基于LLM实现的线上项目自我修复与智能容灾的原理与实践

艾瑞坤

腾讯高级前端工程师

艾瑞坤（小蝌蚪），先后就职于阿里文娱、阿里淘宝、腾讯新闻前端团队，参与淘宝app和腾讯新闻页的开发，为千万用户提供方便快捷网购渠道和优质新闻媒体内容的分发与展示。专注于前端基建、AI智能化开发、前端工程化、webcontainer、多人协同编辑技术等。热爱技术，活跃于github、掘金等开源社区，热衷于技术交流与分享。

目录

CONTENTS

PART 01

前言与背景

Introduction and Background

各大公司降本增效，迫切需要更少的人力支撑现有业务

落地

研发提效

结合前端工程化，将重复枯燥事情进行自动化、智能化整合并解决

智能审核

智能审核稿件、负向评论等，成功率达到 73%，由人工+机审转向智能审核

智能 oncall

重复问题、同类问题、相关问题，chatgpt都可以理解并做出解释，覆盖率 30%

智能容灾

基于LLM实现线上项目具有自我修复能力，自动修复线上问题，实现真正意义上的LLMOps

300人 + AI > 1000人

PART

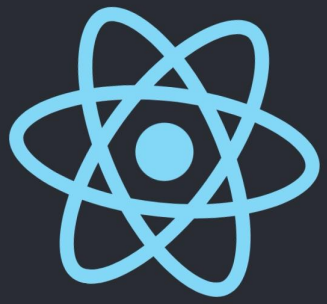
02 智能容灾&智能修复

Intelligent disaster recovery

星球大战中的机器人 R2-D2

智能修复

当部件损坏，可以自行定位问题，
找出解决方案，并自我修复



基于LLM的线上项目智能容灾与自我修复的实现

触发崩溃

Elements Performance insights Sources Network Performance Console 1

Preserve log Disable cache No throttling

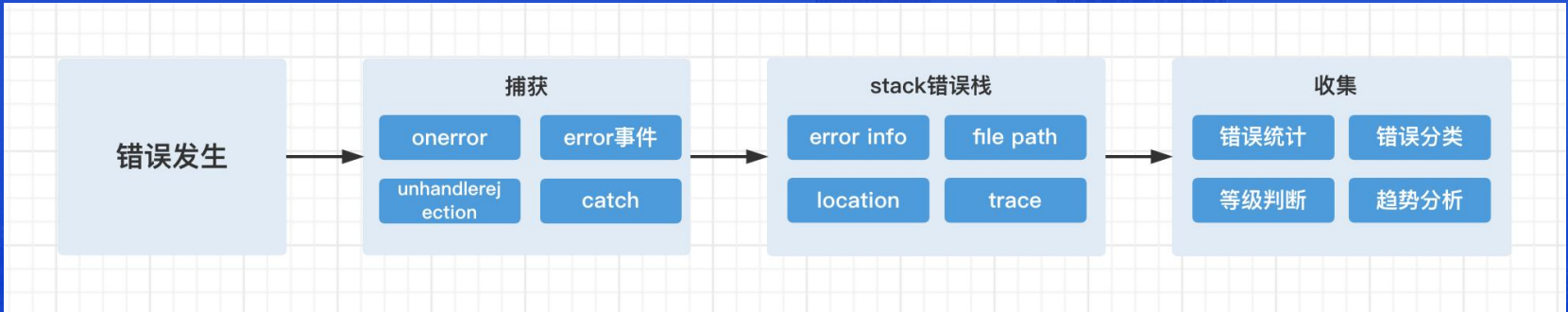
tnpm (node)

```
(base) ➜  server git:(master) ✗
(base) ➜  server git:(master) ✗ tnpm

> @tencent/leah-server@1.2.3 start /l
> cross-env NODE_ENV=development node

[nodemon] 1.19.4
[nodemon] to restart at any time, ent
[nodemon] watching dir(s): *.*
[nodemon] watching extensions: js,mjs
[nodemon] starting `node ./bin/www`
Server started, listening on port: 50
(node:76097) Warning: Accessing non-e
(Use `node --trace-warnings ...` to s
(node:76097) Warning: Accessing non-e
(node:76097) Warning: Accessing non-e
(node:76097) Warning: Accessing non-e
```

