低代码平台AIGC开发落地实战

腾讯 前端技术专家 / 姜天意

About me

2013-2020

阿里: B2B，盒马

前端工程链路
数据可视化与低代码

Node.js framework eggjs

2020-2023

腾讯: 医疗健康事业部-云产品二部

ToB大数据及数字孪生产品研发与前端架构
开源低代码平台逻辑可视化方向PMC

https://github.com/jtyjty99999

Hi there, this is Adams👋 Follow 610

About Me

jtyjty99999 | jtyjty99999 | jtyjty99999@126.com | Profile views 148

- 🔭 I'm currently working on Tencent Healthcare big data front-end team.
- 🌱 I'm a full stack developer with experience in building websites.
- 👯 I'm familiar with large-scale website development and front-end architecture, also experience in design and development of data products.
- 🤔 I'm familiar with product manage , responsible for team management in both Alibaba and Tencent.
- 💬 I'm experienced in HTML, CSS 3, React , Node.js and Data visualization.

Languages and Frameworks

JavaScript | React | HTML5 | CSS3 | Node.JS | Graphql | Amazon AWS | D3.js | ThreeJs

Public Contributions or Maintains

- core team of eggjs Stars 18k
- bizGoblin: react mobile charts based on f2 Stars 129
- next: react components and theme roller Stars 4.2k
- egg plugins like egg-security Stars 226, egg-logrotator Stars 51, egg-mysql Stars 302, egg-mongoose Stars 417, egg-redis Stars 255, egg-graphql Stars 371, egg-oss Stars 81, egg-sequelize Stars 579, egg-view-react Stars 89
- mobile dev knowledge Stars 3.3k

Public Share

- 2022 GIAC Logic lowcode programming best practise
- 2018 imweb Fragile Node.js
- 2017 GIAC Bpmn based front-end architecture and devops
- 2013 NYC open meetup New York Subway data visualization
- 2013 webrebuild step into data visualization
- 2012 w3cplus YY public share f2e auto test tools

Translations or Writes

- Web Component 实战
- SVG 动画
- 知乎

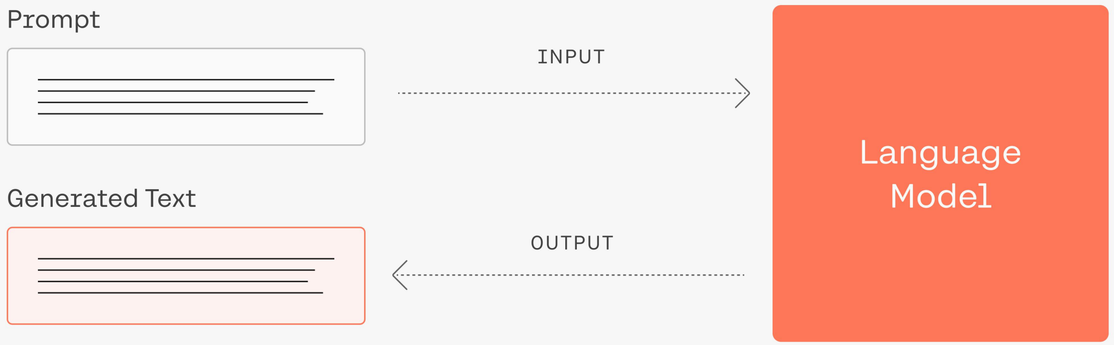
目录

- LLM Prompt engineering 与低代码技术介绍
- 低代码模块接入 GPT 实战
- GPT 高级技巧与个人感想

低代码与 LLM 提示工程介绍

提示工程–什么叫提示工程–人说人话，GPT 说 GPT 话

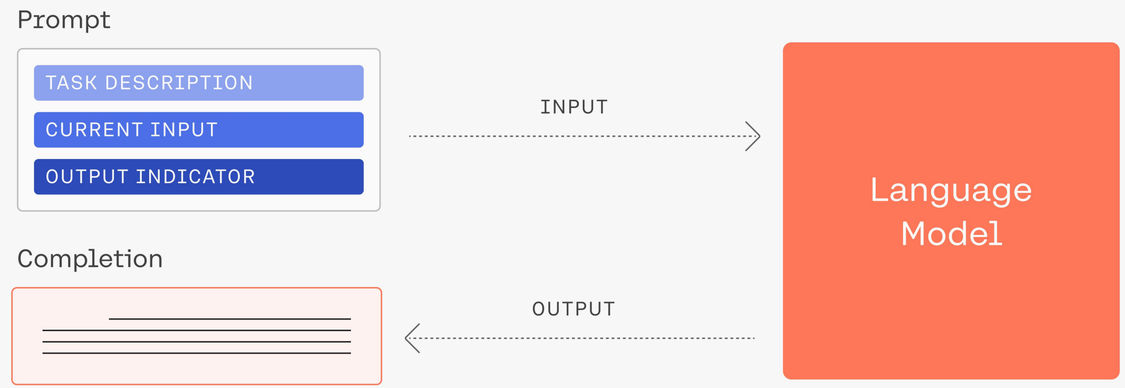
提示工程是通用技术，适合于几乎所有大语言模型（Large Language Models，简称LLMs）



设计最佳提示以指导模型执行任务的方法就是所谓的提示工程(prompt engineering)

prompt engineering 做得好 不仅可以提升回答的质量，也可以限制回答的格式

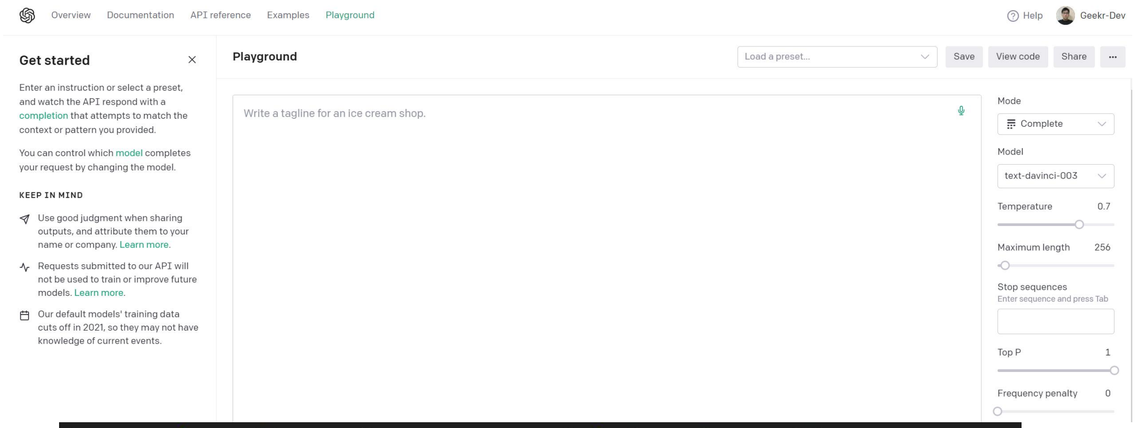
这对后续跟其他系统的集成非常重要



- 指令 – 希望模型执行的特定任务或指令
- 上下文 – 可以包含外部信息或额外的上下文，以引导模型更好地进行响应
- 输入数据 – 我们感兴趣并希望找到响应的输入或问题
- 输出指示符 – 指示输出的类型或格式

如何使用你的openai/chatgpt做提示工程测试

https://platform.openai.com/playground




```
import { Configuration, OpenAIApi } from "openai";
const configuration = new Configuration({
    apiKey: config.apiKey,
});
const openai = new OpenAIApi(configuration);
const prompt = '请告诉我世界上最帅的人是谁';

const res = await openai.createCompletion({
    // 对话模型
    model: "text-davinci-003",//  dialogue-babi-001 对话模型
    prompt: prompt,
    max_tokens: 4000,
    temperature: 0.2
})

console.log(res);
```

- temperature（温度） ß ß 简言之，温度越低，结果越具确定性，因为总是选择概率最高的下一个词。提高温度可能导致更多的随机性，鼓励产生更多样化或富有创意的输出**对基于事实的问答任务使用较低的温度值**，以鼓励更加准确和简洁的回答。对于诗歌生成或其他**创意任务，提高温度值可能会更有益**。
- top_p ß ß 类似地，通过 top_p 调节名为 nucleus sampling 的温度抽样技术，可以控制模型在生成回应时的确定性。**如果需要准确和事实性的答案**，请保持较低的 top_p 值，如果**希望获得更多样化的回应**，请增加到较高的 top_p 值。

https://medium.com/@basics.machinelearning/temperature-and-top-p-in-chatgpt-9ead9345a901

```
In [7]: prompt = f"""
        说明穿秋裤的好处，不超过50个字
        """
        response = get_completion(prompt, temperature=0)
        print(response)
```

穿秋裤可以保暖，防止感冒，提高身体免疫力，增强体力，舒适度高，适合户外活动。

```
In [10]: prompt = f"""
         说明穿秋裤的好处，不超过50个字
         """
         response = get_completion(prompt, temperature=1.5)
         print(response)
```

穿秋裤可以御寒保暖，增加舒适度，特别是对于凉易脱的人，还能避免感冒等卫生问题。

text-davinci-003 和 gpt-3.5-turbo 有啥区别?

text-davinci-003 和 gpt-3.5-turbo 都是 OpenAI GPT-3 系列中的两个模型，区别在于性能和价格性能：gpt-3.5-turbo 相对于 text-davinci-003 在性能方面有所提高。根据 OpenAI 的说法，gpt-3.5-turbo 在许多任务中的性能与 text-davinci-003 相当或更好。这意味着，与 text-davinci-003 相比，gpt-3.5-turbo 可以在保持相同质量输出的同时更有效地完成任务。价格：gpt-3.5-turbo 的价格相对于 text-davinci-003 更具竞争力。使用 gpt-3.5-turbo 的成本约为使用 text-davinci-003 的 1/4。这使得 gpt-3.5-turbo 成为许多应用和开发场景中更实惠的选择。

https://blog.apilayer.com/chatgptapi-vs-davinci-which-ai-api-is-right-for-you/

openai中 role 的解释与4096 token 的限制

chat.js

```
import { Configuration, OpenAIApi } from "openai";

const configuration = new Configuration({
  apiKey: process.env.OPENAI_API_KEY,
});
const openai = new OpenAIApi(configuration);

const completion = await openai.createChatCompletion({
  model: "gpt-3.5-turbo",
  messages: [
    { role: "system", content: "你是一个非常实用的 AI 助手" },
    { role: "user", content: "2022世界杯在哪儿举行？" },
    { role: "assistant", content: "2022年世界杯将在卡塔尔举行。" },
    { role: "user", content: "冠军是谁？" },
  ],
});

console.log(completion.data.choices[0]);
```

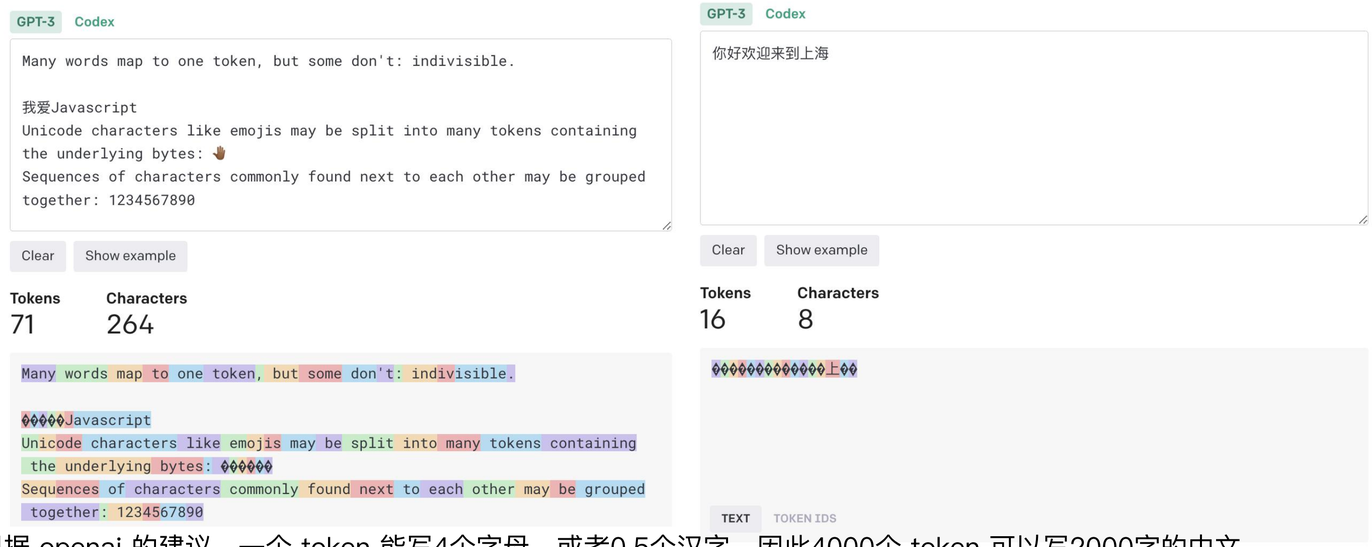
系统消息(role = system)一般用于定义 GPT 对话的行为，比如：你是一个 SQL 工程师，擅长写 SQL。
gpt-3.5-turbo-0301 并不会把这个系统消息做很高的优先度关注。未来的模型将被训练为更加关注系统消息。

用户消息(rule=user)一般是用户自己的输入以及开发者给 GPT 的一些指令。

助手消息(rule=assistant) 可以帮助存 GPT 自己的响应。

当对话场景需要引用之前的消息时，就需要维护好这个message数组，这就是所谓 GPT 对话的上下文 or 历史记录。
大模型对过去的请求没有记忆，所以你如果想保持这个上下文，必须维护好这个 message，然后在每次对话过程中，把完整的 message 传过去。
因此，如果这些记录超过了模型的 token 限制，就需要以某种方式缩短它。

GPT 家族的大语言模型会使用 Token 作为处理文本的标识，用来标识一段文本中的“词”。大语言模型会理解这些 Token 之间的关系，并能够预测一系列 token 中的下一个



根据 openai 的建议，一个 token 能写4个字母，或者0.5个汉字。因此4000个 token 可以写2000字的中文。

输入和输出都会计算 Token 。比如API 调用在消息输入中使用了 10 个Token，而您在消息输出中收到了 20 个 Token，则需要支付 30 个 Token 的费用。如果在调用时候超了，会调用失败。输出的时候超了，会被截断。因此你自己的 GPT 应用， 一定要实现“试算 Token”的功能。

模型名称	文本最长输入
gpt-3.5-turbo	4096个tokens（大约5页）
gpt-4	8192个tokens（大约10页）
gpt-4-32k	32768个tokens（约40页）

https://platform.openai.com/docs/models/moderation

在一段时间之内，“省 Token ”都是 AI 应用需要关注的重要问题

提示工程进阶–少样本提示(few shot)

通过多次提供 shots，可以发现gpt的回答更加精准

The three settings we explore for in-context learning

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
cheese =>                           ← prompt
```

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← example
cheese =>                           ← prompt
```

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

```
Translate English to French:        ← task description
sea otter => loutre de mer          ← examples
peppermint => menthe poivrée        ←
plush girafe => girafe peluche      ←
cheese =>                           ← prompt
```

大模型自身
具有很强的零样本能力，
但在复杂的任务上表现并不好

大语言模型
具备强大的少样本学习能力
通过提供很少的例子
可以自我学习

few shot 的方式，很适合某些可枚举场景的问答，学习

提示工程进阶–思维链

解决更复杂的算数/推

(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

(Output) The answer

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?
A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?
A:

