IRBL迭代二：算法设计文档

作者：程荣鑫

日期：2021.4.13

预处理

Java代码文件

1. 使用Java语法树生成工具Java Parser将代码文件转换为语法树，提取出Java文件中的变量名、方法名、类名，以及注释
2. Java代码中的类名是非常关键的信息，它最能代表一个类文件，所以我们拉高了文本中类名的权重，使得类名占整个Java处理后文件词语数的10%
3. 接下来的处理方法就是转换小写、切词、去停用词、词形还原

Bug report

1. summary相比description有更富集的信息，基于这样的假设我们拉高了report预处理文本中的summary的权重

   $$report' = report[summary] * beta + report[description]$$

   这里的beta取值4时效果较好

2. 同样，接下来的处理方法就是转换小写、切词、去停用词、词形还原

向量化

向量化没有太多的新意，仍然使用tf-idf方法向量化

$TF = log(f) + 1, f$为词语在文本中的频数

$IDF = log\frac{docn}{containsDocn}$　　$docn$：文档总数；$containsDocn$：含有该词语的文档总数

为了计算bug之间的相似度（下面的算法会用到），我们单独为bug reports建立了语料库，计算bug reports之间的相似度

相似度计算

rVSMScore

原来的VSM偏好长文本，这里使用G值修正，即rVSM模型，预测效果比VSM更好

$g = \frac{1}{1+e^{-N}}$, N为该文本的词语数

$sim' = sim \times g$, sim即VSM模型计算结果

SimiBugs

“近期修改的文件可能仍有bug”，基于这样的假设，我们参考历史修复的bug涉及文件来修正预测

计算公式为：

$$SimiScore = \sum_{\substack{\text{All } S_i \text{ that} \\ \text{connect to } F_j}} (Similarity(B, S_i) / n_i)$$

FinalScore

$FinalScore = (1-\alpha) \times N(rVSMScore) + \alpha \times N(SimiBugScore)$

2021.4.15补充：在实验中，我们发现alpha取0.2时能达到最好效果（实验数据见alpha_log.txt）