后的验证集进行相似度计算,相似度计算之前需要先去除不在词汇表中的词汇。

基于 Word2vec 的克隆代码研究根据词向量可以很好地计算出代码的相似度,从而找到文件中的克隆代码,为后续代码优化提供数据支持。

参考文献:

[1] KAMIYA T,KUSUMOTO S,INOUE K. CCFinder:a multilinguis tic token-based code clone detection system for large scale source code[J]. IEEE Transactions on Software Engineering,2002,28(7):654-670.

[2] BETTENBURG N,SHANG W,IBRAHIM W,et al. An empirical study on inconsistent changes to code clones at release level[C]//2009 16th working conference on reverse engineering. Lille,France:IEEE,2009:85-94.

[3] ROY C K,CORDY J R,KOSCHKE R. Comparison and evaluation of code clone detection techniques and tools:a qualitative approach[J]. Science of Computer Programming,2009,74(7):470-495.

[4] ELMATARAWY A, ELRAMLY M, BAHGAT R. Code clone detection using sequential pattern mining[J]. International Journal of Computer Applications,2015,127(2):10-18.

[5] JIANG L,MISHERGHI G,SU Z,et al. DECKARD:scalable and accurate tree-based detection of code clones[C]//ICSE. [s. l.]:[s. n.],2007:96-105.

[6] ROY C K,CORDY J R. Towards a mutation-based automatic framework for evaluating code clone detection tools[C]//Canadian conference on computer science & software engineering. Montreal,Quebec,Canada:ACM,2008.

[7] KAWAGUCHI S,YAMASHINA T,UWANO H,et al. SHINOBI:a tool for automatic code clone detection in the IDE[C]//16th working conference on reverse engineering. Lille,France:[s. n.],2009.

[8] 于冬琦. 基于抽象语法树和静态分析的克隆代码自动重构[D]. 上海:复旦大学,2009.

[9] 甘水滔,秦晓军,陈左宁,等. 一种基于特征矩阵的软件脆弱性代码克隆检测方法[J]. 软件学报,2015,26(2):348-363.

[10] 折蓉蓉,张丽萍,侯 敏,等. 基于决策树推荐克隆重构的方法[J]. 计算机应用,2018,38(7):2037-2043.

[11] 冯江辉. 基于 K-最近邻的 C 克隆代码重构方法研究[D]. 哈尔滨:哈尔滨工业大学,2011.

[12] 蒋勇青,于 洋. 文献相似性检测技术及其应用[J]. 情报工程,2018,4(3):96-104.

[13] 吴庆辉,蔡海洋,吕精巧. 基于改进型遗传神经网络的相似重复记录检测[J]. 计算机测量与控制,2011,19(5):1021-1023.

[14] 张峰逸,彭 鑫,陈 驰,等. 基于深度学习的代码分析研究综述[J]. 计算机应用与软件,2018,35(6):9-17.

[15] 孟祥逢,鲁汉榕,郭 玲. 基于遗传神经网络的相似重复记录检测方法[J]. 计算机工程与设计,2010,31(7):1550-1553.