

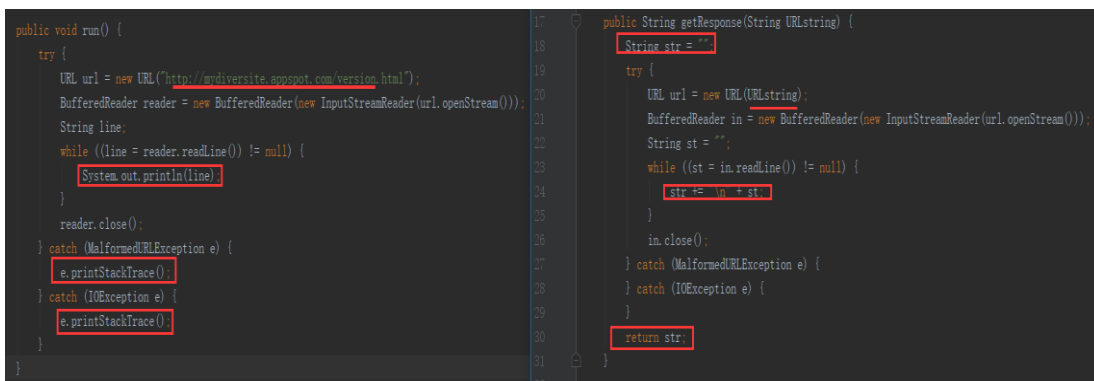
参数 a 调到 0.1,权重不同对最后向量距离的影响变小,

重新跑了之前的 ant 数据集, 定了一个克隆阈值 0.015

Int 和 float 的例子跑出来的距离是 0.01.

之前收集的没有跑出来的 ST3, 总共有 6 个 case. 用现在的方法能跑出来 4 个在阈值之内.但是仍然有两个特殊 case 没有跑出来.

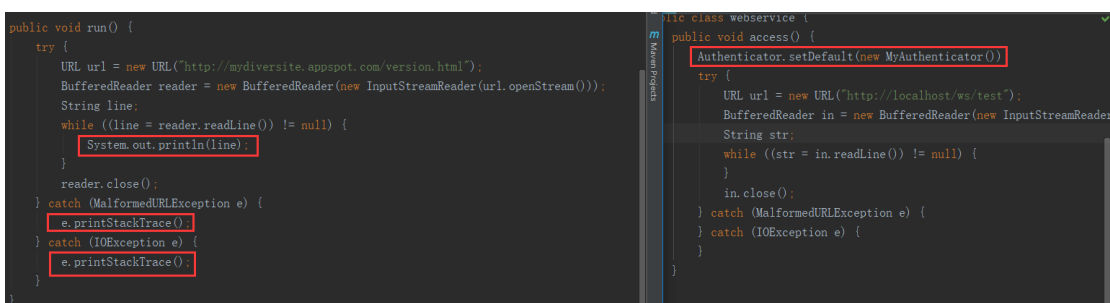
它们距离分别是 0.05 和 0.03,但是通过 token 相似度得出它们分别是 0.8 和 0.8



```
public void run() {
    try {
        URL url = new URL("http://mydiversite.appspot.com/version.html");
        BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        reader.close();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

public String getResponse(String URLstring) {
    String str = "";
    try {
        URL url = new URL(URLstring);
        BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
        String st = "";
        while ((st = in.readLine()) != null) {
            str += "\n" + st;
        }
        in.close();
    } catch (MalformedURLException e) {
    } catch (IOException e) {
    }
    return str;
}
```

dist:0.5, beta:0.8



```
public void run() {
    try {
        URL url = new URL("http://mydiversite.appspot.com/version.html");
        BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));
        String line;
        while ((line = reader.readLine()) != null) {
            System.out.println(line);
        }
        reader.close();
    } catch (MalformedURLException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

class webservice {
    public void access() {
        Authenticator.setDefault(new MyAuthenticator());
        try {
            URL url = new URL("http://localhost/ws/test");
            BufferedReader in = new BufferedReader(new InputStreamReader(url.openStream()));
            String str;
            while ((str = in.readLine()) != null) {
            }
            in.close();
        } catch (MalformedURLException e) {
        } catch (IOException e) {
        }
    }
}
```

dist:0.3, beta:0.8

上面这个两个 case, 通过遍历 AST 生成的两个序列之间差异就比较大(约三分之一不同)

考虑过滤 stop-node, 调整权重,阈值参数等等方式都无法从根本上解决这种 case,

还是因为两个方法 AST 部分子树差异比较大. 算法将这些差异带到了最后生成的方法向量.

但设定的向量距离高于 0.02 将认为不是克隆.这么做的原因是距离超出 0.02 的数据量很庞大,

举个例子, ant 数据集 1205 个 java 文件,检测的出的距离在:

0-0.015 之间的数据量有 1000 个,

0.015-0.02 之间的数据量有 7000 多个.

这个数值如果是大数据集,数据量会非常庞大.

所以结论是虽然通过按行相似度比较可以将它们检测出来,但是这么做效率上会明显降低.

Token 求相似度过程的复杂度比较高...

两个方法按行比较(两层 for), 两个代码行按字符比较(两层 for)

所以对于这两种需解决的 case:

最好是在检测的时候只考虑两个 AST 的相似部分.

如果检测过程只考虑相似部分,而不考虑两者差异部分, 根据最后算出相似部分所占比例来求出相似度, 这样能比较好解决这种 case,

比如按代码行求结点序列, 这样做类似于 token 做法, 但是因为用了一层 sent2vec,最后只有两层 for 比较距离,而不会有 4 层.

同时省去了求 token 相似度这部分.