如何借助AI的帮助，编写今年的开源大事记？

汇报人：庄表伟

时间：2024.12

目录

CONTENTS

01 如何理解这一轮AI浪潮?

02 SmartPrompt的设计思路

03 SmartPrompt功能示例

04 编写开源大事记的思路

05 今后的发展规划

如何理解这一轮AI浪潮？

01

什么是LLM?

- **定义：**
  - LLM是通过深度学习训练的大型神经网络模型，专门用于理解和生成自然语言。
- **工作原理：**
  - 通过大规模的文本数据学习语言结构、语法和语义。
  - 具备处理多种语言任务的能力，如翻译、文章生成、问题回答等。
- **应用场景：**
  - AI对话系统
  - 自动翻译
  - 内容创作
  - 自然语言处理（NLP）

什么是提示词?

- **定义：**
  - 提示词是输入给LLM的文本，用于引导模型生成相关输出或执行特定任务。
- **功能与重要性：**
  - 清晰、具体的提示词能提高模型的回答质量和准确性。
  - 模糊、不明确的提示词可能导致不准确的输出。
  - 不同的模型（开源的/闭源的，小的/大的），针对不同的任务，不同的提示词，可能有不同的效果
- **示例：**
  - 提示词1：“请解释什么是机器学习。”
  - 提示词2：“翻译以下句子为法语：‘Hello, how are you?’”
  - 提示词3：“告诉我这个出错信息是什么意思，我要怎么解决？”

什么是可编程的提示词?

- **定义：**本质上是一种传统架构与AI架构集成起来的**架构模式**
  - 并非所有的问题，都适合用LLM来解决
  - 在选择调用何种LLM时，如何才能灵活切换?

SmartPrompt的设计思路

02

Ruby DSL

- **什么是DSL?**
  - 领域特定语言（DSL）是一种专门为特定问题域设计的编程语言或语法。
  - 与通用编程语言（如Ruby、Python等）不同，DSL通常具有简化的语法，更贴近业务领域的需求。
- **Ruby中的DSL特点：**
  - 简洁性与可读性：Ruby的语法非常简洁、优雅，适合快速开发领域特定的语言。
  - 高效的抽象：通过Ruby的元编程功能，可以创建简洁的DSL，隐藏复杂性。
  - 自然语言风格：Ruby的语法使得DSL通常看起来像自然语言，易于理解和使用。
- **常见应用：**
  - 配置文件：DSL常用于编写配置文件，提升可读性和简洁性。例如，Ruby on Rails的路由配置。
  - 测试框架：RSpec是Ruby中的一个DSL，用于编写易于理解的测试用例。
  - 查询语言：ActiveRecord的查询接口提供了一个Ruby风格的DSL，用于构建数据库查询。

```ruby
# 示例：RSpec测试DSL
describe "Calculator" do
  it "adds two numbers" do
    expect(1 + 1).to eq(2)
  end
end
```

```ruby
# 使用DSL定义任务
scheduler = TaskScheduler.new

scheduler.task("task_1") do
  puts "Executing task 1..."
end

scheduler.task("task_2") do
  puts "Executing task 2..."
end

scheduler.task("task_3") do
  puts "Executing task 3..."
end

# 运行所有任务
scheduler.run
```

用ERB作为提示词模板

- **什么是ERB?**
  - ERB（Embedded Ruby）是Ruby语言中的一种模板引擎，它允许将Ruby代码嵌入到HTML或其他文本格式中。
  - 通过ERB模板，开发者可以将动态内容嵌入到静态页面中，生成动态HTML页面或其他格式的输出。
- **ERB 的工作原理：**
  - 模板语法：在ERB模板中，Ruby代码通过特定的标签（<% %>）嵌入到文本中。
  - <%= %>：执行Ruby代码并将结果插入到输出中。
  - <% %>：执行Ruby代码但不将结果插入到输出中。
  - ERB解析器会将模板中的Ruby代码解析并执行，生成最终的HTML或文本输出。
- **常见应用场景：**
  - Web开发：在Rails等Web框架中，ERB广泛用于生成动态HTML页面，特别适合与数据库或用户输入交互。
  - 邮件模板：生成动态电子邮件内容时，使用ERB模板可以嵌入动态数据（如用户名称、订单信息等）。
  - 报告生成：根据数据生成动态报告，ERB模板可以根据输入的数据结构动态生成报告内容。

模板示例：

Translate the following text from <%= source_language %> to <%= target_language %>:

<%= text %>

The translation should be accurate, maintain the original meaning, and be appropriate for the cultural context of the target language.

