AutoDroid-V2: Boosting SLM-based GUI Agents via Code Generation

Hao Wen[1], Shizuo Tian[1], Borislav Pavlov[1], Wenjie Du[1,*], Yixuan Li[1,*], Ge Chang[1,*], Shanhui Zhao[1,*],Jiacheng Liu[1,*], Yunxin Liu[1,2], Ya-Qin Zhang[1], Yuanchun Li[1,2,3,†]

[1] Institute for AI Industry Research (AIR), Tsinghua University
[2] Shanghai Artificial Intelligence Laboratory
[3] Beijing Academy of Artificial Intelligence (BAAI)

MobiSys '25

Presenter: Chuanhua Fan

Dictionary

- **Background**
- Related work
- Design
- Evaluation
- Conclusion

Background

目标：通过自然语言自动完成复杂任务

Background

当前主流：

依赖云端大语言模型（GPT-4o、Claude等）的 Step-wise GUI Agents

Step-wise GUI Agents：分布式操作，每个UI界面状态都会查询LLM并得出相应的操作

当前痛点：

隐私风险：用户数据泄露

成本与部署难题：推理成本高、闭源

端侧SLM适配性差：推理能力弱、频繁调用计算资源

Background

现有技术优势:

- SLM (Small language models)可以被训练的擅长编码，而且一些SLM可以实现和闭源LLM相当的精度
- 收集大规模高质量的源代码数据比收集其他类型（图片等）的高质量数据更简单

想法:

GUI分步决策问题

代码生成问题

Background

解决办法：

AutoDroid-V2——生成并执行多步脚本（Script-based GUI Agent），特点如下：

- **效率高**：生成单个脚本完成任务
- **能力强**：大量关于轻量级编码助手的研究证明SLM具有很强的编码能力

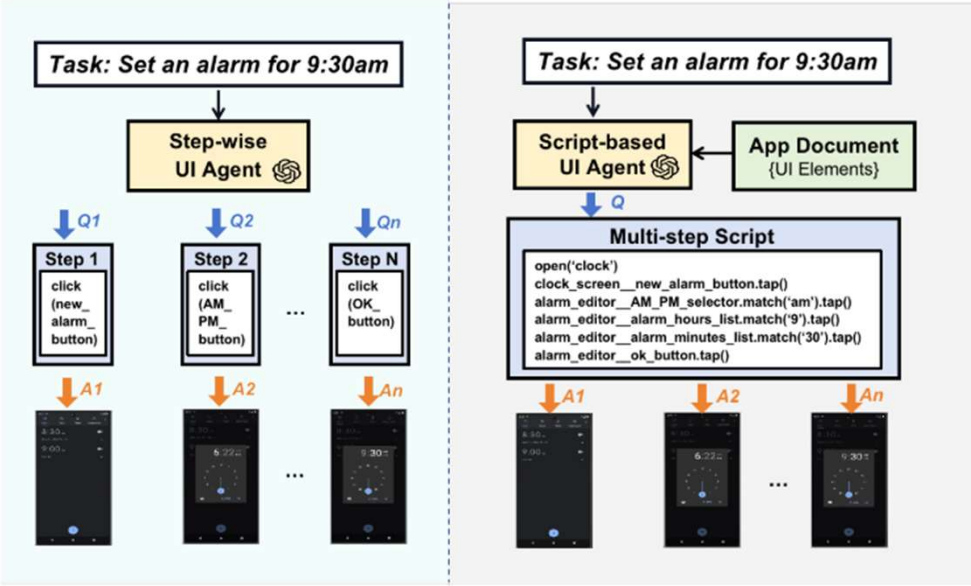


Figure 1: Comparison of conventional step-wise GUI agent and our script-based GUI agent. The step-wise GUI agent includes the user's task and the current GUI state in the LLM prompt ($Q1, \ldots Qn$), asking the model to generate a GUI action for each step ($A1, \ldots An$). The script-based agent inputs the task and app document (Q), prompting the LLM to generate a multi-step script for execution.