智能容灾实现 - 捕获并上报问题



捕获

通过全局错误事件进行错误捕获

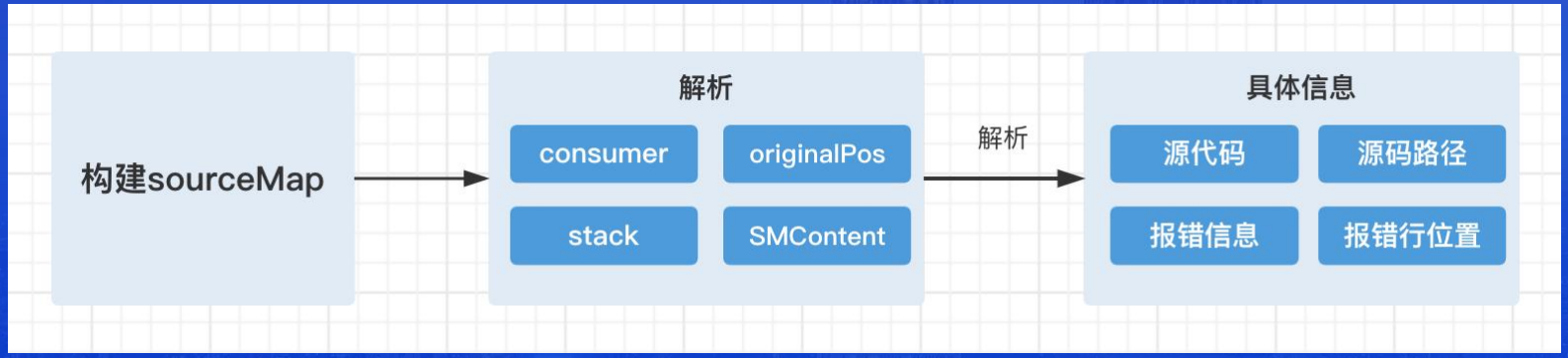
获取错误栈

获取错误栈得到报错信息、编译后的文件路径和trace等

收集

将错误进行归类统计和收集，判断趋势和紧迫程度等

智能容灾实现 - 问题分析与定位



sourceMap

构建sourceMap文件，按照指定规则上传内网cdn

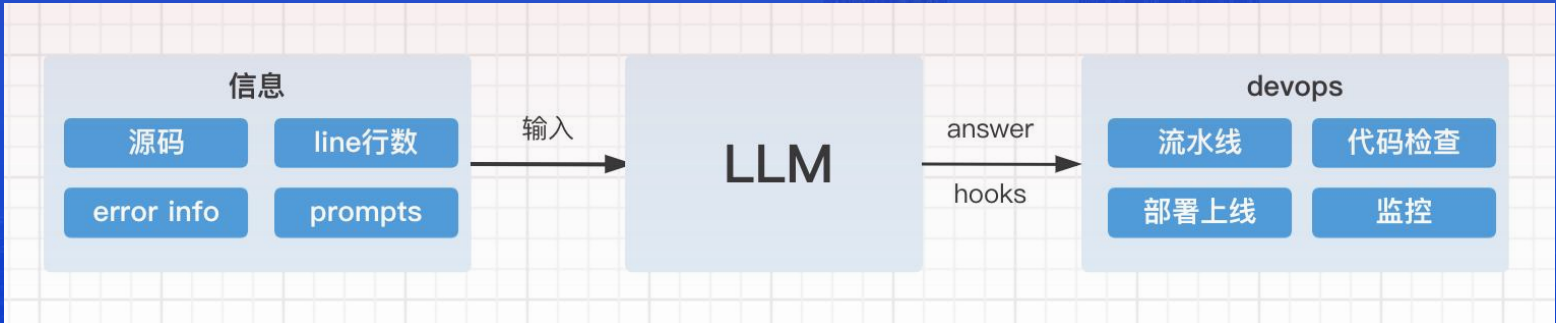
解析

通过sourceMap得到consumer结合错误栈等信息进行解析

具体信息

解析后得到源码、路径和报错所在的具体代码行，这些都是LLM所需要的必要信息

智能容灾实现 – 基于LLM进行智能修复



数据标准化

将具体信息格式化、标准化后，透传给LLM，进行智能分析

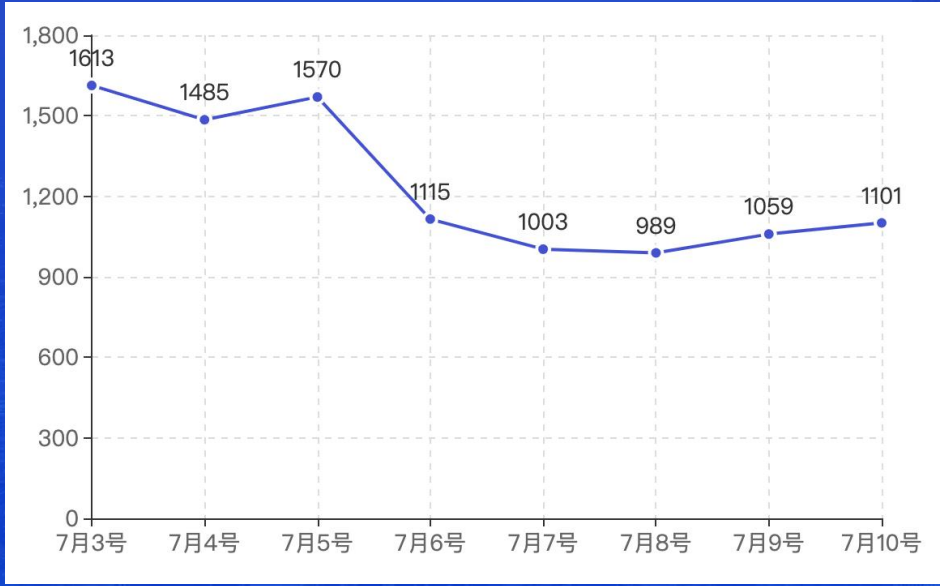
解析

LLM通过源码、报错信息、具体行数、prompts，进行问题分析并给出修复后的代码

devops

将生成的代码替换源码，通过hooks触发构建流水线，执行整套devops流程并自动上线，完成智能修复

效果 + 收益