原始提示中添加a逐步思考iG请一步步思考问题C

少样本思维链提示

16 balls. Half of the balls are golf lf balls. Half of the golf balls are golf balls. The answer is 4. ✓

Q: A juggler can jug and half of the golf b there?
A: The answer (arab

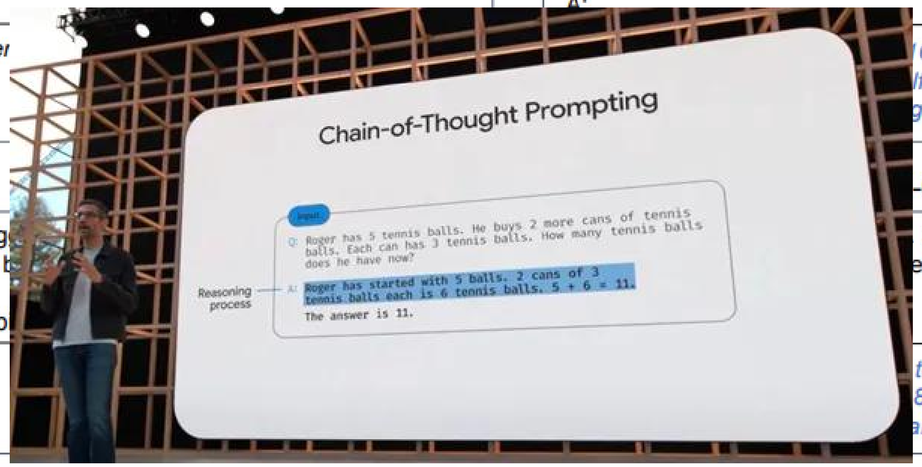
(Output) 8 ✗

-CoT (Ours)

Half of the balls are golf balls, e. How many blue golf balls are

零样本思维链提示

total. Half of the balls are golf 8 golf balls. Half of the golf balls are 4 blue golf balls. ✓



, 13, 82, 7, 1。这是对的吗，请回答是或不是

答案为是。奇数15、

82, 7, 1。这是对的吗，请回答是或不是
奇数，将它们相加，并指出结果是奇数还是偶

所以，原来的说法是不正确的。

May 10, 2023 2:12 PM

一共有12只动物，一半是鸡，一半是雄鸡，问雄鸡的数量

由题可知，一共有12只动物，其中一半是鸡，即6只是鸡。又因为一半是雄鸡，所以雄鸡的数量也是6只。因此，雄鸡的数量为6只。

May 10, 2023 4:49 PM

4:50:22 PM

一共有12只动物，一半是鸡，一半是雄鸡，问雄鸡的数量，请一步步思考下

首先，我们知道一共有12只动物，那么一半就是6只动物。

其次，我们知道一半是鸡，那么就有6只鸡。

再次，我们知道一半是雄鸡，那么就有3只雄鸡。

因此，雄鸡的数量是3只。

May 10, 2023 4:50 PM

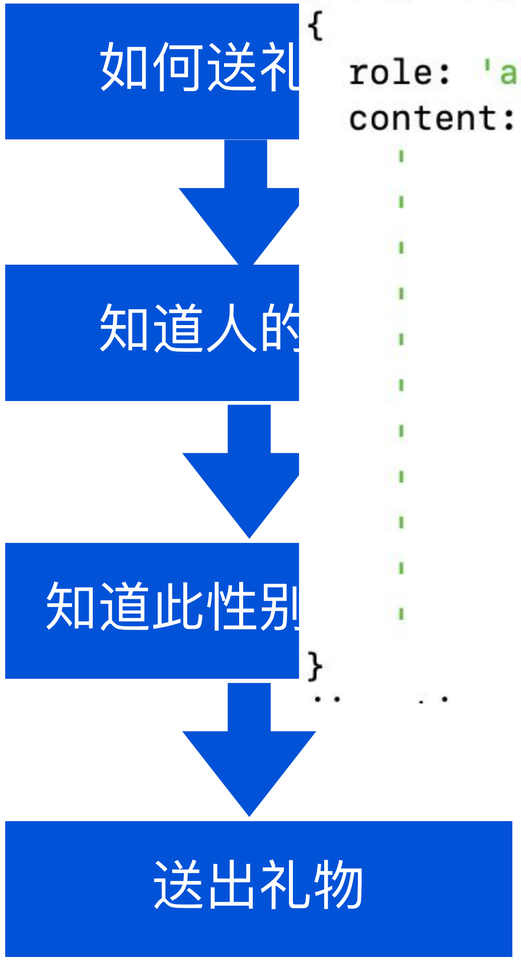
少样本提示不一定能获得推理类问题的可靠答案。

介绍的任务类型涉及到更多的推理步骤。
换句话说，如果将问题分解成多个步骤并向模型演示，可能会有所帮助。

提示工程进阶–思维链高级应用–决策器

最大化的让 AI 一步一步的思考

https://tsmatz.wordpress.com/2023/03/07/react-with-openai-gpt-and-langchain/




```
const msg = [
    {
        "role": "system", "content": `
        >>> 按照给定的格式回答以下问题，你可以使用下面这些工具:
```

```
{
  role: 'assistant',
  content: '          ---\n' +
    '          Question: 我应该送什么礼物给小帅？ \n' +
    '          Thought: 要回答这个问题，我需要先知道小帅的性别\n' +
    '          Action: 查询人物性别\n' +
    '          Action Input: 小帅\n' +
    '          Observation: 小帅是男性\n' +
    '          Thought: 男性可能会对某些特定的物品更感兴趣，需要根据性别推荐商品\n' +
    '          Action: 根据性别推荐商品\n' +
    '          Action Input: 男性\n' +
    '          Observation: 男性可能会对高达、手办、相机等物品更感兴趣\n' +
    '          Thought: 综合上述结果，我认为送小帅高达、手办、相机等物品较为合适\n' +
    '          Final Answer: 送小帅高达、手办、相机等物品'
}
```

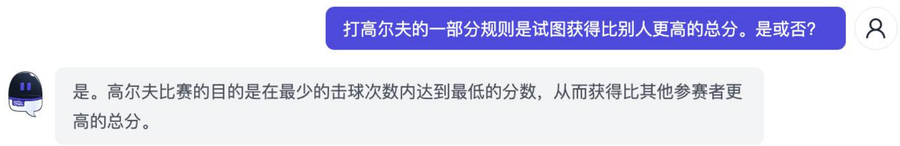
```
    `},
]
```

Reasoning + Acting：ReAct 每次让 LLM 输出一个当前的【思考】和【要做的动作】，这个动作并不只限于检索信息，可以是调用任何工具，类似 ChatGPT plugin。LLM 通过 few shot 的例子和工具自带的描述、还有它自己学到的常识来决定何时调用工具以及如何调用工具。

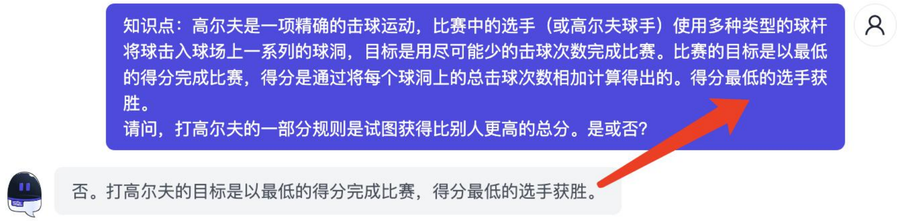
这样 ReAct 引入了外部工具的概念，让 LLM 能够通过这种步进式的方式逐步思考并调用外部工具，根据结果进一步思考循环。同时也可以仅仅是输出一步思考，并继续下去，类似CoT。

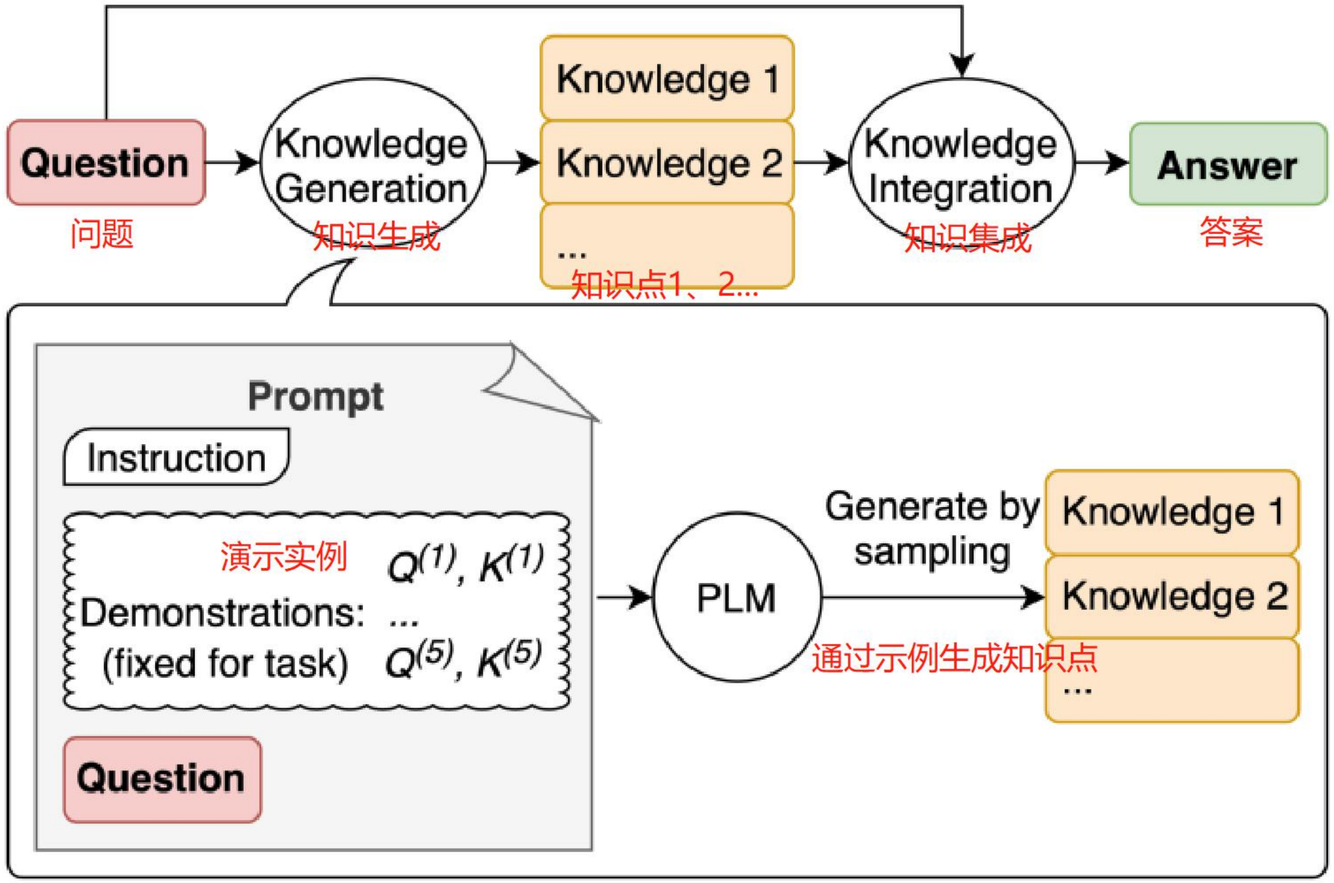
提示工程进阶–生成知识

让模型懂得更多



生成知识提示是一种利用语言模型自动生成知识并整合到常识推理中的方法，这种方法利用了大语言模型的优势，将其作为改进常识推理的、灵活的外部知识来源。通过使用通用的提示格式直接从语言模型中生成知识陈述，然后选择与给定任务相关的知识，可以提高常识推理的准确性。这种方法在语言生成、自然语言推理和问答系统等领域具有广泛的应用前景。





生成知识技术结合 few shot 是很多 AI +低代码技术的基石，也是构建问答系统、知识库等产品的重要技术依赖

提示工程进阶–生成知识

让模型懂得更多

请用某种语法表达a流转到b流转到c的流程

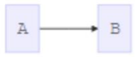
a -> b -> c

graph LR
A –> B

表示如下一个从左到右的图。

```
graph LR
    A --> B
```

mermaid



GPT 学会了我的技能（代替了我）
造 DSL 再也不能称为门槛了！

我现在使用一段json数据表示流程，其中node属性表示流程中的每个节点，link属性表示节点之间的关系, link属性中source表示起点，target表示终点。 举例来讲，流程中a流转到b可以表示为 {nodes:[{name:'a'},{ name:'b'}], links: [{source:'a', target: 'b'}]}

非常好，您已经使用了JSON格式来表示流程中的节点和节点之间的关系。这种表示方法非常直观和易于理解。其中，节点可以用一个对象来表示，对象中可以包含节点的名称等属性。节点之间的关系可以用一个数组来表示，数组中的每个元素表示一个关系，其中source属性表示起点，target属性表示终点。这种表示方法非常灵活，可以用于各种类型的流程。

May 17, 2023 3:29 PM

请用某种语法表达a流转到b流转到c的流程

如果使用JSON格式来表示a流转到b流转到c的流程，可以这样表示：

SQL

```
{
  "nodes": [
    { "name": "a" },
    { "name": "b" },
    { "name": "c" }
  ],
  "links": [
    { "source": "a", "target": "b" },
    { "source": "b", "target": "c" }
  ]
}
```

其中，nodes属性表示流程中的节点，links属性表示节点之间的关系。在这个例子中，有三个节点，分别是a、b、c，a流转到b，b流转到c。因此，links属性中有两个元素，分别表示a到b的关系和b到c的关系。

May 17, 2023 3:29 PM

如何编写提示工程的总结

普通原则

- 从最基本，最原子的任务做起，将大任务分解成多个子任务
- 提示内容最好包含：指令、上下文、输入、输出格式
- 对于希望模型执行的指令和任务要非常具体。提示越具描述性和详细性，结果越好。最好带上例子
- 不要写太多废话，尽量精炼，不要带前后矛盾的提示。

进阶技巧

- 少样本提示（Few-Shot Learners）。给出一些样例，引导模型按照样例回答。
- 思维链（Chain-of-Thought（CoT）prompting）。通过提供推理步骤，让大语言模型具备分析能力。过程中也提到了（Zero-Shot CoT），引导模型在回答的时候，给出推理步骤，会更容易获得理想的结果。
- 生成知识（Generated Knowledge Prompting）。对于一些大模型不掌握的知识。我们可以通过提示的形式输入进去，从而获得更准确的答案。

https://github.com/promptslab/Awesome-Prompt-Engineering

关于提示工程更多的介绍可以学习 https://learnprompting.org/zh-Hans/

langchain–每个g

```
from langchain.prompts import PromptTemplate
from langchain.llms import OpenAI
from langchain.chains import LLMRequestsChain, LLMChain

llm = OpenAI(model_name="gpt-3.5-turbo", temperature=0)

template = """在 >>> 和 <<< 之间是网页的返回的HTML内容。
网页是新浪财经A股上市公司的公司简介。
请抽取参数请求的信息。

>>> {requests_result} <<<
请使用如下的JSON格式返回数据
{{
  "company_name":"a",
}}
Extracted:"""

prompt = PromptTemplate(
    input_variables=["requests_result"],
    template=template
)

chain = LLMRequestsChain(llm_chain=LLMChain(llm=llm, prompt=prompt))
inputs = {
  "url": "https://vip.stock.finance.sina.com.cn/corp/go.php/vCI_CorpInfo/stockid/600519
}

response = chain(inputs)
print(response['output'])
```

```
{
  "company_name":"贵州茅台酒股份有限公司",
  "company_english_name":"Kweichow Moutai Co.,Ltd.",
  "issue_price":"31.39",
  "date_of_establishment":"1999-11-20",
  "registered_capital":"125620万元(CNY)",
  "office_address":"贵州省仁怀市茅台镇",
  "Company_profile":"公司是根据贵州省人民政府黔府函〔1999〕291号文,由中国贵州茅台酒厂有限责任公司作为主发起人,联合贵州茅台酒厂技术开发公司、贵州省轻纺集体工业联社、深圳清华大学研究院、中国食品发酵工业研究院、北京市糖业
}
```