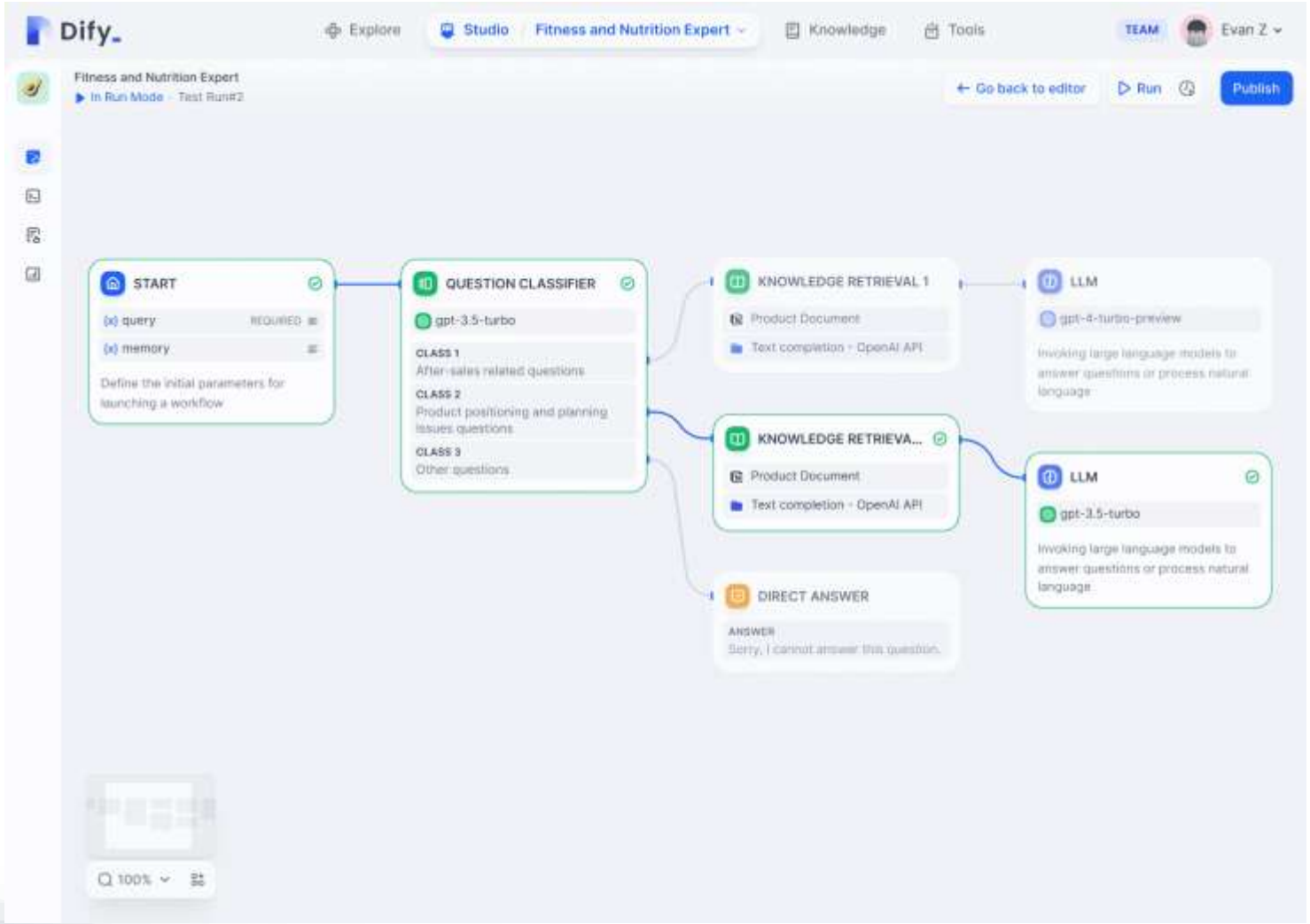
多模型切换

workers > get_code.rb

```ruby
SmartPrompt.define_worker :get_code do
  use "siliconflow"
  sys_msg "You are a helpful programmer."
  model "Qwen/Qwen2.5-Coder-7B-Instruct"
  prompt :generate_code, {
    name: params[:name],
    description: params[:description],
    input: params[:input],
    output: params[:output]
  }
  code_text = safe_send_msg
  if code_text.include?("Failed to call LLM after")
    code_text
  else
    model "meta-llama/Meta-Llama-3.1-8B-Instruct"
    prompt :get_code, {code_text: code_text}
    safe_send_msg
  end
end
```

选择不同的模型

Worker + Workflow



低代码的工作流看起来好看，其实还不如自己写代码来得快。

SmartPrompt功能示例

03

应用的目录结构

- config
  - llm_config.yml
- log
  - log.txt
- templates
  - *.erb
- workers
  - *.rb
- hello.rb
- code.rb
- translat.rb
- embedding.rb
- rag.rb

- ……

```
# gem install smart_prompt
```

config > ! llm_config.yml

```yaml
adapters:
  openai: OpenAIAdapter
  ollama: OllamaAdapter
logger_file: ./log/log.txt
llms:
  siliconflow:
    adapter: openai
    url: https://api.siliconflow.cn/v1/
    api_key: ENV["APIKey"]
    default_model: Qwen/Qwen2.5-7B-Instruct
  llamacpp:
    adapter: openai
    url: http://localhost:8080/
  ollama:
    adapter: ollama
    url: http://localhost:11434/
    default_model: qwen2.5
default_llm: siliconflow
worker_path: "./workers"
template_path: "./templates"
```

