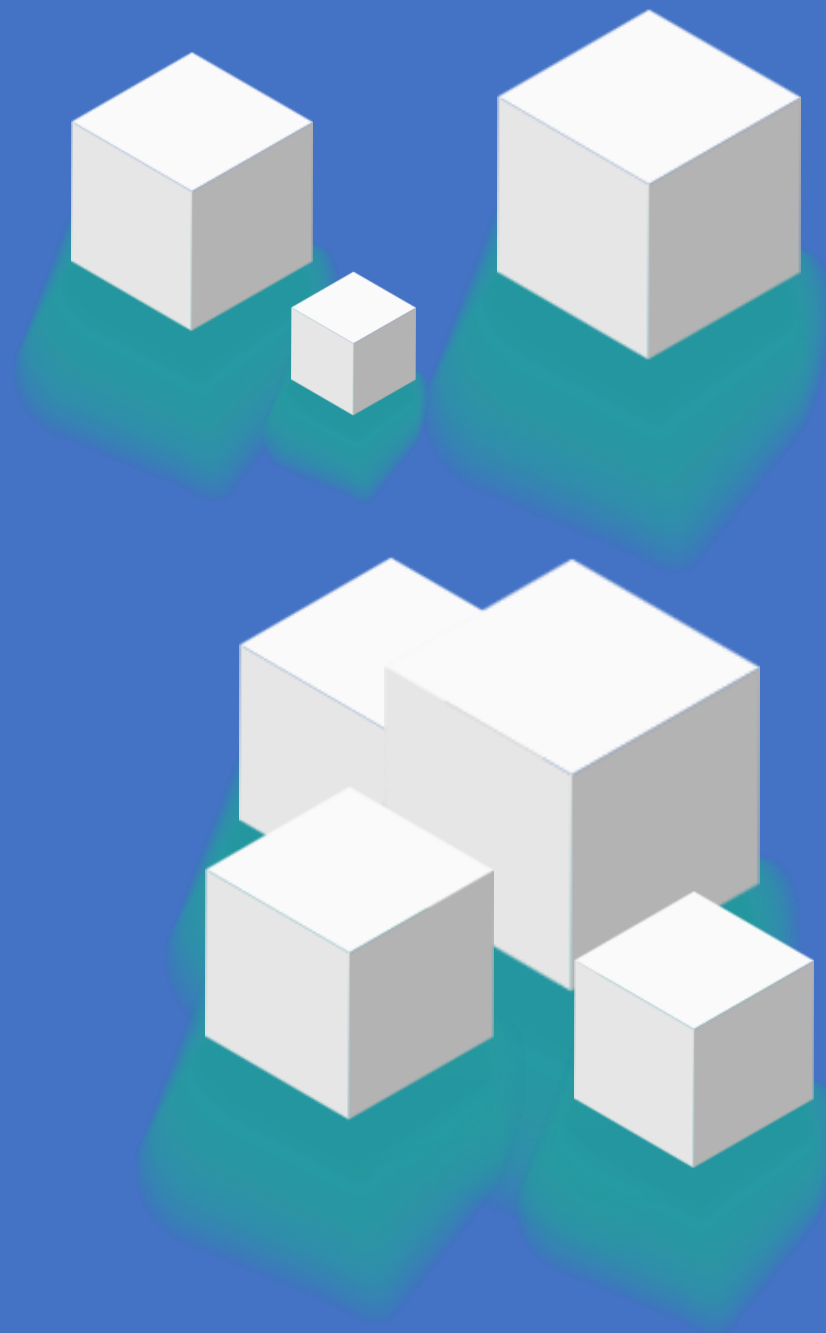
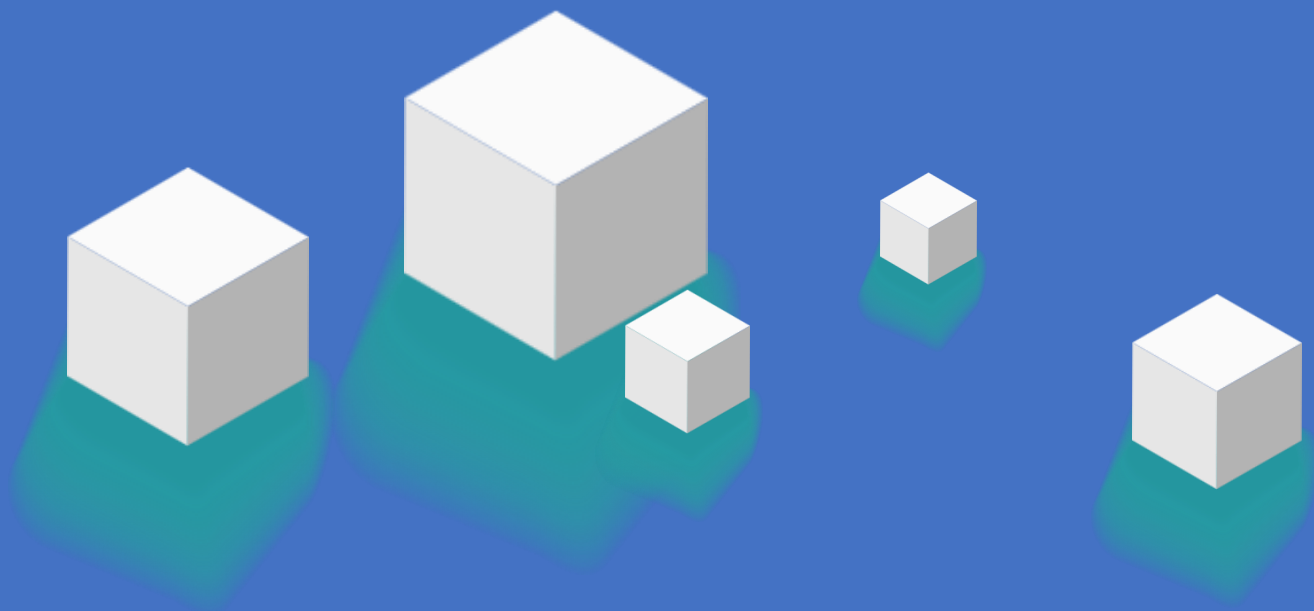