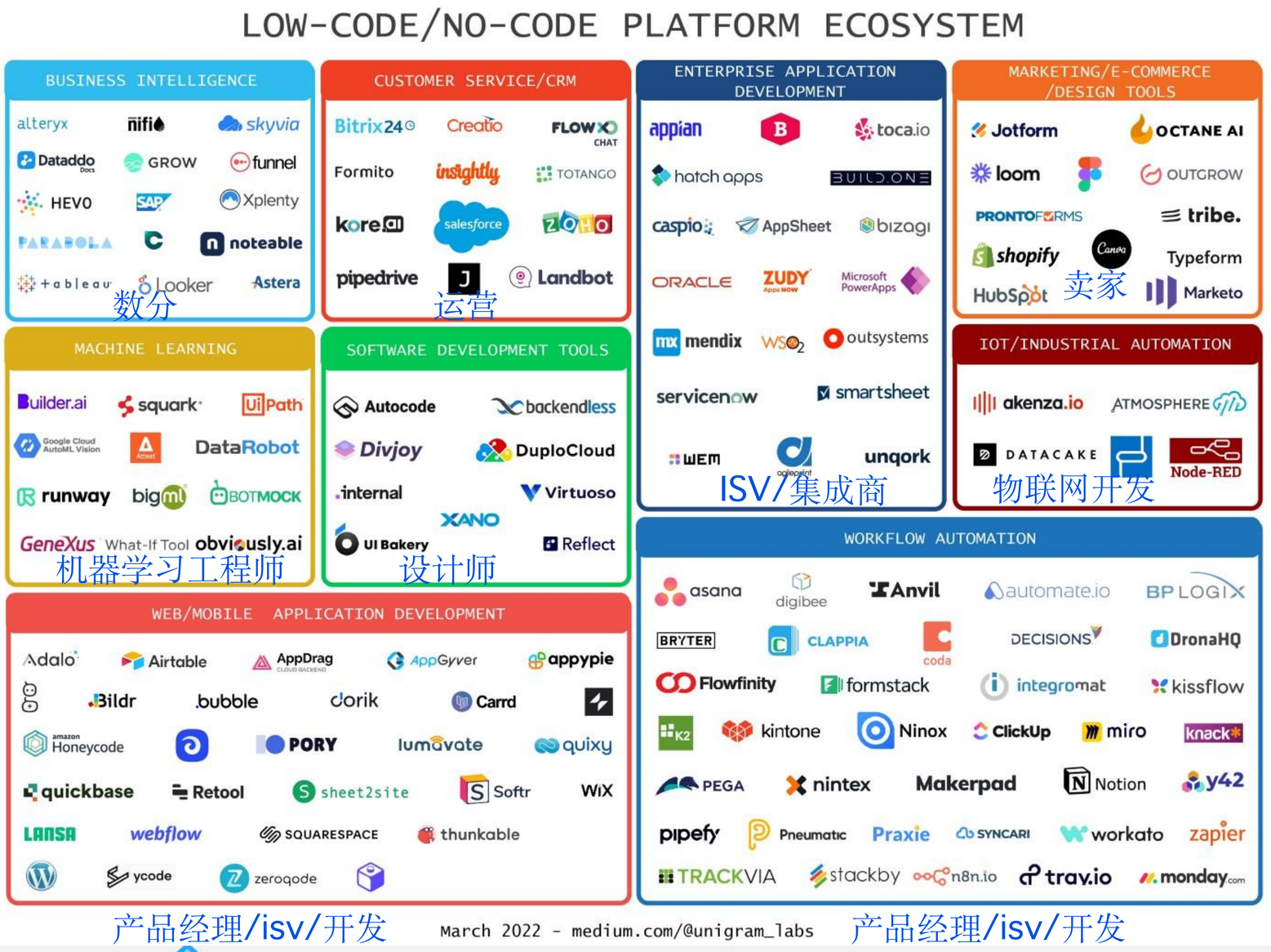
Chains 是 LangChain 中最重要的
如，我们可以创建一个链，它接收
组件或者其他链。我们也可以通过

LangChain 实现了很多现成的链，
Answering with Sources 链，用
解析网页的 LLMRequestsChain

某个特定任务的应用程序。例
然后将LLM 的输出传递给后端

Answering/Question
with Sources 链，用于获取并

低代码介绍-产品矩阵



在国外，低代码类产品矩阵非常庞大。medium.com 将低代码类产品按照功能分为以下几类：

BI 数据可视化图表搭建

CRM 类应用搭建

机器学习类应用搭建

企业应用搭建

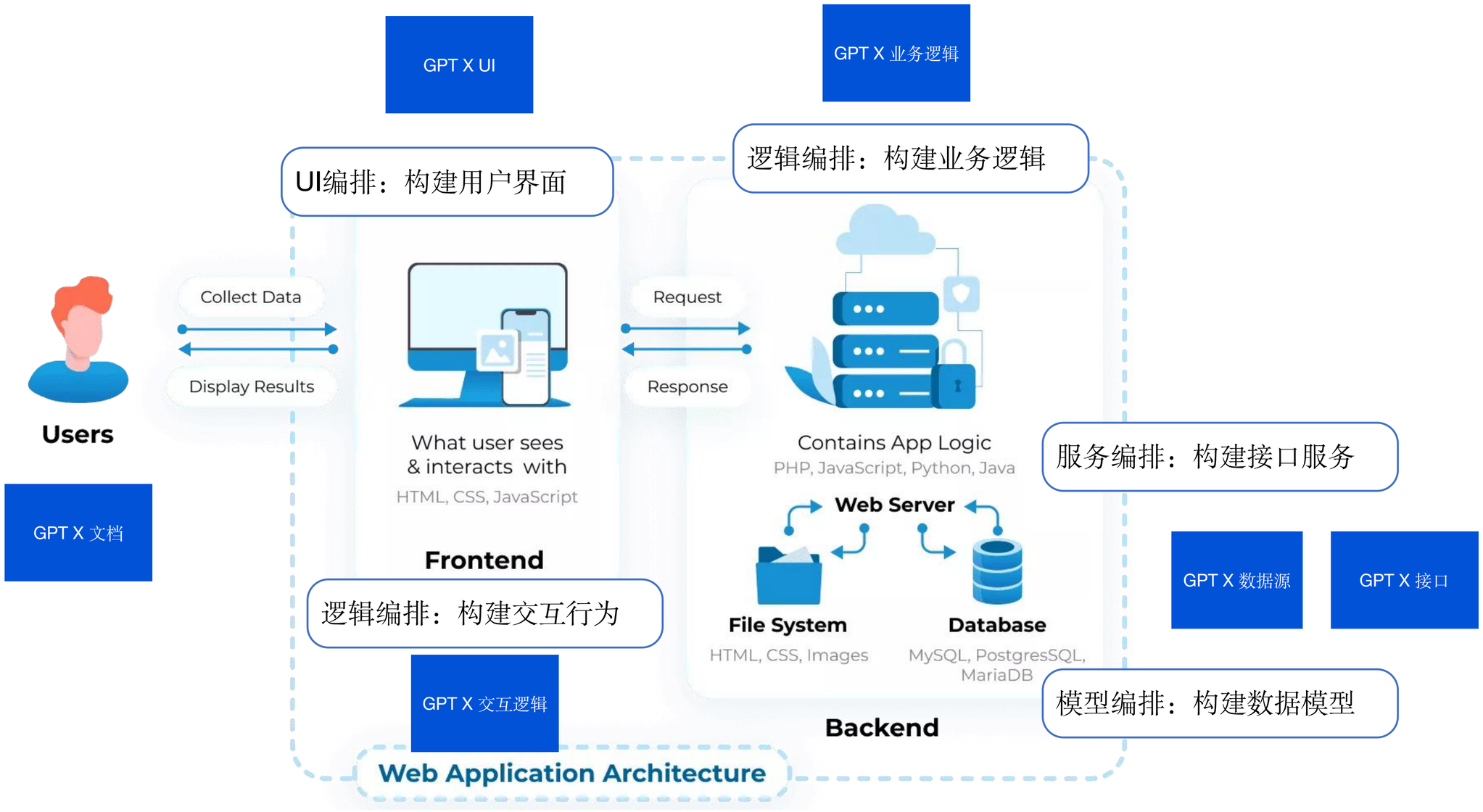
店铺装修工具

物联网、IOT 构建工具

Web、Mobile 应用开发

工作流系统

低代码与GPT


GPT X UI
GPT X 业务逻辑
UI编排：构建用户界面
逻辑编排：构建业务逻辑
Collect Data
Display Results
Request
Response
Users
What user sees
& interacts with
HTML, CSS, JavaScript
Contains App Logic
PHP, JavaScript, Python, Java
服务编排：构建接口服务
Web Server
GPT X 文档
GPT X 数据源
GPT X 接口
Frontend
逻辑编排：构建交互行为
File System
HTML, CSS, Images
Database
MySQL, PostgresSQL,
MariaDB
模型编排：构建数据模型
GPT X 交互逻辑
Backend
Web Application Architecture

实战：根据低代码不同场景构建 Prompt

分解任务

构造 Promopt

获取openai的输出并解析

GPT X 数据源开发–prompt 的 hello world

分解任务

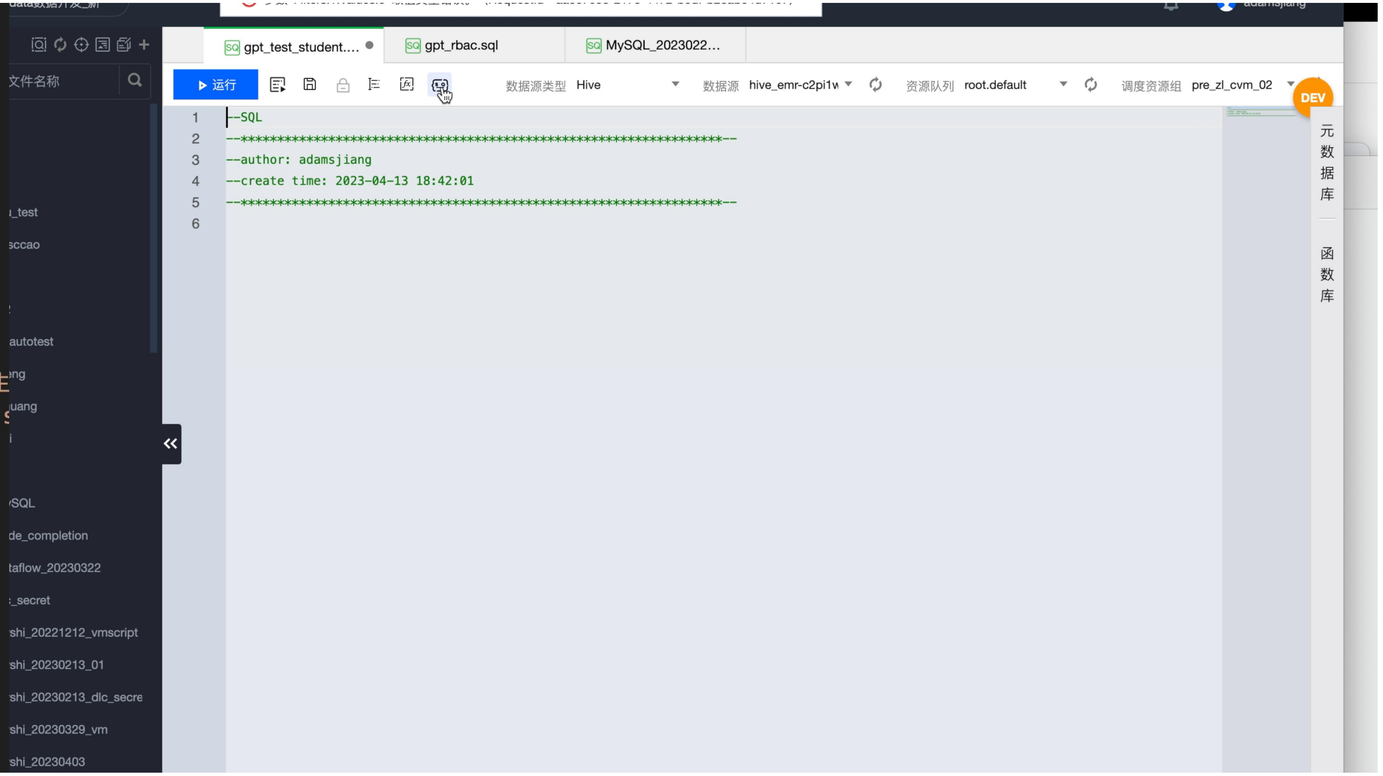
- 目标：希望能够根据需求自动生成 SQL
- 我需要获取目前的库表信息，之后根据 SQL 知识构建 SQL

构造 Promopt

- 指令 ß ß 希望模型输出 SQL
- 上下文 ß ß 当前在哪个库，哪个表
- 输入数据 ß ß 表结构 –DDL
- 输出指示符 ß ß 我希望输出纯正 sql，不想解析一堆内容

