Fastbot-Android 开源版用户手册

Fastbot介绍

> 基于Monkey二次开发结合机器学习、强化学习的稳定性测试工具

优势：

1. **Android多os兼容：**
   同时兼容Android5-11，兼容国内各厂商定制化的Android系统及原生Android系统；
2. **事件快速注入：**
   继承原生Monkey的优势，快速点击，每秒最高可发送12个事件；
3. **专家系统：**
   不同业务线支持不同的个性化需求，业务深度定制化；
4. **智能化测试**：
   基于model-based 边遍历边建模，利用强化学习等算法做高收益决策；
5. **跨平台**：
   支持非标准化控件，*YOLOv3*、ocr、cv分割等UI图像识别能力；

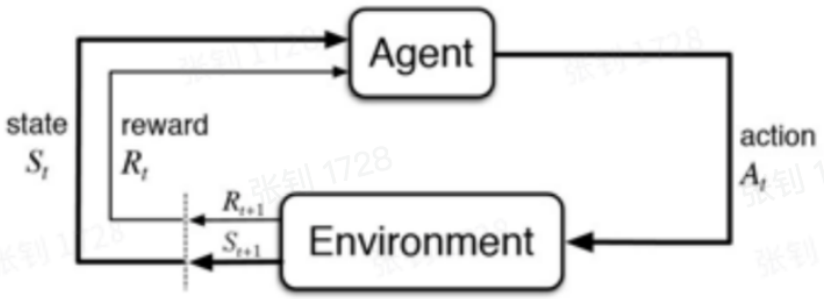
Fastbot-iOS 敬请期待

算法原理

> 在app遍历过程边遍历边自学习，学习app ui transition model同时结合强化学习算法决策出最优路径，提高单位时间遍历效率，通常用Activity覆盖率或代码覆盖率来衡量效率；

想了解更多 -> https://mp.weixin.qq.com/s/3t4H2bfDjei4vXkj_Cz2pg

1. 知识：
   a. 什么是强化学习：



i. 训练一个Agent，不停的和环境（environment：E）进行交互，每次交互得到环境的抽象状态（state：St），以及上一步操作（action：At）执行后带来的收益（reward：Rt）

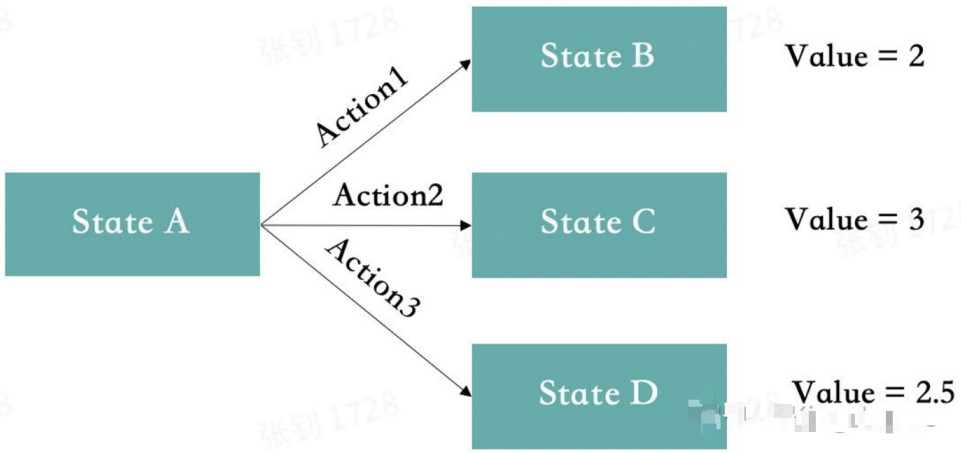
b. 强化学习训练过程：

i. Markov Decision Process（MDP）

1. 马尔可夫决策过程核心思想就是下一步的state只和当前的状态state以及当前状态将要采取的action有关，只回溯一步。比如上图state3只和state2以及action2有关，和state1以及action1无关。我们已知当前的state和将要采取的action，就可以推出下一步的state是什么，而不需要继续回溯上上步的state以及action是什么，再结合当前的（state，action）才能得出下一步state。实际应用中基本场景都是马尔可夫决策过程，比如AlphaGo下围棋，当前棋面是什么，当前棋子准备落在哪里，我们就可以清晰地知道下一步的棋面是什么了

ii. 基于value-based训练

1. 基于每个state下可以采取的所有action，这些action对应的value, 来选择当前state如何行动。强调一点这里面的value并不是从当前state进入下一个state，环境给的reward。因为我们实际训练时既要关注当前的收益，也要关注长远的收益，所以这里面的value是通过一个计算公式得出来的，而不仅仅是状态变更环境立即反馈的reward。这个计算公式取决于用的是单步还是N步；而且这个value是采样得到的，需要经历多轮迭代，loss收敛后才能认为训练结束；
2. 学习到的**策略P**：简单来说，选择当前state下对应value最大的action。选择能够带来最大value加成的action。比如下图stateA状态下，可以采取的action有3个，但是action2带来的value最大，所以最终agent进入stateA状态时，就会选择action2。*（强调一点这里面的value值，在强化学习训练开始时都是不知道的，我们一般都是设置为0。然后让agent不断去尝试各类action，不断与环境交互，不断获得reward，然后根据我们计算value的公式，不停地去更新value，最终在训练N多轮以后，value值会趋于一个稳定的数字。才能得出特定state下，选择某个action，会得到怎样的value）*



