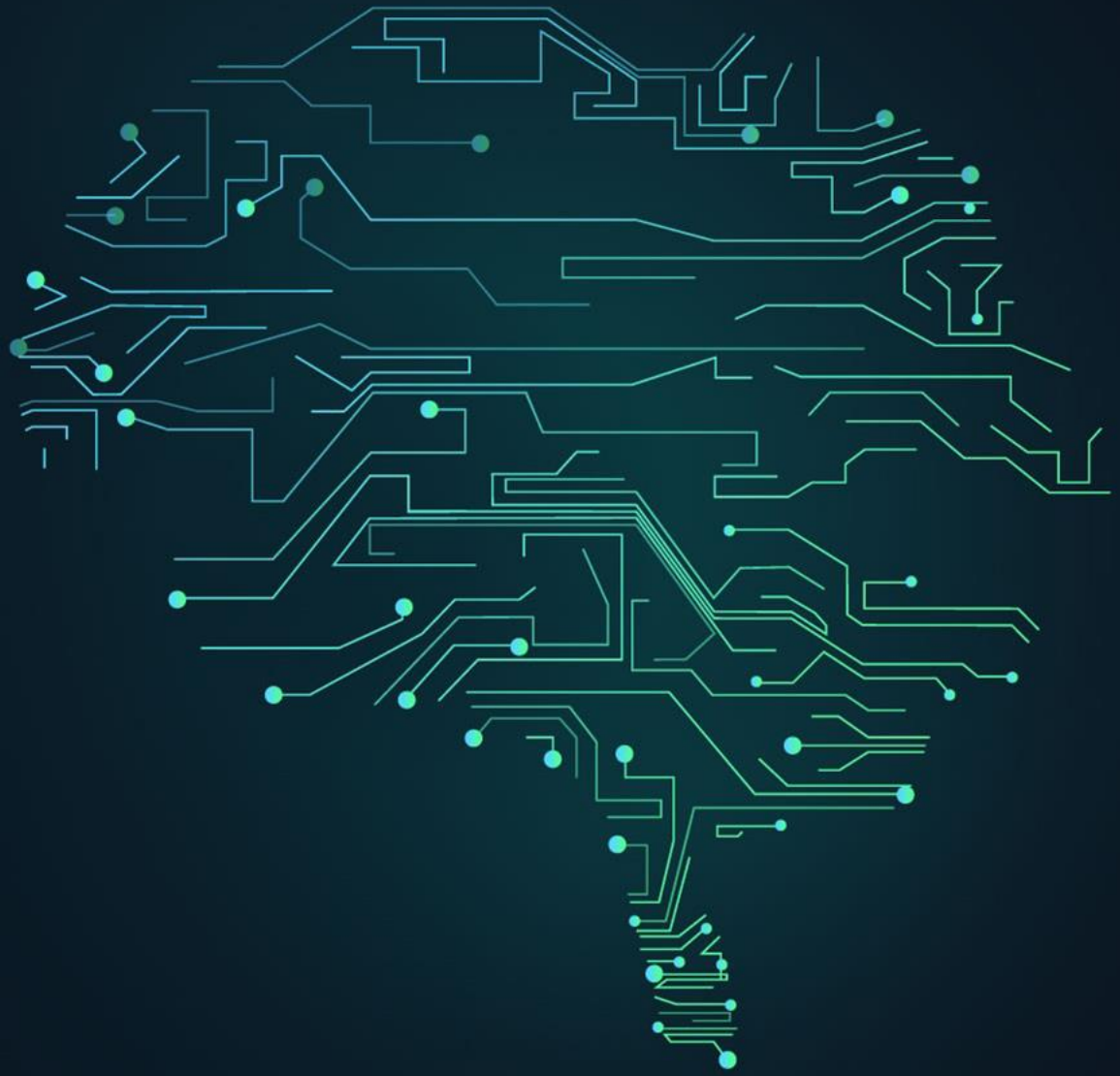
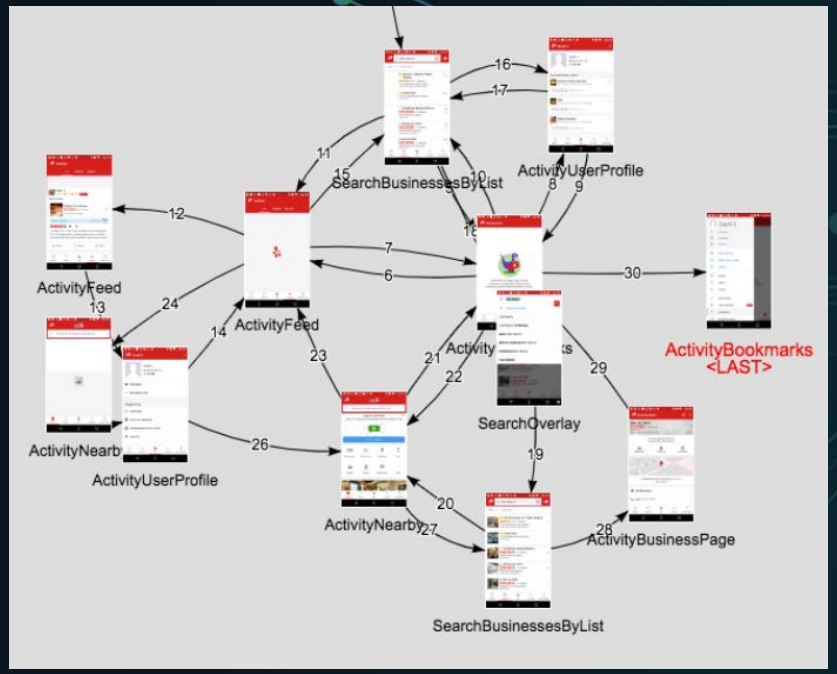
# 深度学习在GUI测试中的应用