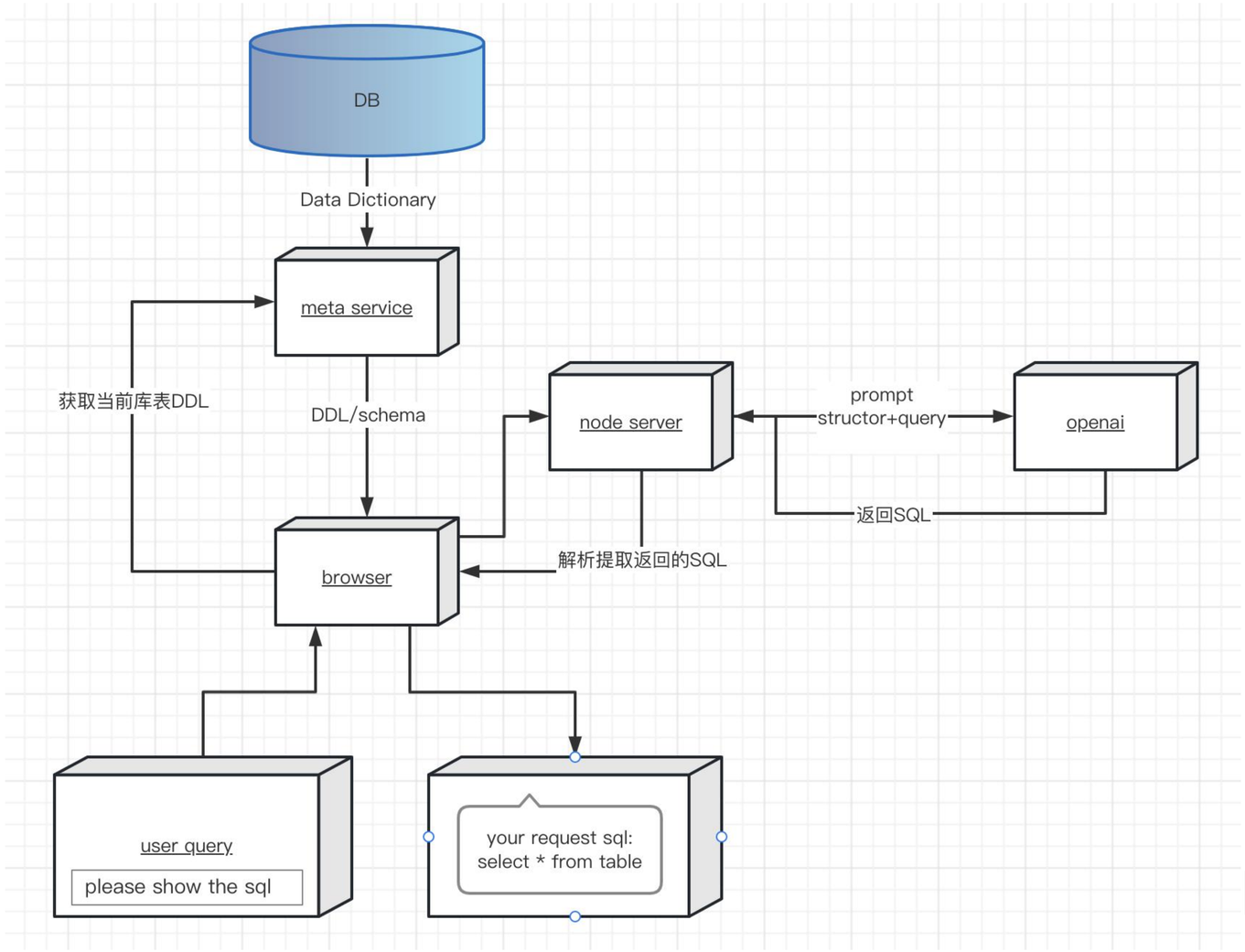
```
import { Configuration, OpenAIApi } from "openai";
import config from "./config.js";

// https://platform.openai.com/docs/api-reference/images

const configuration = new Configuration({
    apiKey: config.apiKey,
});
const openai = new OpenAIApi(configuration);

const msg = [
    { "role": "system", "content": "你是一个很厉害的工程师,可以根据表结构和用户的输入,自动生
    { "role": "user", "content": "我现在有一张表,表名hello,有六个字段,分别是id,name,age,
    { "role": "user", "content": "请按照sex分组,按照rank排名" },
    { "role": "user", "content": "请直接输出sql" }
];

const completion = await openai.createChatCompletion({
    model: "gpt-3.5-turbo",
    messages: msg
});

console.log(completion.data.choices[0].message);
```

输出指示



在构建 GPT 产品时，主要的工作都在组织 prompt 上，因此对 prompt 进行设计可以有效达到目的
SQL 是 GPT 非常擅长编写的语言
利用生成知识（Generated Knowledge Prompting） 这一技巧，让 GPT 掌握了他不知道的东西

GPT X 数据源开发的 prompt 技巧
如何实现跟业务系统的集成



在实际产品中，GPT 通常是一个服务

人机交互由产品进行处理，同时通过代理访问 GPT

系统的元数据可以“喂给”GPT当作语料

通过对元数据的有效缩减可以减少Token

提示：

```
"""
数据表 departments，列 = [DepartmentId, DepartmentName]
数据表 students，列 = [DepartmentId, StudentId, StudentName]
为计算机科学系的所有学生创建一个 MySQL 查询
"""
```

ps://github.com/sqlchat/sqlchat

GPT X SQL开发–使用 langchain

```python
from langchain.chains import SQLDatabaseSequentialChain

db = SQLDatabase.from_uri("mysql://user:pass@localhost:3306/student")

llm = OpenAI(temperature=0, verbose=True)

chain = SQLDatabaseSequentialChain.from_llm(llm, db, verbose=True)

chain.run("How many employees are also customers?")
```

构建 chain 需要的 DB

构建 chain 需要的 LLM

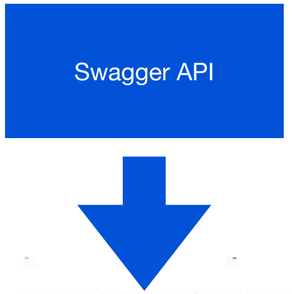
构建 chain

跑起来了！

```
> Entering new SQLDatabaseSequentialChain chain...
Table names to use:
['Employee', 'Customer']

> Entering new SQLDatabaseChain chain...
How many employees are also customers?
SQLQuery:SELECT COUNT(*) FROM Employee e INNER JOIN Customer c ON e.EmployeeId =
c.SupportRepId;
SQLResult: [(59,)]
Answer:59 employees are also customers.
> Finished chain.
```

Langchain的开发过程：
选择一个 or 多个 chain 贴合你的场景
初始化 chain 需要依赖的东西
Run！

GPT X 接口服务–传统接口
通过生成知识技术构建接口查询网关



分解任务

- 通过目前系统的接口如何获得数据

构造 Promopt

- 指令 ß ß 希望告诉我如何组装接口，获取有效的数据
- 上下文 ß ß 当前的接口数据
- 输入数据 ß ß 用户需要的字段
- 输出指示符 ß ß 指示输出的类型或格式

我们仍然利用万能的“生成知识”把 swagger 的接口信息给到 GPT 去学习

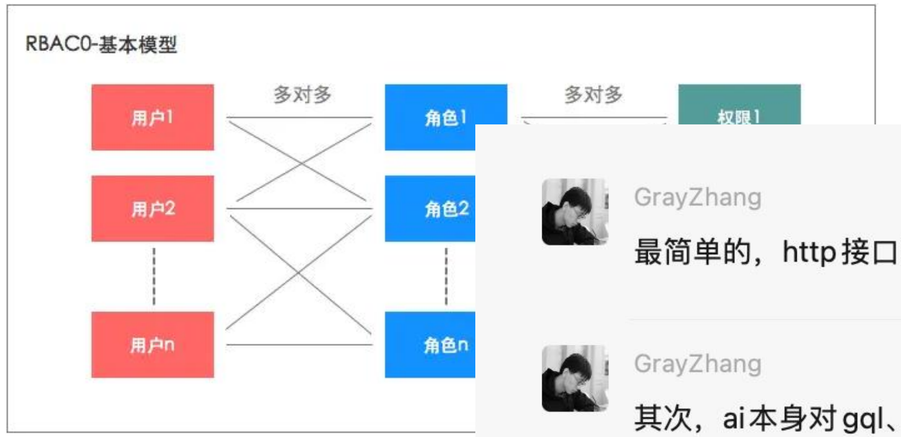
GPT 的学习能力能够理解很多接口操作

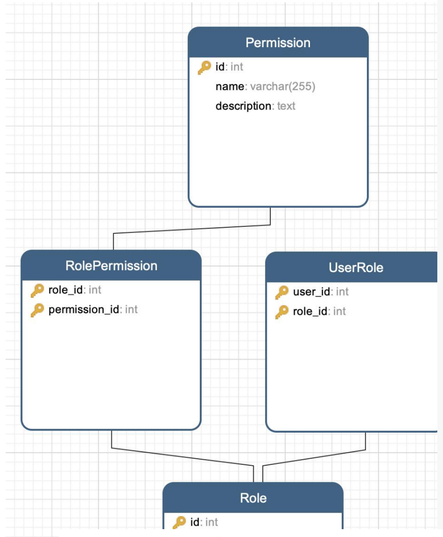
接口数据仍然占据大量的 Token

接口的能力会限制服务能力，用户的需求是枚举不完的

https://github.com/obi1kenobi/trustfall

GPT X 接口服务–gpt X graphql 做 AI 友好的接口设计





这个模型是一个rbac模型，他里面包含哪些restful 查询接口呢

基于这个RBAC模型，可以设计以下RESTful查询接口：

GrayZhang 2023/4/13 20:21

最简单的，http接口需要多轮的事情gql一轮就可以搞定，多轮的关系描述对ai是很麻烦的

GrayZhang 2023/4/13 20:21

其次，ai本身对gql、sql这种学习就比你业务http接口丰富很多

GrayZhang 2023/4/13 20:21

等你们后端降本完了不给你写接口了你继续不需要多轮

GrayZhang 2023/4/13 20:24

你可以这么理解，现在的gpt是个很成熟的sql、gql、es query的开发者，但不是成熟的任何一个http库的使用者，也不是成熟的curl使用者，也不是成熟的swagger阅读者

GrayZhang 2023/4/13 20:24

ai的知识量差距很大

查询用户拥有的所有角色

可以通过以下查询语句获取用户拥有的所有角色：

```
SQL
query {
  1) {
{
```

查询所有的权限列表

语句获取所有的权限列表：

```
{
on
```

s 字段表示要查询所有的权限列表，返回结果中包含了每个权限的 id、ion 三个字段。如果需要查询每个权限所属的角色和用户，可以在中嵌套查询 roles 和 users 字段，例如：

```
{
on
```

这样就可以查询到每个权限所属的角色和用户，返回结果中包含了每个权限的 id、name 和 description 三个字段，以及每个权限所属的角色和用户的 id 和 name 两个字段。

```
  role(id: ID!): Role
  permission(id: ID!): Permission
}
```

AI友好，即在可以使用各种 prompt engineer 的基础上， 你仍然可以选择 AI 最会的技能

GPT X 低代码搭建–基本思路

分解任务

- 通过输入文字，自动生成相应的页面
- 页面是什么东西？页面用什么构建？

构造 Promopt

- 指令 ß ß 希望告诉我如何生成一个页面，使用低代码平台的DSL
- 上下文 ß ß 当前使用的技术栈
- 输入数据 ß ß ？
- 输出指示符 ß ß ？

获取openai的输出并解析

基本思路：
1、利用生成知识技术，设计一个 prompt，使 GPT 返回自己的 DSL
2、利用相关的系统 API，将页面依赖的数据源，系统的组件等信息加载进来

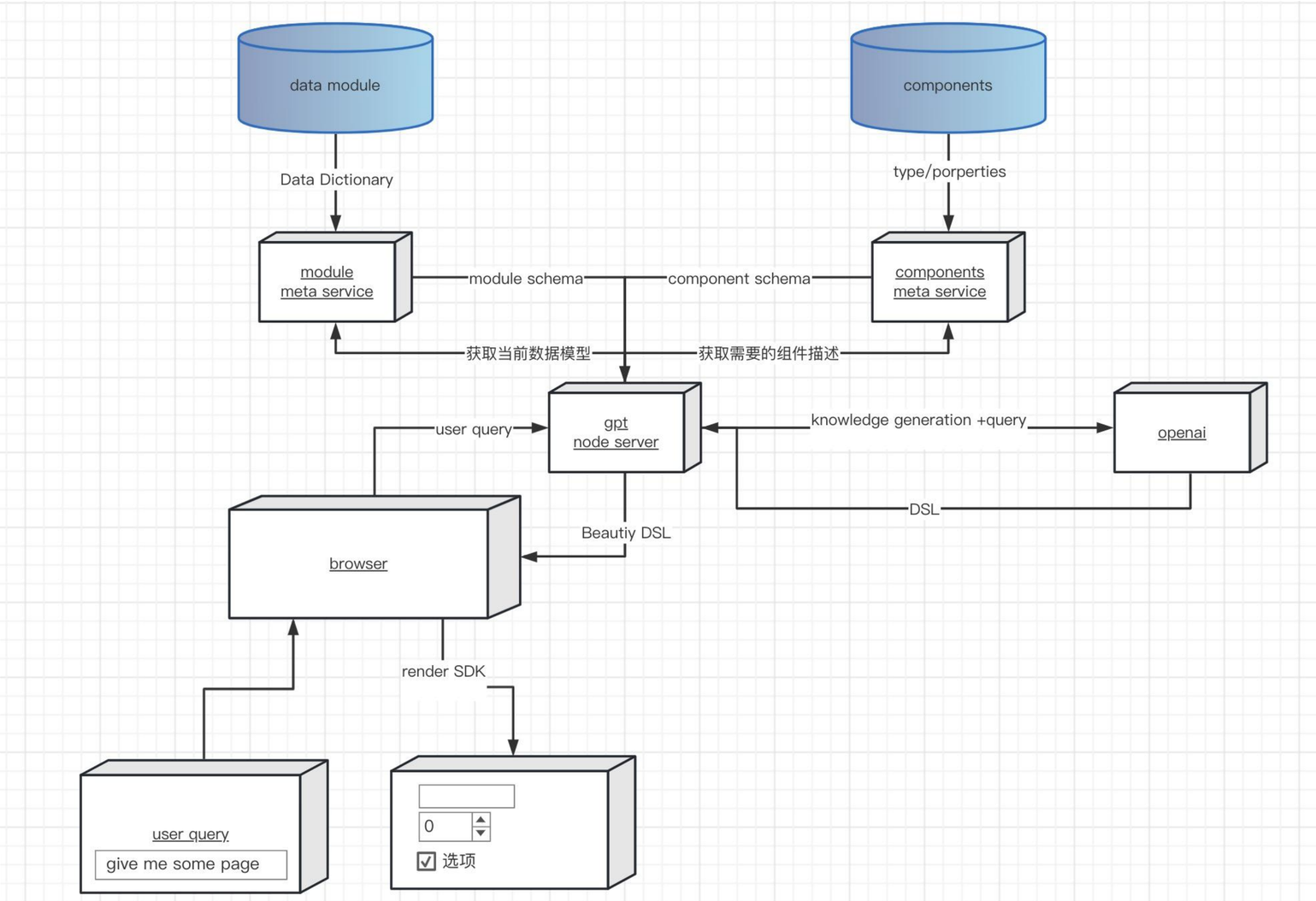
存在的问题：
1、组件层面：组件很多，属性很多，如何全部丢到 prompt 中?
2、大部分的 AIGC 类需求，都以一次生成为主，但低代码这种高频编辑（需求高频变动的）如何解决更新问题

GPT X 低代码搭建–prompt的设计与系统架构

构建一份完整的 Prompt 模板，将相关代码说明传入

```
const msg = [
    {
        "role": "s
    你是一个强大的页
    你首先会在当前的模型中选择数据字段
    {
        type:"root",
        children:[]
    }
    type字段表示组件的类

    你现在拥有以下一系列

    列表组件:type为lis
    columns属性为数组,
    filterColumns表示
    operation属性表示列
    selectable属性表示
    举例来讲,用户要求一
    你会返回如下json:
    {
        "type": "list
        "properties":
          "dataModule
          "columns":
          "filterColu
          "operation"
          "selectable
        }
    }
```

```
{
  role: 'assist
  content: '根据
    '\n' +
    '```json\n'
    '{\n' +
    '  "type":
    '  "propert
    '    "dataM
    '    "colum
    '    "filte
    '    "opera
    '    "selec
    '    "batch
    '  }\n' +
    '}\n' +
    '```\n' +
    '\n' +
    '解释: \n' +
    '\n' +
    '- `type`:
    '\n' +
    '- `propert
    '\n' +
    '  - `dataM
    '\n' +
    '  - `colum
    '\n' +
    '  - `filte
    '\n' +
    '  - `opera
    '\n' +
    '  - `selec
    '\n' +
    '  - `batch
}
```




```
hu",
", "key", "说明"],
["name"],
it", "delete"],
```

GPT X 低代码搭建–如何做update

JSON patch的使用

JSON Path 可以描述 JSON 文档变化. 使用它可以避免在只需要修改某一部分的时候发送整个文档内容. 补丁(Patch)内容的格式也是JSON.JSON Patch由IETF在RFC 6902中规范.