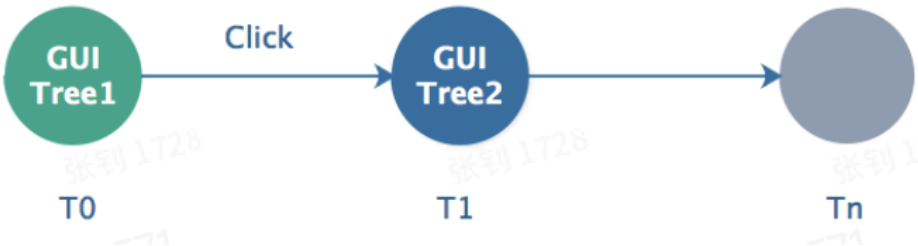
iii. Explore & Exploit(参考https://zhuanlan.zhihu.com/p/21388070)

3. 如上图stateA的状态下，最开始action1&2&3对应的value都是0，因为训练前我们根本不知道，初始值均为0。如果第一次随机选择了action1，这时候stateA转化为了stateB，得到了value=2，系统记录在stateA下选择action1对应的value=2。如果下一次agent又一次回到了stateA，此时如果我们选择可以返回最大value的action，那么一定还是选择action1。因为此时stateA下action2&3对应的value仍然为0。agent根本没有尝试过action2&3会带来怎样的value?
4. 所以在强化学习训练的时候，一开始会让agent更偏向于探索Explore，并不是哪一个action带来的value最大就执行该action，选择action时具有一定的随机性，目的是为了覆盖更多的action，尝试每一种可能性。等训练很多轮以后各种state下的各种action基本尝试完以后，我们这时候会大幅降低探索的比例，尽量让agent更偏向于利用Exploit，哪一个action返回的Value最大，就选择哪一个action。

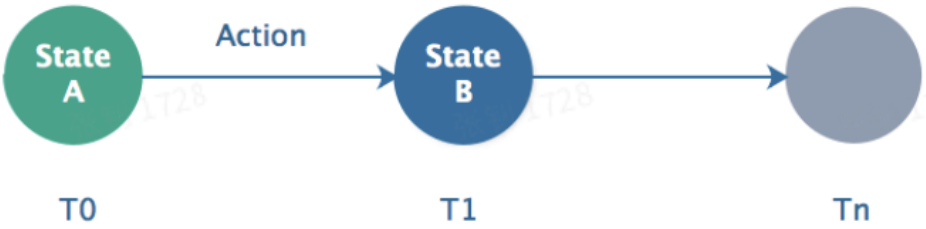
2. 本遍历算法选用n步资格迹强化学习，主要分如下三步：
   a. app遍历时fastbot构建状态转移模型：
      i. 这个模型是一个基于时间轴的App有向图，会记录App每个时刻的状态以及状态间的迁移。
      ii. 首先每个时刻可以从Accessibilityserice中提取出当前Tree结构， Tree经过一个基于抽象函数f0处理逻辑生成State