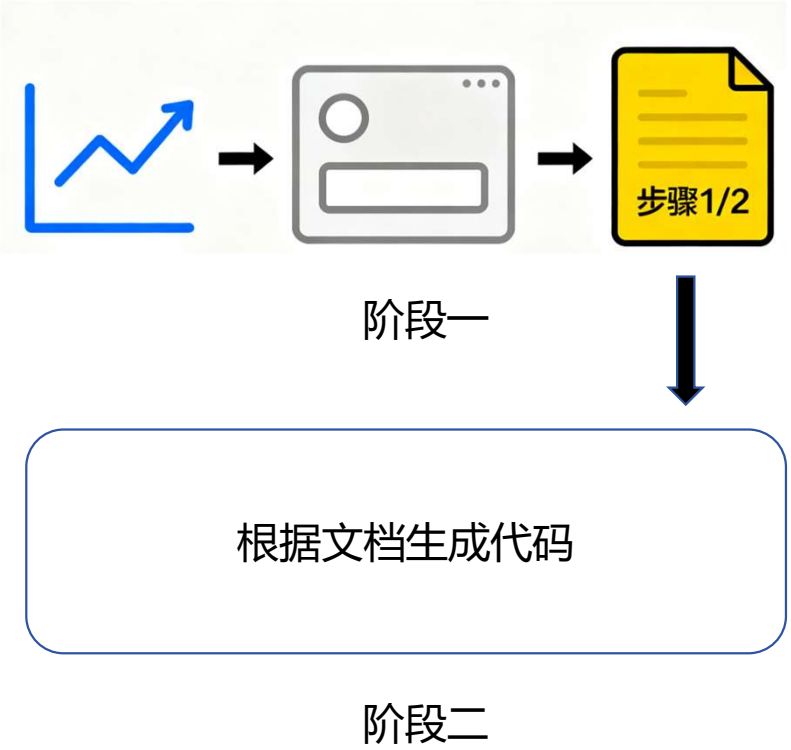
Background

为什么APP运行前生成多步脚本执行任务可行?

应用程序的状态和功能是有限的

阶段:

- 应用程序文档生成阶段
- 根据文档生成代码阶段



Dictionary

- Background
- **Related work**
- Design
- Evaluation
- Conclusion

Related work

基于GUI的移动任务自动化相关定义

术语	定义
task	用户想要完成的多步骤**功能请求**
GUI State	当前**界面状态**，表现为由图像和文本控件构成的“**GUI树**”
GUI element	可以和用户交互的**控件**如按钮、滑块
GUI action	屏幕上进行的**操作** (GUI element, action)