原始文档

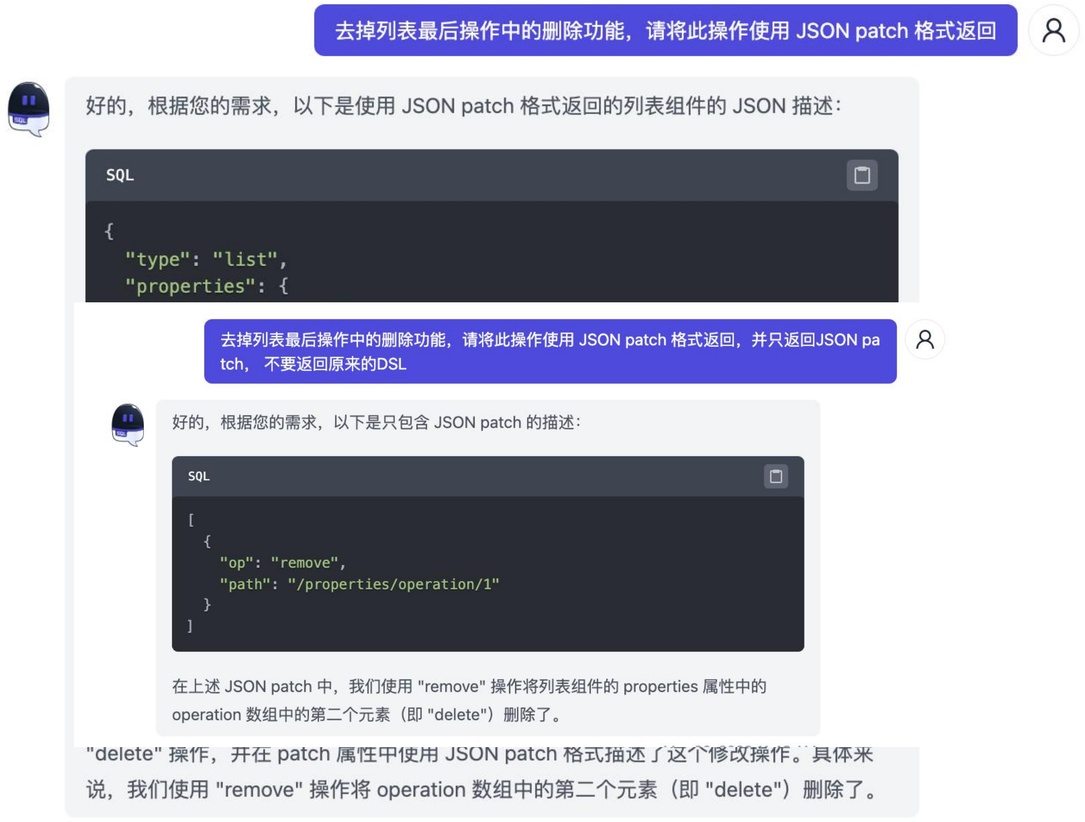
```
{
  "baz": "qux",
  "foo": "bar"
}
```

补丁

```
[
  { "op": "replace", "path": "/baz", "value": "boo" },
  { "op": "add", "path": "/hello", "value": ["world"] },
  { "op": "remove", "path": "/foo"}
]
```

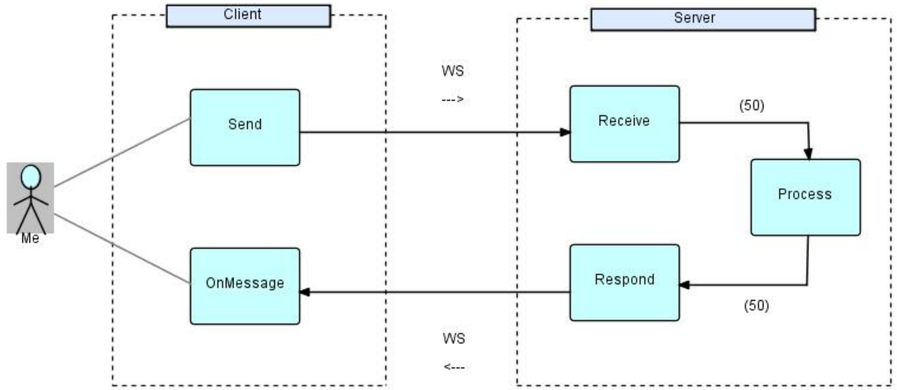
结果

```
{
  "baz": "boo",
  "hello": ["world"]
}
```



GPT X 低代码搭建–逻辑可视化低代码

Flow Based Programing (https://github.com/flowbased)是 J. Paul Rodker Morrison很早以前提出的一种编程范式。FBP 把应用看作一个黑盒子，它能够看到的是一张进程（processes）组成的网，而进程间通过连接（connection）进行通信，进程通过端口（port）来访问连接（这种抽象类似网络编程）。

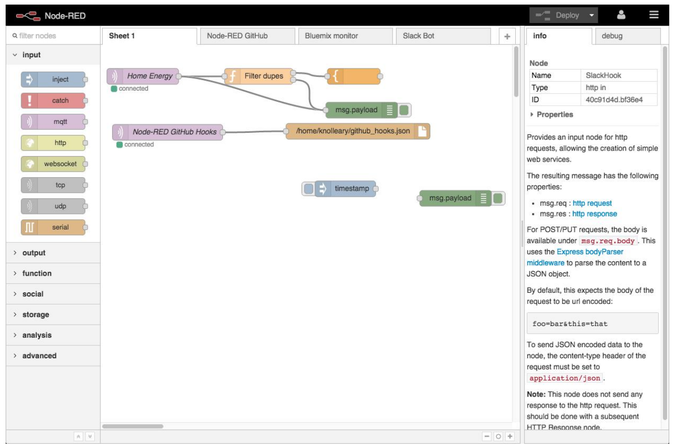


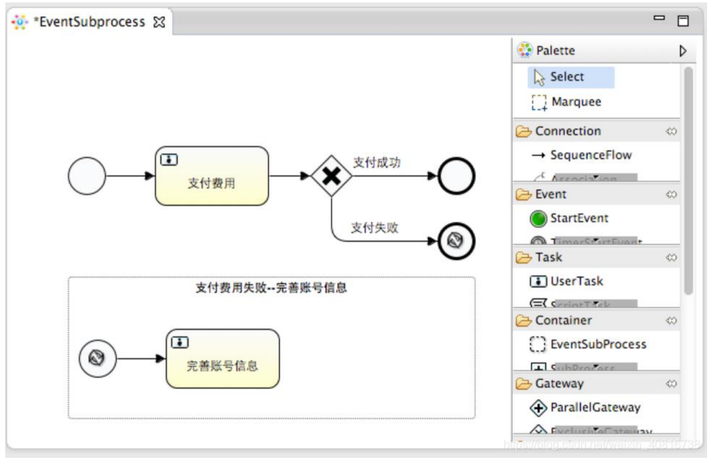
FBP 的三大概念：

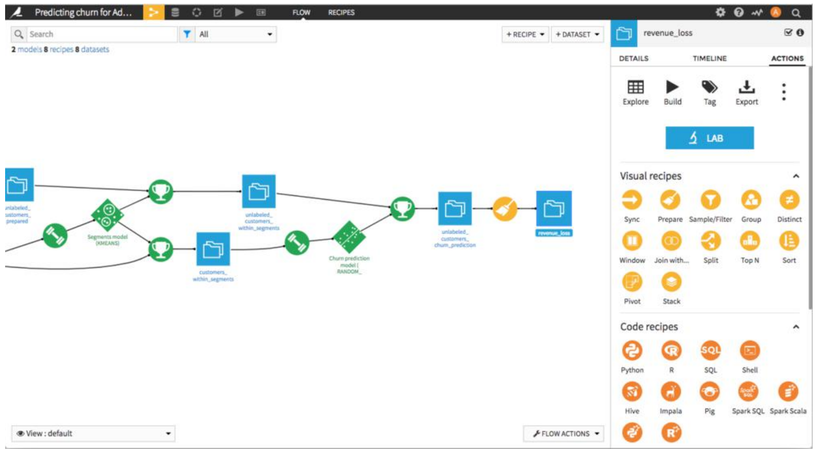
1、进程：组件（component）的一个实例，可以跟其他进程并行运行。其他进程可以是同个组件的其他实例。

2、网络：表示为一个有向图，其中进程作为节点，而连接作为边

3、组件：对于应用开发者，通常可以看作黑盒；当要使用传统高级语言来创建组件或者组件本身是个子图时，它就是白盒。





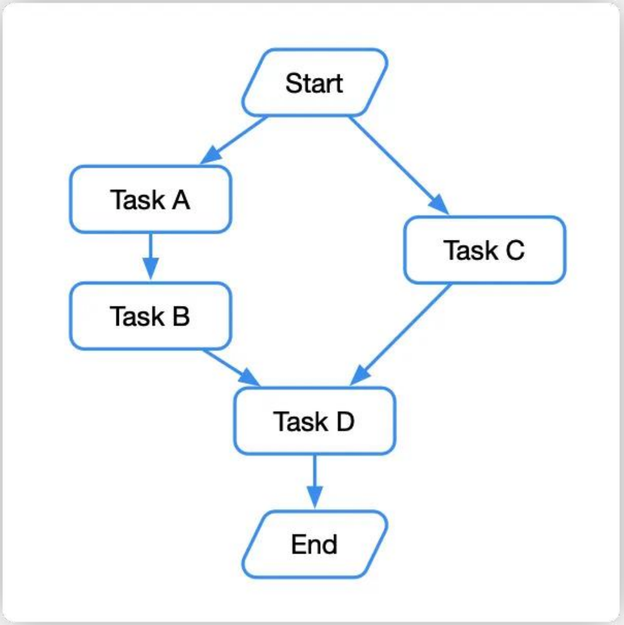


GPT X 低代码搭建–逻辑 DSL设计

设计一个 schema 来存储“图”，兼顾绘图和逻辑。

Logic schema 由三个主要的实体结构构成：

Flow 流程主体, Node 节点描述，Link 节点关系描述




```
export interface Flow = {
    //  逻辑唯一标识，保证 flow 之间唯一
    id: number,
    //  逻辑名称
    name: string,
    // 逻辑描述
    desc: string,
    // 实体,所有的逻辑节点描述集合
    nodes: Node[],
    // 关系，所有的节点间关系描述集合
    links: Link[],
    // 表示在节点之间共享的上下文信息，包括
    context?: Content//Context另行定义
}
```

```
export interface Node = {
    //节点唯一标识，保证单 flow 内唯一即可。
    id: number;
    //节点的标识符，每种节点唯一
    key: string;
    //  节点名称
    name: string;
    // 节点描述
    desc: string;
    // 节点类型；//NodeType可枚举，另行定义
    type: NodeType;
    //节点在前端二维绘制画布中的表现形式(位置，尺寸，填充内容等);NodeDrawAttrs另行定义
    drawAttrs:NodeDrawAttrs;
    //节点的逻辑相关属性(包括表单入参、输出参数解构、节点执行类型、其它扩展属性等);NodeLogicAttrs另行定义
    logicAttrs:NodeLogicAttrs;
}
```

```
export interface Link = {
    //关系标识，此id非必须，由 source target 保证唯一即可
    id: number;
    // 关系名称
    name: string;
    // 关系描述
    desc: string;
    // 关系类型，可枚举，LinkType另行定义
    type: LinkType;
    // 关系起始节点Node的id
    source : number;
    // 关系目标节点Node的id
    target: number;
    // 关系在前端二位绘制画布中的表现形式数据(包括粗细，颜色，内容等);LinkDrawAttrs另行定义
    drawAttrs: LinkDrawAttrs;
    // 关系上的附加属性，可以添加任何属性;LinkLogicAttrs另行定义
    logicAttrs: LinkLogicAttrs;
}
```

在数据库中可以通过 JSON 方式存储完整的 Flow
同样可以存储三张表：Node，Link 表与 Flow 表来将三个实体结构分开存储。

GPT X 低代码搭建–逻辑 DSL prompt

我现在使用一段json数据表示数据处理逻辑，其中node属性表示所有的数据处理节点，link属性表示节点之间的关系, link属性中source表示起点，target表示终点。 举例来讲，我从一个数据源类型为hive，表名为data1的表中抽取数据，输入到类型为mysql, 表名为data2的数据库，可以表示为 {nodes:[{type:'hive', name:'data1'},{type:'mysql', name:'data2'}], links: [{source:'data1', target: 'data2'}]}

这段JSON数据表示了一个数据处理流程，其中包含两个节点，一个是数据源节点，类型为hive，表名为data1，另一个是数据目标节点，类型为mysql，表名为data2。这两个节点之间有一条连接，表示从数据源节点抽取数据后输入到数据目标节点中。这种数据处理流程可以用于数据抽取、转换、加载等场景。

Apr 26, 2023 10:38 AM

请返回一段json数据，表示我从一个tdsql数据源中的table表取数据，之后将数据放入mysql数据源的table2和table3表中

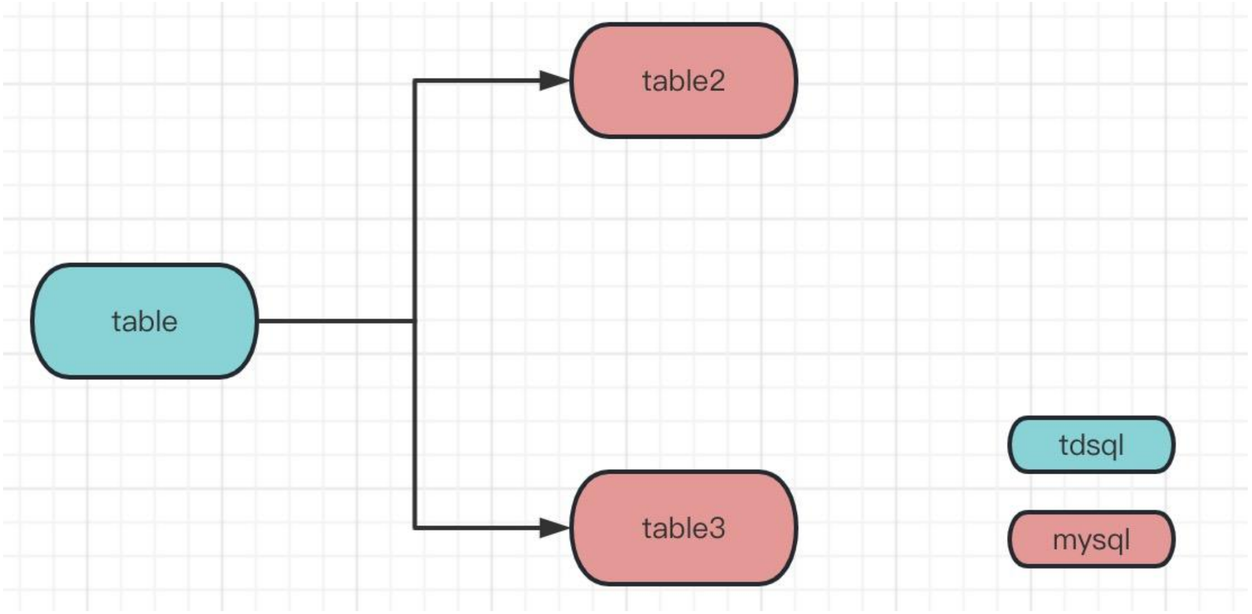
以下是一段JSON数据，表示从一个tdsql数据源中的table表取数据，之后将数据放入mysql数据源的table2和table3表中：