Tree -> f0 -> State

iii. T0 时刻Tree1经过对widgetA 做actionA 跳到 Tree2

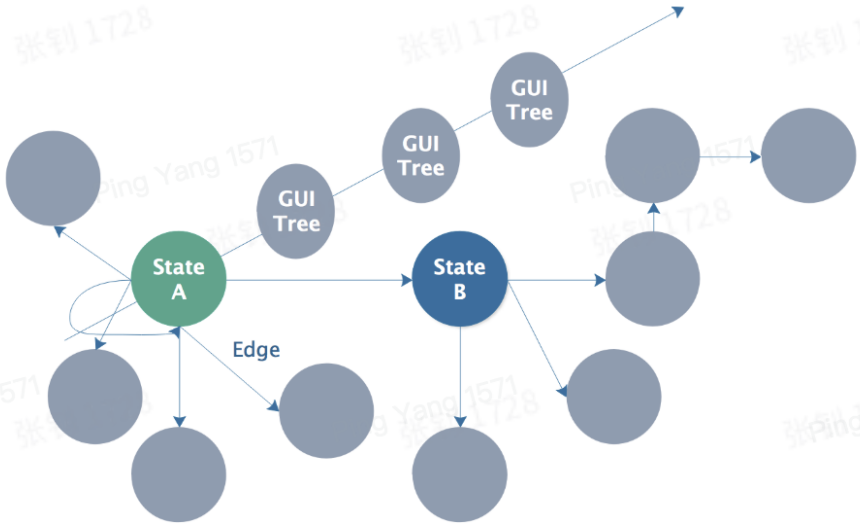


State对应2的过程



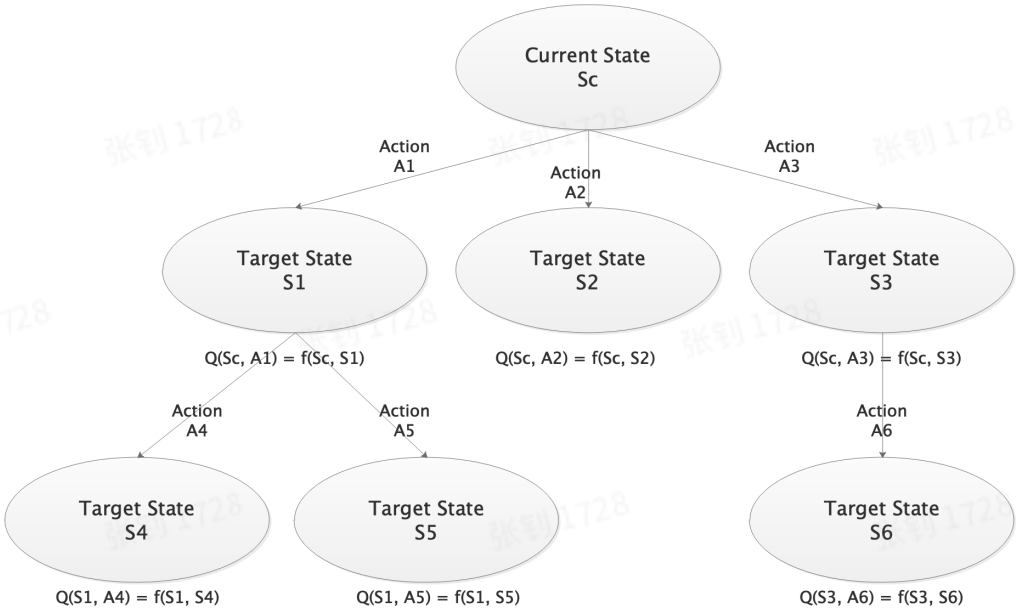
iv. 经过n个时刻 生成高纬度的graph

第4维度：一个State 可以有多个GUITree （图中椭圆形），各Tree之间存在跳转关系（沿虚线第四维度观察，关系图中省略了）

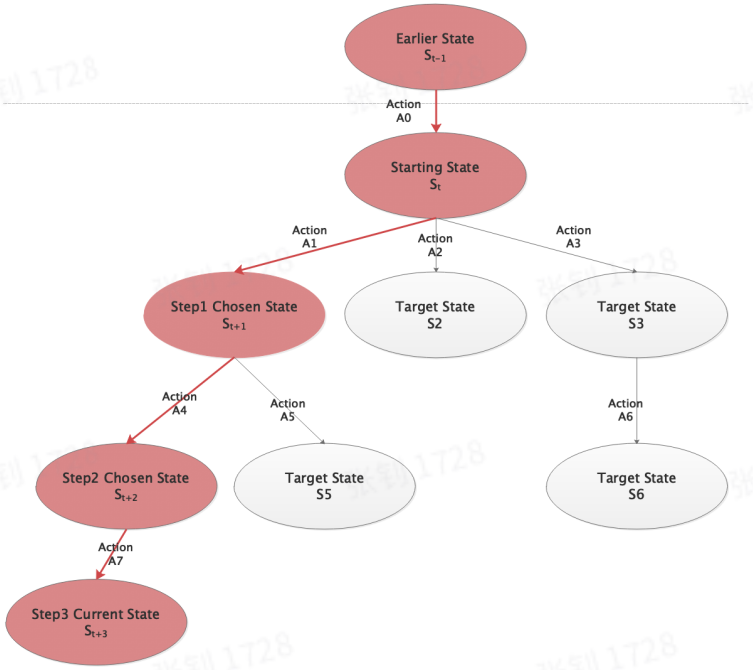


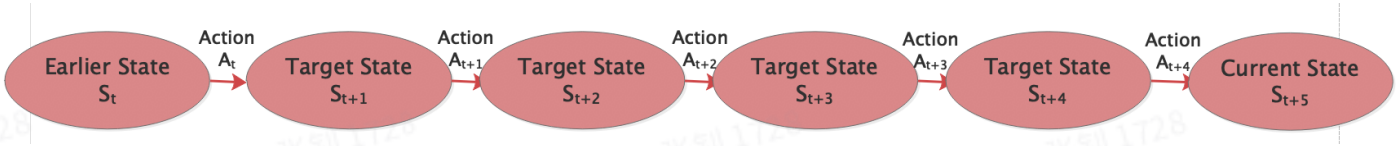
b. 边遍历边更新n步价值收益（Q值表示收益）：

i. 资格迹是指 通过一个堆栈保存一条路径上的G（n 是一个超参数，n越大学习到方差越小，收敛速度越快，但计算量越大）



- 计算N步之内的state-action的Q值，以路径正向进行获取reward，路径以下图红色节点路径为例：





- Q值计算公式：

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n})$$

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha[G_{t:t+n} - Q_{t+n-1}(S_t, A_t)]$$

- 其中：
  - G 为n步折扣收益； r 为步长衰减系数；
  - alpha 为学习率；负向线性衰减；
  - Reward = f(new Activity, new state, new action, revisited state) （以新发现的activity，新访问的state，新访问的action和重复访问的次数的负反馈构成reward function）

c. Explore & Exploit

iii. 通常算法上一般使用Epsilon-Greedy算法来决策Explore & Exploit；这是一个朴素的算法，也很简单有效，有点类似模拟退火：

3. 选一个(0,1)之间较小的数epsilon

4. 每次以概率epsilon（产生一个[0,1]之间的随机数，比epsilon小）则执行maxq；反之执行其他action
5. epsilon随时间而增大，最终增涨到1；

iv. Epsilon-Greedy算法的问题是epsilon增长是个超参数，增长速度快、慢实际可能都不合适，需要综合看测试时长及app的复杂程度，总的来说比较难以衡量及泛化；

v. 本算法使用UCB 公式代替 epsilon-greedy算法来实现 Explore & Exploit；

1. 对于state下存在未访问的action，则随机选取一个unvisited action执行，并计算Q值；
2. 对于state下所有action都已经访问过，则利用UCB公式来平衡已访问action的Q值和访问次数：

$$UCB\ value\ of\ visited\ Action = \frac{P_t}{V_t} + C\sqrt{\frac{\ln V_c}{V_t}},\ C: Constant$$