Related work

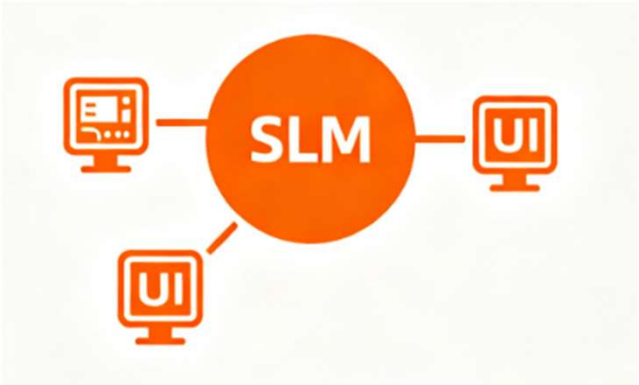
两类主流GUI Agents的局限性

- *基于LLM的 GUI Agents*

特点：逐步决策方式完成任务灵活性更强
缺点：隐私风险高、成本高
启发：使用端侧LLM支持的agent

- *基于SLM的 GUI Agents*

特点：可以通过定制SLM来提高任务自动化的效果
缺点：推理能力弱、消耗大量端侧资源
启发：引入由文档引导的脚本生成



Dictionary

- Background
- Related work
- **Design**
- Evaluation
- Conclusion

Design

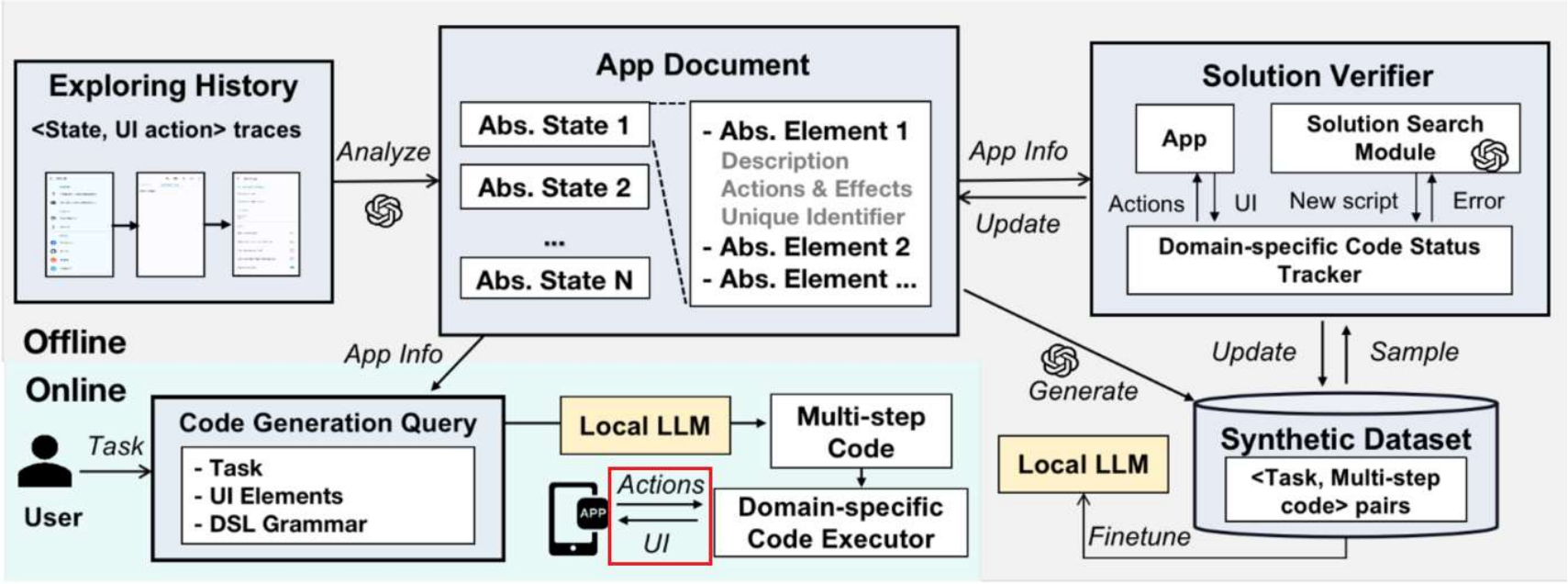


Figure 2: The architecture of AutoDroid-V2.