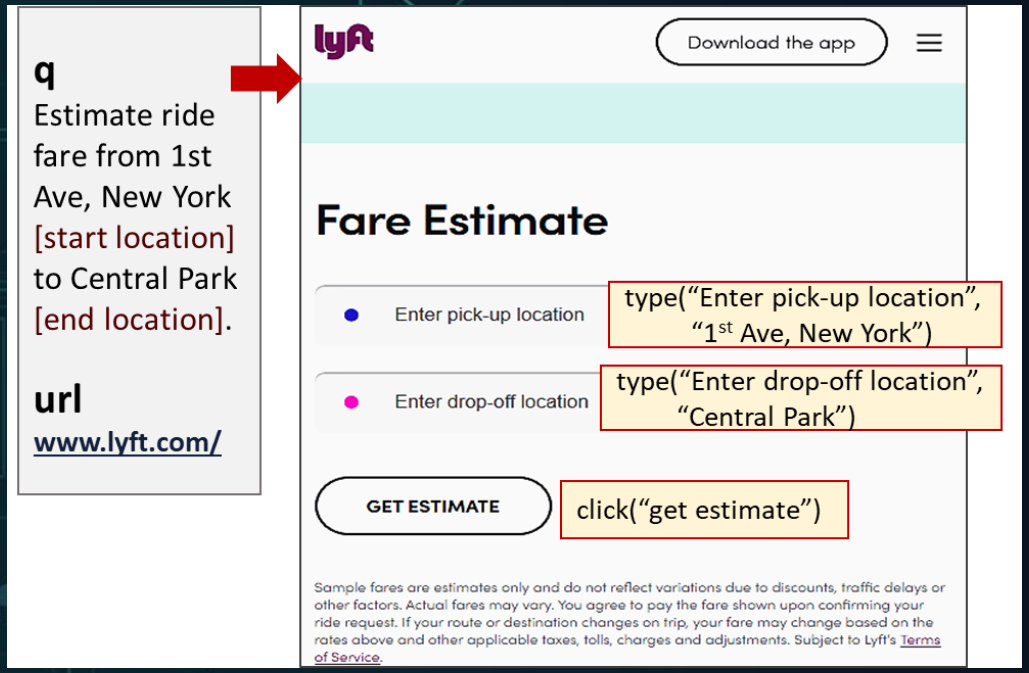
主讲人：李元春
北京大学、微软亚洲研究院

# 背景：GUI测试

基于软件的图形界面（GUI），生成交互动作序列（如点击、文本输入、拖拽、滑动等）作为测试用例。




遍历测试：探索软件GUI，在有限时间内获取更高的代码覆盖率、触发更多bug




功能测试：生成GUI动作序列，测试软件的特定功能是否正常

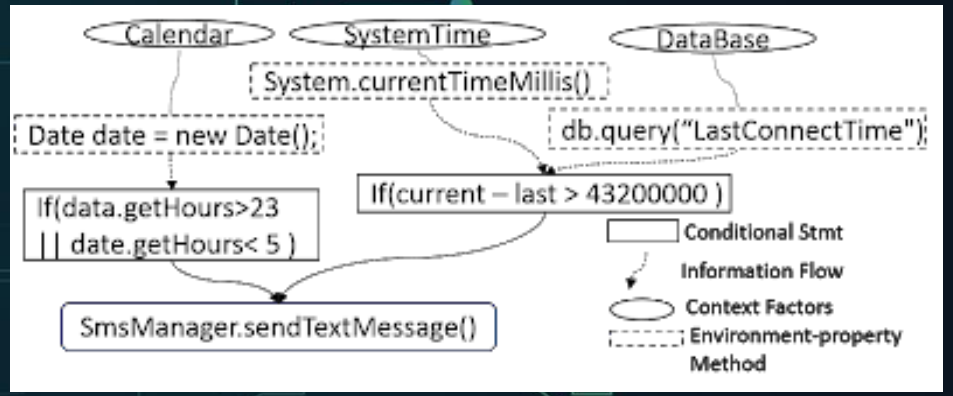
# 目录
Contents

01 章节 PART

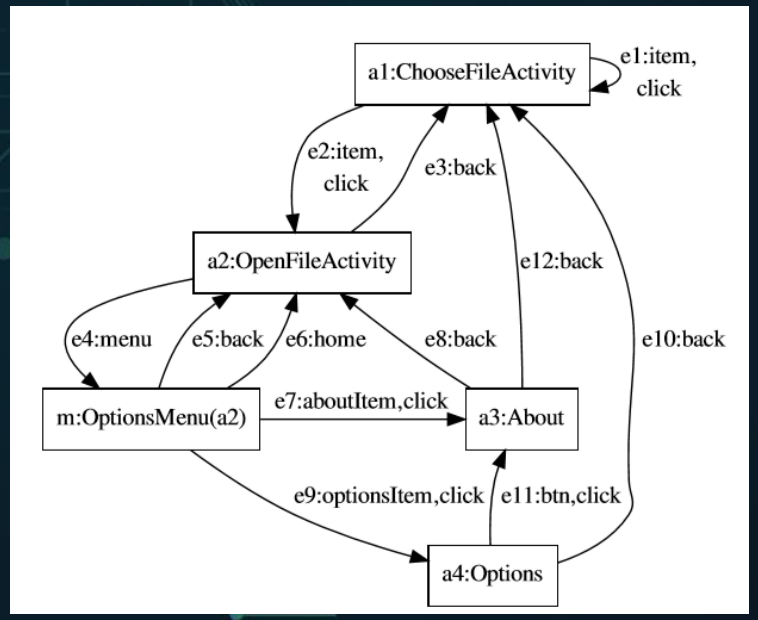
# 基于深度学习的移动应用GUI遍历测试

# 背景：Android应用GUI遍历测试

- 随机策略
  - 生成随机交互动作
    - Monkey, DynoDroid, DroidFuzzer
- 基于模型的策略
  - 模型用于定向探索未知状态
    - AppContext, Gator, ACTEve
  - 模型用于记忆已探索状态
    - MobiGUITAR, DroidBot, DroidMate




[AppContext, ICSE 2015]




[Gator, ASE 2015]

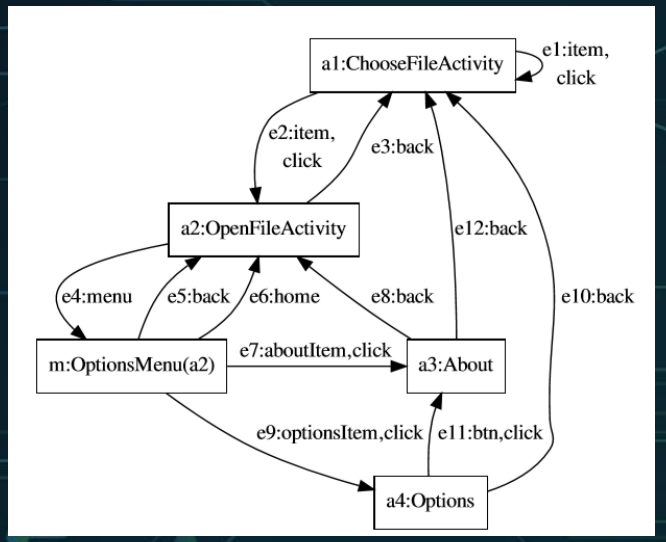

[DroidBot, ICSE 2017]