AI大模型原理与API使用



Self-introduction

和计算机、算法相关

(10岁，清华计算机博士，NOI，ACM比赛，ACM，IEEE，中国人工智能协会，阿里云MVP，CCF专委)

和培训、企业服务相关

(专栏付费订阅人数超过17万，企业客户包括：蚂蚁金服，美的，汽车之家，上汽大众，晶科能源，航天信息，中国银行，中国银联，花旗银行，汇丰银行，杭州银行，泰隆银行，中原银行，长沙银行，渤海银行，平安银行，平安保险，平安科技，阳光保险，兴业银行，兴业数金，上海银科，尚诚金融，嘉银金科，马上金融，中泰证券，富达基金，大恒集团，华润集团，中国外汇交易中心，联通软研院，梅赛德斯奔驰，雀巢，麦当劳，西门子等)



学习方法

- Thinking: behind the theory, original from the real problem
- Action: solve problems by tools, present the results

>> 今天的学习目标

AI大模型原理与API使用

- AI的分类
- LLM是如何训练的
- LLM中的Token
- Temperature与Top P 的作用
- AI Chat产品的超能力

联网搜索、读取文件、记忆功能

- 大模型API使用

系统提示词 与 用户提示词

LLM的输入与输出Token限制

CASE-情感分析-Qwen

CASE-天气Function-Qwen

CASE-表格提取-Qwen

CASE-运维事件处置-Qwen

什么是AI

AI的核心目标是让机器能够执行通常需要人类智能的任务，例如语言理解、图像识别、复杂问题解决等

- 早期阶段：以规则为基础的专家系统，依赖预设的逻辑和规则。
- 机器学习时代：通过数据训练模型，使机器能够从数据中学习规律。
- 深度学习时代：利用神经网络模拟人脑的复杂结构，处理更复杂的任务。
- 大模型时代：以大规模数据和算力为基础，构建通用性强、性能卓越的AI模型

AI的分类

分析式AI

- 也称为判别式AI，其核心任务是对已有数据进行分类、预测或决策。
- 优势在于其高精度和高效性，但其局限性在于仅能处理已有数据的模式，无法创造新内容。

生成式AI

- 专注于创造新内容，例如文本、图像、音频等。
- 突破在于其创造性和灵活性，但也面临数据隐私、版权保护等挑战

大语言模型LLM

1. 大型语言模型 (LLM)

- LLM 是基于海量文本数据训练的深度学习模型，属于生成式AI的一种。它能理解和生成类人类的自然语言，常见模型如GPT系列、DeepSeek, Qwen等。
- 具备强大的文本理解、摘要、翻译、问答及内容创作能力。通过上下文关联，能进行连贯且富有逻辑的对话与写作。并且通过少量示例可以进行下游任务的学习。

场景示例：

- 智能客服：电商网站导入基于LLM的聊天机器人，能即时理解客户复杂的售后问题，提供个性化的解决方案，大幅提升服务效率与客户满意度。
- 内容创作：营销团队使用LLM，输入产品关键字和目标受众，快速生成多版本的广告文案、社交媒体帖文与博客文章，有效降低人力成本。

生图/生视频模型

2. 生图/生视频模型 (Text-to-Image/Video)

- 专门将文字描述转换为全新的图像或视频。它们学习了图像、视频与其对应文字标签之间的关联，代表模型有DALL-E、Midjourney及Sora。
- 能够根据用户输入的文字提示（Prompt），创造出符合描述且风格多样的视觉内容。模型能融合不同概念、属性和风格，生成前所未有的原创作品。

场景示例：

- 产品设计：设计师输入“一款具有未来感的流线型运动跑鞋，采用回收海洋塑料材质”，模型可快速生成多款概念图，加速产品可视化与迭代过程。
- 影视预览：导演利用文字生成视频模型，将剧本中的关键场景转换为动态预览片段，以便在实际拍摄前，评估镜头、光影和场景布局的可行性。

