**主要改动:**

根据L老师的提议, 主要改动为:

Sent2vec模型的第一个步骤, 求对词向量序列加权平均, 初步得到一个句子向量.



的含义是考虑单词的权重. a是超参数, 固定为10e-3, 如果一个单词在语料库中出现的越频繁, 那么这个单词的权重越低.

1. **sent2vec模型的用法**: 模型中有个参数p(w), 这个参数含义是单词w在语料库中的词频. 之前的做法一直是使用默认值, 即为定值1. 因为如果考虑到结点的词频, 那么最后的检测效果跟被检测项目有很大关系,

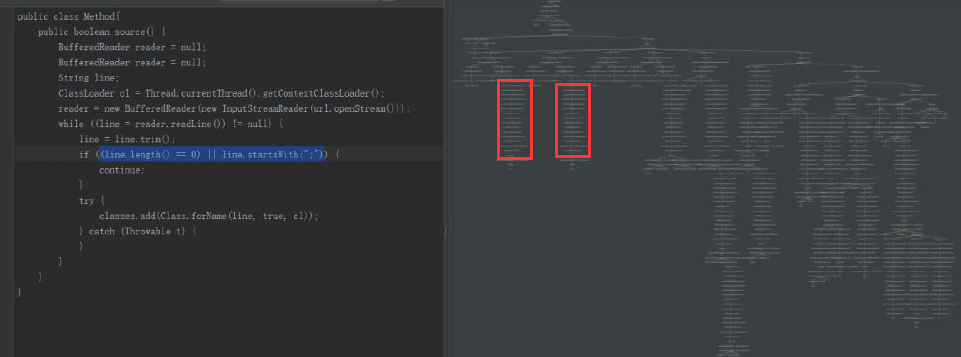
举个例子, 假如现在有两个方法代码,如果只对它们进行检测, 那么最后得出的距离可能是5, 另一方面如果还是这两个方法,但是把它们放到一个大数据集再跑一遍, 那么最后它们的距离可能会是1,2,3等等(这个值可能会大也可能会小, 今天跑过实验刚发现到.). 而判断克隆的阈值是固定的. 所以会影响到对克隆的判断.

1. **stop node问题**

现在不过滤结点了….考虑所有的结点类型

当初为什么做过滤

当时考虑到, 方法AST最后形成的结点序列, 序列中的stop-node大量存在, 如下图红圈内结点.



如果两个方法序列大量含有这些序列, 那么在检测它们时考虑这些结点会对准确率造成较大影响,

尤其是以前用过的传统字符匹配做法, 本质是计算两个序列中相似结点所占的比例, 最终得出两个序列的相似度.

然后很自然想到把这些结点过滤掉, 这样能够做到很好区分不同的语法结构.

但是现在老师说, 不考虑这些结点也可以, 因为sent2vec模型能够出现过于频繁的结点的权重降低. 让 试试不过滤结点的实验效果如何.

所以重新生成了一次random walk语料库, 这次是把这部分单独作为一个独立项目, 减少代码混乱.

对现在的实验效果影响很大. 最关键的是不知道效果是变得越好还是越差.

现在方法法的检测效果 , 准确率是可以的. 但是recall效果不好,

之前和刘老师讨论得很频繁, 关于方法上的改动做得特别多,

结点词向量上, AST遍历方式上, 如何生成结点序列上, token序列上,sent2vec模型上等等.

总之我感觉到的是, 这中间的改动, 没有一次是对效果提升特别明显的, 老师考虑的出发点始终是从理论入手, 这很毛病, 但是实验效果总是不尽如人意.感觉总是在原地徘徊.

所以想大家一起来讨论方法, 或许会对效果提升有新发现新思路新方法.

目前结症在于:

Recall问题

基于AST的优势体现的不是特别明显

检测效率过慢

**重要不紧急:**

无法实现增量检测, 实际应用价值大打折扣. 如果能够检测一个开发人员提交的代码是否会对系统造成代码克隆,从而在源头规避克隆,或许这会是比较有意义的一点.

现在有一些工具和方法实现了增量检测, 实时检测,大规模检测.

或许可以从这些文章中借鉴一些经验.

