基于词频直方图和感知神经网络的 C 代码克隆检测

陈奕宇[1]

[1](南京大学 计算机科学与技术系, DZ1933004)

摘 要: 本次大作业要求自主实现对 C++ 代码数据集的功能查重。所谓功能查重是指，若两份代码是重复的，等价于这两份代码实现的功能相同。该大作业采用 Kaggle 比赛的形式，每位学生单独组队提交结果，以 F1 分数评分。本文基于词频直方图和感知神经网络建立了简洁高效的算法，辅以长尾截断、随机采样、样本分布均衡等技巧，实现算法在比赛中得到 A 榜第 7，B 榜第 6 的名次（共 193 队），最高可达到 0.67 的 F1 分数。

关键词: C++代码查重;Kaggle;词频直方图;感知神经网络

中图法分类号: TP301 **文献标识码**: A

本次大作业要求自主实现对 C++ 代码数据集的功能查重。所谓功能查重是指，若两份代码是重复的，等价于这两份代码实现的功能相同。该大作业采用 Kaggle 比赛的形式，时限为 6 月初到 6 月 25 日，每位学生单独组队提交结果，提交的结果为 csv 格式，以 F1 分数评分。

1 数据简介

1.1 数据集结构

该 C++代码数据集共分为三部分：训练集、测试集、测试列表。

1.1.1 训练集

训练集文件位于./train/train 目录下，共有 83 个类，每个类对应目录中一个文件夹。每个类中所有样本均实现同一任务，不同类代表不同任务，因此两样本若来自同一类则判为克隆，若来自不同类则不判为克隆。每个类中均有 502 个样本，每个样本为一份实现该类任务的 C++代码。

1.1.2 测试集

测试集文件位于./test/test 目录下，共有 10000 个样本，每个样本为一份实现未知任务的 C++代码。大作业中实现的算法要求，测试集样本中任选两个样本，能够判别这两份样本是否克隆。

1.1.3 测试列表

测试列表文件为 sample_submission.csv。该文件为列表格式，其首行为列名，其余行给出了 200000 个测试样本对。其中每一行第一列内容形如“name1_name2”，给出了该行测试样本对中两样本的文件名称；第二列标识样本对是否判为克隆，内容为{0,1}中的一个，为 0 表示不是克隆，为 1 表示判为克隆。本作业需按照该列表给定的测试样本对测试，并在 Kaggle 平台上提交形如该列表的结果。

1.2 数据结构

每个样本为一份 C++代码。在调试过程中，我发现每份代码与完整的 C++代码比较，均经过不完善的预处理。其特性如下：

1）大部分全局变量与局部变量被替换为以 VAR_ 开头的独特名称，但有部分变量未被替换名称，如 month year 等。所有函数名、结构名、常量未被替换名称；

2）所有原本以#开头的行被删除。这主要产生了一部分未知的宏定义变量、语句，并为代码运行分析带来障碍。.

2 算法设计

本次实验代码平台为 Python 3.7，深度学习工具包为 PyTorch。

2.1 任务设定

在任务设定上，我们有两条思路可以考虑。第一种设定，是看作一个 83 类的分类问题，在判断两份代码是否克隆时，现将代码各自判为某一类，再比较两份代码所属类是否相同。第二种设定，是看作一个 2 类的分类问题，将两份代码同时作为输入，输出它们是否克隆。

其中相比较，第一种设定的可解释性较强，学习难度低，但唯一的缺点是无法判别未知类的样本。因题目未给出测试集的任何类别信息，保险起见我采用第二种设定。

2.2 算法流程

按照一般的机器学习思路，我将算法流程列为：

数据预处理 -> 特征提取 -> 分类器

2.3 数据预处理 & 特征提取

因数据预处理与直方图特征直接相关，这里一同说明。

经初步调研[1]， C++ 代码的查重方法主要基于三类特征提取方法：直方图分析、代码结构分析、NLP 算法。本实验中样本代码的不完整特性为代码结构分析带来困难，因此不考虑；现有 NLP 算法的预训练模型多针对自然语言环境，且 C++ 代码在 NLP 算法中难以断句，实现较困难；因此我采用直方图分析的特征提取方法。

我选择词频直方图作为每个代码的特征。词频特征简单、易提取、并有基础的效果保证，缺点是忽略了代码的时序信息。建立词频直方图的步骤是：

1) 从所有训练样本中找出有价值的词组成词列表；

2) 对每个代码，关于词列表中的词进行计数，列表中所有位置的词频就组合成该代码的词频直方图。

注意以上步骤需遍历所有训练样本两次。其中，根据 C++语言特征，我将空格作为分隔符用来分词。

实验发现，当不对数据进行任何预处理时，所有训练样本中共有 2w+个不同词，也就是说每个样本的特征有 2w+维，这明显高出预期复杂度。在对所有词按总词频降序排列后，我发现两点特征，并针对进行改进：

词频分布为长尾分布。有大量的词总词频在 50 以下，而每类就有 502 个样本，可见这样的词在统计意义上并不构成有效特征。因此我从词列表中去除总词频在一定数量以下的词，实验中经测试定为 50。

C++语言允许部分语法不隔空格，使部分不应出现的新词产生，如 for(i=1;i<2;i++)将被分为一个词，而这个词出现概率极低，不具有统计意义，同时产生 for 词的漏记录。为弥补这一漏洞，我在预处理中在代码的特定字符前后插入空格。这些特定字符包括：'{', '}', '(', ')', '[', ']', '+', '-', '*', '/','&', ':', '.', ',', ';', '!', '<', '>', '\'', '\"'；结合 re 模块的模板匹配功能，该预处理不会产生大量空词。