- 报错数平均下降 28%
- 串行结构、三方服务、依赖崩溃
- 内测项目容灾兜底成功率 100%
- 年中 5min 30min 1h

H2规划：智能化测试平台

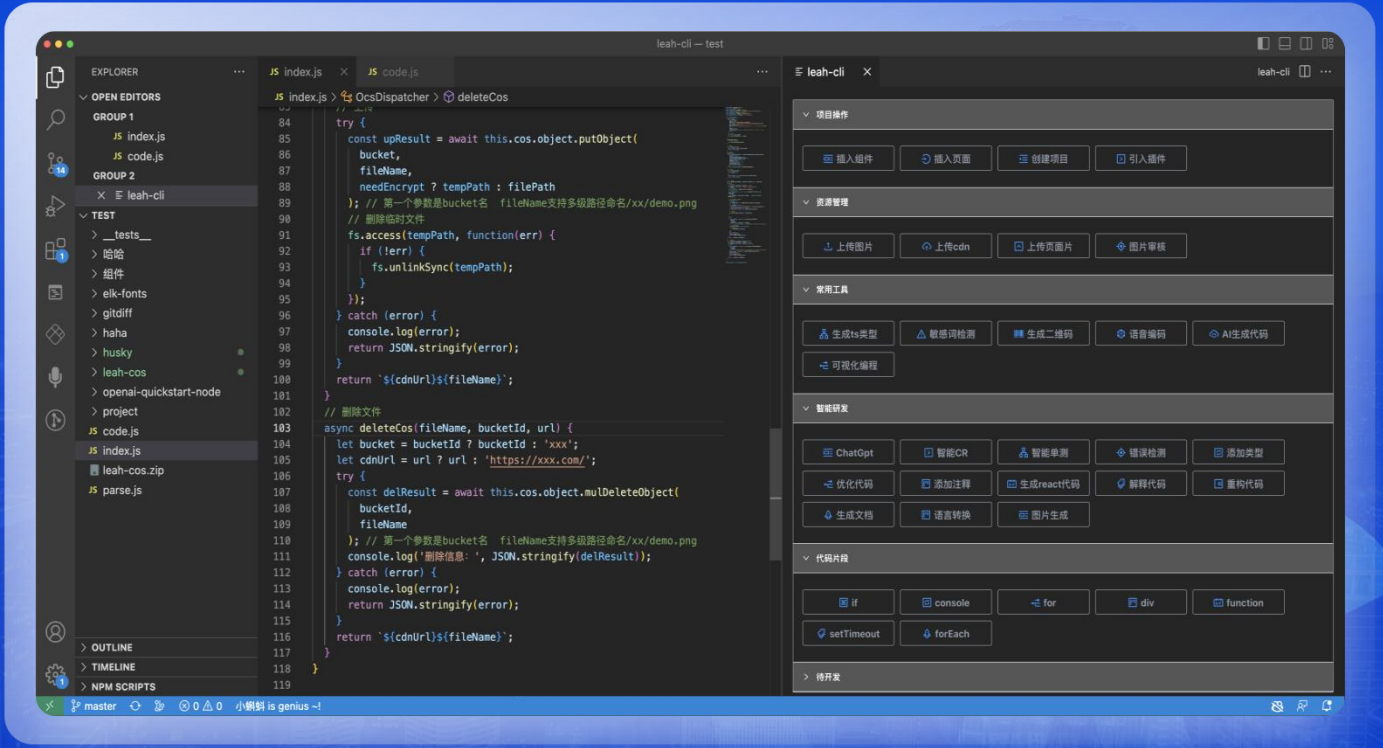
智能化测试 = 自动化测试 + 智能修复 + 基建体系

PART

03 智能研发提效

R&d efficiency and Intelligent improvement

> IDE + 工程化



代码：https://github.com/airuikun/smart-ide

通过AST精准修改代码

AST

通过 parser 将代码转化成
AST 抽象语法树

定位

通过 traverse 遍历 AST 找到
代码需要插入的结构位置

生成代码

通过 Declaretion 和 Specifer 生成对应的
代码语法树，Generator 生成对应的代码

结合chatgpt

智能 CR

智能单测

错误检测

解释代码

重构代码

优化代码

代码优化

代码生成

添加类型

添加注释

生成代码

语言转化

生成文档

图片生成

奇效