视觉识别模型

3. 视觉识别模型 (Computer Vision Model)

- 视觉识别模型让计算机能“看懂”并解析图像与视频内容，属于计算机视觉领域。主要任务包括图像分类、物体检测、图像分割等，模型如YOLO、ResNet。
- 能准确辨识影像中的物体、人脸、文字或特定场景。其核心在于从像素中提取特征，并与已知模式进行比对，以完成识别、定位或追踪等任务。

场景示例：

- 智能制造：在生产线上部署视觉识别系统，能即时检测产品外观的微小瑕疵，如刮痕或缺件，自动剔除不合格品，确保出厂品质，准确率远超人眼。
- 医疗影像分析：医院导入AI辅助判读系统，分析X光或CT扫描影像。模型能快速标记出疑似肿瘤或病变的区域，协助放射科医生提高诊断效率与准确性。

视觉识别模型

4. 自动驾驶模型 (Autonomous Driving Model)

- 一套复杂的AI系统，整合了视觉识别、传感器融合、决策规划等多种模型。其目标是让车辆在无需人类干预下安全行驶，是AI技术的高度整合应用。
- 通过摄像头、激光雷达(LiDAR)等传感器，即时感知周遭环境，识别行人、车辆与交通标志。模型会预测其他物体的动态，并规划出最佳的行驶路径与操作。

场景示例：

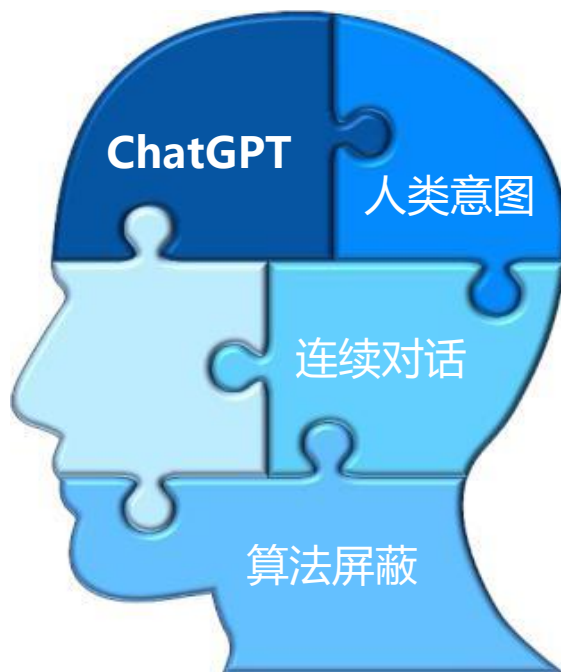
- 无人配送：物流公司采用自动驾驶货车，在特定园区或高速公路进行货物运输。系统能自主导航、避开障碍物并遵守交通规则，实现24小时不间断的物流运作。
- 高级辅助驾驶：现今许多市售车辆搭载的辅助驾驶系统，能在高速公路上自动跟车、维持车道居中。这背后就是自动驾驶模型在识别车道线与前车距离，并控制方向盘与加减速。

大语言模型

大语言模型是一种通用自然语言生成模型，使用大量预料数据训练，以实现生成文本、回答问题、对话生成等

基本能力

- 语言生成
- 上下文学习
- 世界知识



“超”能力

- 响应人类指令
- 泛化到没有见过的任务
- 代码生成和代码理解

ChatGPT是如何训练出来的？

步骤1：收集数据，微调监督模型

从问题库中选择问题

香蕉是什么？

人工编写答案

香蕉 (banana)
是芭蕉科、芭蕉属植物...

在GPT3.5
上做微调，
得到**监督学习模型**



步骤2：收集比较数据，训练奖励模型

从问题库中选择问题

香蕉是什么？

重复生成
4次回答

A: 一种酸
水果...

B: 装饰
品...

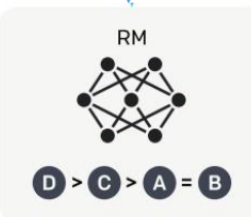
C: 一种猴
子爱吃的...

D: 香蕉是
一种黄色...

人工排序

D > C > A = B

利用排序
结果训练
奖励模型



步骤3：收集数据，强化学习优化模型

重新选一个
问题

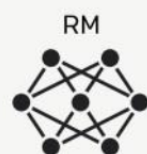
写一个水獭的故事

强化学习
模型生成
回答



很久很久以前...