在实行以上改进后，每个样本的特征维数降到 1455 维，已初步符合深度学习分类的要求。

此外，深度学习要求输入特征的值在[-1,1]区间内，而词频是自然数，要求归一化。经实验，我选择将绝对词频数转为和为 1 的相对词频值，并添加一维特征记录代码的总长度 min(0.001*L, 1)。

因训练集总样本对数约有 8*10^8 组，样本总特征数约有 6*10^7，做 PCA 等全局数据挖掘速度较慢，这里不选择进行处理。

2.4 分类器

在该二分类问题设定中，输入两个样本的特征，输出两类的概率。经测试，将输入样本的特征相减不能达到好的效果，猜测是损失了一些信息，因此我将输入样本的特征原汁原味地输入分类器，输入总长度为 2912。

对于如此高维的输入，传统机器学习算法如 SVM、kNN 等要么出现维度爆炸要么拟合能力不足，因此

只能使用深度神经网络作为分类器。其中 MLP 是最高效简洁的深度神经网络模块，被广泛采用。

网络结构方面，一开始我打算用孪生网络[2]（Siamese Network），其流程如下：

输入 1 -> 特征 1 | 输入 2 -> 特征 2

特征 1 ⊕ 特征 2 -> 输出

孪生网络是 CV 中做对比学习的经典思想，其优势在于可利用预训练的编码器，将特征不明显的输入图像转化为特征向量。在实验过程中，我逐渐想到词频直方图特征已是明显的特征，且并无预训练编码器可用，因此不如裁剪编码器部分。最后，我使用的网络结构如下：

输入 1 ⊕ 输入 2 -> 输出

该结构仅用一个 MLP 即可实现。MLP 的结构如下：

输入 1⊕2 -> FC1 -> ReLU -> FC2 -> ReLU -> FC3 -> Softmax -> 输出

其中隐藏层大小经试验定为 1024–512，激活函数经试验定为 LeakyReLU。

2.5 训练过程

损失函数采用二分类问题中经典的 BCE loss。为契合该损失函数，分类器 MLP 的输出层加入 softmax 层，以输出每类判别概率，再将‘1’类概率作为预测来计算损失函数。

因训练集总样本对数约有 8*10^8 组，普通地做循环训练是不现实的。我采用了随机抽取 Batch 的方法做训练，每 iter 抽取一个 Batch。该方法类似 SGD，能够在保证收敛的同时提供泛化性能，但与循环训练相比样本选取概率的方差更大，因此学习更加不稳定。为提高稳定性，我将 batch size 在不影响速度的情况下尽可能调大，提高 iter 数，降低学习率，并避免使用 SGD。

接下来考虑样本均衡问题。训练集总样本对数约有 8*10^8 组，但其中只有约 2*10^7 组是克隆样本，占比仅为 2.5%，简单随机抽取样本将带来训练结果的严重偏差。因此我在每次采样 Batch 中分别存储克隆样本对和非克隆样本对，并将非克隆样本对随机筛至克隆样本对数量，以此保证分类器同时学习克隆样本和非克隆样本的判别。

我还加入了一定比例的自克隆样本对，但效果不明显。

模型泛化性是一个很严重的问题。截止封榜，我的算法在训练集上能够达到 99%的 F1 分数，但测试时还只有 66%。为提升泛化性，我尝试了一些常用措施，包括提高 iter 数，降低学习率，使用学习率调节器，调整 MLP 层数及大小，这些可以提升一定效果，而加入 dropout 和 weight decay 会使效果降低。

3 实验结果

至提交截止时，A 榜第 7，B 榜第 6（共 193 队），提交结果最高可达到 0.67276 的 F1 分数。模型大小仅 13M，在单个 GTX 2080Ti 上训练 10 分钟可达到 0.60 分数，训练 2~3 小时达到最优，是非常简洁轻便的算法。

期间在一些参数上做了一些对比实验，结果如下：

表 1 词频直方图长尾截断效果对比（其它参数不变）

词频阈值	10	30	50	100
F1 分数	0.50996	0.60096	0.60094	0.58881

表 2 学习率效果对比（同时调整迭代轮数，其它参数不变）

学习率	0.002	0.001	0.0005	0.0002
F1 分数	0.63194	0.65038	0.67276	0.64531

4 结束语

本次大作业我从零开始实现了一个基于词频直方图和 MLP 的代码克隆检测算法，并达到了不错的排名。为一个任务从头开始设计算法是不容易的，这之中需要参考已有文献，也需要结合任务特征，选择最适合的方法。一开始，我选择词频直方图 + MLP 方法是看中它必然可实现、简单有效的特性，不期望能够得到靠前的名次；对算法的改进也是顺藤摸瓜，像这样有体系的报告只有在最后才会形成；我想做研究、做创新其实都是这样一条探索的路。另外，我发现任何任务上都免不了数据清洗，只有数据中的错误信息尽量减少，学习得到的模型才能更加可靠，这使我也更加体会到数据挖掘方法的重要性。.

References:

[2] Chopra S, Hadsell R, LeCun Y. Learning a similarity metric discriminatively, with application to face verification[C]//2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, 2005, 1: 539-546.

附中文参考文献:

[1] 朱林琴.C 语言查重与自动评分算法研究[硕士学位论文].湘潭大学,2017.