基于 PCFG 口令猜测算法分析

基于 PCFG 的方法允许用户创建自己的口令并且自动从训练集中导出编码规则。将数据集通过随机数随机选取分成两个部分。

训练集：60%

测试集：40%

（1）PCFG 方法得出基础结构并计算概率

将所有训练集中的口令划分成相应字符的序列片段，并且获得相应的**基础结构（LDS 形式）**及其相关的发生概率。

在程序中，对于给定的训练集中的每一个口令，将口令变换成基础结构类型，并将这样的基础结构加入专门保存基础结构的字典（mode）中，同时计算相同基础结构类型出现的次数（key:value）。

除此之外计算出所有基础结构类型出现的总次数，每个口令中存在的字母、数字和特殊字符的概率，以个数来分开存储。

例如，口令"zhangsan@123"中"zhangsan"将划分为 L 片段，"@"将划分成 S 片段，"123"将划分成 D 片段。它的基础结构是 L8S1D3。L8S1D3 的概率如下:

$$\frac{\#count(L8S1D3)}{count(base\_structure)}$$

"123"代表的 D3 片段在所有的口令中的概率如下：

$$\frac{\#count("123")}{\#count(D3)}$$

同理"@"代表的 S1 片段在所有口令中出现的概率如下，这些信息用于生成概率上下文无关文法。

$$\frac{\#count("@")}{\#count(S1)}$$

对得出的概率**降序排列**。

pcfg.py 程序将基础结构概率存于 base_struct 目录下

将字母各类型概率存于 base_alpha 目录下

将数字各类型概率存于 base_digit 目录下

将特殊字符各类型概率存于 base_special 目录下

（注：本次设计考虑去除空字符）

（2）生成字典+猜想算法实现

在生成新的训练集的程序中，将存储基础结构类型字典中的每个基础结构片段与**字母生成的字典**、在训练程序中生成的数字和特殊字符存储的文件进行比对，生成新的口令训练集。

以基础结构 L8S1D3 为例， L8 片段表示此结构中 8 个连续的字母，那么就在字母字典中寻找字母长度为 8 的口令，并保存到用来存储字母的列表中；S1 同样的表示此结构中有一个特殊字符，那么就将保存相应个数的特殊字符文件中的内容保存到用来存储数字的列表中，同样的可以将数字保存到相应的用来存储数字的列表中。

在这些列表生成完了之后，以 L8S1D3 的顺序对每个列表进行遍历，将遍历中匹配到的内容相结合，这就生成了新的训练口令集。比如，L8S1D3 在字典中匹配到字符串"zhangsan,lisi "，在数字和特殊字符文件中匹配到"123"，

"@"，那么生成的口令就应该是"zhangsan@123"和"lisi@123"。而任何一个猜测的概率都是其内部各个片段概率的乘积。比如，"zhangsan@123"的概率就是：

P("zhangsan@123")=P(L8S1D3)*P(L8→zhangsan)*P(S1→@)*P(D3→123)

生成的新的训练集中，口令也是按照概率由高到低排列。数字片段和特殊字符片段的概率从训练集中得出，而字母字符片段的概率可以从训练集中得到也可以从外部输入字典中得到。

文件提供的算法解析（guess_mine.py 文件的 Guess 类）：

1. initqueue()方法在**优先队列**中依次放入训练集每个模式概率最高的值，其中字母保留缩写，比如 5678L35678#@5678#@78L3；
2. queueinsert()则针对每个模式，每次取出队首位置的模式，对每个模式同类型段的值进行等价字符替换，每次只替换一位，并依次插入队列。比如取出 5678L35678#@5678#@78L3 后，在以下循环中依次向队列插入新值

```
for i, s in enumerate(base):
```

1234L3——————————— ①
5678L31234——————————— ②
5678L35678..——————————— ③
5678L35678#@1234——————— ④
5678L35678#@5678..——————— ⑤

第二次调用 queueinsert()方法时，则会取出优先级较高的①号值，同上进行遍历，注意#和.符号由于#的 ASCII 编码较小，所以在比较时，#优先级较高。

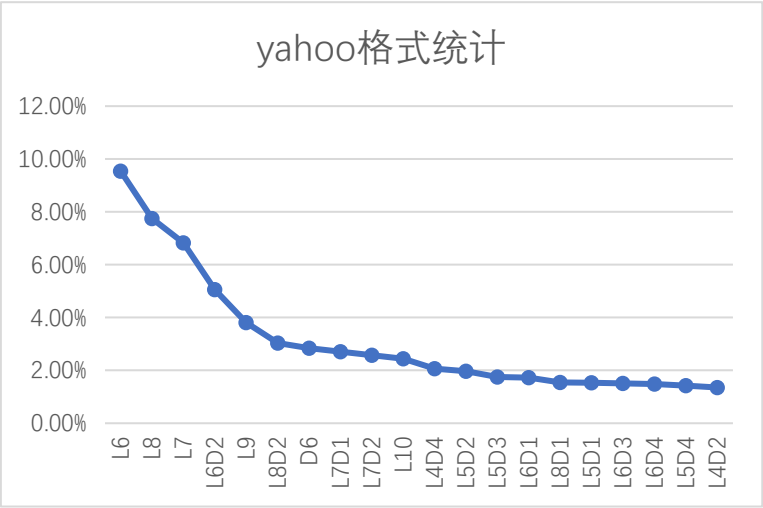
（3）guesspw()方法实现口令猜解

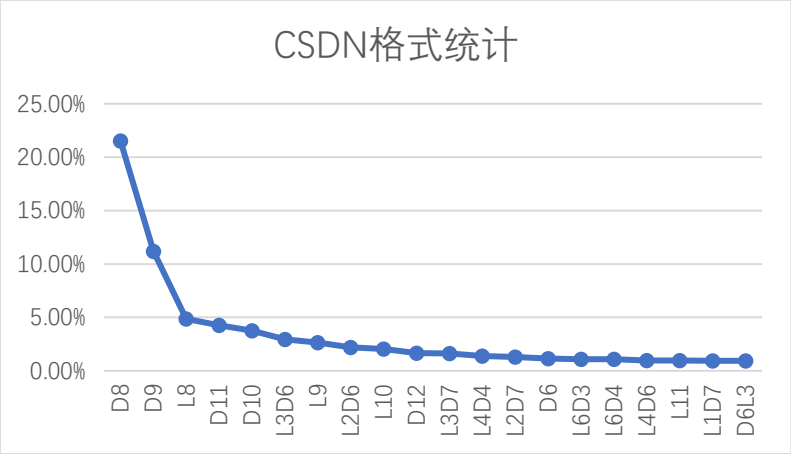
该方法会依次获取到优先队列的模式值、模式的字母位置（列表存储，可能多个）、以及模式概率，没有字母直接跳过，否则通过 replacements 列表列出每个位置所有可能对应情况，比如 5678L35678#@5678#@78L3，假设 L3 有两种可能情况:asf 和 abc，则最终调整的 replacements 为[(asf, asf), (asf, abc), (abc, asf), (abc, abc)]，将所有可能结果写入 txt 文件。

口令在测试集中，视为猜测正确，每个猜测口令可能对应多个测试集口令。

该程序将口令概率值小于 0.000000001 的值过滤掉。

（4）统计结果




CSDN格式统计
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
D8
D9
L8
D11
D10
L3D6
L9
L2D6
L10
D12
L3D7
L4D4
L2D7
D6
L6D3
L6D4
L4D6
L11
L1D7
D6L3