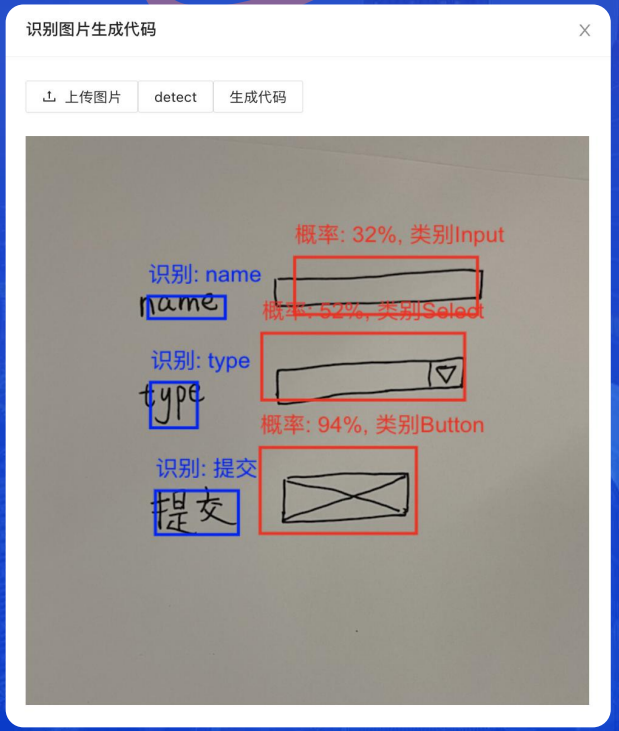
LLM + tensorflow

不局限于LLM，而是结合AI技术，实现技术创新

AI识别手画稿生成代码

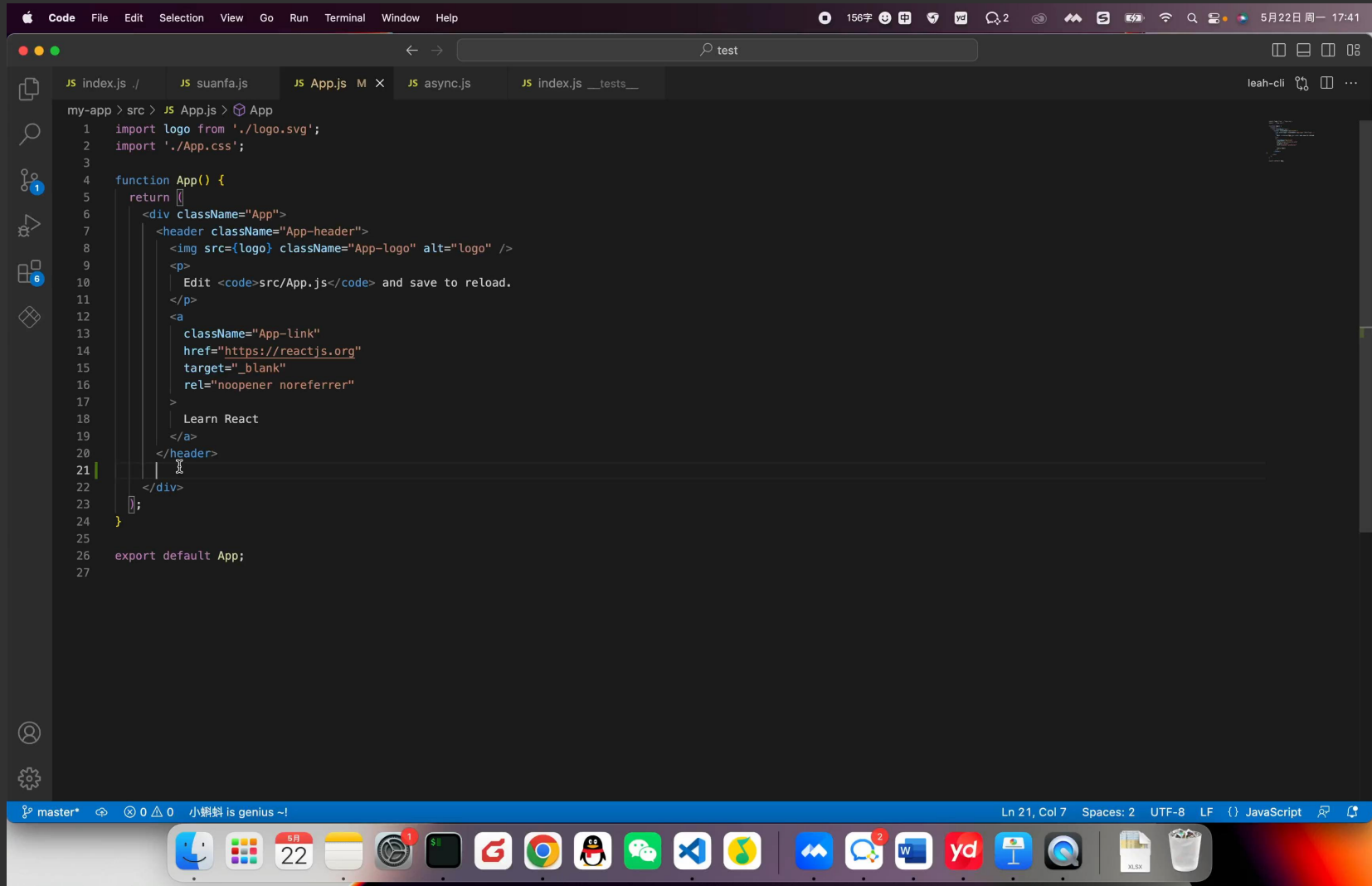
AI

识别

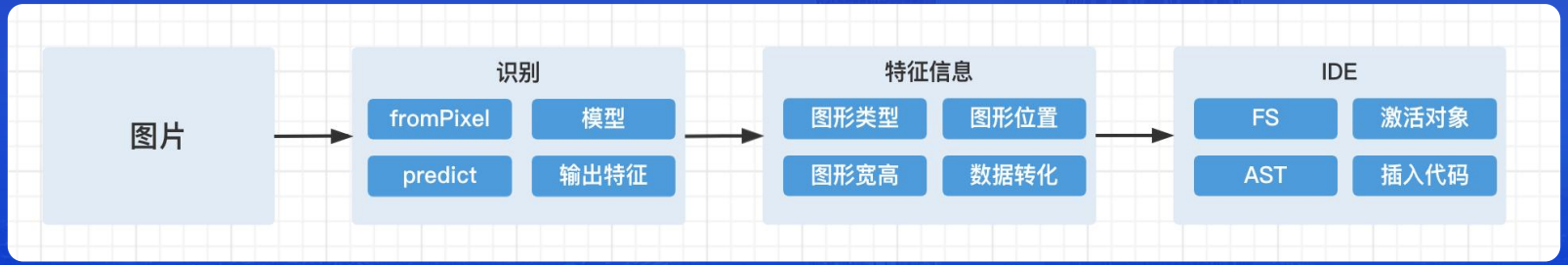


转化

code


Code File Edit Selection View Go Run Terminal Window Help
5月22日 周一 17:41
test
index.js ./
suanfa.js
App.js M
async.js
index.js __tests__
leah-cli
my-app > src > App.js > App
1 import logo from './logo.svg';
2 import './App.css';
3
4 function App() {
5 return (
6
7 <header className="App-header">
8
9
10 Edit <code>src/App.js</code> and save to reload.
11
12 <a