喂给奖励
模型打分



生成分数，
并**持续迭代模型**

r_k

ChatGPT微调 (基于Reward Model)

ChatGPT是如何训练出来的？



用排序任务 代替打分任务，更容易让标注员给出统一的标注结果

Rank List 标注平台

用于生成模型生成 Rank List 的标注。

标注思路参考自 [InstructGPT](#)。

RLHF 更多介绍: [想训练ChatGPT? 得...](#)

Model Config

当前标注配置（可在源码中修改）：

```
{
  "model_name" :
  "uer/gpt2-chinese-
  cluecorpusmall"

  "device" : "cuda:0"

  "dataset_file" :
  "data/human_labeled/total_dataset.t

  "rank_list_len" : 4

  "max_gen_seq_len" : 40

  "random_prompts" : [
    0 : "今天我去了"
    1 : "这部电影很"
    2 : "刚收到货, 感觉"
    3 : "这部电影很"
    4 : "说实话, 真的很"
    5 : "这次购物总的来说体验很"
```

Label Dataset

Setting Prompts

随机 prompt

prompt:

今天早晨我去了

Generate Results

句子1排名

4

今天早晨我去了，一大早出来时还是一位穿着灰蓝色t恤的老奶奶。这时门口一个白发苍苍

句子2排名

3

今天早晨我去了车站，等她下面等车，我看到她手里拎着小提着小箱子在走，大家都不会

句子3排名

1

今天早晨我去了教堂，那里是我的大学。你从我们学校走出走到我的课堂，我仿佛看到了我

句子4排名

2

今天早晨我去了美国，想和你出去玩。我先去了美国，想去了之后，又打电话问他的情况时

Rank Results

Rank 1:

今天早晨我去了教堂，那里是我的大学。你从我们学校走出走到我的课堂，我仿佛看到了我

Rank 2:

今天早晨我去了美国，想和你出去玩。我先去了美国，想去了之后，又打电话问他的情况时

Rank 3:

今天早晨我去了车站，等她下面等车，我看到她手里拎着小提着小箱子在走，大家都不会

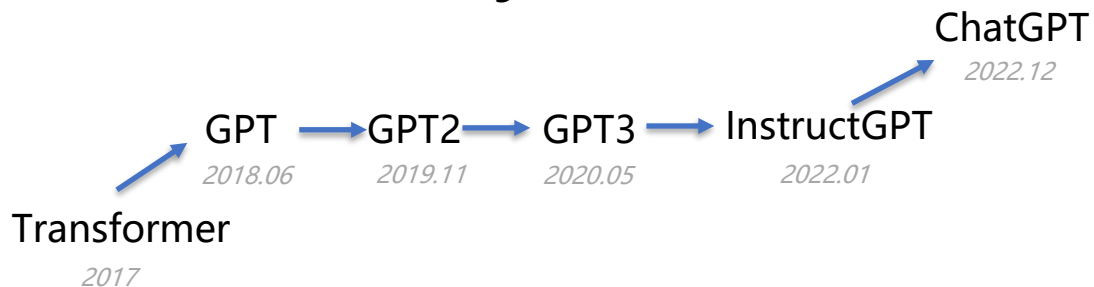
Rank 4:

今天早晨我去了，一大早出来时还是一位穿着灰蓝色t恤的老奶奶。这时门口一个白发苍苍

存储当前排序

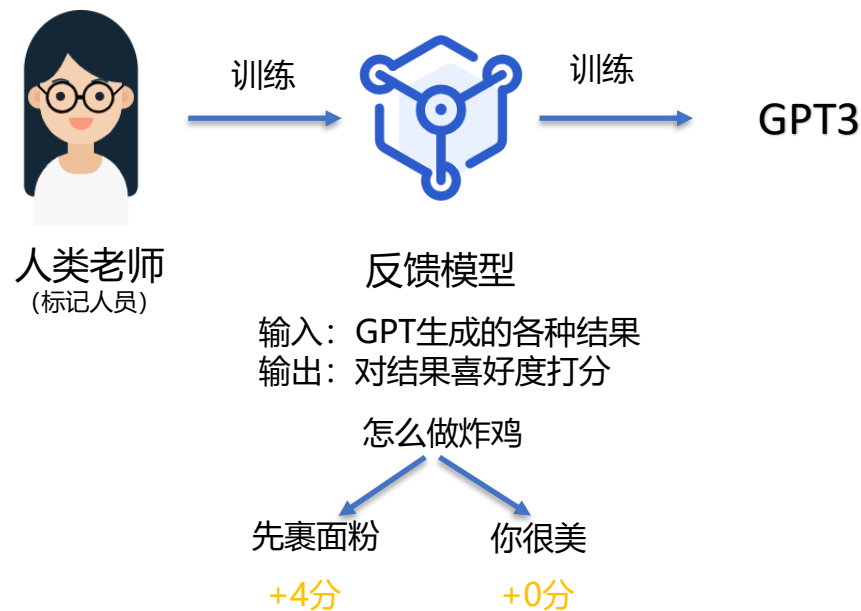
ChatGPT的优势

- 模型量级的提升
Boost in model magnitude



模型	发布时间	参数量	预训练数据量
GPT	2018 年 6 月	1.17 亿	约 5GB
GPT-2	2019 年 2 月	15 亿	40GB
GPT-3	2020 年 5 月	1,750 亿	45TB

- 基于人类反馈的强化学习 (RLHF)
Reinforcement Learning from Human Feedback



对用户真实意图的理解
Understanding of the user's true intent



上下文衔接能力
Ability to link context



对知识和逻辑的理解能力
Comprehension of knowledge and logic

不同语言模型的Token都是如何定义的

Thinking: 不同语言模型的Token是如何定义的?