# GUI遍历测试方法对比



**随机探索**

+ 简单易用，输入速度快
- 大量无效、冗余输入



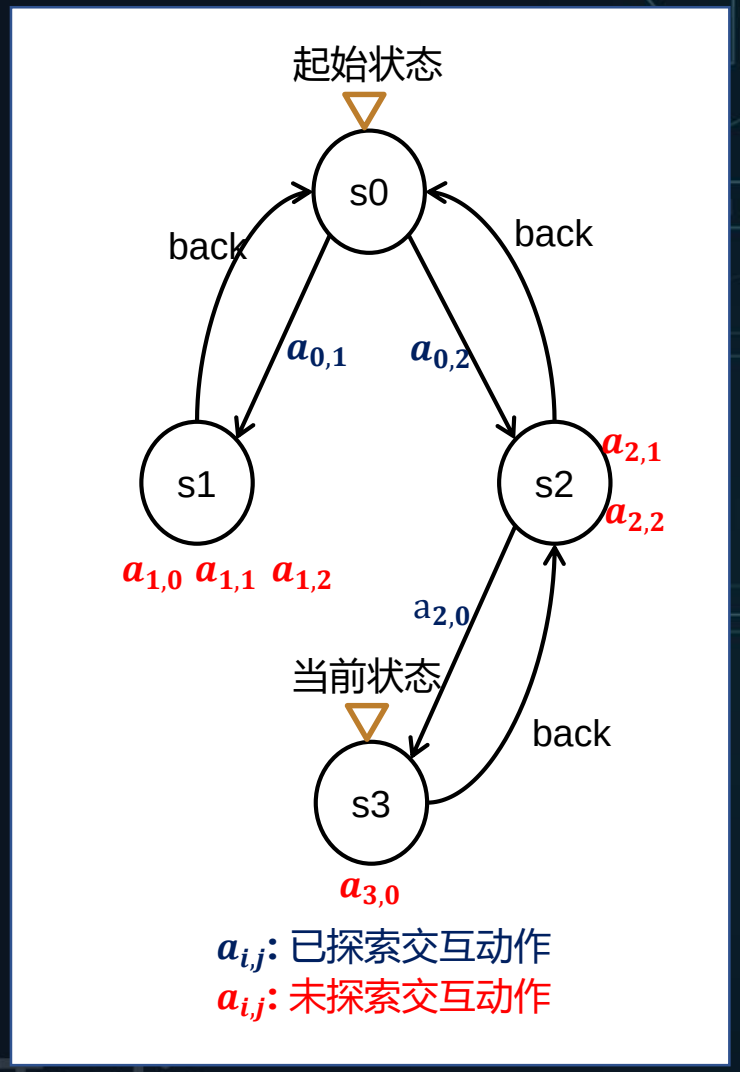

**基于模型**
**模型用于分析目标状态可达路径**

+ GUI探索效率高
- 需要分析代码，易用性差


**基于模型**
**模型仅记忆已探索状态**

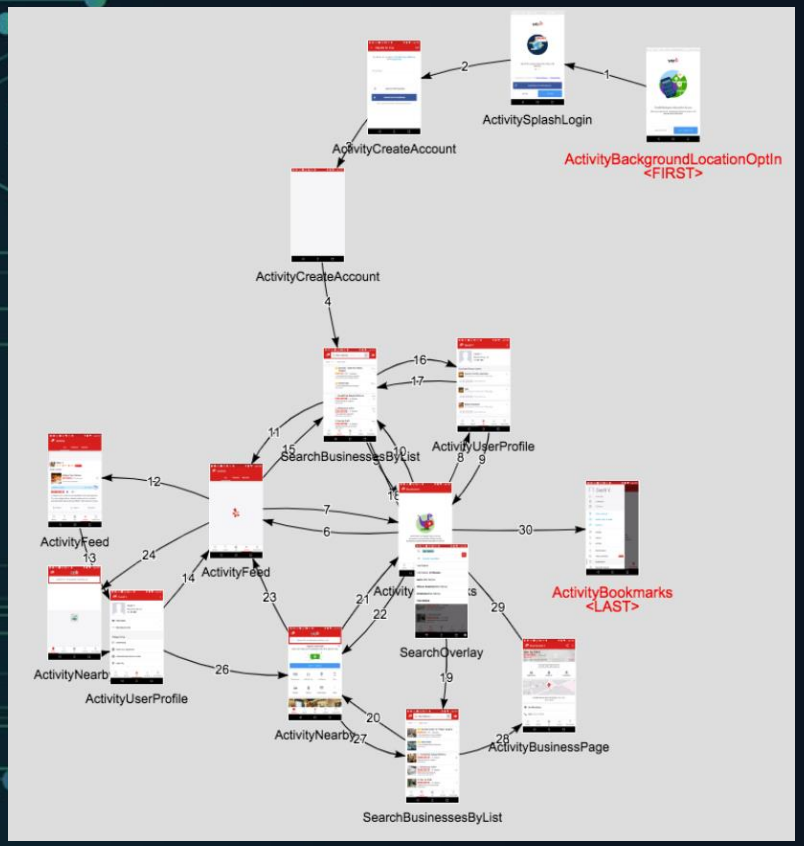
+ 可以避免大部分冗余输入
+ 不需要代码分析，易用性高
**（主流方法）**

# 界面转换图（UTG）模型



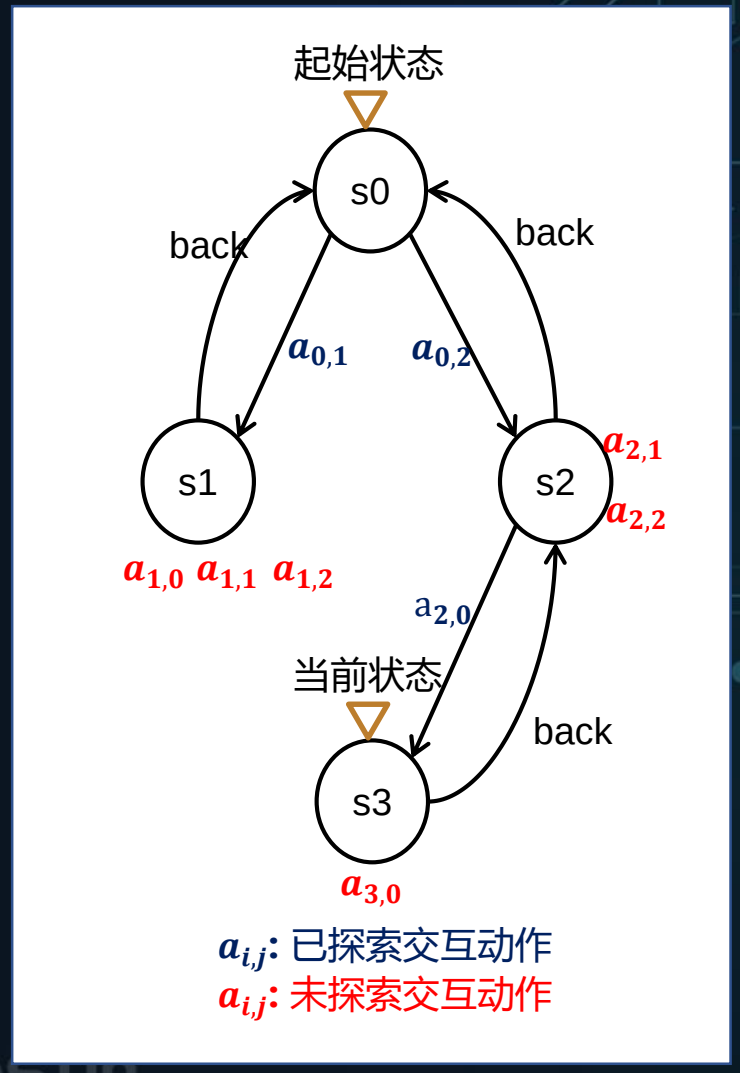

将应用已探索界面建模为有向图G

- 节点：界面状态
  - UI树
  - 可接受动作集合
- 边：交互动作
  - 动作类型（click/type/scroll/...）
  - 动作位置（x, y）或目标界面元素
  - 动作值（输入文本、选择id等）




实例：DroidBot生成的UTG

# 基于界面转换图的GUI测试策略



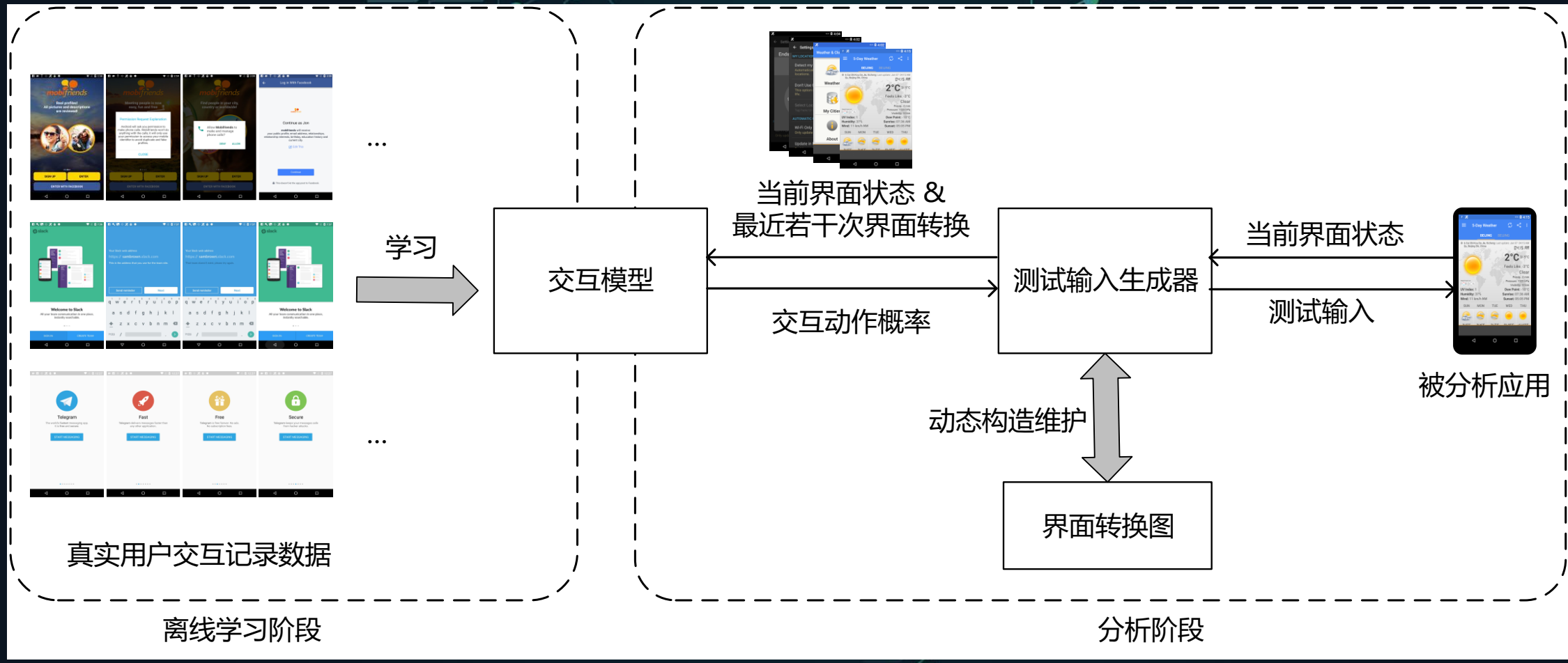

- 策略目标
  - 基于当前界面转换图，选择一个交互动作，使得执行该交互动作的未来收益（覆盖率、触发bug数量等）最大
    - “不确定状态空间搜索”问题
- 常用策略
  - 深度优先
  - 宽度优先
  - 随机探索
  - 启发式规则
- **GUI探索效率低**

# 机会和挑战



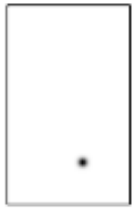
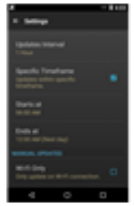
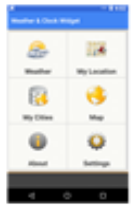
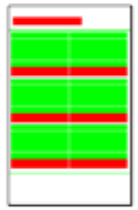
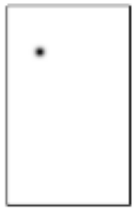
- 观察
  - 终端用户和测试人员尽管没有应用代码相关信息，却能对应用进行高效的探索测试
- 原因
  - GUI交互存在通用模式：很多应用有相似的功能，很多功能和UI组件的交互方式类似
  - 用户和测试人员有大量使用应用的经验
- 机会
  - 如何用人工智能自动学习交互模式知识，进而将学习到的模式应用到自动化测试中，提高GUI探索效率
- 挑战
  - 如何表示GUI状态和动作
  - 如何设计模型学习GUI状态与动作的对应模式

# Humanoid: 基于深度学习的拟人交互输入策略



# Humanoid: 交互界面状态和动作的表示



| Original data | | Extracted feature | |
|---|---|---|---|
| State | Action | UI skeleton | Action heatmap |
|  | Touch "CANCEL" |  |  |
|  | Touch "Back" |  |  |
|  | Touch "Weather" |  |  |
|  | NULL |  |  |

- 假设界面大小为w*h
- GUI状态s
  - 表示为UI结构图，shape=(w, h, 2)
    - 信道1（红色）：如果UI对应位置为文字，设为1，否则为0
    - 信道2（绿色）：如果UI对应位置为图片，设为1，否则为0
  - 忽略了具体文本和图像信息，便于泛化
- GUI交互动作a
  - 动作类型：shape=(7,)，表示交互动作a属于7类中每一类的概率
  - 动作位置：shape=(w, h, 1)，表示交互动作a发生在UI对应位置的概率
    - Trick：训练时将孤立的点转换为以该点为中心的概率分布，做平滑处理

# Humanoid: 深度学习模型的输入和输出


| Action Type | UI Element | Probability |
|---|---|---|
| touch | ☰ | 0.7 |
| touch | ⋮ | 0.15 |
| touch | HAIDIAN QU | 0.1 |
| touch | ↻ | 0.02 |
| touch | ⋖ | 0.015 |
| touch | ◁ | 0.002 |
| touch | Instagram | 0.001 |
| touch | HAIDIAN QU | 0.001 |
| swipe_left |  | 0.001 |
| swipe_right | | 0.001 |
| long_touch | ☰ | 0.0005 |
| ... | ... | ... |

