基于词频统计与多层感知机的代码克隆检测[[1]](#footnote-0)\*

陈诚1+

1(南京大学 匡亚明学院，江苏省南京市 210046)

**Code Clone Detection based on and Token Frequency and Multi-layer Perceptron**

CHENG Chen1+

1(Kuang Yaming Honors School, Nanjing University, Nanjing 210046, China)

+ Corresponding author: Phn +86-186-5148-9273, E-mail: 799371209@qq.com, http://dii.nju.edu.cn

**Abstract:** Software clone, means that software developers reuse existing codes by copy-and-paste, or implement functions very similar to existing ones. Code clone usually leads to software defects and infringement disputes. Therefore, it is necessary to design a system which can automatically detect whether two source code files are a clone pair. To achieve this goal, we have to employ a series of data mining algorithms. We first utilize a lexer to scan source codes and transfer code sequences into token sequences. Afterwards, we generate feature vectors for each source code using token frequency. Finally, by training on plenty of clone and non-clone pairs we build our MLP detection model, which can predict whether two source code files are a clone pair to some extent.

**Key words:** Software mining; Code clone; Token frequency; Multi-layer perceptron

摘 要: 软件克隆，即软件开发者通过复制粘贴已有代码进行开发，或者功能的实现与已有代码十分类似。代码克隆通常会留下软件漏洞，且会导致侵权纠纷，因此，设计出一款自动检测两份源代码是否为克隆对的系统显得十分必要。为了达成该目标，需要使用到一系列数据挖掘算法。我首先使用词法分析器扫描源代码并将代码序列转化为Token词序列，再通过词频统计生成源代码对应的特征向量，随后运用多层感知机对大量克隆对与非克隆对源代码特征向量进行训练，获得自动检测模型，可以在一定程度上预测出两份源代码是否为克隆对。

关键词: 软件挖掘;代码克隆;词频统计;多层感知机

# 任务理解

挖掘任务所提供的训练集包括83个文件夹，每个文件夹中的500个源代码文件样本两两为克隆对，不同文件夹中的源代码文件则为非克隆对。我们需要在该训练集上用数据挖掘技术学习预测模型，随后对由测试集中10000份源代码生成的200000组代码对进行克隆预测，使用F1分数评价模型的优劣。

该任务属于监督学习，我认为其中存在三个难点。一，数据挖掘算法的输入不能是源代码的形式，我们需要将其转化为对应的特征向量，那么如何提取特征才能在最大程度上保留源代码的信息呢？二，通过训练集我们可以生成组克隆代码对，与组非克隆代码对，约为克隆代码对的80倍，可见样本类别分布极不均衡，符合我们的常识----绝大部分的代码对并不相似，克隆代码对是较为稀有的。三，F1分数综合考虑了precision精准度和recall召回率，而这两者是互相矛盾的性能度量指标，不能厚此薄彼，高F1分数就意味着高鲁棒性的模型。

于是我将任务划分为四个主要步骤。第一，将训练集与测试集均预处理转化为向量形式；第二，从训练集中生成合理数量的克隆代码与非克隆代码向量对；第三，设计合理的模型对大量特征向量对进行挖掘；第四，对取自测试集的200000组代码对进行预测。

# 特征提取

## 数据预处理

源代码文件可以理解为一种文本文件：本身可以看成一个长字符串，由一连串的单词组成。其特殊之处在于，源代码的书写有其独特的格式，包括标识符（变量名）、关键字（while、for、int等）、常量（数值常量、字符串常量）、运算符（>=、&、\*等）、界符（[]、{}、;等）五种。自然语言处理中以空格为界对文本进行分词，而代码中可能一整个语句都不包含空格，因此这种分词方法显然是行不通的。经过编译原理的学习我知道，在将源代码编译转化成可执行文件的过程中，第一步便是词法分析（Lexical Analysis），将代码序列分解为token单词序列。因此，我使用clang编译器将源代码文件转化成单词序列文件，命令为

clang -cc1 -dump-tokens ' + src + ' 2> ' + dst

## 词频统计

词频统计固然忽视了源代码中的结构性、上下文信息，但也很大程度的保留了源代码的信息，可以用于评判两份代码的相似程度。

因此，我首先遍历训练集中的单词序列文件，统计出所有出现的单词，剔除一些冗余的单词（如左右括号必成对出现且数量相等，二者取其一即可），余下的73个单词便是73个特征维度，包括‘numeric\_constant’、‘string\_literal’等。随后第二次遍历训练集中的单词序列文件，对每一个文件统计单词词频形成对应的1×73特征向量。

至此，我完成了特征提取的步骤。

# 训练集与测试集的处理

通常情况下，训练集的格式应当为大量的数据样本与样本标记对，而任务提供的数据集却不是这种可以直接投入网络训练的格式。因此，对属于同一个文件夹的特征向量，我随机抽取2个组成2×73的样本向量并标记为1，重复多次；随机抽取2个不同文件夹，各随机抽取一个特征向量组成2×73的样本向量并标记为0，重复多次。将样本标记对洗牌shuffle后生成最终的训练集。从中可以分离出少部分作为验证集。