离线阶段：微调本地大语言模型

在线阶段：接收用户指令，自动完成任务

Design

离线阶段1：APP文档生成阶段

目标：从 APP 的随机探索轨迹中，提炼简洁、精准的“APP 文档”，为 SLM 提供清晰的交互逻辑。

随机探索轨迹抽象为 <GUI State, UI Action> 对

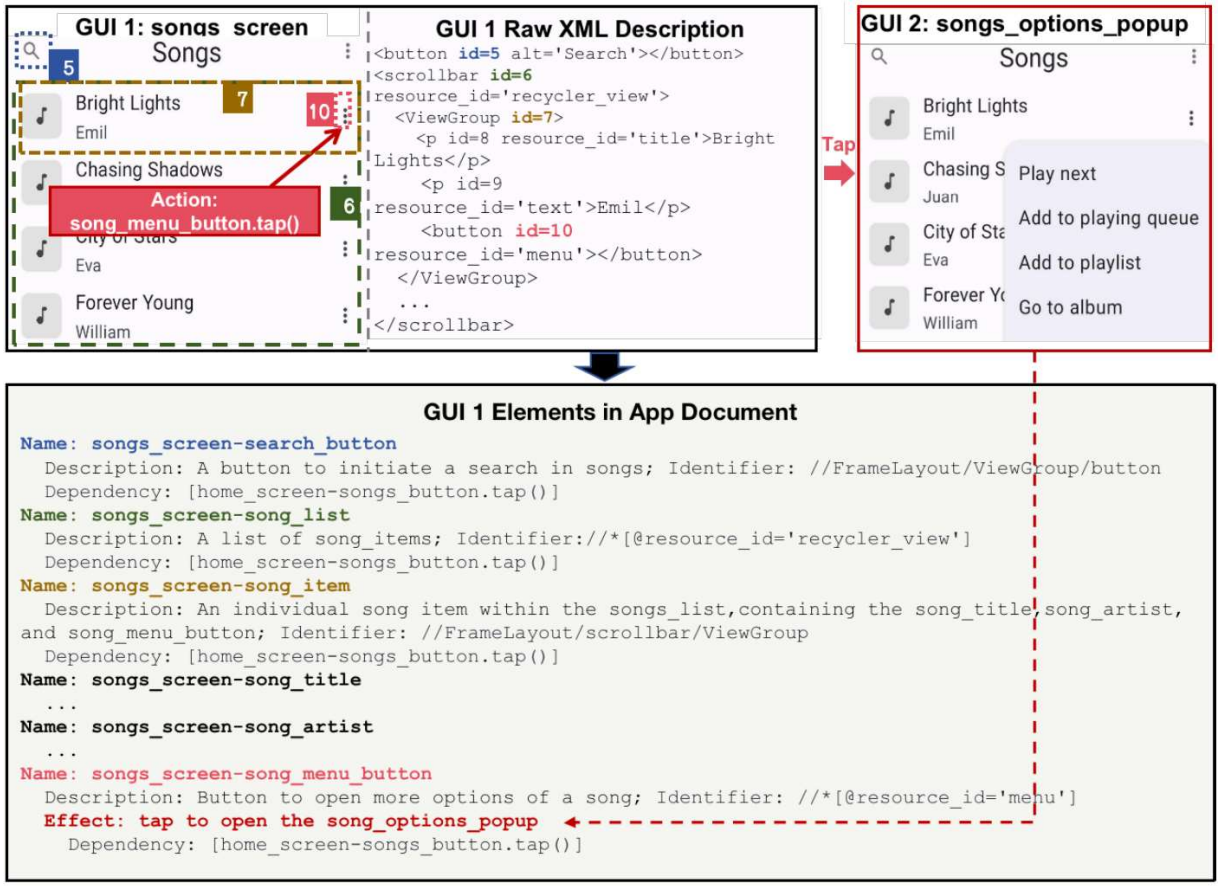


Figure 3: The document of an app containing essential abstract elements in one GUI.

Design

离线阶段1：APP文档生成阶段

- **（1）功能感知GUI分组**

目标：解决GUI数量多和冗余问题

问题：相同布局的GUI可能功能不同

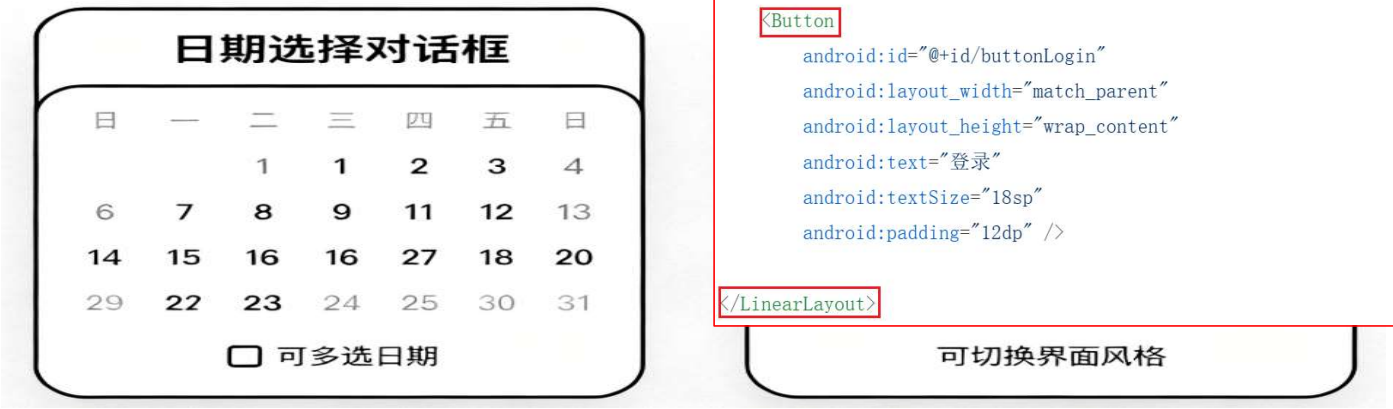
布局相似性分组：移除 GUI 树的具体内容，保

JI

功能相似性分组：调用 GPT-4o 判断分组后的

分

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="24dp"
    android:background="#FFFFFF"
    tools:context=".MainActivity">

    <!-- 登录按钮 -->
    <Button
        android:id="@+id/buttonLogin"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="登录"
        android:textSize="18sp"
        android:padding="12dp" />