翻译

translate_txt.rb > ...

```ruby
require "../smart_prompt/lib/smart_prompt"
engine = SmartPrompt::Engine.new("./config/llm_config.yml")
Words = {}

def translate_text(engine, text)
  checked = true
  count = 0
  result = ""
  while checked == true && count < 10
    puts "try #{count}"
    result = engine.call_worker(:categorized_translation, {
        text: text,
        source_language: "English",
        target_language: "Chinese"})
    checked =
      (result.include?("{") && result.include?("}")) ||
      (result.include?("```") || result.include?("``` json")) ||
      result.include?("**Output**") ||
      result.include?("**输出**") ||
      (result.split("\n").size > text.split("\n").size)
    count += 1
  end
  result
end

book = File.read("../OpenLife.txt")
new_book = File.new("../OpenLife_new.txt", "w+")

book.split("\n").each do |line|
  if line.strip.empty?
    result = line
  elsif Words.has_key?(line)
    result = Words[line]
  else
    result = translate_text(engine, line)
    Words[line] = result
  end
  new_book.puts result
end
new_book.close
```

```ruby
SmartPrompt.define_worker :categorized_translation do
  use "siliconflow"
  model "Qwen/Qwen2.5-Coder-7B-Instruct"
  prompt :cat_trans, {text: params[:text], source_language: params[:source_language]}
  json = send_msg
  model "Qwen/Qwen2.5-72B-Instruct-128K"
  if word?(params[:text])
    prompt :senior_translator, {
      json: json,
      text: params[:text],
      source_language: params[:source_language],
      target_language: params[:target_language]
    }
    result = send_msg
  else
    prompt :senior_translator2, {
      json: json,
      text: params[:text],
      source_language: params[:source_language],
      target_language: params[:target_language]
    }
    result = send_msg
  end
  result
end
```

翻译

templates > cat_trans.erb

```
# Command:
**Input**: <%= source_language %>
**Output**: Analysis
1. **Category**: Based on the content of a given text, determine to which of t
    - Abbreviations and acronyms
    - Dates and times (No translation required)
    - Numerical value (No translation required)
    - Monetary units
    - Units of measurement
    - Chapter headings
    - Proper nouns
    - Literature citations
    - Fixed expressions, phrases, idioms and expressions
    - Titles of literary and artistic works
    - Legal and regulatory provisions
    - Technical terms
    - File name (No translation required)
    - Dialogue, inflections, slang and colloquial expressions
    - Historical and cultural contexts
    - Formatted texts, formulas and symbols
    - URI/URL (No translation required)
    - Other complex texts
2. **Instances**: Under this category, enter actual examples from the text.

# Example:

**Input**: NASA's mission was launched on March 1, 2023, at 10:30 PM.
**Output**(JSON format):
{
  "category": ["Abbreviations and acronyms", "Dates and times"],
  "Instances": {
    "Abbreviations and acronyms": ["NASA (name of organization)"],
    "Date & times": ["March 1, 2023", "10:30 PM"]
  }
}

**Input**: <%= text %>
**Output**(JSON format):
```

templates > senior_translator.erb

```
# Analysis:
<%= json %>

# Command:
1. Output only translated content
2. Please follow the input file format strictly and do not add anything else.
3. Based on the results of the above analysis, translate the following <%= source_language %>, into <%= target_language %>.

# Example:
**Input**: , John Wiley & Sons.
**Output**: ，约翰·威利父子。

# Task:
**Input**: <%= text %>
**Output**(Plain text):
```

templates > senior_translator2.erb

```
# Analysis:
<%= json %>
# Command:
0. The input just one word
1. Output only translated content
2. Please follow the input file format strictly and do not add anything else.
3. Based on the results of the above analysis, translate the following <%= source_language %>, into <%= target_language %>.
# Example:
**Input**: world
**Output**: 世界
# Task:
**Input**: <%= text %>
**Output**(Plain text):
```

抓取新闻

workers > get_news.rb

```ruby
SmartPrompt.define_worker :get_news do
  url = params[:text]
  html = call_worker(:download_page, {url: url})
  text = call_worker(:html_to_text, {html: html})
  use "siliconflow"
  sys_msg "You're a journalist familiar with open source-related news coverage."
  model "Qwen/Qwen2.5-7B-Instruct"
  prompt :analyzing_news_content, {news: text}
  news_json = safe_send_msg
  f = File.open("news.json", "w+")
  f.puts news_json
  f.close
  prompt :generate_sql2, {news: news_json}
  sql = safe_send_msg
  sql
end
```

templates > analyzing_news_content.erb

```
Based on the content of the page provided below, the content of N news items
was analyzed and presented in the following format:

```json
[
    {
        original_title: "",
        chinese_title: "",
        date: "",
        summary: "",
        categories: ["",""]
    },
]
```

Input news:
<%= news %>
```