SQL

```
{
  "nodes": [
    { "type": "tdsql", "name": "table" },
    { "type": "mysql", "name": "table2" },
    { "type": "mysql", "name": "table3" }
  ],
  "links": [
    { "source": "table", "target": "table2" },
    { "source": "table", "target": "table3" }
  ]
}
```

这段JSON数据中，nodes属性表示所有的数据处理节点，包括tdsql数据源中的table表、mysql数据源中的table2和table3表。links属性表示节点之间的关系，其中source表示起点，target表示终点。这里有两条连接，分别表示从tdsql数据源中的table表抽取数据后，分别输入到mysql数据源中的table2和table3表中。

Apr 26, 2023 10:39 AM



GPT X 数据可视化–基础用法

分解任务

- 生成一个柱状图，横轴是year，纵轴是count，数据是 XXX

我们使用一个json来描述图表结构，type表示图表类型，data表示图表数据，dimension 表示图表维度，measure表示图表指标。 例子：使用柱状图展示每个year的stock: { type: 'bar', data: [{year:2001, stock: 100}, {year:2002, stock: 200}, {year:2003, stock: 300}], dimension: ['year'], measure: 'stock'}

这个JSON描述了一个柱状图，展示了每个年份的股票数量。具体解释如下：

- type: 'bar' 表示这是一个柱状图。
- data: [...] 表示图表的数据，这里是一个数组，每个元素都是一个对象，表示一个数据点。每个数据点都有一个year和一个stock属性，分别表示年份和股票数量。
- dimension: ['year'] 表示图表的维度是年份，也就是说，x轴上会显示年份。
- measure: 'stock' 表示图表的指标是股票数量，也就是说，y轴上会显示股票数量。

May 4, 2023 3:28 PM

现在返回一个json，利用柱状图显示每月的工资， 其中月的字段为month, 工资为salary

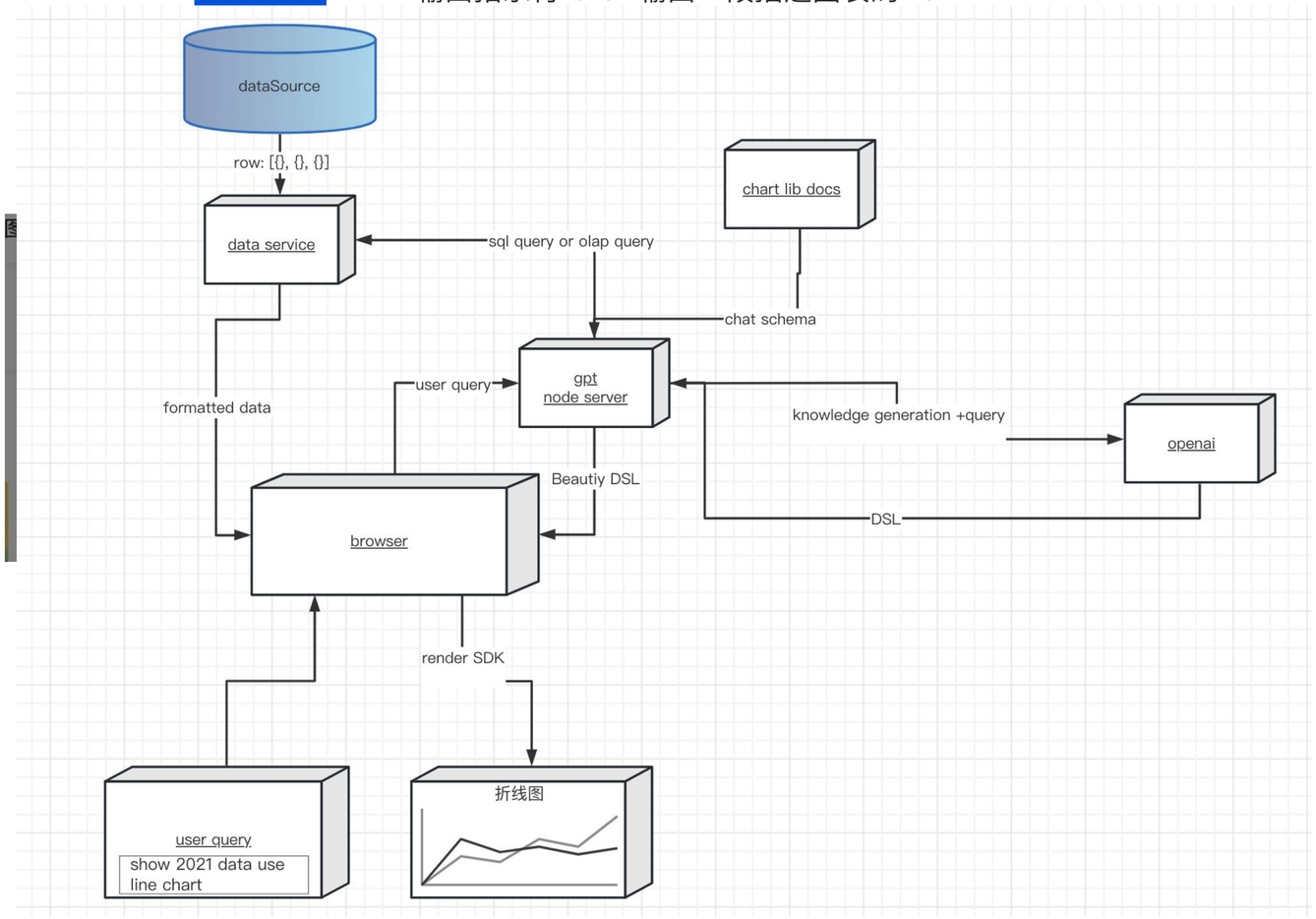
好的，以下是一个示例JSON，用于展示每月的工资：

SQL

```
{
  "type": "bar",
  "data": [
    {"month": "Jan", "salary": 3000},
    {"month": "Feb", "salary": 3500},
    {"month": "Mar", "salary": 4000},
    {"month": "Apr", "salary": 4500},
    {"month": "May", "salary": 5000},
    {"month": "Jun", "salary": 5500},
    {"month": "Jul", "salary": 6000},
    {"month": "Aug", "salary": 6500},
    {"month": "Sep", "salary": 7000},
    {"month": "Oct", "salary": 7500},
    {"month": "Nov", "salary": 8000},
    {"month": "Dec", "salary": 8500}
  ],
  "dimension": ["month"],
  "measure": "salary"
}
```

构造 Promopt

- 指令 ß ß 生成一段图表的描述
- 上下文 ß ß 图表DSL的规范
- 输入数据 ß ß 当前的数据
- 输出指示符 ß ß 输出一段描述图表的DSL



https://www.graphmaker.ai/

GPT X 数据可视化–进阶–异常识别

你的可视化真的有用吗?

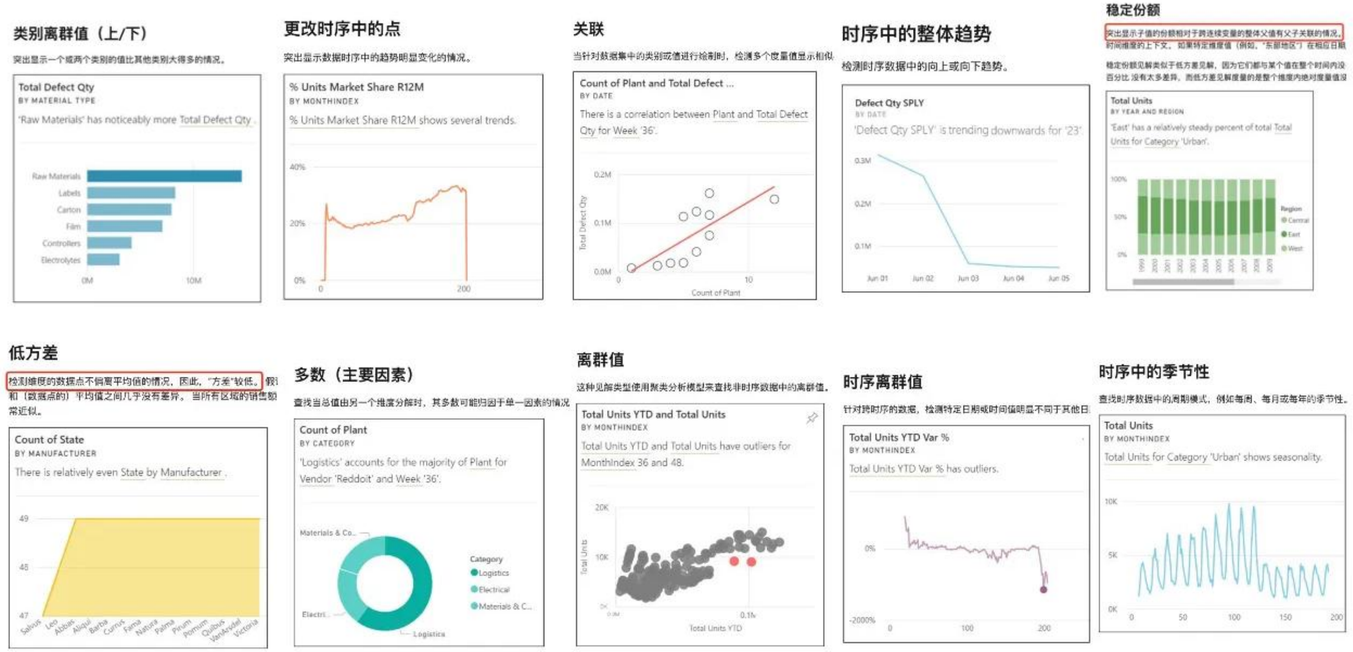
分解任务

- 展示袜子每年的销量数据趋势，并分析其中的异常，标注在图表上

构造 Promopt

- 指令 ß ß 生成图表描述
- 上下文 ß ß 当前的库表字段，趋势要用什么图表，如何分析异常
- 输入数据 ß ß
- 输出指示符 ß ß 输出一段描述图表的DSL

能够构成自动化分析,解读的一些特征:



我现在有一份数据现有数据[{x:1, y:1}, {x:2, y:2}, {x:3, y:4}, {x:4, y:4}] 请基于线性回归的思路，找出这组数据中的异常数据，请直接返回异常数据的值， 不要写出推理过程

根据线性回归的思路，可以通过计算每个数据点与回归直线的残差来判断是否为异常值。在这组数据中，回归直线的斜率为1，截距为0，因此可以计算每个数据点的残差：

- 对于{x:1, y:1}，残差为0
- 对于{x:2, y:2}，残差为0
- 对于{x:3, y:4}，残差为-1
- 对于{x:4, y:4}，残差为0

因此，可
线预测的

我有数据 [2, 4, 6, 8, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 50, 52, 54, 56, 58, 60, 62, 64, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 90, 92, 94, 96, 98, 100] 请基于Z-score算法检测其中的异常

根据Z-score算法，我们需要计算数据集中每个数据点的Z-score值，然后找到Z-score值大于3或小于-3的数据点，这些数据点可以被视为异常值。

以下是基于Python的Z-score算法实现：

```python
import numpy as np

def z_score(data):
    threshold = 3
    mean = np.mean(data)
    std = np.std(data)
    z_scores = [(x - mean) / std for x in data]
    return np.where(np.abs(z_scores) > threshold)

data = [2, 4, 6, 8, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36

outliers = z_score(data)
print("Outliers:", outliers)
```

输出结果：Outliers: (array([4]),)

解释：根据Z-score算法，数据点11被视为异常值，因为它的Z-score值为3.06，超过了阈值3。

GPT X 数据可视化–进阶–自动化可视化

你的可视化真的有用吗?

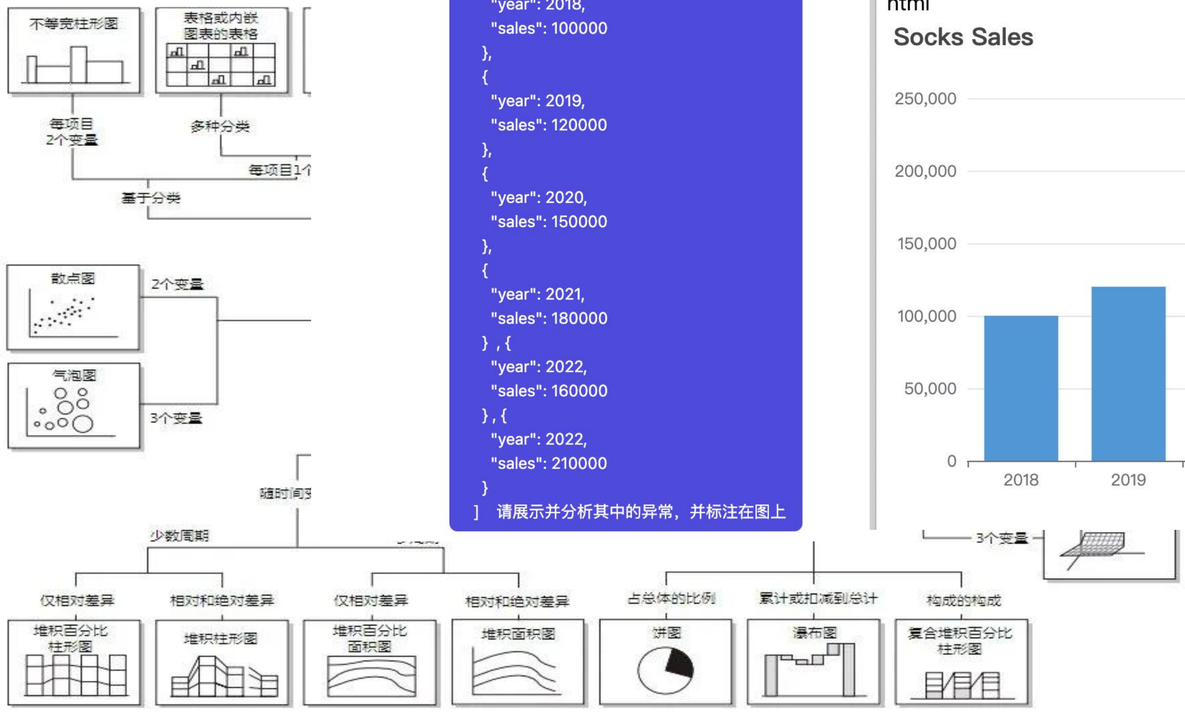
分解任务

- 展示袜子每年的销量数据趋势，并分析其中的异常，标注在图表上

构造 Promopt

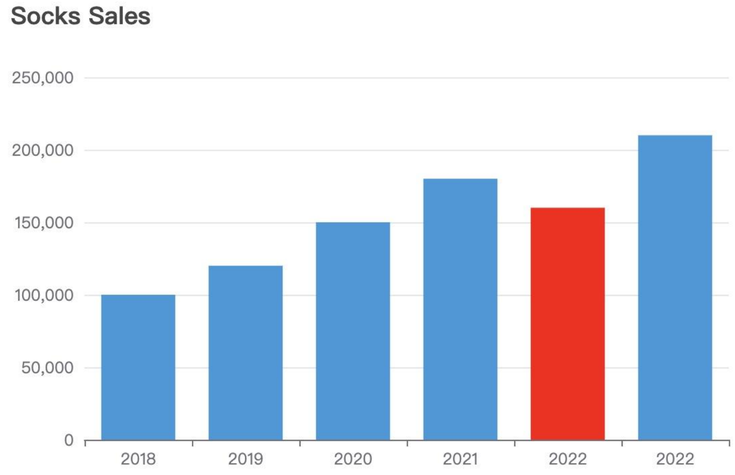
- 指令 ß ß 生成图表描述
- 上下文 ß ß 当前的库表字段，趋势要用什么图表，如何分析异常
- 输入数据 ß ß
- 输出指示符 ß ß 输出一段描述图表的DSL