</LinearLayout>
```



Design

离线阶段1：APP文档生成阶段

- **（2）抽象GUI元素的定义**

目标：减少需要描述的元素数量

将同类元素抽象为：

- 动态元素
- 静态元素
- 元素列表

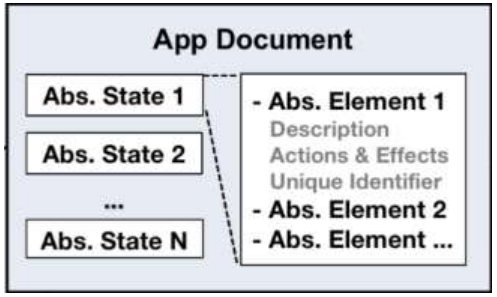


Table 1: Components of Abstract GUI Elements

Component	Description	Example
Name	A concise name for the element formatted as *state_name-element_name*.	songs_ui-song_list
Description	A brief description of the element's purpose and functionality.	A list of song_items
Identifier	Uniquely identifies and locates a specific GUI element within the app's GUI tree.	//*[@resource id='recycler view']
Options	All the possible keywords of the items in this element list. The items can be chosen by using '<element_list>.match(Options[i])' statement, which will be explained in Section 3.2.1.	<song_names>
Effect	describes the effect of performing actions on it.	tap to open the song_options_popup
Dependency	The navigation path for one GUI element from the other GUI screens.	Tap main_screen-open_song

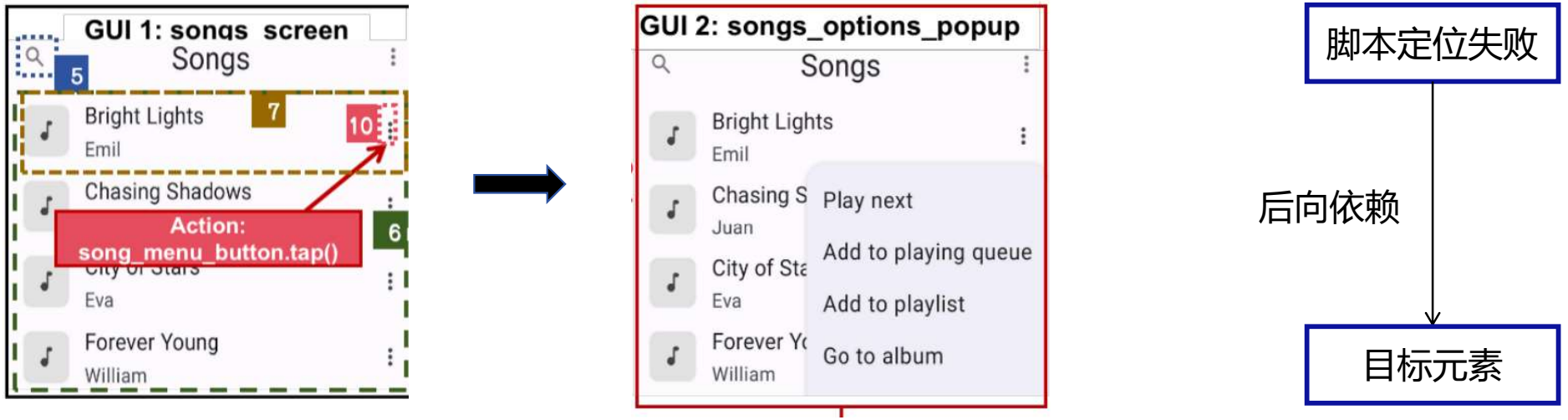
Design

离线阶段1：APP文档生成阶段

- **（3）前后项依赖分析**

目标：梳理交互逻辑，方便脚本生成

- **构建ETG**（Element Transition Graph）转换关系 (e_i ,a_i ,e_i + 1)
- **后向依赖**：到达目标元素的所有可能路径（如 “如何从首页到歌单列表”）
- **前向依赖**：目标元素操作后可触发的后续元素（如 “点击歌单列表可进入歌曲详情”）



Design

离线阶段2：SLM定制的数据合成

目标：生成高质量的训练数据集

- **（1）大规模任务与脚本生成**

思路：从 ETG 图中采样子图 G_s（对应一组可完成的交互），反向生成用户任务，并基于 APP 文档生成脚本

工具：自定义DSL(Domain-specific Language) Python库，包含两类API：

- GUI操作APIs：操作元素
- 信息检索 APIs：定位元素

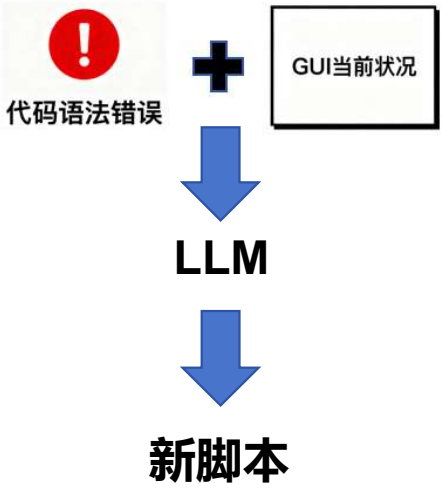
```
'''
User task: Add the songs of Artist Emil to the Playlist 'Emil'.
'''
# Navigate to the Songs section
home_screen-songs_button.tap()
# Flag to control the scrolling loopdone = False
while not done:
  for song_item in songs_screen-song_list:
    song_artist = song_item.get_text(songs_screen-song_artist)
      if song_artist == 'Emil':
        song_item.tap(songs_screen-song_menu_button)
        song_options_popup-add_to_playlist_button.tap()
        add_playlist_screen-playlists.match('Emil').tap()