生成代码

code.rb

```ruby
require "../smart_prompt/lib/smart_prompt"
engine = SmartPrompt::Engine.new("./config/llm_config.yml")
result = engine.call_worker(:get_code, {
  name: "calculate_triangle_area",
  description: "calculates the area of a triangle",
  input: "the base and height of the triangle",
  output: "the area as a float"
})

if result.include?("Failed to call LLM after")
  puts result
else
  code_str = result + "\n" + "puts calculate_triangle_area(8, 10)"
  eval(code_str)
end
```

workers > get_code.rb

```ruby
SmartPrompt.define_worker :get_code do
  use "siliconflow"
  sys_msg "You are a helpful programmer."
  model "Qwen/Qwen2.5-Coder-7B-Instruct"
  prompt :generate_code, {
    name: params[:name],
    description: params[:description],
    input: params[:input],
    output: params[:output]
  }
  code_text = safe_send_msg
  if code_text.include?("Failed to call LLM after")
    code_text
  else
    model "meta-llama/Meta-Llama-3.1-8B-Instruct"
    prompt :get_code, {code_text: code_text}
    safe_send_msg
  end
end
```

tes > generate_code.erb

```
Write a Ruby function called "<%= name %>" that <%= description %>.
The function should accept <%= input %> as input and return <%= output %>.
Just return the source code to me, no other explanations needed.
```

tes > get_code.erb

```
Remove all non-code text and return only the code. Not even characters like ```.

<%= code_text %>
```

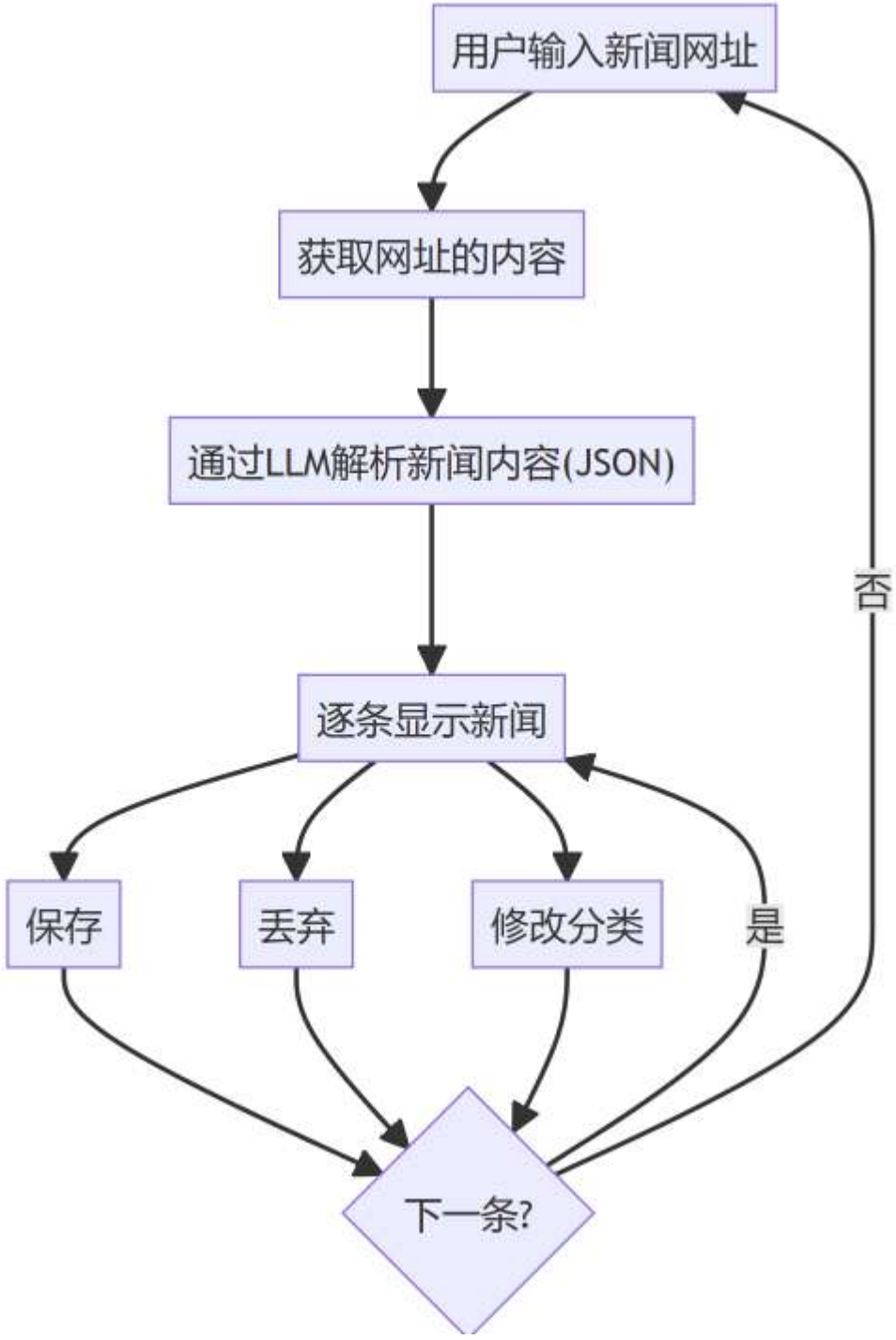
```ruby
def calculate_triangle_area(base, height)
  (base * height) / 2.0
end
puts calculate_triangle_area(8, 10)
40.0
```