Pt 为action的Q值；
Vc 定义为当前action的访问次数；
Vt 为当前state的访问次数；
C 为常数，用以平衡action价值带来的正反馈和访问次数带来的负反馈，通常选取根2；

本地接入

1. 环境预备

- 支持 Android 5，6，7，8，9，10，11真机及模拟器
- 将 framework.jar , monkeyq.jar push 到手机上某个目录中，建议 `/sdcard`

```
adb push framework.jar /sdcard
adb push monkeyq.jar /sdcard
```

2. shell运行

```
adb -s 设备号 shell CLASSPATH=/sdcard/monkeyq.jar:/sdcard/framework.jar exec app_process /system/bin com.android.commands.monkey.Monkey -p 包名 --agent robot --running-minutes 遍历时长 --throttle 事件频率 -v -v
```

参数说明

```
-s 设备号 #多个设备需要指定设备号，单独设备无需此-s参数
-p 包名 # 遍历app的包名，-p+包名
--agent robot # 遍历模式，无需更改
--running-minutes 遍历时长(分钟) # 遍历时间：--running-minutes 时间
--throttle 事件频率 # 遍历事件频率，建议为500-800

（可选参数）
--bugreport #崩溃时保存bug report log
--output-directory /sdcard/xxx #log\crash 另存目录
```

结果说明

a. Crash、ANR 捕获
   - 捕获到Java Crash、ANR、Nativie Crash会以追加方式写入/sdcard/crash-dump.log文件
   - 捕获的Anr 同时也会写入/sdcard/oom-traces.log文件

b. Activity覆盖率统计
   - 正常跑完Fastbot会在当前shell中打印totalActivity（总activity列表），ExploredActivity（遍历到的activity列表）以及本次遍历的总覆盖率
   - 总覆盖率计算公式： coverage = testedActivity / totalActivities * 100

注意事项

- totalActivities：通过framework接口 PackageManager.getPackageInfo 获取，这包含app中所有的Activity，其中也包含了很多废弃、不可见、不可达等Activity

3. 专家系统

a. 自定义输入法（自动输入+屏蔽输入栏）

> ADBKeyBoard在输入栏自动输入内容，屏蔽UI输入法
> 适用需求：遇到搜索栏乱输入，想要输入指定字符

- 下载ADBKeyBoard，并在手机端中设置为默认输入法 ADBKeyBoard下载地址
  - 生效后，当遇到输入栏ADBKeyBoard不会弹起ui输入栏，会显示ADB Keyboard{ON}
    tarbar
- 随机输入字符串：
  - 配置max.config 中 max.randomPickFromStringList = false
    (1) 在pc端新建**max.config**文件（文件名称不可更改）
    (2) 输入 max.randomPickFromStringList = false
    (3) 通过以下命令将max.config文件push到手机端

```
adb push max.config /sdcard
```

- 从文件中随机读取字符串输入
  - 配置max.config 中 max.randomPickFromStringList = true
    (1) 在pc端新建**max.strings**文件（文件名称不可更改）
    (2) 输入想要输入的字符串，字符串结束要换行

(3) 通过以下命令将文件push到手 机端

```
adb push max.strings /sdcard
```

b. 自定义事件序列

> 手动配置Activity的路径（UI自动化用例）
> 适用需求：场景覆盖不全，通过人工配置到达Fastbot遍历不到的场景

i. 在pc端新建 **max.xpath.actions**文件（文件名称不可更改）
ii. 编写事件序列配置（case）：

max.xpath.actions

```
[
{
        "prob": 1,
        "activity":"com.ss.android.account.v3.view.NewAccountLoginActivity",
        "times": 1,
        "actions": [
            {
                "xpath": "//*[@resource-id='com.ss.android.article.news:id/bwa']",
                "action": "CLICK",
                "text":"[redacted]",
                "clearText": false,
                "throttle": 2000
            },
            {
                "xpath": "//*[@resource-id='com.ss.android.article.news:id/bwi']",
                "action": "CLICK    ",
                "text":"[redacted]",
                "clearText": false,
                "throttle": 2000
            },
            {
                "xpath": "//*[@resource-id='com.ss.android.article.news:id/q5']",
                "action": "CLICK",
                "throttle": 2000
            }
        ]
}
]
```