  # Check if the end of the song_list has been reached
  bottom = songs_screen-song_list.scroll("down")
  if bottom:
    done = True
```

Listing 3: LLM generated Python code for the user task.

Design

离线阶段2：SLM定制的数据合成

- **（2）基于验证的脚本修正**



问题：LLM 生成的脚本可能因 “语法错误”“元素不存在” 等无法执行

方案：

- 在模拟器上执行脚本，捕获错误类型
- 将 “错误信息 + 当前 GUI 状态” 反馈给 LLM，重新生成脚本，直至可执行

- **（3）基于树的脚本质量提升**

问题：脚本可执行不代表可以完成任务

方案：

- 用云端 LLM 构建**奖励模型**：输入 “任务 + 脚本 + 执行轨迹”，判断脚本是否完成任务
- 构建**DFS 树**：根节点为初始脚本，子节点为基于奖励模型修改后的新脚本，迭代搜索最优脚本

输出：过滤后的 “任务 - 代码” 对，用于 SLM 微调

Design

在线阶段：基于脚本的任务执行

- **（1）运行时的动态处理**

➢ 通过后向依赖定位元素
➢ 失败后重新生成新脚本

Algorithm 1 Dependency-Aware Element Locating

Input: Target element E, current app instance App, dependency paths $\mathcal{D} = [D_1, D_2, \ldots, D_k]$, app document Doc
Output: Execution status S