至于测试集，需要将20万对待测试的测试源代码文件同样转化为200000×2×73的向量。

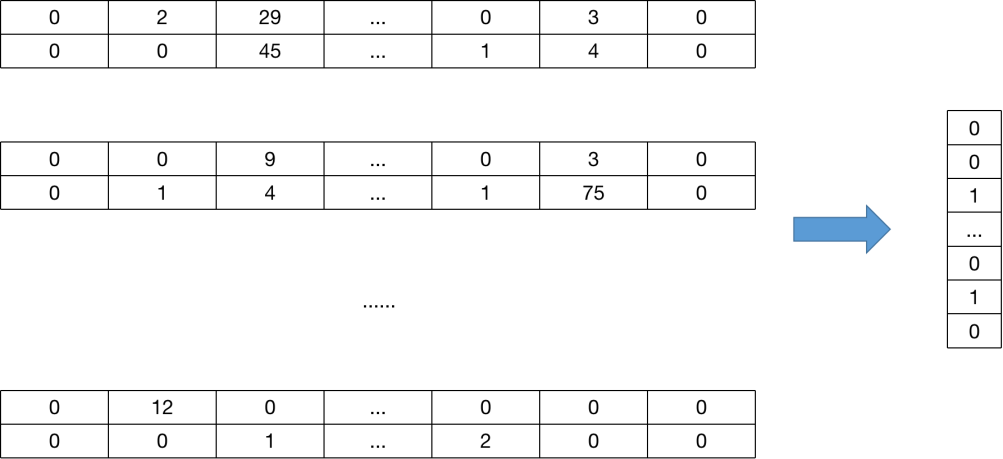


图 4-1 训练集的表示

# 实验结果

## 多层感知机

至此，已将源代码经特征提取成向量形式，并且已经构建了有标记的样本向量的训练集，下一步便是搭建模型并加以训练，学习样本向量与标记之间的映射，也即2×73到1的映射。最容易且最先想到的便是抢分类器神经网络，多层感知机MLP（Multi-layer Perceptron）模型。

我使用的是MXNet/Gluon深度框架，构建四层MLP，每层分别有512、256、128、1个结点感知机，最后一层的一个感知机的输出为0-1间的浮点数，取值大于0.5判定为1（表示是克隆代码），反之判定为0。我使用MSE作为损失函数Loss Function，学习率Learning Rate设置为0.01。

最开始的时候，训练集与验证集中正负样本的比例我设置约为1:1，训练短短几轮后，验证集的F1分数已经很高。但是，使用这些模型对测试集进行预测，得到的F1分数始终无法超过0.25，与在验证集上所得结果不符。

于是，我试图对测试集中的正负样本比例进行研究。根据F1分数的定义，以及全1的预测取得0.09529的F1分数，可以计算出测试集200000对样本中约有10000对为正样本。这也就意味着正负样本比例约为1:19。

于是，我将训练集与验证集中的正负样本比例设置为1:10，设置训练轮数为70-100，重新进行训练，可以看到验证集的F1分数初始极低，随着训练逐渐提升。训练所得模型的在测试集上的F1分数最高接近0.58（0.57744/0.57759）。

## 其他分类算法

数据挖掘的算法还有许多，为了体现MLP算法的优越性，我将算法结果与朴素贝叶斯算法、决策树算法进行了比较。

|  |  |  |
| --- | --- | --- |
| 算法 | 参数 | Public-F1 |
| MLP | (146-512-256-128-1), lr = 0.01 | **0.57744** |
| Decision Tree | max\_depth = 15 | 0.12717 |
| Naive Bayes | No parameters | 0.05155 |

# 总结与展望

## 总结

1. 特征的提取是至关重要的一步，对于源代码文件使用对应的词法分析器是很好的选择。但通过词法分析得到的信息在转化为
2. 实际上，由于克隆代码对（正样本）的相对稀有性，合理设置训练集与验证集中的正负样本比例，可以更好地指示当前模型的实际效果。同样的，对于测试集设置不同的正负样本比例可以更好地检测当前模型的鲁棒性。
3. 神经网络算法，较其他机器学习算法，尽管需要更长的训练时间，有更多的参数需要调试，但通常能取得更好的分类预测结果。

## 展望

对于二分类问题而言，0.58的F1分数显然是不够理想的，我认为可以从以下几个方面入手尝试改进。

1. 词频统计实际上忽视了源代码中的结构性、上下文信息，尝试对于这些信息构建对应的特征。

2. 对词频特征向量进行正则化、归一化等处理，也许可以提升模型的训练效果。

3. 运用信息论的思想，衡量不同特征的重要程度，并据此赋予不同的权重。

4. 对MLP模型而言，如何设置隐层数、每层的结点数、学习率等等，也许有更佳的参数选择。

致谢 非常感谢一学期以来老师的教导与助教的答疑解难。

1. 作者简介: 陈诚(1998-),男,江苏省盐城市人,学士,主要研究领域为机器学习、图像超分辨。 [↑](#footnote-ref-0)