**输入：**

当前界面情境$context_i$，包含：

- 当前界面$s_i$
- 最近若干次交互

$s_{i-1}, a_{i-1}, s_{i-2}, a_{i-2}, s_{i-3}, a_{i-3}$

**交互模型**

**输出：**

交互位置概率分布

$$p_{loc}(x, y \mid context_i)$$

交互动作类型概率分布

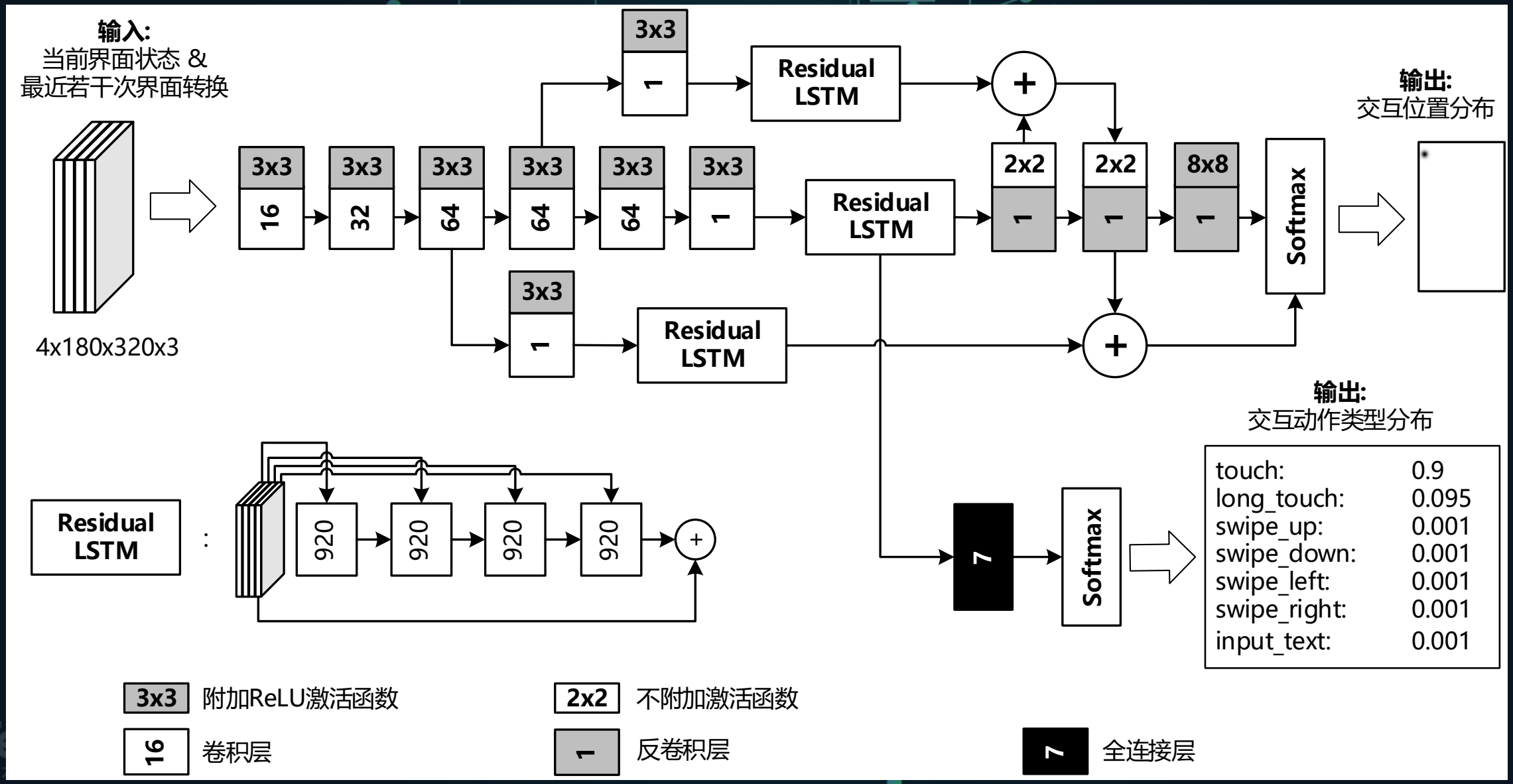
$$p_{type}(t \mid context_i)$$

每个交互动作的概率

$$p(action) = p_{type}(action.type) * \sum_{x,y \text{ in action.element}} p_{loc}(x, y)$$

# Humanoid: 深度学习模型结构





# Humanoid: 测试策略伪代码

**Algorithm 1** The test generation strategy of Humanoid.

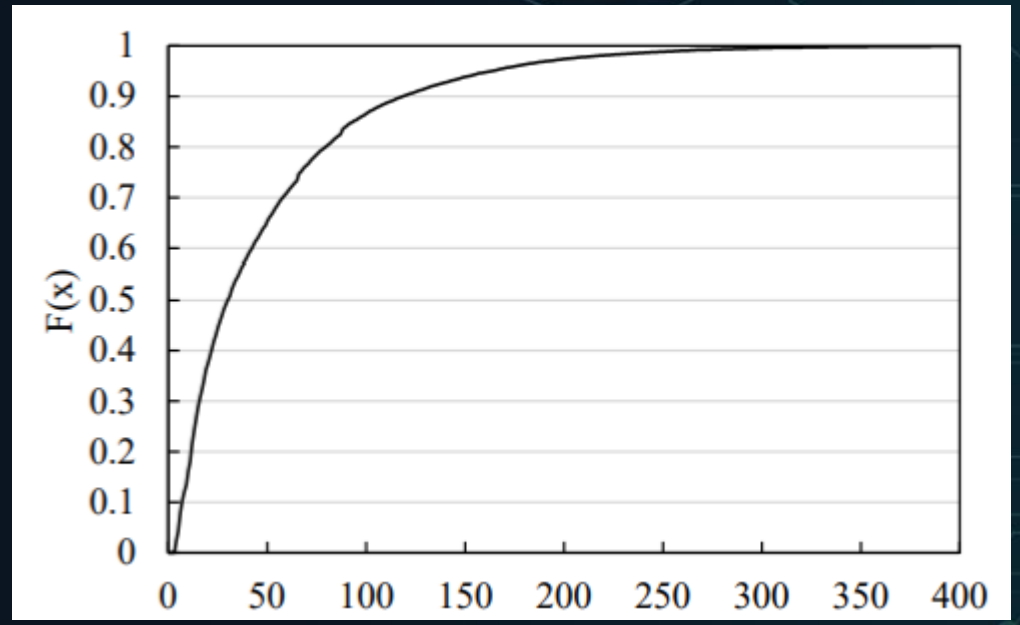
Load the pretrained interaction model $M$
Create an empty UTG $G =< S, E >$
Start the app under test
Observe current UI state $s$ and add $s$ to $S$
**repeat**
    Extract all unexplored actions in $s$ as $A$
    **if** $A$ is not empty **then**
        Choose an action $a$ from $A$ based on the probabilities predicted by $M$
    **else**
        Get a state $s'$ in $S$ that has unexplored actions
        Get the shortest path $p$ from $s$ to $s'$ in $G$
        Choose the first action in $p$ as $a$
    **end if**
    Perform action $a$
    Observe the new UI state $s_{new}$ and add $s_{new}$ to $S$
    Add the edge $< s, a, s_{new} >$ to $E$
**until** all actions in all states in $S$ have been explored

# Humanoid: 实验评估



- 在公开数据集上训练
  - Rico dataset：http://interactionmining.org/rico
  - 众包收集的交互记录，9.3k+应用, 10k+交互序列
- 与6个代表性工具对比
  - Monkey
  - PUMA
  - Sapienz
  - Stoat
  - DroidBot
  - DroidMate
- 在两个应用集合上测试
  - AndroTest评测集中的开源应用 68 个，每个测试1小时
  - Google Play流行应用 200 个，每个测试3小时

# Humanoid: 实验评估结果




每个UI状态中的action个数CDF图

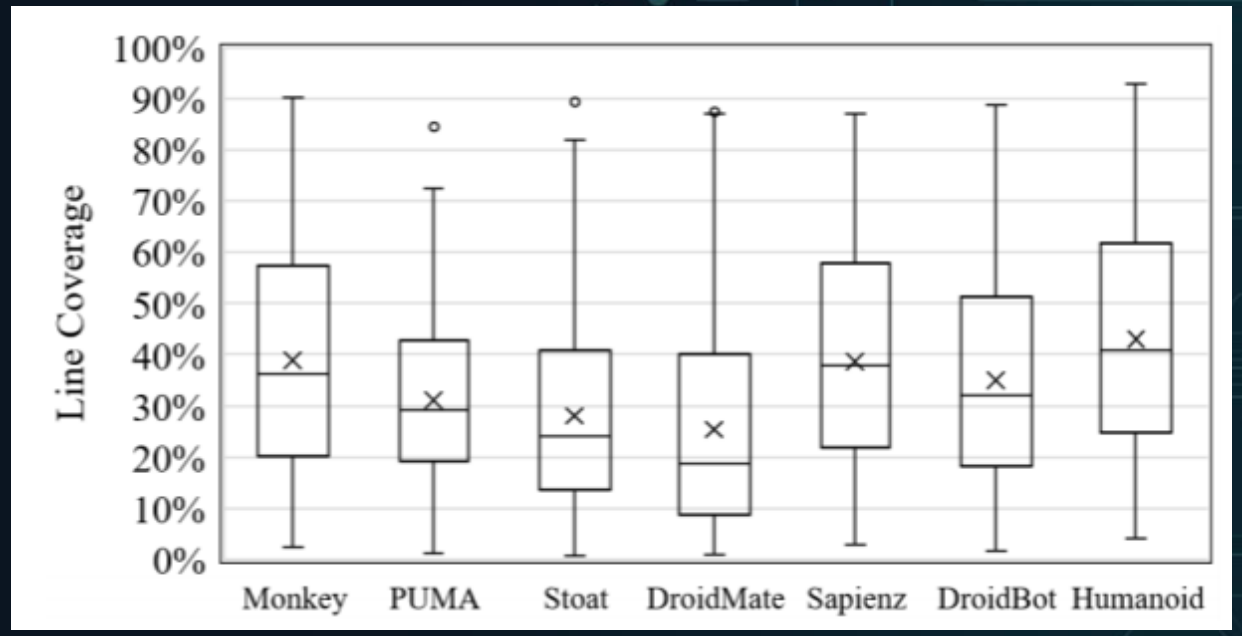
| N | Random top-N accuracy | Humanoid top-N accuracy |
|---|---|---|
| 1 | 5.8% | 51.2% |
| 3 | 17.5% | 67.6% |
| 5 | 29.1% | 74.8% |
| 10 | 58.3% | 85.2% |