- prob：发生概率，"prob"：1,代表发生概率为100%
- activity：所属场景，详见：三.获取当前页面所属的Activity
- times：重复次数，默认为1即可
- actions：具体步骤的执行类型
- throttle：action间隔事件（ms）

action 支持以下类型：**action必须大写

1. CLICK：点击，想要输入内容在action下补充text，如果有text 则执行文本输入
2. LONG_CLICK：长按
3. BACK：返回
4. SCROLL_TOP_DOWN：从上向下滚动
5. SCROLL_BOTTOM_UP：从下向上滑动
6. SCROLL_LEFT_RIGHT：从左向右滑动
7. SCROLL_RIGHT_LEFT：从右向左滑动

iii. 存在切换页面情况：

1. activity会跳转，actions也应该拆分（同一个activity不需要做拆分）

格式为图下：从prob开始写下一个activity

```
{
        "prob": 1,
        "activity":"com.ss.android.article.news.activity.MainActivity",
        "times": 1,
        "actions": [
            {
                "xpath": "//*[@resource-id='com.ss.android.article.news:id/bob']",
                "action": "CLICK",
                "throttle": 2000
            },
            {
                "xpath": "//*[@resource-id='com.ss.android.article.news:id/gi'and @text='开直播']",
                "action": "CLICK",
                "throttle": 2000
            }
        ]
},
{
        "prob": 1,
        "activity":"com.ss.android.live.host.livehostimpl.LiveBroadcastBeforeActivity",
        "times": 1,
        "actions": [
            {
                "xpath": "//*[@resource-id='com.ss.android.article.news:id/startlive_preview_btn']",
                "action": "CLICK",
                "throttle": 2000
            }
        ]
},
```

iv. 编写好文件后，在json.cn中检查无误后，推送到手机端中

```
adb push 路径+max.xpath.actions /sdcard   #mac直接拖动文件即可
```

v. 有用的经验：

- 包名的获取方式（需要配置好ADB命令）：

```
aapt dump badging .apk文件   #mac系统直接拖动apk文件
```

```
localhost% aapt dump badging /Users/luyizhi/Documents/Fastbot[redacted]/头条
/15317-TT_Android_Multi_SCMs_v4.apk
package: name='com.ss.android.article.news' versionCode='[redacted]' versionName='[redacted]
6'
sdkVersion:'21'
targetSdkVersion:'28'
uses-permission: name='com.android.launc[redacted]'
uses-permission: name='com.and[redacted]her3.[redacted]'
uses-permission: name='android.per[redacted]'
uses-permission: name='android.perm[redacted]E'
```

- 使用Maxim获取当前控件所属的Activity

```
adb shell CLASSPATH=/sdcard/monkey.jar:/sdcard/framework.jar exec app_process
/system/bin tv.panda.test.monkey.api.CurrentActivity
```

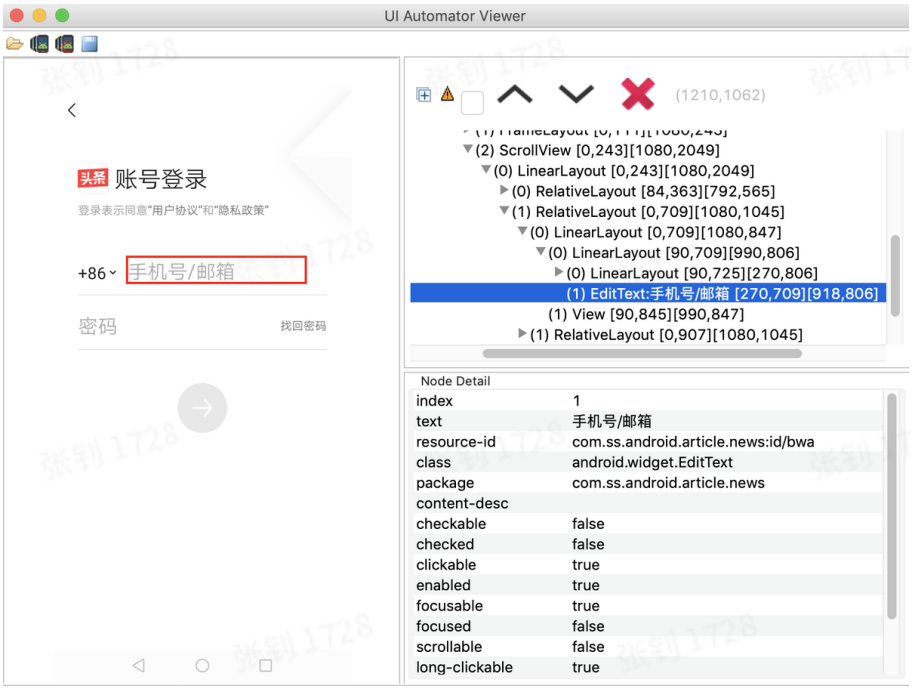
头条登录页面的所属activity

```
localhost% adb shell CLASSPATH=/sdcard/monkey.jar:/sdcard/framework.jar exec app
_process /system/bin [redacted].monkey.api.CurrentActivity
[Fastbot] current activity:
[Fastbot] // com.ss.android.account.v3.view.NewAccountLoginActivity
```