寻找合适的图表形态展现数据



我有一份袜子每年的销售数量的数据 [
{
"year": 2018,
"sales": 100000
},
{
"year": 2019,
"sales": 120000
},
{
"year": 2020,
"sales": 150000
},
{
"year": 2021,
"sales": 180000
} ,{
"year": 2022,
"sales": 160000
} ,{
"year": 2022,
"sales": 210000
}
] 请展示并分析其中的异常，并标注在图上

html



当我想查看分布数据时，可以使用直方图、散点图、正态分布图等，当我想看趋势数据时，我可以使用曲线图，柱状图，当我想看数据的构成时，我可以使用饼图。现在我有一份数据 [
{
"year": 2018,

我有另一份数据 [
{
"label": "产品A",
"value": "15000"
},
{
"label": "产品B",
"value": "10000"
},
{
"label": "产品C",
"value": "7500"
},
{
"label": "产品D",
"value": "5000"
},
{
"label": "其他",
"value": "2500"
}
] 我想看我的产品构成，请选择一个合适的图表

对于这份数据，可以使用饼图来展示产品构成。饼图可以直观地展示各个产品在总体中所占的比例，帮助我们更好地了解产品的构成情况。

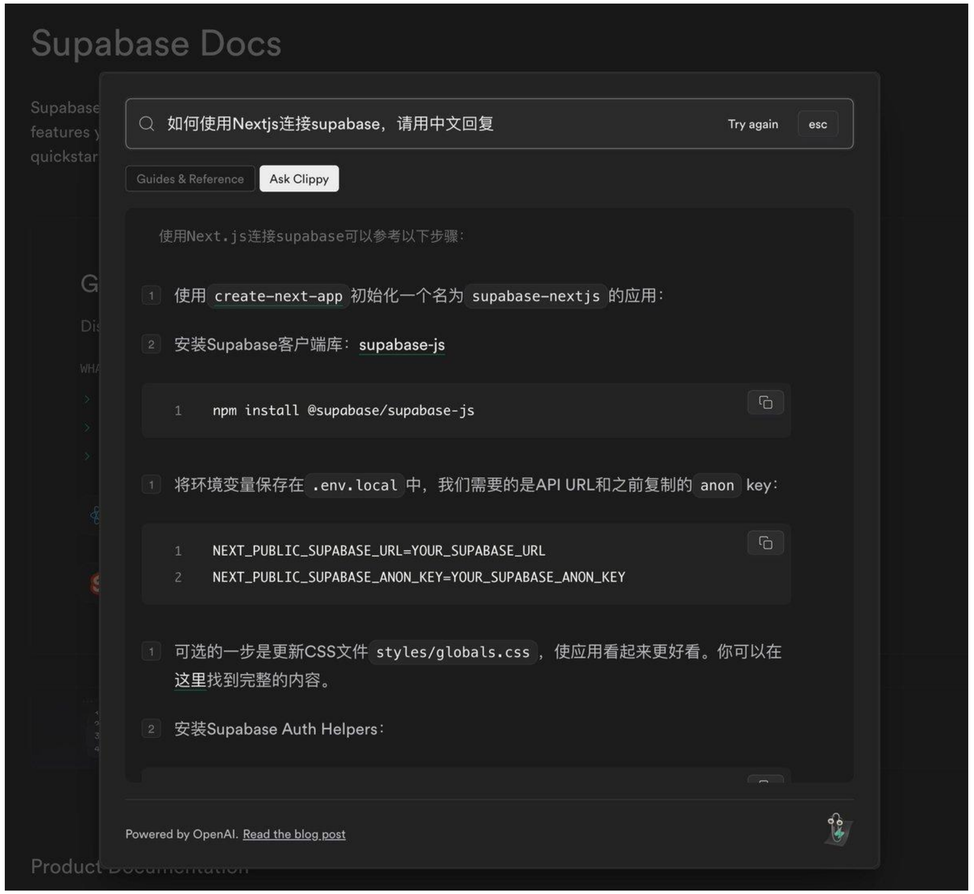
GPT X 文档-让 AI 更智能
问答系统构建

分解任务

- 提问：平台如何支持在线打包发布

构造 Promopt

- 指令 ß ß 返回打包发布的文档
- 上下文 ß ß 全部文档
- 输入数据 ß ß 我需要哪方面的文档
- 输出指示符 ß ß 输出一段文档内容

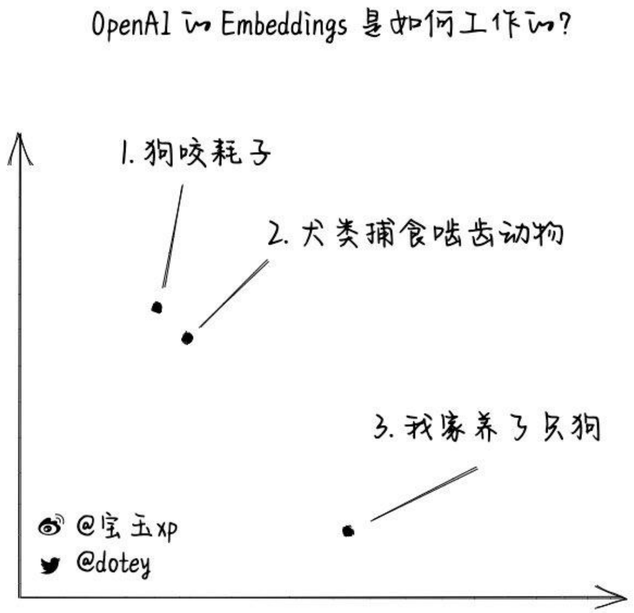


- token不够，把全部文档喂给 GPT 需要大量 Token
- 如何判断用户的提问跟你的文档内容吻合?
- 我输入的是中文，这文档是英文啊，如何搜索?
- 如何去管理和存储你的文档?

GPT X 文档–让 AI 更智能
如何判断两个字符串“类似”–万物皆可 embedding

openai官网 embedding 使用范例 https://platform.openai.com/docs/guides/embeddings/use-cases

1. "狗咬耗子"
2. "犬类捕食啮齿动物"
3. "我家养了只狗"



搜索（结果按查询字符串的相关性进行排序）
聚类（将文本字符串按相似性分组）
推荐（推荐具有相关文本字符串的项目）
异常检测（识别相关性较小的异常值）
多样性测量（分析相似度分布）
分类（文本字符串按其最相似的标签进行分类）

在自然语言处理和机器学习领域，"embeddings" 是指将单词、短语或文本转换成连续向量空间的过程。这个向量空间通常被称为嵌入空间（embedding space），而生成的向量则称为嵌入向量（embedding vector）或向量嵌入（vector embedding）。
嵌入向量可以捕获单词、短语或文本的语义信息，使得它们可以在数学上进行比较和计算。这种比较和计算在自然语言处理和机器学习中经常被用于各种任务，例如文本分类、语义搜索、词语相似性计算等。
在中文语境下，"embeddings" 通常被翻译为 "词向量" 或者 "向量表示"。这些翻译强调了嵌入向量的特点，即将词汇转换成向量，并表示为嵌入空间中的点。

GPT X 文档–让 AI 更智能
如何实现和存储 embedding–向量数据库技术介绍

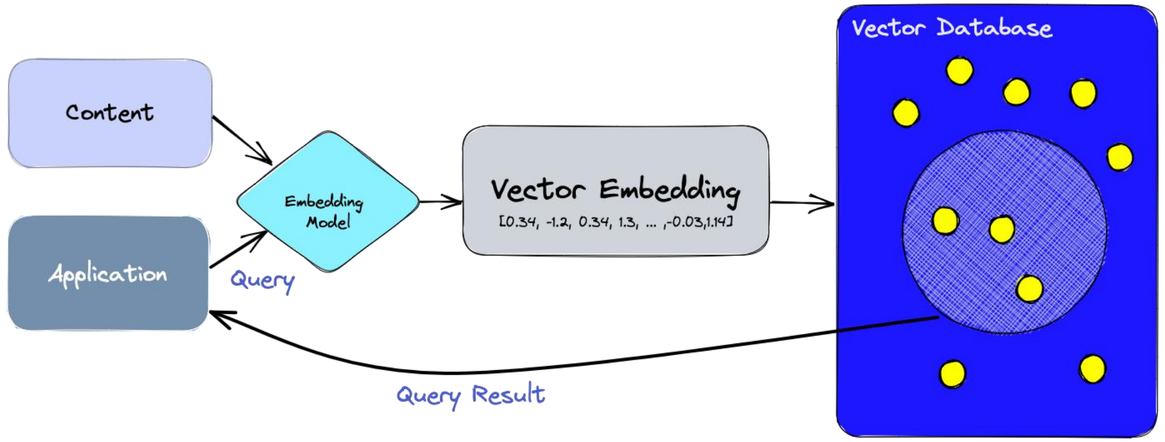
Example: Getting embeddings

```
curl https://api.openai.com/v1/embeddings \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $OPENAI_API_KEY" \
  -d '{
    "input": "Your text string goes here",
    "model": "text-embedding-ada-002"
  }'
```

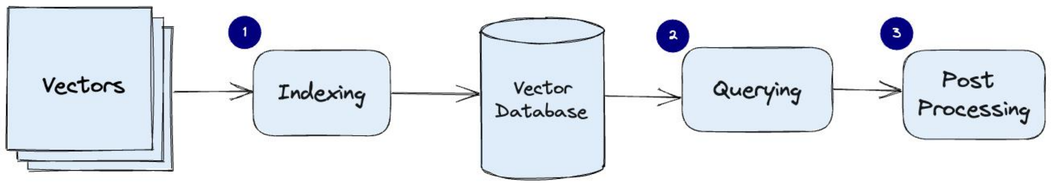
Example response:

```
{
  "data": [
    {
      "embedding": [
        -0.006929283495992422,
        -0.005336422007530928,
        ...
        -4.547132266452536e-05,
        -0.024047505110502243
      ],
      "index": 0,
      "object": "embedding"
    }
  ],
  "model": "text-embedding-ada-002",
  "object": "list",
  "usage": {
    "prompt_tokens": 5,
    "total_tokens": 5
  }
}
```

官方推荐使用ada–002，便宜又好用。



1. **索引**：向量数据库使用 PQ、LSH 或 HNSW 等算法(https://www.6aiq.com/article/1587522027341)对矢量进行索引
2. **查询**：向量数据库将索引查询向量与数据集中的索引向量进行比较，找到最相近的结果
3. **后处理**：某些场景下向量数据库从数据集中检索最相近的结果后，对其进行后处理以返回最终结果。比如通过使用不同的相似性算法重新排列所有结果。

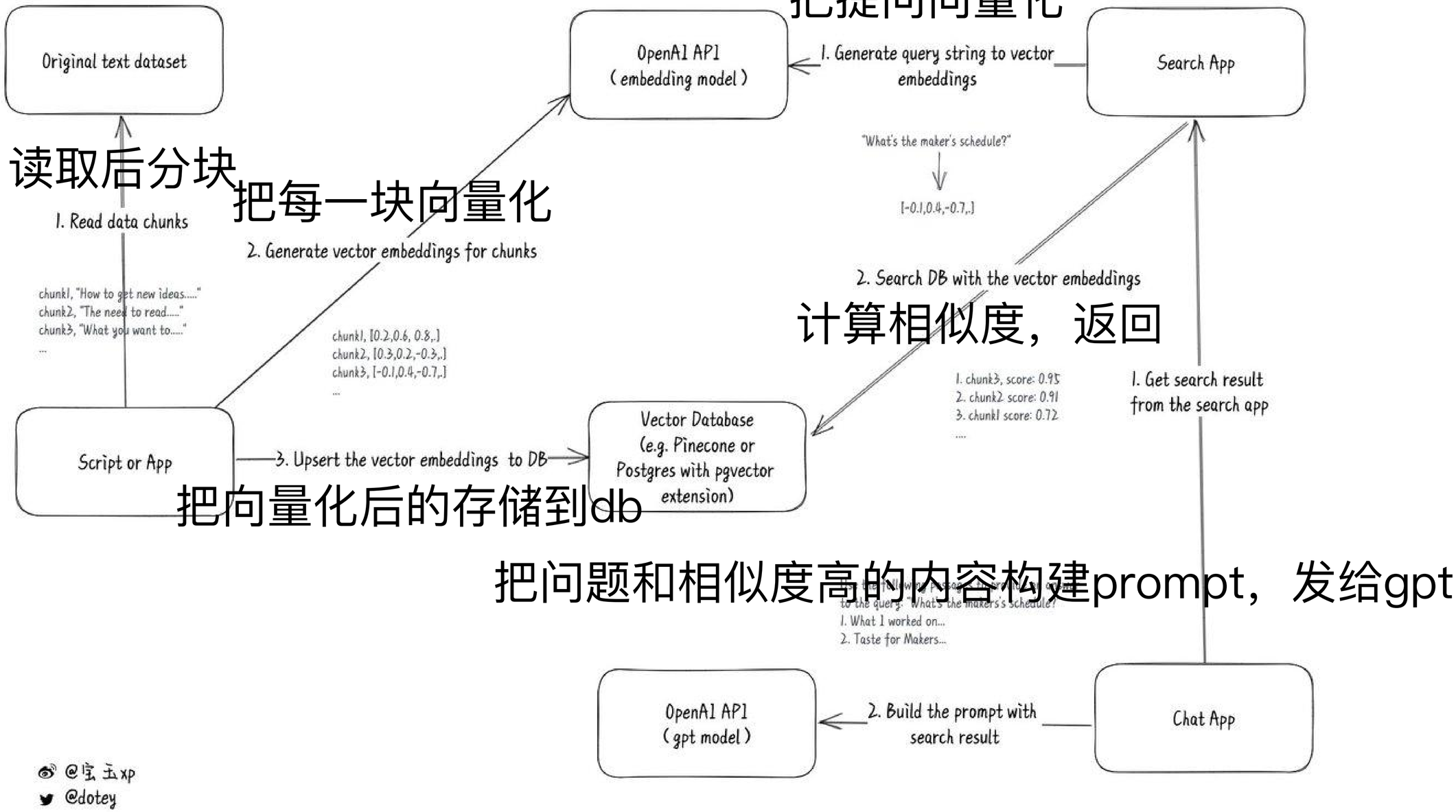



```
SELECT * FROM items WHERE id != 1 ORDER BY embedding
<-> (SELECT embedding FROM items WHERE id = 1) LIMIT 5;
```

https://supabase.com/docs/guides/database/extensions/pgvector

普通搜索和向量搜索的异同 https://cloud.tencent.com/developer/beta/article/1941250?areaSource=106005.13