Token是大型语言模型处理文本的最小单位。由于模型本身无法直接理解文字，因此需要将文本切分成一个个Token，再将Token转换为数字（向量）进行运算。不同的模型使用不同的“分词器”（Tokenizer）来定义Token。

例如，对于英文 Hello World：

GPT-4o 会切分为 ["Hello", "World"] => 对应的 token id = [13225, 5922]

对于中文“人工智能你好啊”：

DeepSeek-R1会切分为 ["人工智能", "你好", "啊"] => 对应的token id = [33574, 30594, 3266]

分词方式的不同会直接影响模型的效率和对语言细节的理解能力。

可以通过 <https://tiktokenizer.vercel.app/> 这个工具看到不同模型是如何切分你输入的文本的。

模型的常见特殊Token

为了让模型更好地理解文本的结构和指令，开发者会预设一些具有特殊功能的Token。

这些Token不代表具体词义，而是作为一种“标点”或“命令”存在。

- 分隔符 (Separator Token):

用于区分不同的文本段落或角色。比如，在对话中区分用户和AI的发言，可能会用 `<|user|>` 和 `<|assistant|>` 这样的Token。

- 结束符 (End-of-Sentence/End-of-Text Token):

告知模型文本已经结束，可以停止生成了。常见的如 `[EOS]` 或 `<|endoftext|>`。这对于确保模型生成完整且不冗长的回答至关重要。

- 起始符 (Start Token):

标记序列的开始，例如 `[CLS]` (Classification) 或 `[BOS]` (Beginning of Sentence)，帮助模型准备开始处理文本。

Temperature、Top P 的原理与作用

Thinking: 大模型中的Temperature, Top P的原理和作用是什么?

=> 控制LLM 生成文本的多样性, 但原理不同。

- Temperature (温度):

原理: 在模型计算出下一个Token所有可能的概率分布后, Temperature会调整这个分布的“平滑度”。

高Temperature (如 1.0+): 会让低概率的Token更容易被选中, 使生成结果更具创造性, 可能出现不连贯的词语。

低Temperature (如 0.2): 会让高概率的Token权重更大, 使生成结果更稳定、更符合训练数据, 但会更保守。

Temperature、Top P 的原理与作用

Top P (核采样):

原理: 它设定一个概率阈值 (P) , 然后从高到低累加所有Token的概率, 直到总和超过P为止。模型只会在这个累加出来的“核心”词汇表中选择下一个Token。

- 高Top P (如 0.9): 候选词汇表较大, 结果更多样。
- 低Top P (如 0.1): 候选词汇表非常小, 结果更具确定性。

假设模型要完成句子: “今天天气真...”

模型预测的下一个词可能是: 好(60%)、不错(30%)、糟(9%)、可乐(0.01%)。

高Temperature: 会提升所有词的概率, 使得“可乐”这个不相关的词也有机会被选中。

Top P (设为0.9): 会选择概率总和达到90%的词。这里 好(60%) + 不错(30%) = 90%, 所以模型只会从“好”和“不错”中选择, 直接排除了“可乐”这种离谱的选项。

相比Temperature, Top P能更动态地调整候选词的数量, 避免选到概率极低的离谱词汇 => 产生更高质量的文本。

AI大模型聊天产品的“超能力”

超能力1：联网搜索

弥补LLM训练数据截止日期的限制 => 获取外部信息

当用户提问涉及最新资讯时，系统会识别出这一需求，自动调用搜索Tool，并将问题转化为多个简洁的搜索关键词。接着，程序调用搜索引擎API（如Google搜索）获取信息。

最后，这些实时信息会作为上下文提供给模型，由模型进行总结和提炼，生成精准且与时俱进的回答。

例如，当你询问“黄金的涨跌和哪些因素有关？”

LLM会调用一个搜索工具，输入你刚才的问题，然后获取相关的信息

=> 整理到回答中。

AI大模型聊天产品的“超能力”

超能力2：读取文件

基于“检索增强生成”（Retrieval-Augmented Generation, RAG）的技术。

当你上传一个文件（如PDF、Word文档）时，系统首先会将其内容分割成小块（Chunks）。

然后，通过Embedding技术将这些文本块转化为数学向量，并存储在专门的“向量数据库”中。

当你针对文件内容提问时，系统会将你的问题也转化为向量，并在数据库中快速找到最相关的文本块，最后将这些文本块连同你的问题一起交给模型，生成答案。

比如，上传一份公司财报后，提问“第二季度的利润是多少？”

RAG系统能精确定位到财报中相关的片段，让LLM直接使用。

AI大模型聊天产品的“超能力”