- 定位当前页面的控件

a. 使用Android SDK自带的页面属性检查工具UiAutomatorViewer（需提前配置好Android SDK）

```
~/Library/Android/sdk/tools//bin/uiautomatorviewer
```



b. 使用Maxim在终端查看当前Tree结构

```
adb shell CLASSPATH=/sdcard/monkey.jar:/sdcard/framework.jar exec app_process /system/bin tv.panda.test.monkey.api.Dumptree
```

```
le="false" password="false" scroll-type="false" bounds="[90,725][270,806]">
          <node index="0" text="+86" resource-id="com.ss.android.article.news:
id/ai5" class="android.widget.TextView" package="com.ss.android.article.news" co
ntent-desc="" checkable="false" checked="false" clickable="false" enabled="true"
 selected="false" focusable="false" focused="false" scrollable="false" long-clic
kable="false" password="false" scroll-type="false" bounds="[90,725][192,806]"/>
        </node>
        <node index="3" text="手机号/邮箱" resource-id="com.ss.android.article
.news:id/bwa" class="android.widget.EditText" package="com.ss.android.article.ne
ws" content-desc="" checkable="false" checked="false" clickable="true" enabled="
true" selected="false" focusable="true" focused="false" scrollable="false" long-
clickable="true" password="false" scroll-type="false" bounds="[270,709][918,806]
"/>
        <node index="4" text="" resource-id="com.ss.android.article.news:id/bw
h" class="android.widget.RelativeLayout" package="com.ss.android.article.news" c
ontent-desc="" checkable="false" checked="false" clickable="false" enabled="true
" selected="false" focusable="true" focused="false" scrollable="false" long-clic
kable="false" password="false" scroll-type="false" bounds="[0,907][1080,1045]">
          <node index="0" text="密码" resource-id="com.ss.android.article.news
:id/bwi" class="android.widget.EditText" package="com.ss.android.article.news" c
ontent-desc="" checkable="false" checked="false" clickable="true" enabled="true"
 selected="false" focusable="true" focused="false" scrollable="false" long-click
able="true" password="false" scroll-type="false" bounds="[90,907][630,1004]"/>
          <node index="1" text="找回密码" resource-id="com.ss.android.article.
news:id/bwk" class="android.widget.TextView" package="com.ss.android.article.new
```

- 尽量使用resource-id作为xpath路径，也可以组合使用比如"xpath": "//*[@resource-id='xxx'and @text='xx']"

c. 场景细粒度控制

> 手动配置黑、白名单配置
> 适用需求：单独覆盖几个场景或屏蔽一些不必要场景

i. Activity白名单配置（只覆盖白名单内的activity）

- 在PC端新建awl.strings文件（名称固定为：awl.strings）
- 在文件中写入Activity的名称，例如

```
com.panda.videoliveplatform.activity.WelcomeActivity
com.panda.videoliveplatform.activity.SplashWakeActivity
com.panda.videoliveplatform.activity.MainFragmentActivity
com.panda.videoliveplatform.activity.LiveRoomActivity
```

- 将awl.strings文件push到手机端的sdcard目录下

```
adb push awl.strings  /sdcard # 目录必须为sdcard
```

- 运行命令时添加以下参数：--act-whitelist-file /sdcard/awl.strings

```
adb -s 设备号 shell CLASSPATH=/sdcard/monkeyq.jar:/sdcard/framework.jar exec
app_process /system/bin com.android.commands.monkey.Monkey -p 包名 --agent robot
      --act-whitelist-file /sdcard/awl.strings --running-minutes 遍历时长 --
throttle 事件频率 -v -v # 标红部分为添加白名单配置的参数
```

ii. Activity黑名单配置（黑名单内的activity不覆盖）

- 在PC端新建abl.strings文件（名称固定为：abl.strings）
- 在文件中输入Activity的名称，同白名单方法一致
- 将abl.strings文件push到手机端的sdcard目录下

```
adb push abl.strings  /sdcard # 目录必须为sdcard
```

- 运行命令时添加以下参数：--act-blacklist-file /sdcard/abl.strings

```
adb -s 设备号 shell CLASSPATH=/sdcard/monkeyq.jar:/sdcard/framework.jar exec
app_process /system/bin com.android.commands.monkey.Monkey -p 包名 --agent robot
      --act-blacklist-file /sdcard/abl.strings --running-minutes 遍历时长 --
throttle 事件频率 -v -v # 标红部分为添加黑名单的参数
```

*注意：白名单和黑名单不能同时设置，按照非白即黑的原则，即设置了白名单则白名单外的都为黑名单。通过hook 可以监控activity 启动和切换，如果启动的是黑名单中的activity，就拒绝启动该activity，从ui上看就是点了跳转没效果。

d. 屏蔽控件或区域

> 手动配置需要屏蔽的控件或区域
> 适用需求：测试过程中“半路”中途退出登录，屏蔽退出登录按钮

i. 黑控件、黑区域

- 在PC端新建max.widget.black文件（名称固定为：max.widget.black）

- 文件内容配置格式如下：
  - 配置activity：当activity与currentactivity一致时执行如下匹配
  - 屏蔽控件或区域共有三种方式：
    - 配置bounds：屏蔽某个区域，在该区域内的控件或坐标不会被点击。
    - 配置xpath：查找匹配的控件，屏蔽点击该控件。
    - 配置xpath+bounds：查找匹配的控件，当控件存在时屏蔽指定的区域。

以根据xpath屏蔽单独控件为例

- 将max.widget.black文件push到手机端的sdcard目录下

```
adb push max.widget.black /sdcard # 目录必须为sdcard
```

ii. 树剪枝屏蔽

- 在PC端新建max.tree.pruning文件（名称固定为：max.tree.pruning）
- 文件内容配置格式如下：
  - 配置activity：当activity与currentactivity一致时执行如下匹配
  - 剪枝方式：
    - 配置xpath：查找匹配的控件，改变控件属性，从而使控件屏蔽