GPT X 文档–让 AI 更智能
核心流程



GPT X 文档–让 AI 更智能
使用 langchain

src > vector > bee.txt

```
1  熊蜂（蜜蜂科熊蜂属，约 260 种）、木蜂（蜜蜂科木蜂亚科，约 500 种）的蛰针是光滑的，蛰刺攻击后可以从人和非人
2
3  无刺蜂（约 550 种）没有蛰针，用口器自卫。这之外还有一些蜜蜂物种所有个体都没有蛰针。蜂类的蛰针从产卵器特化而
4
5  真社会性的群居蜜蜂只占所有蜜蜂物种的约十分之一，而且它们并不都会建造蜂巢。黄蜂之类经常在毒液里配备杀伤力更大
```

```
// https://js.langchain.com/docs/modules/models/embeddings/integrations
const embeddings = new OpenAIEmbeddings();
const model = new OpenAI({ temperature: 0 });
const chain = loadQARefineChain(model);

// Load the documents and create the vector store
const loader = new TextLoader(path.join(__dirname, "./bee.txt"));
const docs = await loader.loadAndSplit();
// https://js.langchain.com/docs/modules/indexes/vector_stores/integrations/memory
const store = await MemoryVectorStore.fromDocuments(docs, embeddings);

// Select the relevant documents
const question = "什么蜜蜂没有蛰针，请用中文回答。";
const relevantDocs = await store.similaritySearch(question);

// Call the chain
const res = await chain.call({
  input_documents: relevantDocs,
```

embedding，转换器，转换为向量

初始化一个 Chain

将文本转化为向量后存储到内存

搜索，默认是余弦相似度

组成好 prompt 调用 openai

```
> langchain-ts-starter@1.0.0 start
> tsx -r dotenv/config src/index.ts

{ output_text: '\n\n无刺蜂和一些蜜蜂物种的所有个体都没有蛰针。' }
```

加载文本

↓

切片、分块

↓

存储到向量数据库

↓

查询相关的文本

↓

文本+query调用openai

https://towardsdatascience.com/4-ways-of-question-answering-in-langchain-188c6707cc5a

GPT X 低代码的总结

AIGC技术，一定是建立在一个好的 DSL 上才会实现开发者层面与机器交互。
从 sql 到 gql/swagger 到 ui schema 到 logic schema 一直到 chart，DSL是人机交互的桥梁。
DSL 的编写才是提效的关键技术。

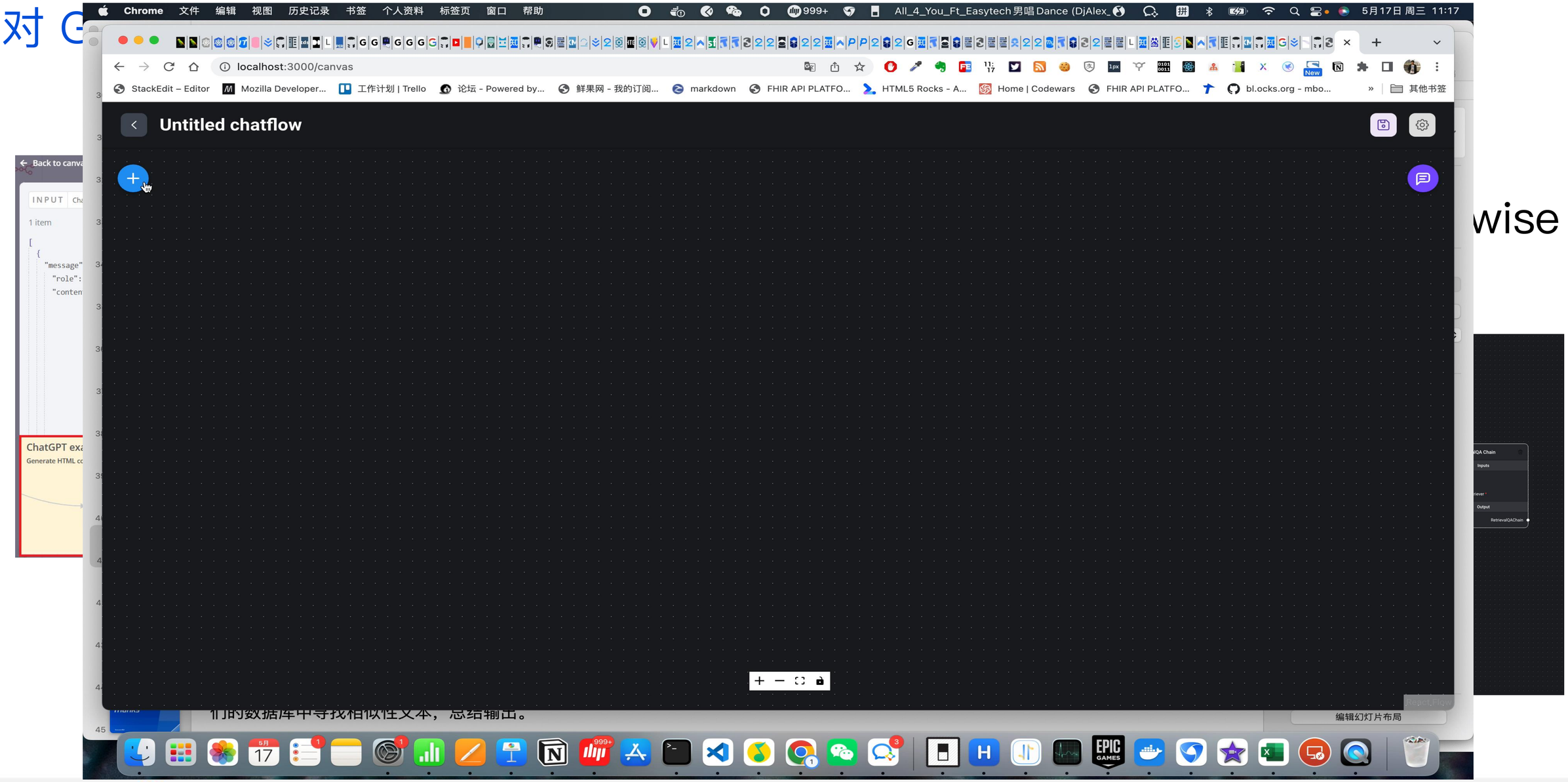
- 在使用 GPT 实现低代码能力时，仍要遵循基本 prompt 原则：指令、上下文、输入输出，其中输出的格式记得明确。
- GPT 具有强大的学习能力，可以将低代码方方面面的知识都灌输给他，给他提供完善的 knowledge，甚至可以“教”他一些他不懂的专业知识
- 对于搭建场景，设计一个完善的 DSL 是重中之重，prompt 技巧只是补充，DSL 设计的好，使用简单的 few shot 就可以实现大多数场景
- 可以通过向量数据库+ embedding 的方式，突破 token 的限制，从而实现大规模文本搜索的功能
- langchain 是实战利器，推荐所有想跟自己系统集成 gpt 能力的开发者使用

关于提示工程更多的介绍可以学习 https://learnprompting.org/zh-Hans/

https://promptingguide.azurewebsites.net/techniques/knowledge

GPT高级使用技巧及思考

Gpt workflow
对 G

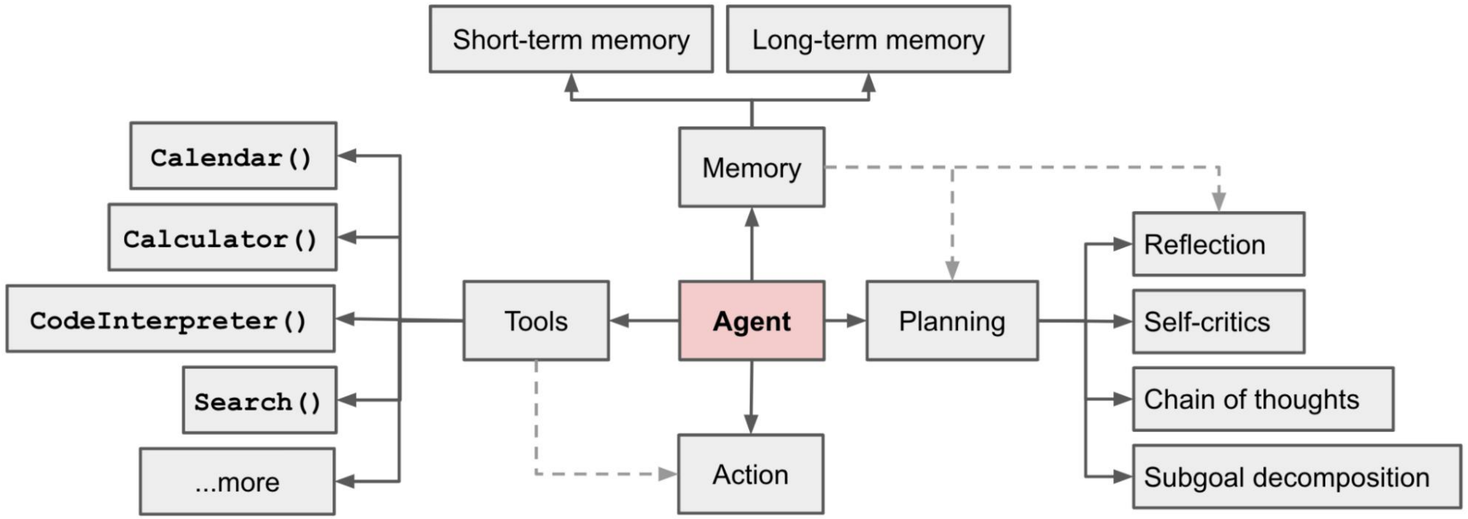


wise

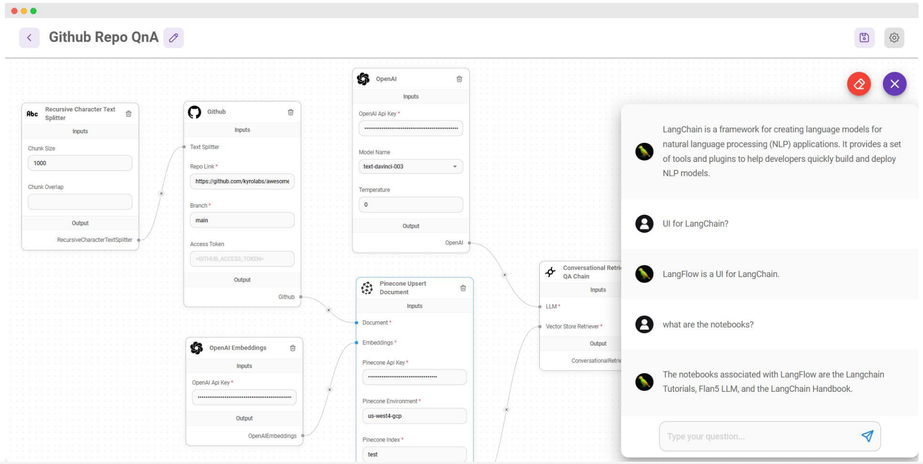
ai based agent 应用架构及其低代码编排能力

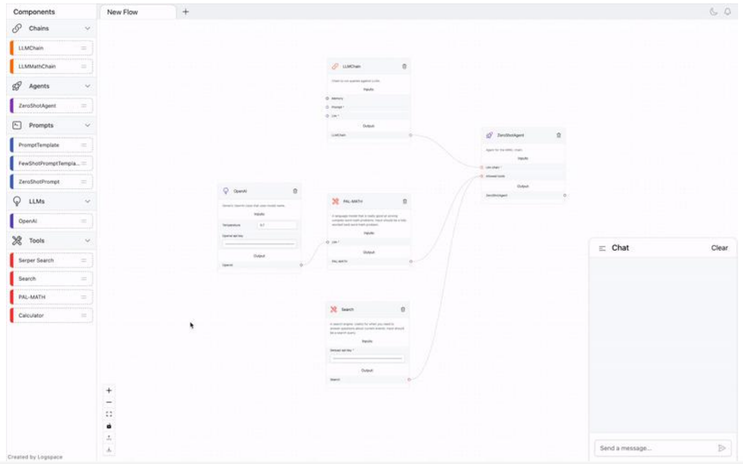
随着 ChatGPT 的发展，不仅仅传统的人机界面交互会被颠覆，维持了很多年的存储–领域模型–服务端–客户端的应用架构也极有可能会被颠覆。

6月23日，openai专家提出了 《LLM Powered Autonomous Agents》概念（https://lilianweng.github.io/posts/2023-06-23-agent/），其正是市面上曾经风靡的langchain agent/reAct/autogpt等方案的思想，其架构设计如右图：



值得注意的是，市面上已经基于这样的架构出现了多种架构解决方案以及低代码编排工具，如：





ai–based system 可能会颠覆传统的系统设计

https://flowiseai.com/

https://github.com/logspace-ai/langflow

GPT 的优势

GPT 绝不是无所不能，通过DSL和prompt的构建才能发挥价值

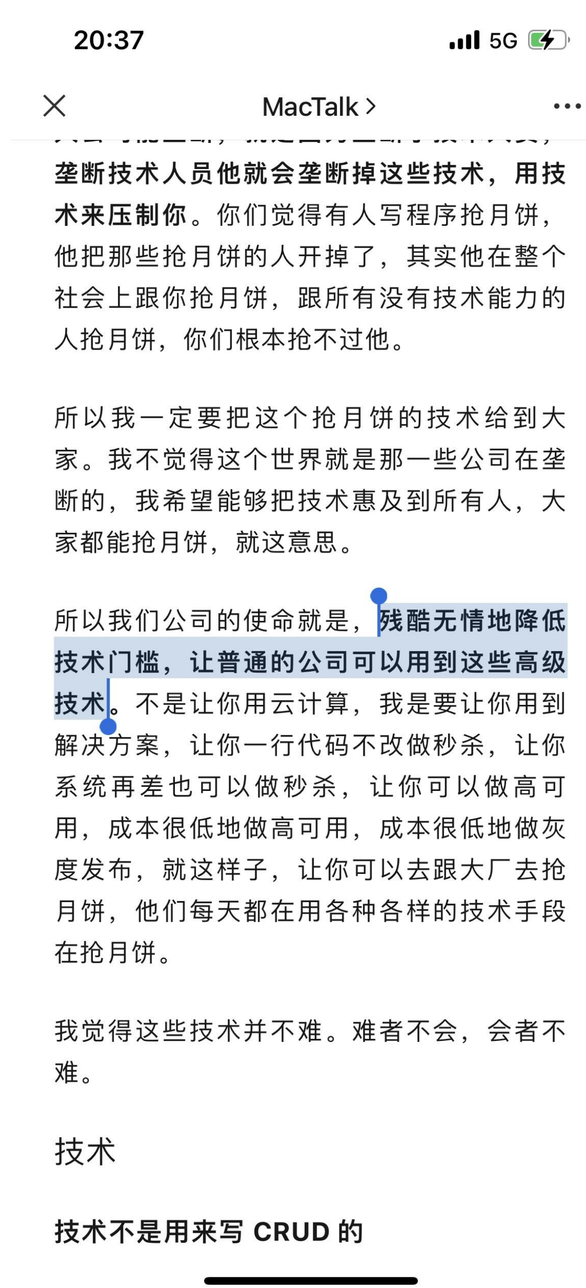
通过工程能力可以大幅度提升GPT的效果，因此绝对不是“无事可做”

强大的学习理解能力

ai一定是建立在一个好的DSL上才会更加方便大家与机器交互，从sql到gql/swagger到form schema到 logic schema一直到chat，我们发现 DSL 是我们跟人机交互的桥梁，DSL 的编写才是提效的关键技术。

工欲善其事 善假于物也

openai/大模型的核心价值：低门槛的ai能力调用



想一想，我该如何把这些技术应用在工作实践中？

THANKS