Humanoid模型预测的action概率精确度

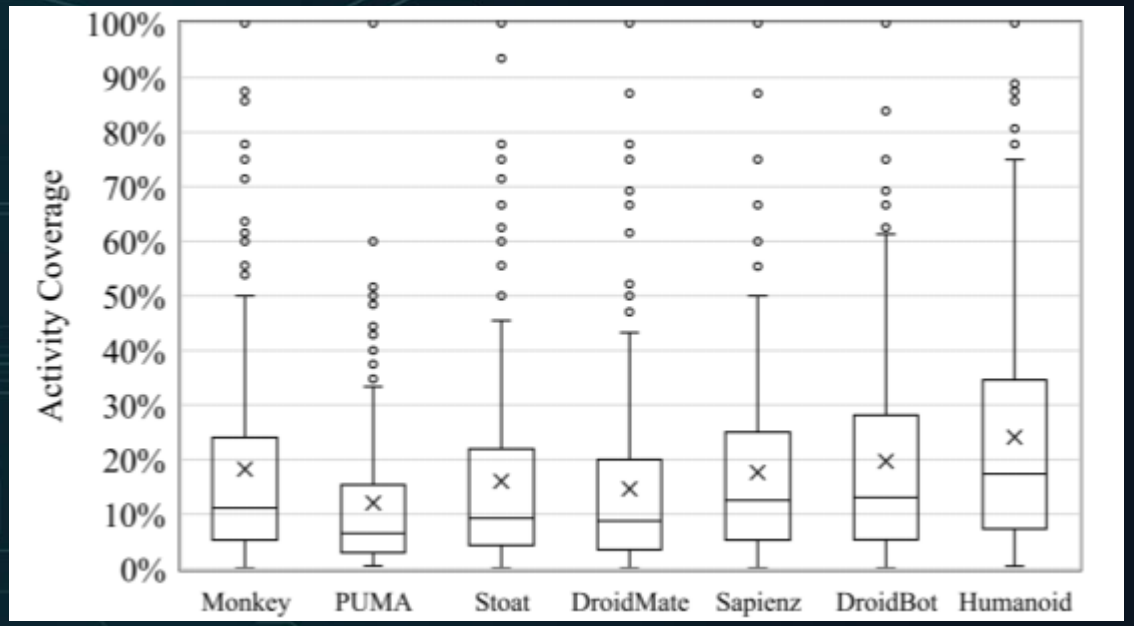
**Humanoid的模型可以有效地将用户的人工输入赋予更高的概率**

# Humanoid: 实验评估结果






各测试工具在68个开源应用中取得的代码行覆盖率

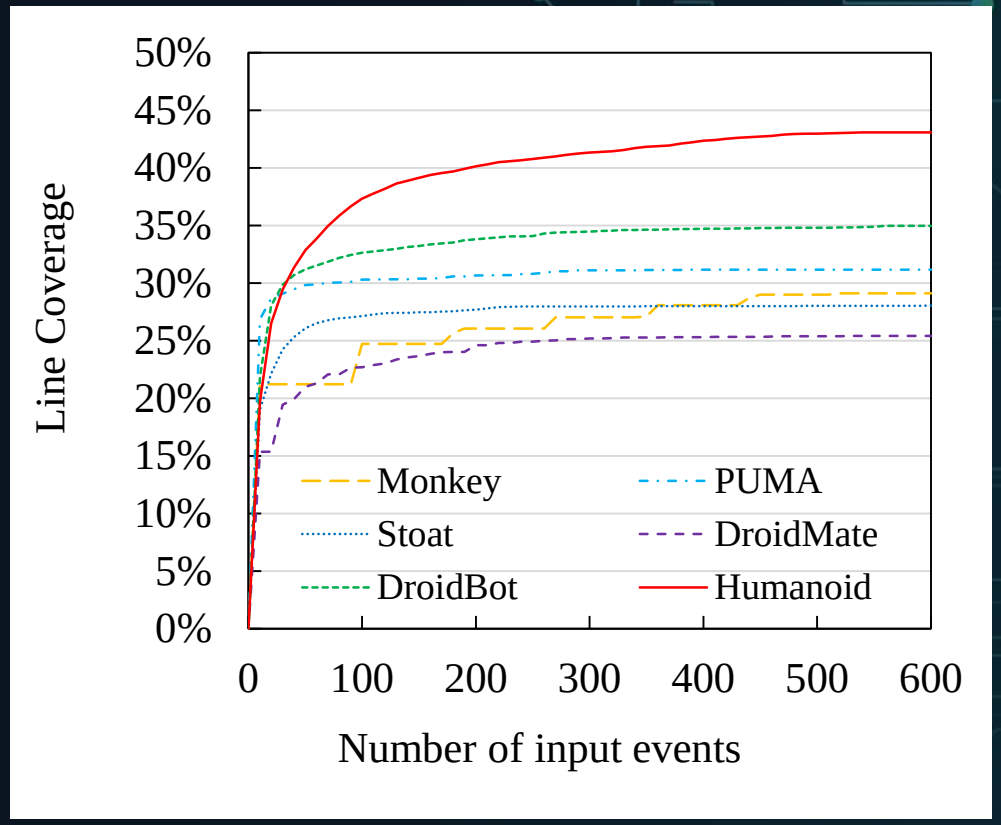



各测试工具在200个市场流行应用中取得的界面覆盖率

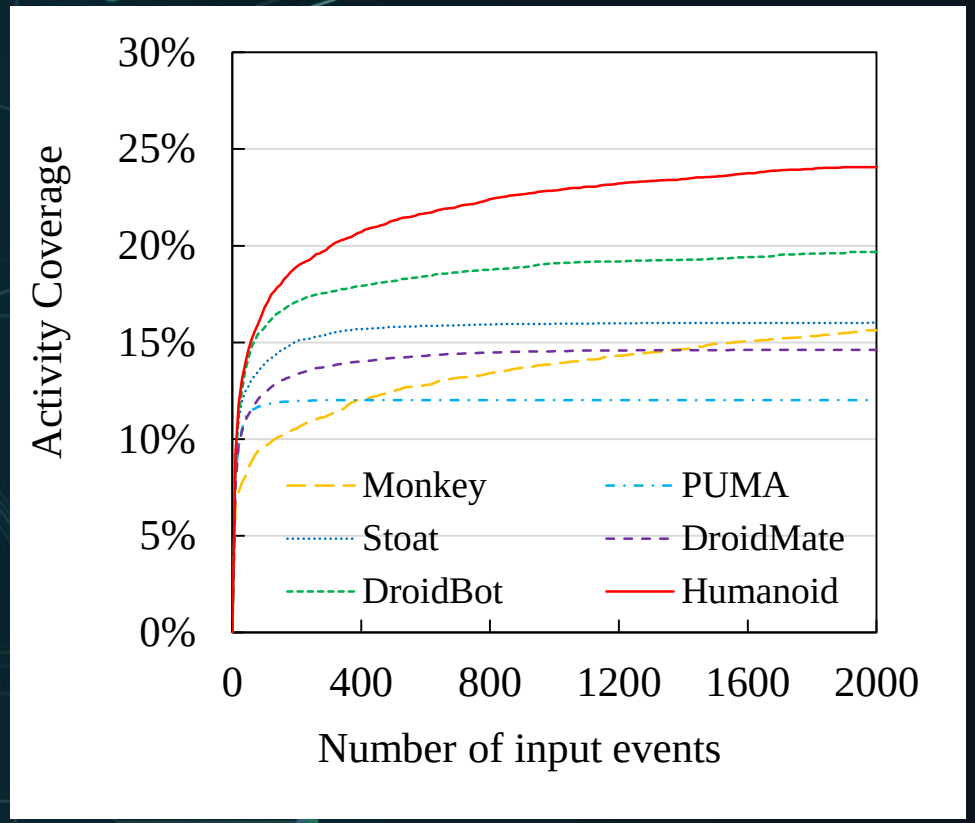
**Humanoid可以在开源应用和市场流行应用上达到更高的覆盖率**

# Humanoid: 实验评估结果






各测试工具对68个开源应用的代码覆盖速度




各测试工具对200个市场流行应用的界面覆盖速度

**Humanoid可以在开源应用和市场流行应用上达到更高的覆盖率**

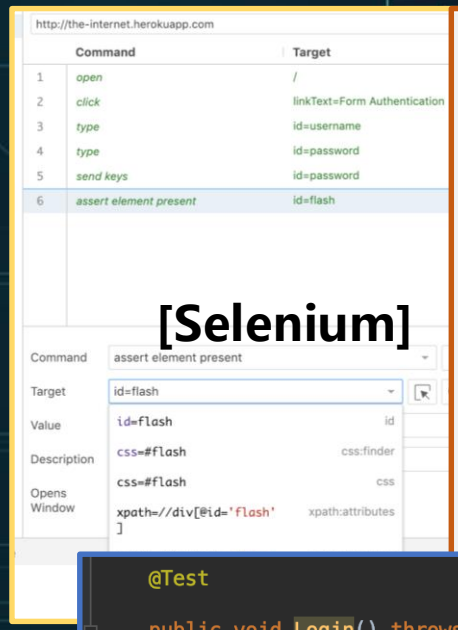
章节 PART 02

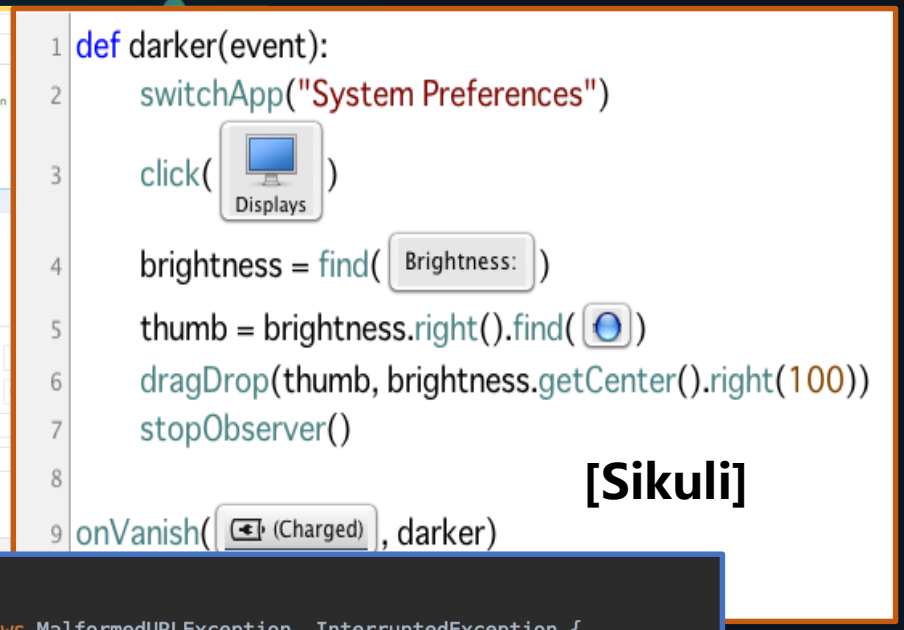
# 基于强化学习的GUI功能测试

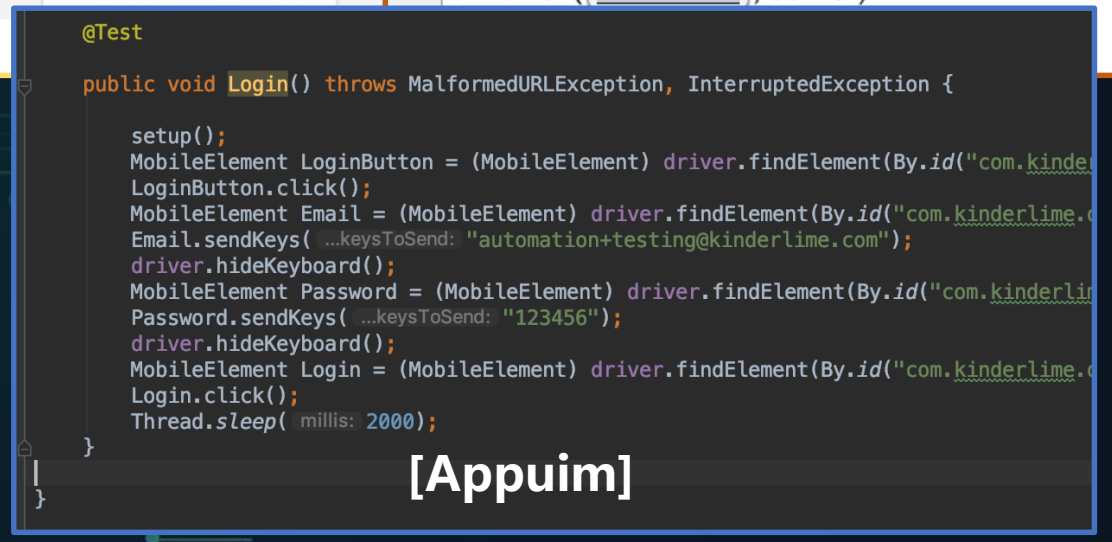
(work in progress)

# 背景：GUI功能测试

- 生成GUI动作序列，对软件的特定功能进行验证
- 例如，测试应用登录功能，须编写脚本：
  - open(<app_name>)
  - type(<input id=user>, ‘my@email.com’)
