1、tf-idf.py:对每个类的api进行筛选。

tf=该类别包含当前api的apk数/该类别总的apk数，读取Desktop\newclass文件夹之下的数据，newclass文件夹下每一个类别中有一个api\_num.txt文件，里面统计了每个api在本类别出现的次数，里面的数据格式如下： "void <clinit>()>": 770,“返回值<函数名>（返回值）”：出现次数。

该脚本使用的另一个数据文件是Desktop\newclass\api-apk.txt,里面统计了每一个api在全类别中出现的apk数。里面的数据格式如下："boolean isInEditMode()>": 4534，“返回值 函数名（返回值）>”,和每个类别中的api\_num.txt中的格式不一样。

结果输出在Desktop\ld代码\tf-idf文件夹中，每个类别选取了tf-idf结果最高的24个api，分别存储在每个类别自己的type-24-ti.json文件中。

2、api\_ti\_select\_.py:对tf-idf筛选之后的结果进行扩充。

统计tf-idf最终筛选了多少不同的api，outcome列表用来存储结果，最终结果存储在Desktop\ld代码\api\_ti\_sum.txt文件中。

进行api集合拓展，用于后续hits算法。将每个类别的拓展结果单独存在Desktop\ld代码\call\_statistic\_each\type\_call.txt文件中。

将筛选出来的api和链接关系得到的api统一存储到Desktop\ld代码\api\_statistic\_each\type\_api.txt文件夹中。

3、HITS.py: 对每个类别的结果进行hits分析。

使用Desktop\ld代码\api\_statistic\_each文件夹和Desktop\ld代码\ call\_statistic\_each文件夹中的数据进行hits算法迭代，迭代结果保存在Desktop\ld代码\hits\tpye\_hits.txt文件中。

4、api\_add\_ti\_hits.py: 将hits筛选出来的特征和tf-idf筛选出来的特征合并。

将两次筛选的结果合并在Desktop\ld代码\api\_1000.txt文件夹中。

5、api\_dic.py: 构建用于tovector的字典。

通过读取Desktop\ld代码\api\_1000.txt中的api和Desktop\ld代码\call\_statistic\_each\type\_call.txt中的调用关系，构建用于生成每个apk的特征向量的api\_dic字典。只保留前个调用。

6、tovector\_1000.py:生成每个apk的机器学习输入向量。

结果保留在Desktop\datas文件夹中。

7、train.py:顾名思义，最后的训练结果。  
其他代码文件：

1、api-apk.py:统计每一个api在多少个apk中出现过。用于tf-idf计算。

2、parse\_all\_smali.py:自己编写的smali代码分析文件，提取api和api之间的调用关系，后期没用了，使用的是flow-droid工具直接生成调用链。

3、api\_statistic.py：用于统计每个类的api在多少个apk中出现。

4、parse\_only\_method.py:用于分析manifest文件，提取activity和permission等信息，最后为每个apk生成permission、api\_call、parse\_smali三个txt文档。