超能力3：记忆功能（从“金鱼”到“伙伴”）

LLM本身是无状态的，每次对话都是一次全新的互动，不记得之前的交流。

为了实现“记忆”，系统会在每次对话时，将最近的几轮问答作为背景信息一起发送给模型

=> 称为“短期记忆”或“上下文窗口”。

对于需要长期记住的关键信息，例如你的名字或偏好，系统会通过特定算法提取这些信息，

=> 将其存储在用户专属的数据库中。

=> 在后续的对话中，系统会先从数据库中读取，为模型提供更个性化的背景知识。

比如，你告诉AI“我喜欢简洁的回答风格”，系统会记录这一偏好。

下次你提问时，它就会倾向于给出更简练的答复。

大模型正广泛应用于各个行业中



保险

保险条款智能解析
文本处理效率提升30倍



金融

金融信贷智能风控
借贷风险判断准确率提升21.5%



医疗

医学病例自动化抽取
病例处理效率显著提升



人力资源

候选人信息智能分类
模型识别准确率达到99%



证券

行业新闻信息抽取
智能分析行业动态



通信

短信内容智能分类与审核
过滤效率显著提升



电商

电商评论观点分析
快速搭建评论数据分析系统



物流

快递单物流地址智能识别与处理

API使用

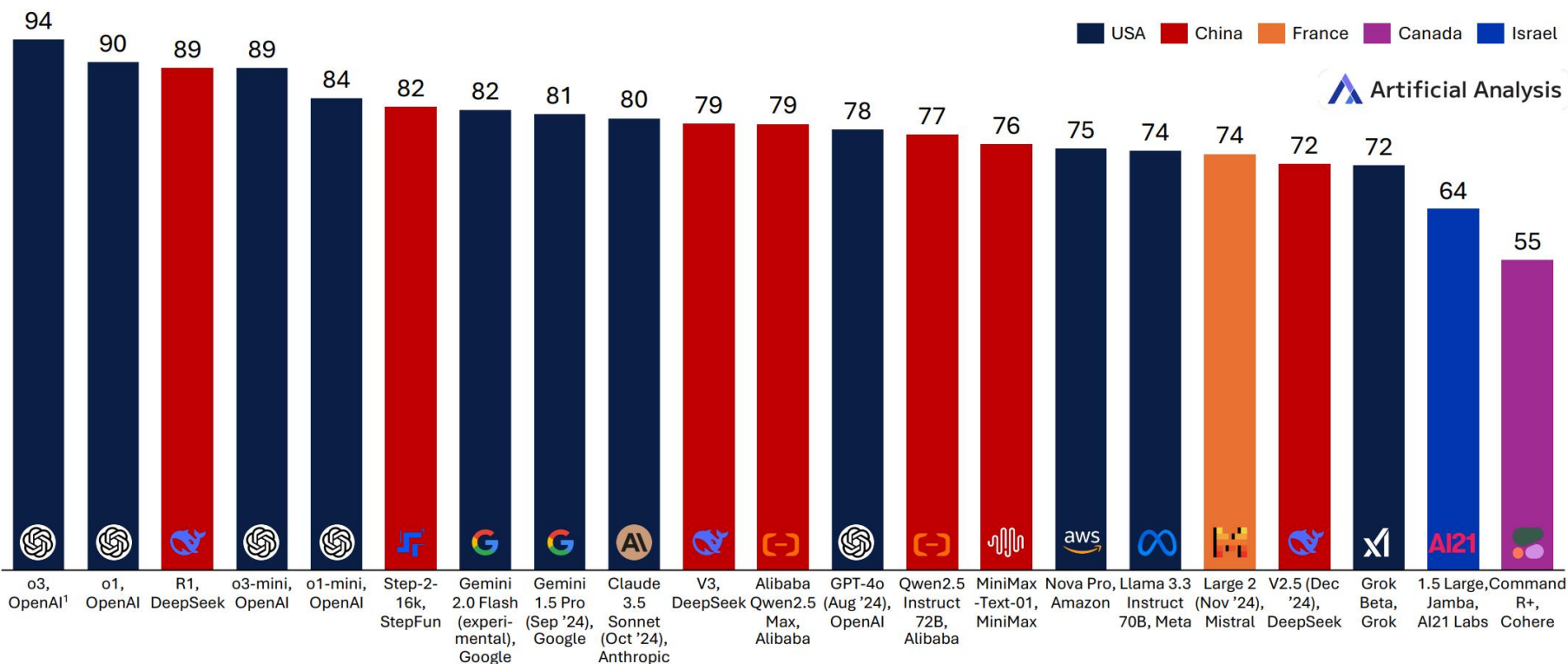
全球AI发展现状

LANGUAGE MODEL COUNTRY OF ORIGIN



While the US maintains an overall lead in the intelligence frontier, China is no longer far behind. Few other countries have demonstrated frontier-class training

The Language Model Frontier: Country of Origin
Artificial Analysis Intelligence Index, Selected Leading Models (Early 2025), Non-exhaustive



1. Estimated based on company claims and comparable results where available, not yet independently benchmarked by Artificial Analysis
2. A number of leading models from Chinese AI labs are excluded due to limited access or evaluation data