  - type(<input id=pwd>, ‘my_password’)
  - click(<button id=login>)
- 人工编写测试用例耗时耗力、维护困难；对普通测试人员有门槛



# GUI功能测试用例生成：问题描述



- 给定测试用例的自然语言描述：

**Lyft/Uber测试用例：**

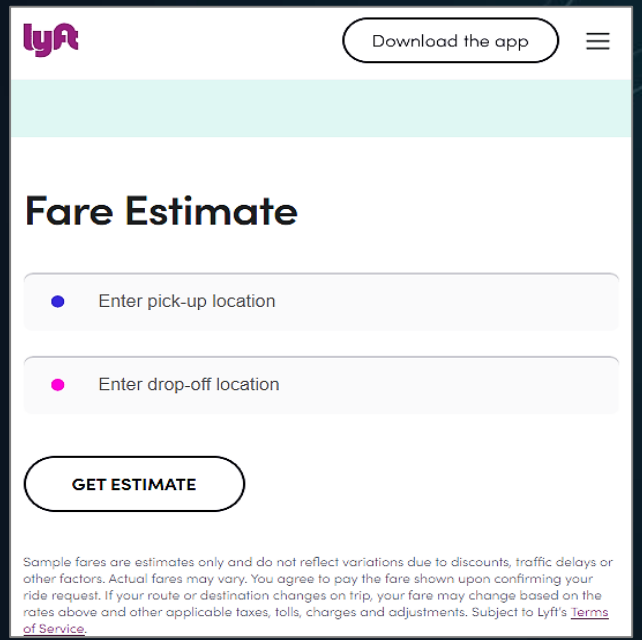
**Estimate ride fare from 1st Ave, New York [start location] to Central Park [end location]**

**param1 param1-annotation param2 param2-annotation**

估算从 纽约一号大街 [出发位置] 前往 中央公园 [目的地] 的打车费用

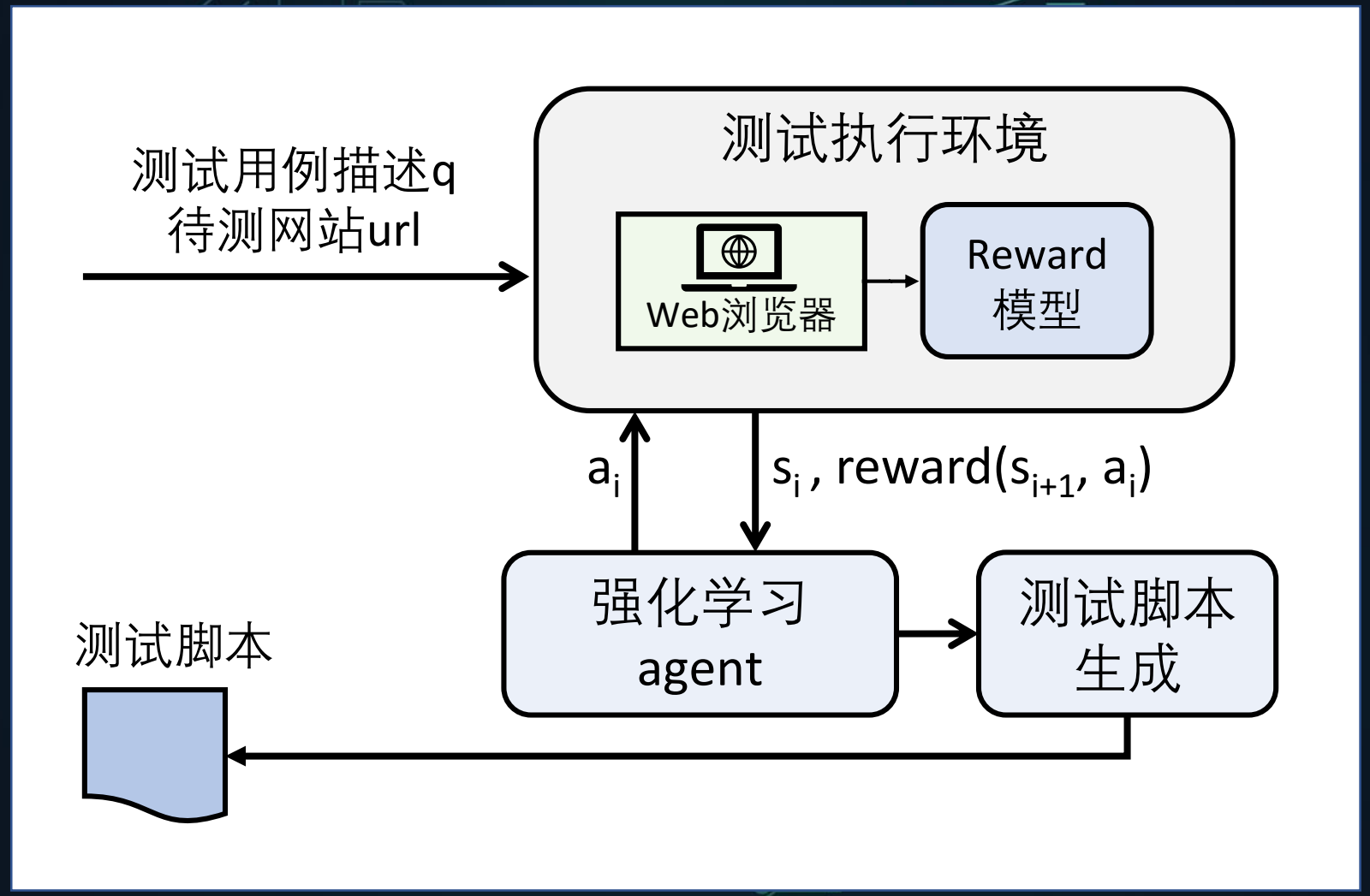
参数1 参数1标注 参数2 参数2标注

- 如何自动生成测试脚本（交互动作序列）：




type("1st Ave, New York") @ id("estimate")/div[1]/div[1]/…
enter @ id("estimate")/div[1]/div[1]/…
type("Central Park") @ id("estimate")/div[1]/div[1]/…
enter @ id("estimate")/div[1]/div[1]/…
click @ id("estimate")/div[1]/div[1]/…