- 将max.tree.pruning文件push到手机端的sdcard目录下

```
adb push max.tree.pruning  /sdcard # 目录必须为sdcard
```

*注意：剪枝屏蔽效率更高，但无法作用于fuzzaction，通常控件区域屏蔽需同时配置黑控件及树剪枝

e. **支持反混淆**

> 手动配置反混淆文件，针对每个包的混淆xpath做处理
> 适用需求：对黑、白名单、屏蔽控件和自定义事件中的xpath做反混淆转换

- 配置混淆映射文件并push到手机端sdcard中

```
adb push resguard_mapping_NewsArticle_beta_version_v7.2.x_?????.txt /sdcard
```

- 配置反混淆文件，以自定义事件为例，在max.xpath.actions中配置**混淆前**的id
- 运行命令时添加以下参数：--resMapping-file/sdcard/混淆映射文件

```
adb -s 设备号 shell CLASSPATH=/sdcard/monkeyq.jar:/sdcard/framework.jar exec app_process /system/bin com.android.commands.monkey.Monkey -p 包名 --agent robot --resMapping-file /sdcard/resguard_mapping_NewsArticle_beta_version_v7.2.x_?????.txt --running-minutes 遍历时长 --throttle 事件频率 -v -v # 标红部分为添加反混淆功能的参数
```

f. 高速截图及打印xml结构

> 保存测试过程中的截图和打印xml结构
> 适用需求：观察测试过程中的截图

- 高速截图保存条件为：
- 在PC端新建max.config文件，增加以下属性
  - max.takeScreenshot = true
  - max.takeScreenshotForEveryStep = true
  - max.saveGUITreeToXmlEveryStep =true
- 将max.config文件push到手机端sdcard中

```
adb push max.config  /sdcard # 目录必须为sdcard
```

- 目录默认保存为手机端sdcard中，如需改变保存位置，在执行命令末尾添加 --output-directory 指定路径

```
adb -s 设备号 shell CLASSPATH=/sdcard/monkeyq.jar:/sdcard/framework.jar exec app_process /system/bin com.android.commands.monkey.Monkey -p 包名 --agent robot --running-minutes 遍历时长 --throttle 事件频率 -v -v --output-directory 指定路径 # 标红部分为保存截图指定位置的参数
```

注：--throttle参数要>200才会截图

g. Schema Event支持

> app需支持允许第三方通过intent方式执行Schema跳转

i. Schema Event （schema跳转）

- 在PC端新建max.schema文件，例如⬇

```
snssdk32://detail?groupid=[redacted]2139
snssdk32://pgcprofile?user_id=83544[redacted]
snssdk32://search?keyword=[redacted]
snssdk32://play_[redacted]
```

- 将max.schema文件push到手机端的sdcard目录下

```
adb push max.schema  /sdcard # 目录必须为sdcard
```

- schema事件默认会在App启动后执行
  - 配置max.config 增加
    - max.execSchema = true
    - max.execSchemaEveryStartup = true 每次启动app先执行schema

h. 权限自动授予

app的权限弹窗处理

默认启动app前会自动授予app所需的所有权限，但如果想测试app运行过程中的动态权限弹窗在max.config配置

```
max.grantAllPermission = false
```

Fastbot启动后不会自动授予各种权限；

shell中增加

```
-p com.android.packageinstaller
-p com.android.permissioncontroller
-p com.lbe.security.miui (miui android 10)
-p com.samsung.android.permissioncontroller (samsung android 10)
```

增加其一弹窗相关package，可在权限弹窗时关闭弹窗；

```
[Fastbot] *** DEBUG *** com.ss.android.ugc.aweme requrested permission android.permission.BROADCAST_PACKAGE_CHANGED
[Fastbot] *** DEBUG *** com.ss.android.ugc.aweme requrested permission android.permission.BROADCAST_PACKAGE_INSTALL
[Fastbot] *** DEBUG *** com.ss.android.ugc.aweme requrested permission android.permission.BROADCAST_PACKAGE_REPLACED
[Fastbot] *** DEBUG *** com.ss.android.ugc.aweme requrested permission android.permission.ACCESS_BACKGROUND_LOCATION
[Fastbot] *** DEBUG *** com.ss.android.ugc.aweme requrested permission android.permission.ACCESS_MEDIA_LOCATION
[Fastbot] [1602232506692] // zhangzhao debug, evolve= false
[Fastbot] [1602232506819]

[Fastbot] *** DEBUG *** The top activity package com.android.permissioncontroller is not allowed.
[Fastbot] [1602232506871] fastbot exec shell:
[Fastbot] *** INFO *** Let's wait for activity loading...
[Fastbot] [1602232506877] fastbot exec schema:
[Fastbot] [1602232506878] Switch: #Intent;action=android.intent.action.MAIN;category=android.intent.category.LAUNCHER;launchFlags=0x102
[Fastbot] [1602232526904]     // Allowing start of Intent { act=android.intent.action.MAIN cat=[android.intent.category.LAUNCHER] flg=0
gUid=2000 } in package com.ss.android.ugc.aweme
[Fastbot] *** INFO *** We are still waiting for activity loading. Let's wait for another 100ms...
```

常见问题

1. 本地测试时，手机的顶部状态栏找不到了，怎么恢复呢?