全球AI发展现状

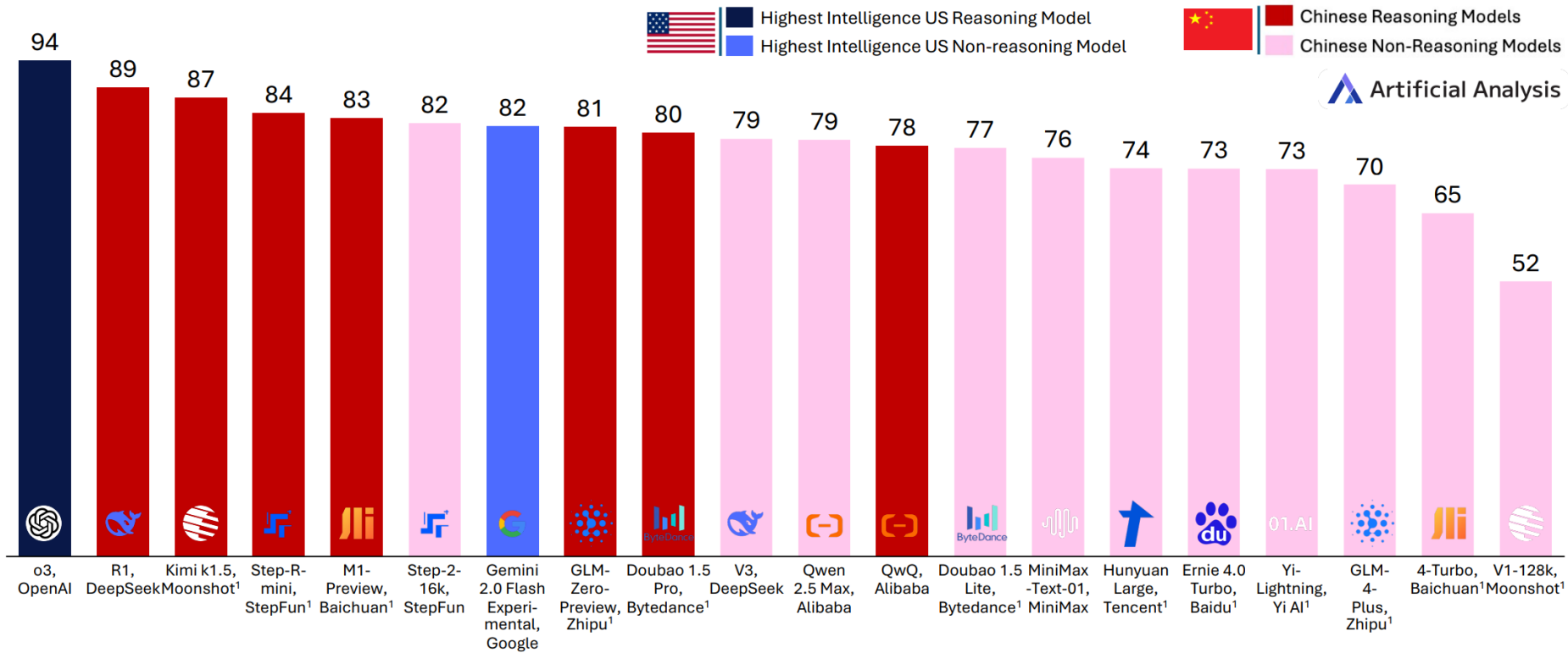
LANGUAGE MODEL COUNTRY OF ORIGIN



As of early 2025, several Chinese AI labs have demonstrated or claimed frontier-level intelligence, with seven releasing models featuring reasoning capabilities

The Language Model Frontier: Models by Chinese AI Labs

Artificial Analysis Intelligence Index, Leading Models (Early 2025), Non-exhaustive



1. Estimated based on company claims and comparable results where available, not yet independently benchmarked by Artificial Analysis

全球AI发展现状

全球AI模型发展现状（中美对比）：

- **美国**：OpenAI、Anthropic、Google、Meta等公司主导前沿模型，如GPT-4o、Claude 4 Sonnet、Gemini 2.5 Flash。
- **中国**：DeepSeek（如R1、V3）、阿里巴巴（如Qwen3）、Moonshot等公司快速追赶，部分模型（如Kimi K2, DeepSeek R1）已接近美国前沿水平。
- **关键趋势**：中国模型在2024年显著缩小与美国的差距，尤其在推理模型和开源模型领域表现突出。
- **其他地区**：法国（Mistral）、加拿大（Cohere）等也有前沿模型，但中美仍是主导力量。

出口限制与硬件影响

美国对华限制：

- 时间线：2022年10月首次限制（H100、A100），2023年10月升级（H800、A800受限），2025年1月新增“AI扩散规则”。
- 当前状态：仅H20、L20等低性能芯片可出口中国，未来可能进一步收紧。
- 影响：中国依赖国产芯片（如华为昇腾）或降级版NVIDIA芯片（如H20，算力仅为H100的15%）。