编写开源大事记的思路

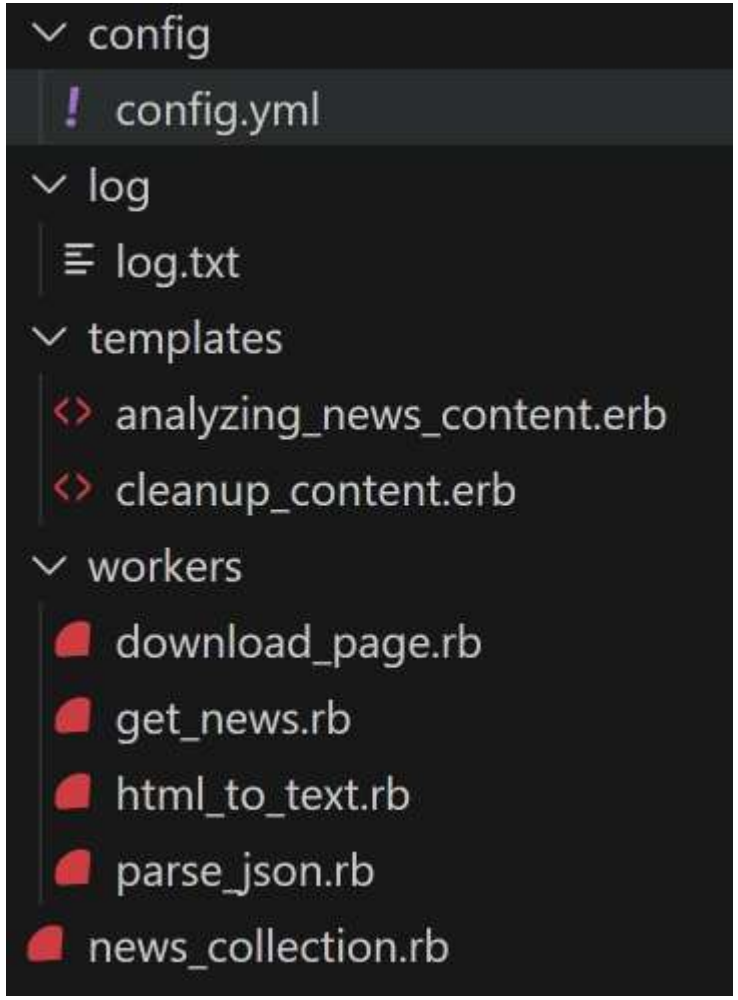
04

news_collection.rb

```
root@zbw-laptop:~/NewsCollection# ruby news_collection.rb
欢迎使用开源新闻收集器
您可以直接输入新闻URL来收集新闻
或输入 'exit' 来退出程序。
> https://mp.weixin.qq.com/s/aYHe9Ra24k2w3csXZr2pVA
无法加载文件：unexpected token at '{
        original_title: "Two Popular Domestic Open Source Front-end Projec
ts Poisoned",
        chinese_title: "两款知名国产前端开源项目被‘投毒’",
        date: "December 19",
        summary: "A severe supply chain poisoning incident has occurred in
 the front-end community, affecting Zan's open-source component library Va
nt and ByteDance's open-source front-end packaging tool Rspack. Multiple v
ersions of these projects were injected with malicious code due to the the
ft of npm tokens. The malicious code included a mining script that mined M
onero on Linux systems and attempted to steal cloud service credentials. B
oth projects have released patched versions to address the issue.",
        category: "开源软件安全"
    }
]
'
> https://mp.weixin.qq.com/s/aYHe9Ra24k2w3csXZr2pVA
原始标题：Two Popular Domestic Open Source Front-End Projects Poisoned
中文标题：两款知名国产前端开源项目被“投毒”
日期：2023-12-19
摘要：前端社区发生了一起严重的供应链投毒事件，有赞开源组件库 Vant 和字节开
源前端打包工具 Rspack 的多个版本被植入了恶意代码。Vant 团队成员的 npm to
ken 被盗用，多个版本被注入恶意脚本并发布至 npm 仓库。Rspack 也因同一 GitHu
b 组织下的另一名维护者的 npm token 被盗，发布了带有相同恶意代码的 1.1.7 版
本。两个项目均已发布修复版本，并清理了相关 token。恶意代码在 Linux 系统上
下载并运行挖矿程序，同时窃取用户的云服务凭据。
分类：开源软件安全
--------------------
请输入操作（S：保存，D：丢弃，E：修改分类，F：修改日期，Q：退出）：
新闻已保存到数据库。
--------------------
>
```



更多代码




```ruby
def call_llm(input)
  result = @engine.call_worker(:get_news, {text: input})
  news_list = @engine.call_worker(:parse_json, {text: result})
  news_list.each do |news|
    unless News.find(title: news[:original_title])
      puts <<~NEWS
        原始标题: #{news[:original_title]}
        中文标题: #{news[:chinese_title]}
        日期: #{news[:date]}
        摘要: #{news[:summary]}
        分类: #{news[:category]}
      NEWS
      puts "--------------------"
      loop do
        puts "请输入操作 (S: 保存, D: 丢弃, E: 修改分类, F: 修改日期, Q: 退出):"
        input = STDIN.getch.upcase
```