13 className="App-link"
14 href="https://reactjs.org"
15 target="_blank"
16 rel="noopener noreferrer"
17 >
18 Learn React
19
20 </header>
21
22
23);
24 }
25
26 export default App;
27
master*
0 0
小蝌蚪 is genius ~!
Ln 21, Col 7
Spaces: 2
UTF-8
LF
JavaScript

识别图形生成代码



识别

通过fromPixel获取图片数据
进行预测，输出图形识别结果

特征

特征包括图形类型、位置、宽高等
转成前端可识别的 schema 数据

代码

IDE拿到数据，结合模板代码
通过AST操作插入代码

理念

让AI帮我们把30%不喜欢的事情做完

捡苹果理论

PART

04 应对内卷

Response to shit

内卷三大形态

没事硬卷

“既要 又要 还要”，骗子才能满足

论文式日报

用AI应对内卷

AI不仅仅是业务和研发提效： AI+工程化+shell脚本

面向卷王编程、面向领导编程、前端点赞工程师

面向 HR 编程：F脚本，在线时长

后厂村的希望 (3)

1 

1 

1 

1 

1 

1 

AI应对内卷

实现了领导的“既要、又要、还要”

实现了 HR 的“既要、又要、还要”

六边形战士

程序员的终局

PART

05 安全问题

security issues

安全风险

黑产

刷数据、点赞、评论等
生成一百个10～100字的搞笑愤怒激进理性的评论

绕过验证机制

识别验证码、滑动模块

代码泄露

核心逻辑与带有 key、token 的代码

数据泄露

训练数据、核心数据

思考

风险与
效率并存

需要内部
制定规范与流程

万无一失
一失全无

代码安全扫描下
部分使用

PART

06 未来畅想

Imagine thie future

趋势

- AI识别生成模板代码，基于描述进行定制化开发
- Lowcode + 原子化开发
- prompt 工程师 = 超级个体 = 研发 + 测试 + 产品 + UI + 客服

未来规划

- 智能化测试
- LLMOps 持续建设
- LLM + AI技术：为AI加上眼、耳、手，实现真正的0.3人力

感谢

第一名的小蝌蚪

https://github.com/airuikun