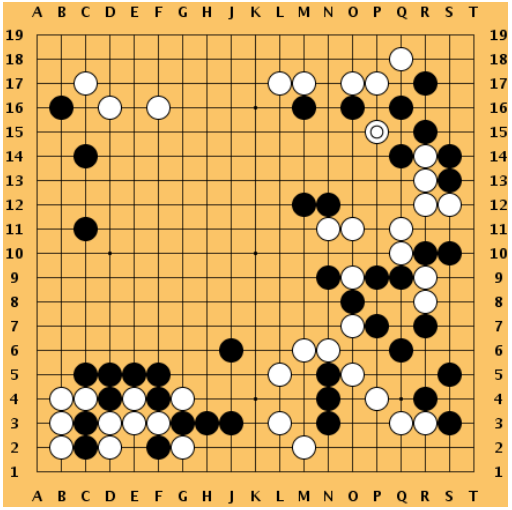
# 使用强化学习自动生成测试用例



# 强化学习: environment与agent



- 强化学习环境：environment
  - env.init()——返回初始状态$s_0$
  - env.getActionSet()——返回当前状态$s_i$中的动作集合$A_i$
  - env.step(a)——在当前状态$s_i$中执行动作$a_i$，返回新的状态$s_{i+1}$，奖励（reward）$r_{i+1}$
- 强化学习目标：agent
  - 通过不断与environment交互，争取最大化累计reward
  - 交互策略p(a|s)，表示在状态s中选择动作a的概率
  - Q-learning学习目标: Q(s, a)



围棋environment

- s: 19*19棋盘状态
- a: 落子位置<x,y>
- r: 胜利+1，否则0

# 强化学习环境reward的设计

- reward设计目标：为正确的动作序列赋予更高的reward
- 正确的动作序列$< a_0, s_0, a_1, s_1, \dots, a_i, s_i >$应与任务描述$q$匹配
  - 思路：动作与任务参数匹配；GUI文本与任务描述句子结构匹配；动作序列符合

$$R(q, w, as) = w_{dist} * num_{ld} + w_{dir} * num_{rd} + w_{task} * task_{sim} + w_{par} * \sum_{p \in P} par_{sim}(p) - k$$

鼓励动作序列空间距离接近

鼓励动作序列从左往右，从上到下

鼓励目标界面与任务描述文本相似

鼓励每个动作与任务参数匹配

惩罚动作序列过长

$$sim(t_1, t_2) = \max\{sim_{1-gram}(t_1, t_2), sim_{n-gram}(t_1, t_2), sim_{date}(t_1, t_2), sim_{number}(t_1, t_2)\}$$

$$sim_{1-gram}(t_1, t_2) = \begin{cases} \frac{max\{|t_1|, |t_2|\} - lev\_dist(t_1, t_2)}{max\{|t_1|, |t_2|\}} & t_1[0] = t_2[0] \\ 0 & t_1[0] \neq t_2[0] \end{cases}$$

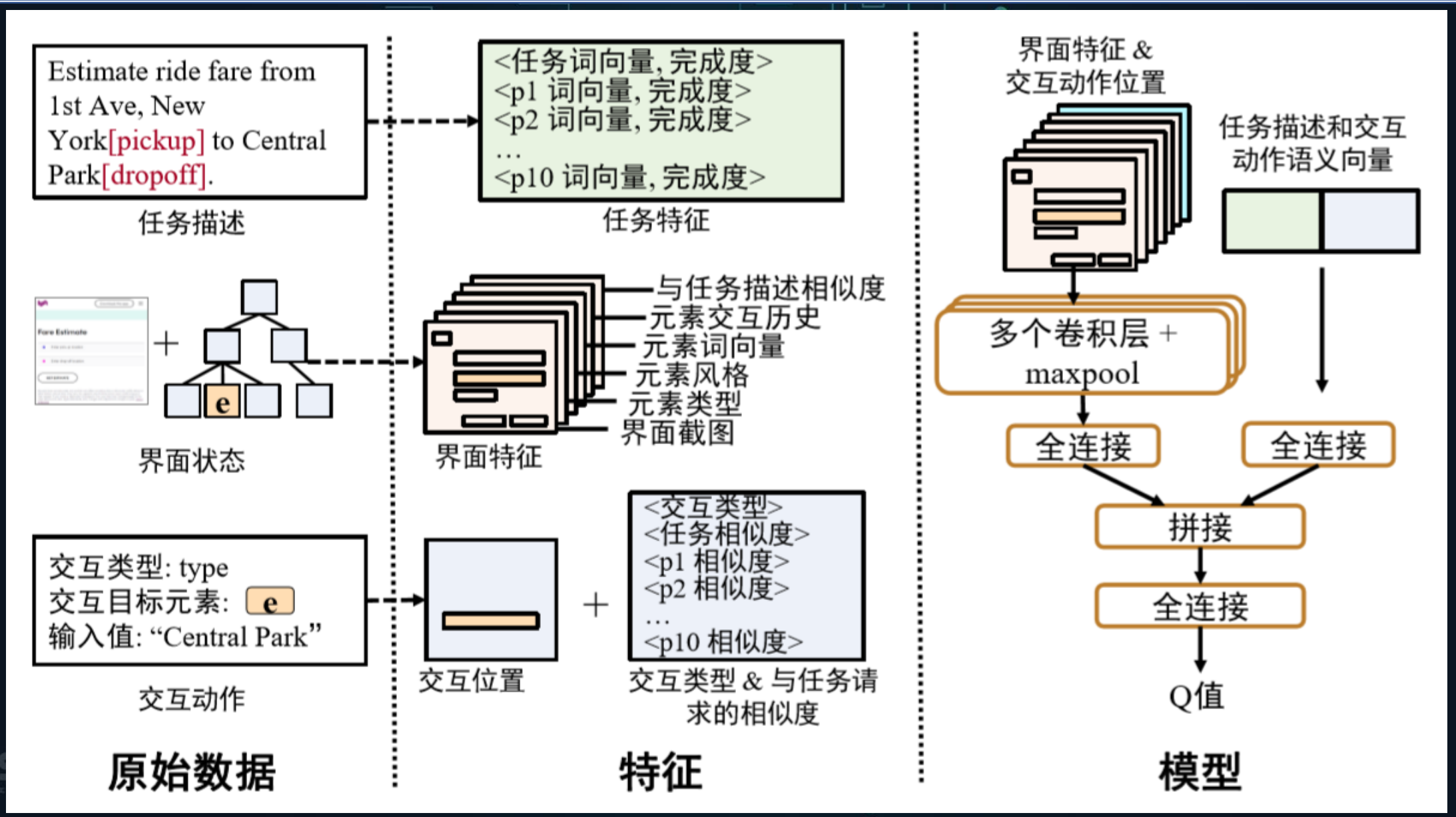
$$sim_{n-gram}(t_1, t_2) = \frac{|common(split(t_1), split(t_2))|}{max\{|split(t_1)|, |split(t_2)|\}}$$

$$task_{sim} = \underset{tw \in q'}{\text{mean}}\{\underset{e \in w^*}{\max}\{sim(tw, e_{text})\}\}$$

$$par_{sim}(p) = \underset{e_i, e_j \in w^*}{\max}\{s_v(p, e_i) + \frac{s_v(p, e_i) * s_m(p, e_j)}{1 + \lambda * dist(e_i, e_j)}\}$$

# Q network: 用来近似Q(s, a)的神经网络





# Q network 训练算法

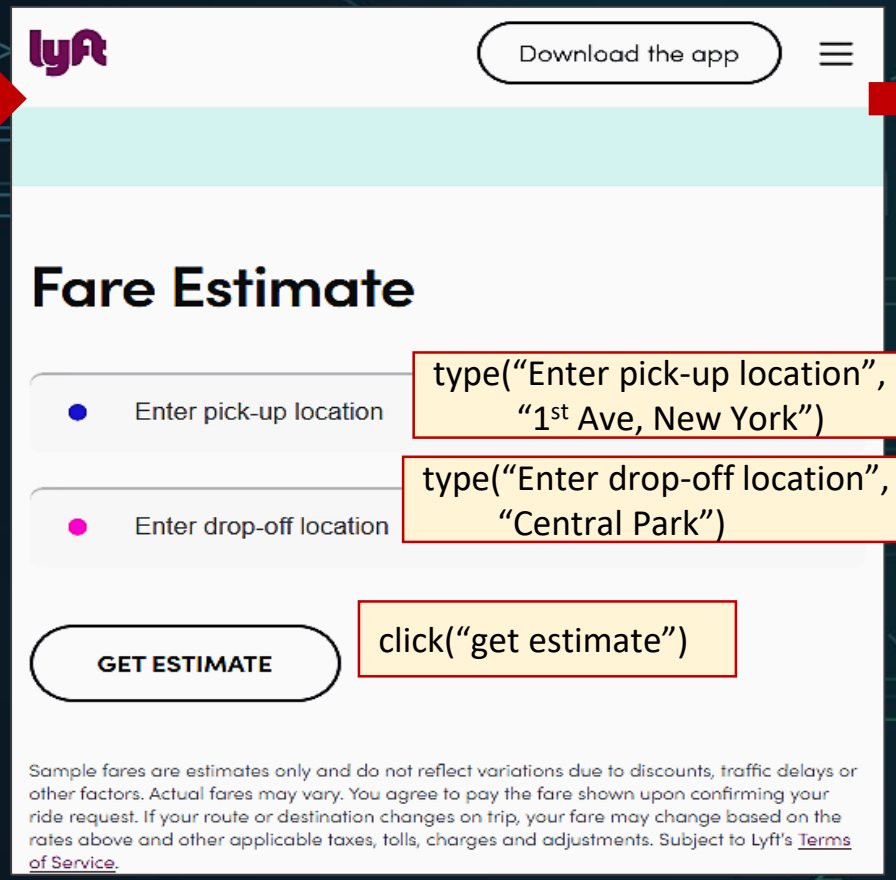
**Algorithm 1** Training the Q network.