更多代码

news_collection.rb | export_db_to_excel.rb

export_db_to_excel.rb > ...

```ruby
class News < Sequel::Model(DB[:news])
end

result = News.all

OUTPUT_FILE = 'news_export.xlsx'
workbook = RubyXL::Workbook.new
sheet = workbook.worksheets[0]
sheet.sheet_name = 'News'
sheet.add_cell(0, 0, 'ID')
sheet.add_cell(0, 1, '标题')
sheet.add_cell(0, 2, '日期')
sheet.add_cell(0, 3, '摘要')
sheet.add_cell(0, 4, '分类')
id = 1
result.each do |news|
  sheet.add_cell(id, 0, news[:id])
  sheet.add_cell(id, 1, news[:title])
  sheet.add_cell(id, 2, news[:date].strftime('%Y-%m-%d'))
  sheet.add_cell(id, 3, news[:summary])
  sheet.add_cell(id, 4, news[:category])
  id = id + 1
end

workbook.write(OUTPUT_FILE)
puts "数据成功导出到 #{OUTPUT_FILE}"
```

演示一下

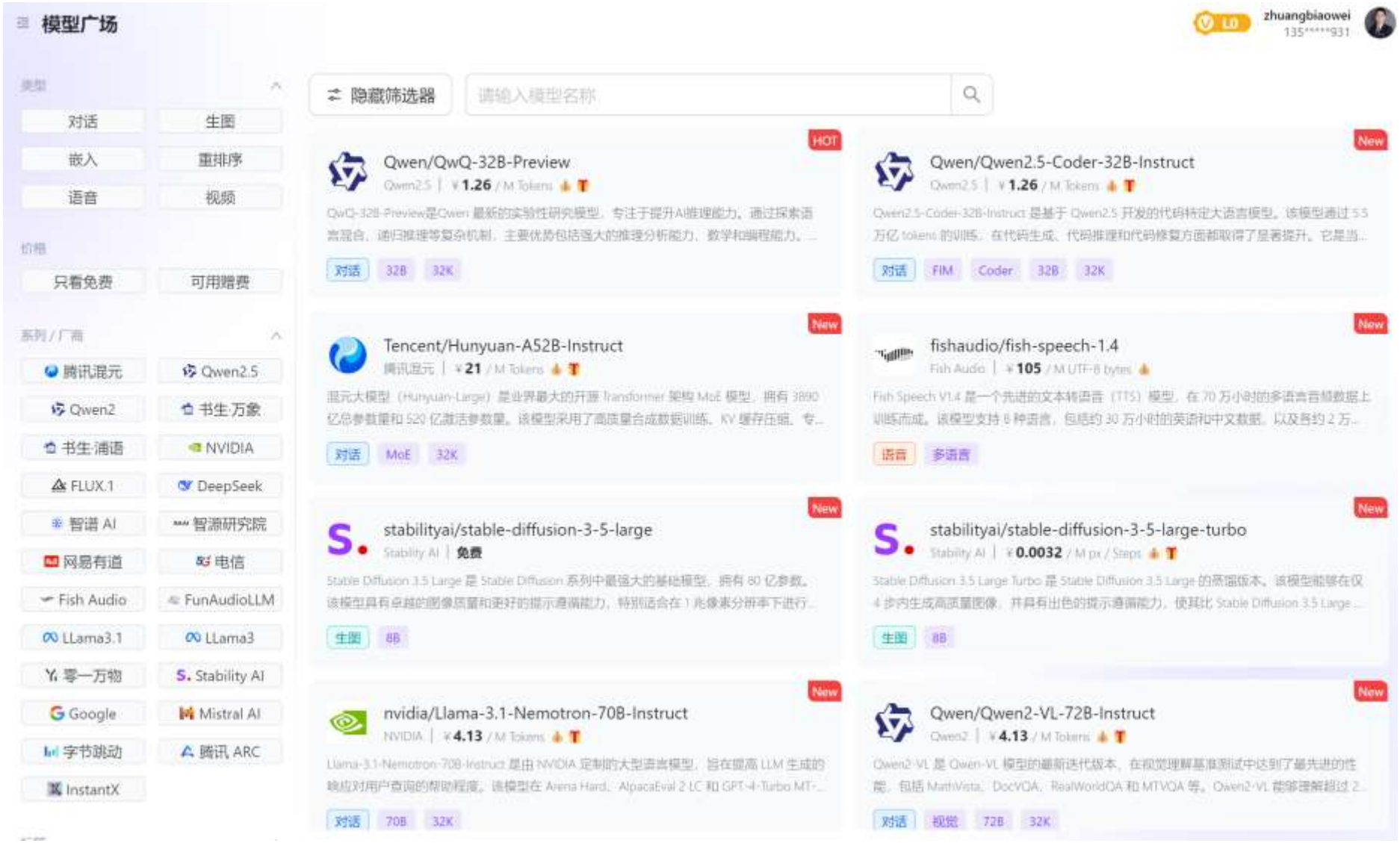