```
function ElementGrounding(E, App, 𝒟)
    while E.identifier ∉ current_ui_tree and attempts < limit do
        Dependency ← Select from 𝒟 based on which dependency includes an action matching the current GUI state
        for action in Dependency do
            current_ui_state ← GetCurrentUIState(App, Doc)
            if action.element.ui_state = current_ui_state then
                App.send_action(Action)
    if E ∉ Current then
        return Error                                   ▷ Section 3.2.2
    else
        return CurrentUITree.FindElement(E.identifier)
```

- **（2）提示词的压缩与重用**

问题：端侧 SLM 上下文短，完整的 APP 文档无法加载
方案：
➢ 提示词压缩：用户任务 + 指令 + APP 元素名称
➢ KV 缓存复用：APP 元素名称预加载到KV缓存中

任务
基本指令
元素名称

prompt

Dictionary

- Background
- Related work
- Design
- **Evaluation**
- Conclusion

Evaluation

实验设置

数据集：

DroidTask：158个任务、13个App

AitW-subset：68个任务、10个App

基线模型：

VLM 类agent：SeeClick（Qwen-VL-9.6B）

CogAgent（CogGLM-17B）

LLM 类agent：Mind2Web（Llama-3.1-8B）

AutoDroid（Llama-3.1-8B-ft）

硬件：

端侧设备：OnePlus ACE 2 Pro（Snapdragon 8 Gen2）

MacBook Pro（M2 Pro）；

训练设备：8×A100 80GB 服务器。

核心指标：

成功率SR： 任务完成率

反向冗余率RRR:人工操作长度 / Agent操作长度，越大效果越好

推理延迟（LLM Inference Latency）： 从输入提示词到输出脚本的时间

Token消耗（Token Consumption）： 输入或输出Token的数量

Evaluation

成功率和反向冗余率

Table 2: Success rate on DroidTask. SR: Success Rate. RRR: Reversed Redundancy Ratio. 'ft' represents LLMs fine-tuned on GUI-specific dataset.

Metric	Method	Launcher	Calendar	Camera	Clock	Contacts	Dialer	File	FireFox	Gallery	SMS	Music	Notes	Recorder	Average
SR	SeeClick (Qwen-VL-9.6B-ft)	0%	0%	0%	9.1%	0%	0%	0%	0%	11.1%	13.3%	0%	0%	0%	2.7%
	CogAgent (CogVLM-17B-ft)	0%	0%	0%	9.1%	0%	0%	0%	0%	11.1%	13.3%	0%	0%	12.5%	3.3%
	Mind2Web (Llama-3-8B)	60.0%	41.2%	60.0%	50.0%	21.4%	33.3%	20.0%	12.5%	22.2%	40.0%	11.1%	35.7%	33.3%	34.4%
	AutoDroid (Llama-3-8B-ft)	80.0%	29.4%	50.0%	50.0%	21.4%	40.0%	**43.8%**	**87.5%**	44.4%	**33.3%**	44.4%	**35.7%**	**66.7%**	43.9%
	AutoDroid-V2 (Llama-3-8B-ft)	**80.0%**	**64.7%**	**66.7%**	**83.3%**	**50.0%**	**60.0%**	31.2%	75.0%	**66.7%**	26.7%	**55.6%**	28.6%	55.6%	**54.4%**
RRR	Mind2Web (Llama-3-8B)	21.1%	26.7%	42.6%	29.6%	39.4%	43.0%	38.9%	66.7%	75.0%	58.3%	23.1%	42.7%	50.7%	41.0%
	AutoDroid (Llama-3-8B-ft)	73.8%	82.4%	**92.3%**	70.8%	**88.9%**	**91.7%**	**95.2%**	67.6%	**100.0%**	95.0%	84.4%	95.0%	88.9%	86.3%
	AutoDroid-V2 (Llama-3-8B-ft)	**93.8%**	**94.7%**	91.7%	**74.7%**	88.1%	88.9%	95.0%	**76.4%**	96.7%	**100.0%**	**95.0%**	**100.0%**	**100.0%**	**90.5%**

- VLM成功率最低

原因：仅依赖屏幕截图，不适用于文本比较多的APP

- AutoDroid成功率高

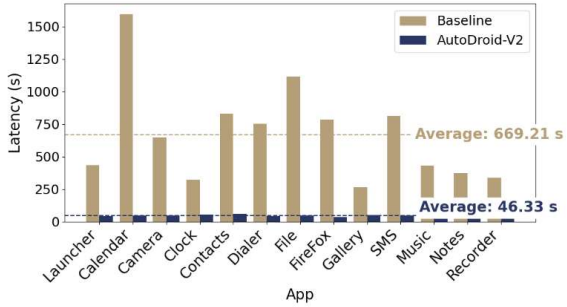
原因：采用了自动任务生成和微调技术，步进式，而Autodroid-v2采用基于脚本所以SR更高

Table 3: Success rate on AitW-subset. SR: Success Rate.

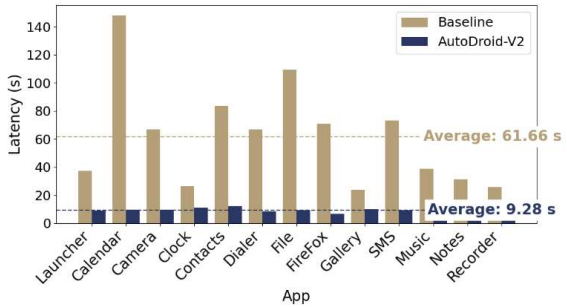
Method	Model	SR
Mind2Web	Llama-3.1-8B	27.1%
SeeClick	Qwen-VL-9.6B	30.8%
CogAgent	CogGLM-17B-ft	36.7%
AutoDroid	Llama-3.1-8B-ft	36.7%
AutoDroid-V2	**Llama-3.1-8B-ft**	**47.1%**