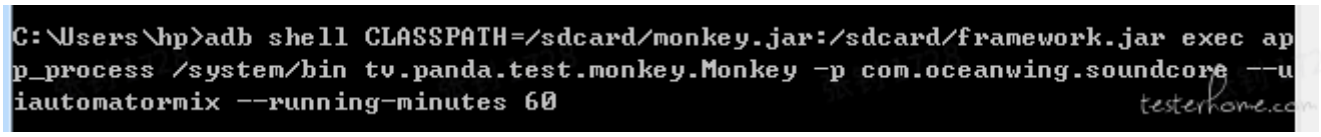
```
adb shell wm overscan reset # ps:为了防止测试时点击到设置，影响测试效果，做的特殊设置
```

2. 小米手机运行Fastbot报错?

答：开启 “开发者选项” -> "USB调试（安全设置）允许通过usb调试修改权限或模拟点击"

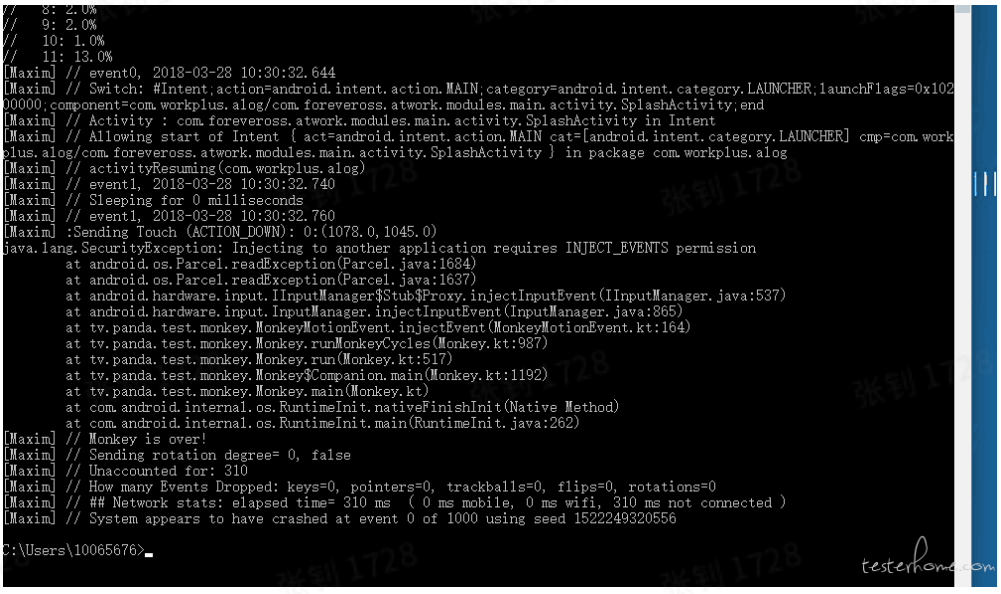
3. 运行Fastbot时无任何log，启动后就退出?

答：需检查/sdcard/是否存在monkey.jar ，framework.jar。部分机型发现adb push过去monkey.jar自动被更名成monkey. 导致无法运行。



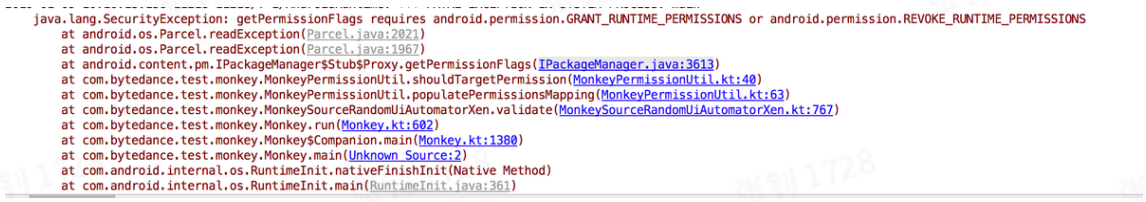
4. vivo7.1运行Fastbot报错?

答：关闭锁屏和开启usb模拟点击即可。



5. oppo运行Fastbot 1.0模式报错?

答：oppo存在权限监控，需要在开发者-> 开启 禁止权限监控 即可



```
java.lang.SecurityException: getPermissionFlags requires android.permission.GRANT_RUNTIME_PERMISSIONS or android.permission.REVOKE_RUNTIME_PERMISSIONS
    at android.os.Parcel.readException(Parcel.java:2021)
    at android.os.Parcel.readException(Parcel.java:1967)
    at android.content.pm.IPackageManager$Stub$Proxy.getPermissionFlags(IPackageManager.java:3613)
    at com.bytedance.test.monkey.MonkeyPermissionUtil.shouldTargetPermission(MonkeyPermissionUtil.kt:40)
    at com.bytedance.test.monkey.MonkeyPermissionUtil.populatePermissionsMapping(MonkeyPermissionUtil.kt:63)
    at com.bytedance.test.monkey.MonkeySourceRandomUiAutomatorXen.validate(MonkeySourceRandomUiAutomatorXen.kt:767)
    at com.bytedance.test.monkey.Monkey.run(Monkey.kt:602)
    at com.bytedance.test.monkey.Monkey$Companion.main(Monkey.kt:1380)
    at com.bytedance.test.monkey.Monkey.main(Unknown Source:2)
    at com.android.internal.os.RuntimeInit.nativeFinishInit(Native Method)
    at com.android.internal.os.RuntimeInit.main(RuntimeInit.java:361)
```