后续工作

- ChatGPT Project？————看来不行，不能读excel文件
- DB-GPT？
- 还可以试试其他的

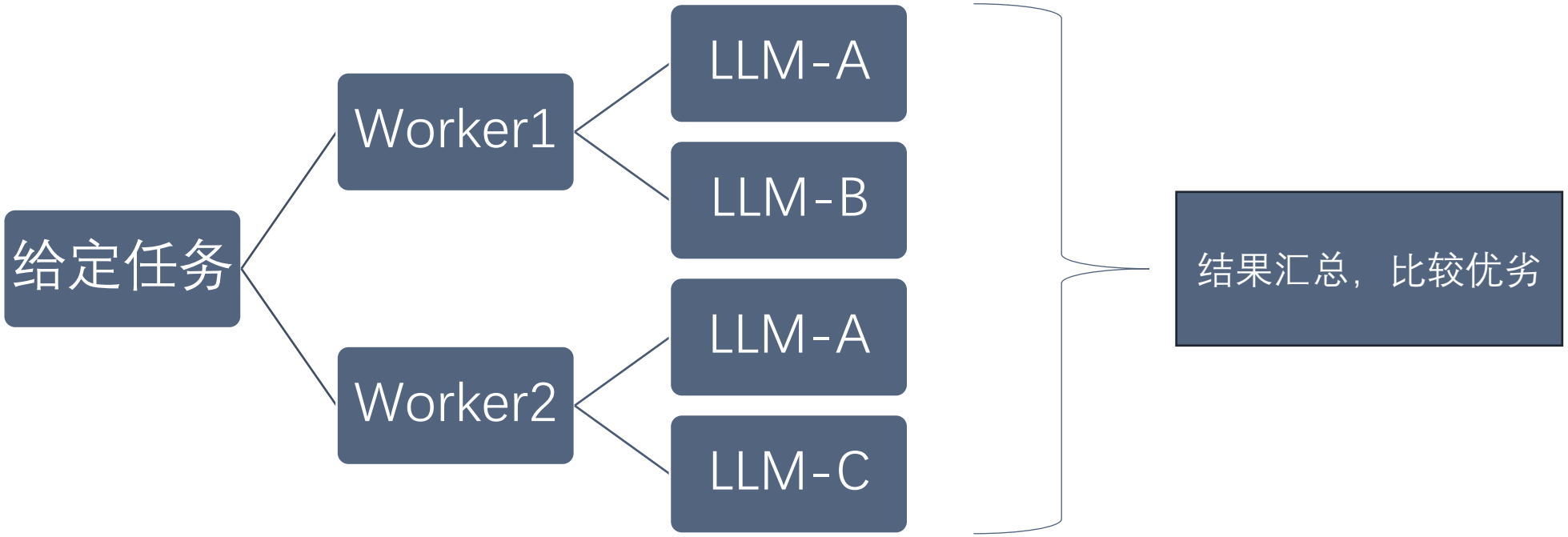
今后的发展规划

04

支持多模态的大模型


模型广场
zhuangbiaowei
隐藏筛选器
请输入模型名称
对话
生图
嵌入
重排序
语音
视频
只看免费
可用赠费
腾讯混元
Qwen2.5
Qwen2
书生·万象
书生·浦语
NVIDIA
FLUX.1
DeepSeek
智谱 AI
智源研究院
网易有道
电信
Fish Audio
FunAudioLLM
LLama3.1
LLama3
零一万物
Stability AI
Google
Mistral AI
字节跳动
腾讯 ARC
InstantX
HOT
Qwen/QwQ-32B-Preview
¥1.26 / M Tokens
New
Qwen/Qwen2.5-Coder-32B-Instruct
¥1.26 / M Tokens
Tencent/Hunyuan-A52B-Instruct
¥21 / M Tokens
fishaudio/fish-speech-1.4
¥105 / M UTF-8 bytes
stabilityai/stable-diffusion-3-5-large
免费
stabilityai/stable-diffusion-3-5-large-turbo
¥0.0032 / M px / Steps
nvidia/Llama-3.1-Nemotron-70B-Instruct
¥4.13 / M Tokens
Qwen/Qwen2-VL-72B-Instruct
¥4.13 / M Tokens