**Inputs:** $Q$: the model to train, $q$: the task query, $url$: the website URL, $ENV$: the task execution environment, $N$: the number of episodes, $M$: the number of steps in each episode, $D^{replay}$: an empty replay buffer. $f_{explore}$: the exploration strategy.
Initialize the set of action sequences $AS$ as $\emptyset$
**for** episodes j from 1 to N **do**
  reset $ENV$ and load $url$ in the browser.
  initialize the action sequence $as$ as empty.
  initialize $s_0$=<$q$, $w_0$, $as$> where $w_0$=$url$'s start page
  calculate $\epsilon$ as $1 - j/N$
  **for** steps $i$ from 0 to M **do**
    set $A_i$ as the set of possible actions in $s_i$.
    break current episode if $A_i = \emptyset$
    generate a random number k in the range [0.0, 1.0)
    **if** $k < \epsilon$ **then**
      select an action $a_i = f_{explore}(A_i)$
    **else**
      select an action $a_i = argmax_{a \in A_i} Q(s_i, a)$
    **end if**
    send $a_i$ to $ENV$ and observe $r_i$ and $s_{i+1}$
    put <$s_i$, $a_i$, $r_i$, $s_{i+1}$> into $D^{replay}$
    put $a_i$ into $as$
  **end for**
  sample a batch from $D^{replay}$ (if enough entries)
  perform a gradient descent step to update $Q$ (as in [58])
  put $as$ into $AS$
**end for**
**Output:** $Q$ and the found action sequences $AS$

# 一个例子

**q**
Estimate ride fare from 1st Ave, New York [start location] to Central Park [end location].

**url**
www.lyft.com/




**Script-1 Total_reward: 11.02**
type($e_{\text{Enter pick-up location}}$, 1st Ave, New York)
type($e_{\text{Enter drop-off location}}$, Central Park)
click($e_{\text{get estimate}}$)

**Script-2 Total_reward: 10.52**
type($e_{\text{Enter drop-off location}}$, Central Park)
type($e_{\text{Enter pick-up location}}$, 1st Ave, New York)
click($e_{\text{get estimate}}$)

**Script-3**
...

**Script-TEMPLATE**
type($e_{\text{Enter pick-up location}}$, start location)
type($e_{\text{Enter drop-off location}}$, end location)
click($e_{\text{get estimate}}$)

# 实验评估



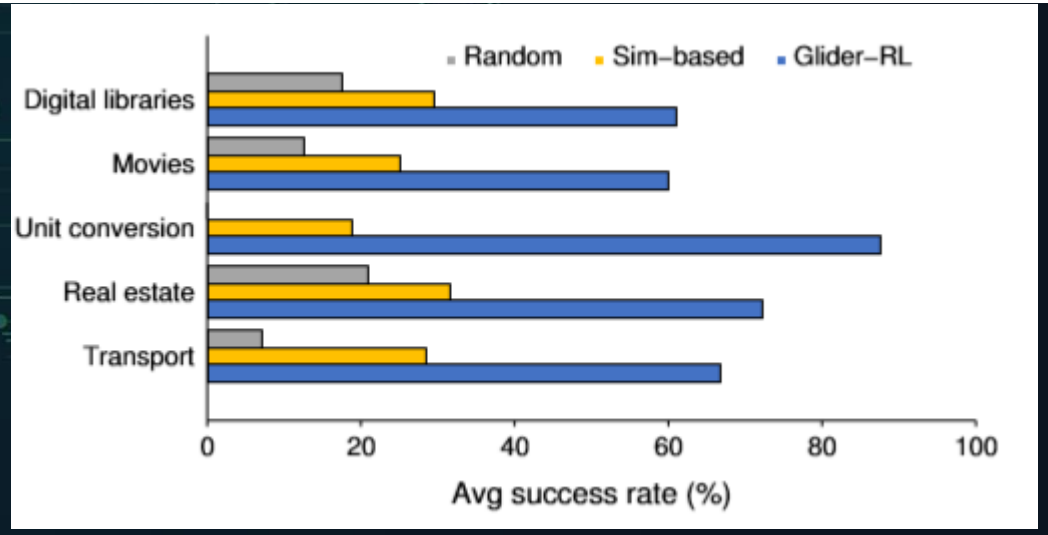
- 评估集
  - 选取10大类网站，共73个网站
  - 41个测试用例描述，例如：
    - 在波士顿订一个下午8点4人的桌子
    - 将“intelligent”从英语翻译到西班牙语
    - 预定9月1日从BOS飞往SFO的飞机
  - 172个<测试用例描述，网站>组合
  - 人工标注正确测试脚本，平均包含3.9个动作
- 对每个<测试用例描述，网站>运行算法
  - 100 episodes (大约1小时)
  - 得到reward最高的5个交互动作序列

| Category | # *q* | # *url* | Tot | Avg steps | *q* and *url* example |
|---|---|---|---|---|---|
| Health | 4 | 1 | 4 | 4 | *q*=Lookup canceled dermatology appointments at Laboratory *url*=demo.openmrs.org |
| Restaurants | 6 | 4 | 16 | 3.2 | q=Reserve a table for 4 people at 8pm in Boston, *url*=opentable.com |
| Transport | 4 | 8 | 12 | 4.6 | *q*=Estimate ride fare from 1st Ave, NY to Central Park, *url*=lyft.com |
| Real estate | 4 | 8 | 18 | 2.9 | *q*=Rent a house in Seattle with 3 bedrooms, *url*= rent.com |
| Movies | 3 | 4 | 10 | 5.3 | *q*=Buy tickets for the Lion King on Friday in 15113, *url*=fandango.com |
| Shopping | 3 | 9 | 15 | 2.4 | *q*=Find women shoes, *url*=ebay.com |
| Books | 3 | 9 | 26 | 5.3 | *q*=Find books by Umberto Eco, sort by publication date, *url*=aadl.org |
| Digital libraries | 4 | 8 | 18 | 2.6 | *q*=Citations of “ImageNet: A Large-Scale...”, *url*=scholar.google.com |
| Dictionary | 3 | 10 | 18 | 2.9 | *q*=Translate intelligent from English to Spanish, *url*=translate.com |
| Unit conversion | 2 | 8 | 16 | 4.3 | *q*=Convert length from 7333 inches to feet, *url*=unitconversion.org |
| Flights | 5 | 4 | 19 | 5.4 | *q*=Flights from BOS to SFO departing 09/01 for 2 adults, *url*=aa.com |
| **Summary** | **41** | **73** | **172** | **3.9** | |

# 实验评估结果

- 正确脚本出现在输出结果top1和top5的准确率为65.7%和76.6%
- 对比baseline有较大优势：
  - MCTS （蒙特卡洛树搜索）
  - Hill Climbing （爬山算法*）
  - Random （随机搜索）
  - Sim-based （基于文本相似性贪心搜索）
- 准确率仍然不是很高，原因：
  - 问题本身的难度
  - 算法的局限性

| Category | Glider-RL | | MCTS | | Hill climbing | |
|---|---|---|---|---|---|---|
| | top | top 5 | top | top 5 | top | top 5 |
| Health | 75.0 | 100 | 75.0 | 100 | 75.0 | 100 |
| Restaurants | 62.5 | 68.8 | 50.0 | 62.5 | 43.8 | 43.8 |
| Transport | 66.7 | 75.0 | 50.0 | 50.0 | 25.0 | 58.3 |
| Real estate | 72.2 | 77.8 | 33.3 | 44.4 | 38.9 | 50.0 |
| Movies | 60.0 | 70.0 | 30.0 | 30.0 | 20.0 | 40.0 |
| Shopping | 66.7 | 80.0 | 35.7 | 42.9 | 21.4 | 35.7 |
| Books | 61.5 | 80.8 | 37.5 | 45.8 | 36.0 | 64.0 |
| Digital libraries | 61.1 | 72.2 | 16.7 | 50.0 | 27.8 | 44.4 |
| Dictionary | 72.2 | 77.8 | 44.4 | 77.8 | 44.4 | 77.8 |
| Unit conversion | 87.5 | 87.5 | 12.5 | 50.0 | 25.0 | 56.3 |
| Flights | 36.8 | 52.6 | 29.4 | 29.4 | 23.5 | 35.3 |
| **Summary** | **65.7** | **76.6** | **37.7** | **53.0** | **34.6** | **55.1** |



# THANK YOU

感 谢 聆 听

## 未来工作

- 更适合GUI的深度学习网络架构
- 更完善的数据集，针对GUI的数据增强方法
- 更稳定、高效的测试环境
- 交互动作序列的安全保证