Evaluation

延迟和成本



(a) LLM inference latency in Snapdragon 8 Gen 2.



(b) LLM inference latency in Apple M2 Pro.

Figure 4: Average LLM inference latency of AutoDroid-V2 and baseline (AutoDroid) on DroidTask.

具有更小的推理延迟

Table 4: Runtime token number of AutoDroid-V2 and AutoDroid. 'Cached' denotes the cached prompt prefix that has been processed and can be reused. 'Remaining' denotes the remaining parts of the input prompt that come after the cached prefix.

	Method	Input Tokens			Output Tokens
		Cached	Remaining	Total	
Per Step	AutoDroid	66.0	452.1	518.1	127.4
	AutoDroid-V2	N/A	N/A	N/A	N/A
Per Task	AutoDroid	431.3	3021.2	3452.5	832.4
	AutoDroid-V2	2760.1	**67.9**	2828.0	**122.9**
			97.8%		85.2%

- Cached Tokens：prompt中可以内重复使用的那部分前缀（APP文档）
- Remaining：prompt中剩余的需要实时处理的动态部分

Evaluation

细粒度性能分析

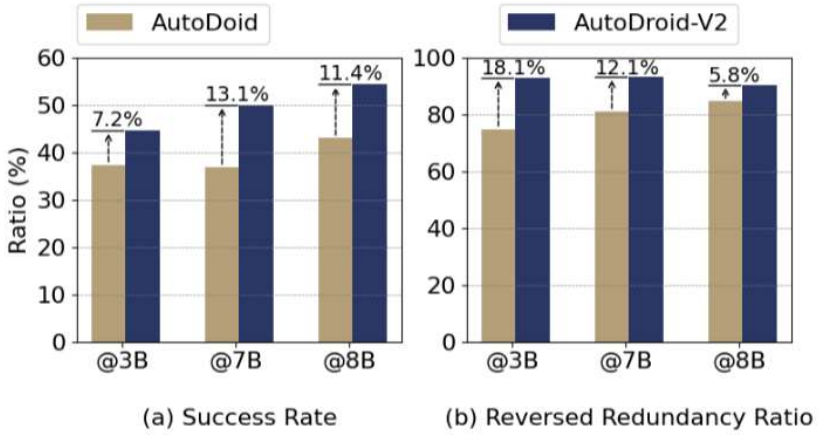


Figure 5: Success rate and reversed redundancy ratio of AutoDroid-V2 and baseline based on different LLMs. llama3.2-3b, qwen2.5-7b, llama3.1-8b

3B参数的小模型，实现了很高的成功率

消融实验：脚本验证的必要性

对比 “有无脚本验证模块” 的成功率：
有验证：54.4%（过滤无效脚本，提升准确性）
无验证：51.9%（部分脚本无法执行或未完成任务）

Dictionary

- Background
- Related work
- Design
- Evaluation
- **Conclusion**

Conclusion

结论

- 提出了一个文档引导的、基于脚本的端到端系统AutoDroid-V2，目的是实现使用设备端SLM实现移动任务自动化
- 可能实现将GUI Agent完全部署在用户设备上

启发

- 本文提到的AutoDroid-v2方法只是完成单个的APP里面用户任务，但是实际情况中的用户任务会涉及到多个APP之间的交互，比如QQ上面的图片可以发到微信上面，针对这种情况我们进行改进，可以在APP文档里面记录两个APP之间进行的交互信息。
- 当前有一部分APP使用户拥有更加灵活的用户界面布置的能力，针对这类动态界面的任务自动化，可以进一步思考解决方案。

Conclusion

Table 2: Success rate on DroidTask. SR: Success Rate. RRR: Reversed Redundancy Ratio. 'ft' represents LLMs fine-tuned on GUI-specific dataset.

Metric	Method	Launcher	Calendar	Camera	Clock	Contacts	Dialer	File	FireFox	Gallery	SMS	Music	Notes	Recorder	Average
SR	SeeClick (Qwen-VL-9.6B-ft)	0%	0%	0%	9.1%	0%	0%	0%	0%	11.1%	13.3%	0%	0%	0%	2.7%
	CogAgent (CogVLM-17B-ft)	0%	0%	0%	9.1%	0%	0%	0%	0%	11.1%	13.3%	0%	0%	12.5%	3.3%
	Mind2Web (Llama-3-8B)	60.0%	41.2%	60.0%	50.0%	21.4%	33.3%	20.0%	12.5%	22.2%	40.0%	11.1%	35.7%	33.3%	34.4%
	AutoDroid (Llama-3-8B-ft)	80.0%	29.4%	50.0%	50.0%	21.4%	40.0%	**43.8%**	**87.5%**	44.4%	**33.3%**	44.4%	**85.7%**	**66.7%**	43.9%
	AutoDroid-V2 (Llama-3-8B-ft)	**80.0%**	**64.7%**	**66.7%**	**83.3%**	**50.0%**	**60.0%**	31.2%	75.0%	**66.7%**	26.7%	**55.6%**	28.6%	55.6%	**54.4%**
RRR	Mind2Web (Llama-3-8B)	21.1%	26.7%	42.6%	29.6%	39.4%	43.0%	38.9%	66.7%	75.0%	58.3%	23.1%	42.7%	50.7%	41.0%
	AutoDroid (Llama-3-8B-ft)	73.8%	82.4%	**92.3%**	70.8%	**88.9%**	**91.7%**	**95.2%**	67.6%	**100.0%**	95.0%	84.4%	95.0%	88.9%	86.3%
	AutoDroid-V2 (Llama-3-8B-ft)	**93.8%**	**94.7%**	91.7%	**74.7%**	88.1%	88.9%	95.0%	**76.4%**	96.7%	**100.0%**	**95.0%**	**100.0%**	**100.0%**	**90.5%**

在有些的APP上面AutoDroid的成功率比AutoDroid-v2高（如文件、短信、笔记、联系人），原因是这些APP高度依赖用户特定的数据，解决方案验证器会在一个没有任何用户数据的模拟环境里面来验证脚本的有效性，假如需要删除一个联系人时，该脚本在模拟环境当中不能正确，执行被验证器判断执行错误，从而使该**正确的高质量脚本被过滤掉**了。

Thank you!