多个Worker、多个模型并发



还需要一种调度多个Worker的更好的DSL

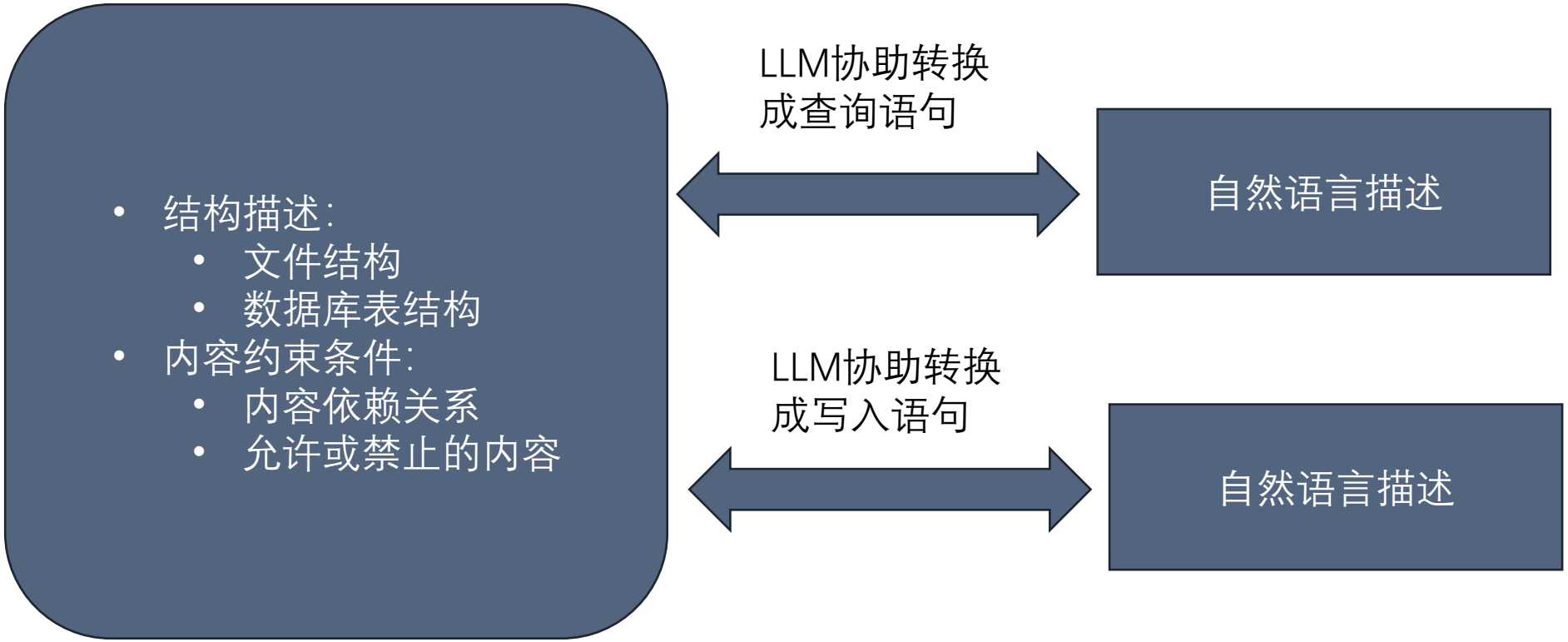
对于Worker的输出结果进行调优

现在的worker里，写死了模型名称，也有明确调用的模板。

如果能够定义：“理想的输出结果”，然后针对可选的模板，反复优化模板的提示词，直到找到最佳组合。

效果就会更加理想。

系统级Worker：文件管理、数据库管理等



从SmartPrompt到SmartAgent

前置说明：我会给你发布清晰的任务说明，我发布的任务分为三类：

- 知识/逻辑类：这类任务的输出结果有对错之分
- 场景/应用类：这类任务的输出结果有好坏之分
- 艺术/创意类：这类任务的输出结果只有美丑之分（而且是因人而异的）

任务类型：场景/应用类

具体任务类型：帮我完善一个系统的架构设计

依赖知识：

- 这是一个Ruby语言框架

任务背景：

- 这个框架用于更好的调用各种大模型
- 输入的是一个大致的设计思路，还比较模糊

任务输入：
我想用ruby语言设计一个名叫SmartAgent的框架，大概的思路如下，请给我一些反馈意见，帮助我完善这个设计。

- 用户能够借助这个SmartAgent的框架，完成一些非常复杂的工作。
- 任务首先被分为简单任务和复杂任务。
    - 简单任务分为：适合用大模型完成的任务，适合用代码完成的任务。
        - 适合用大模型完成的任务，需要借助SmartAgent的框架，判断选择哪一个合适的大模型，选择（或生成）对应的合适的提示词。
        - 适合用代码完成的任务，需要借助SmartAgent的框架，判断是直接调用已经存在的代码，还是下载网络上的某个gem，还是通过LLM生成一段代码。
    - 复杂的任务，需要借助大模型的能力，将任务分解为多个子任务，并以一种适当的DSL，描述这些子任务如何在一起协作
- 代码类的任务，可以再进一步细分
    - 网络类任务：HTTP协议访问Web、HTTP协议调用RESTful API
    - 数据类任务：查询数据、理解数据结构、操作数据结构，操作数据内容（添加、修改、删除）
    - 文件类任务：单个文件的读写、一个文件夹下的多个文件的读写
    - 运算类任务：主要是指各种各样的算法（函数）
- 这个系统对于大模型的应用，主要按照输入输出的信息类型来做分类
    - 无论输入还是输出，信息类型都包括：文本、数据、图像、视频、音频。
    - 例如：输入“请计算1加到10的总和”，系统可以按如下方式处理：自然语言（文本）->生成计算代码（文本）->执行代码得到结果（数据）->自然语言告知结果（文本）

执行思路：请帮我深入分析这个设计，一步一步的考虑各种情况与应用场景，同时考虑可能存在的例外情况，经过你的反思、综合、整理，最后告诉我一个你认为最完善的回答。

输出要求

输出要求--内容约束：

- 一份结构清晰、完整、全面的设计稿
- 这个设计稿，还应该加上你认为必要的评注与说明

输出要求--格式约束：markdown

看看ChatGPT和Claude给我的提示

谢谢大家

汇报人：庄表伟

时间：2024.12