硬件性能对比：

- NVIDIA H100：989 TFLOPs，3.35 TB/s带宽。
- NVIDIA H20：148 TFLOPs，4 TB/s带宽（专为中国市场设计）。
- AMD MI300X：1307 TFLOPs，5.3 TB/s带宽（未受限制）。

AI扩散规则（AI Diffusion Rule）是美国对华芯片出口管制政策的进一步升级，美的在通过三级许可框架 严格限制先进AI加速器流向中国及其他特定国家。

中国AI公司概况

大科技公司:

- 阿里巴巴: 通义千问 (Qwen) 系列, Qwen3
- 百度: 文心一言 (Ernie 4.0 Turbo)
- 腾讯: 混元大模型 (Hunyuan Large)
- 字节跳动: 豆包 (Doubao 1.6 Pro)
- 华为: 盘古5.0 (Pangu 5.0 Large)

初创公司:

- DeepSeek: R1、V3, 开源模型表现优异。
- Moonshot: Kimi K2, 专注长上下文窗口。
- MiniMax: Text-01, 多模态能力突出。
- 其他: 智谱AI (ChatGLM)、百川智能 (Baichuan) 等。

CASE: 情感分析-Qwen

CASE：情感分析-Qwen

TO DO：对用户观点评论进行情感分析，即正向、负向

使用 dashscope中的Qwen-Turbo

针对提取的用户评论，可以进行批量化分析

	用户评论	情感
0	价格还可以，我是6768拿下的（用了一张500返20的券），第二天就涨回到6999了。送的正...	正向
1	非常好，有品质	正向
2	我是韶音忠实粉丝，从上一款AS800头戴旗舰到这一次OpenFit，从未让我失望，音质有了更...	正向
3	韶音的品质一直没问题。	正向
4	这款音效特别好 给你意想不到的音质。	正向
5	佩戴非常舒服，无感佩戴，随便运动不会掉	正向
6	预装的是Linux,不是xp，给客服打电话说不能换操作系统，要不就不保修了，哪有这样的道理	负向
7	牌子够老，够响亮，冲着牌子去的，结果让人很伤心！唉。。。。。。	负向
8	用了几天，结果系统崩溃了，到同方检测，发现30%坏道，已经退回换货了，不知道换来的如何	负向

商品评论观点.xlsx

CASE：情感分析-Qwen

```
import json
import os
import dashscope
from dashscope.api_entities.dashscope_response import Role
dashscope.api_key = "sk-XX"

# 封装模型响应函数
def get_response(messages):
    response = dashscope.Generation.call(
        model='qwen-turbo',
        messages=messages,
        result_format='message' # 将输出设置为message形式
    )
    return response
```

```
review = '这款音效特别好 给你意想不到的音质。'
messages=[
    {"role": "system", "content": "你是一名舆情分析师，帮我判断产品口碑的正负向，回复请用一个词语：正向 或者 负向"},
    {"role": "user", "content": review}
]

response = get_response(messages)
response.output.choices[0].message.content
```

运行结果：

'正向'

CASE：情感分析-Qwen

Thinking：什么是 DashScope API？

DashScope 是阿里云提供的模型即服务（Model-as-a-Service）平台 API。集合了多种AI大模型（比如这里使用的 deepseek-v3）

Thinking：如何使用这个 API？

首先，需要通过 `pip install dashscope` 安装函数库。

然后，设置API 密钥。再按照规定的格式准备输入消息（messages），

最后调用 `dashscope.Generation.call()` 函数，即可向指定的模型发送请求并获得回复。

CASE：情感分析-Qwen

Thinking：调用参数（call 函数）有哪些？

- model: 指定使用的模型，例如 'deepseek-v3'。
- messages: 传递给模型的对话内容，是一个包含多个字典的列表，每个字典代表一则消息。

messages 是一个列表，其中每个元素都是一个字典，包含 role 和 content 两个键。

role: 角色，可以是 'system' (系统指示)、'user' (用户提问) 或 'assistant' (AI回复)。

content: 该角色说的具体内容。

- result_format: 设定返回结果的格式。此处 'message' 表示标准化的消息对象格式。

Thinking：响应参数（response 对象）包含什么？

API 的响应是一个对象。我们主要关注 `response.output.choices[0].message.content`。这层层深入的路径，代表从模型众多可能的响应（choices）中，取出第一个（[0]），并读取其消息（message）中的实际文本内容

DashScope使用方法

1、基本设置：

```
import dashscope
from dashscope.api_entities.dashscope_response import Role

# 设置 API key
dashscope.api_key = "your-api-key"
```

2、模型调用：

```
# 基本调用格式
response = dashscope.Generation.call(
    model='模型名称', # 例如: 'qwen-turbo', 'deepseek-r1' 等
    messages=messages, # 消息列表
    result_format='message' # 输出格式
)
```

3、messages 格式：

```
messages = [
    {"role": "system", "content": "系统提示信息"},
    {"role": "user", "content": "用户输入"},
    # 如果有历史对话
    {"role": "assistant", "content": "助手回复"},
    {"role": "user", "content": "用户新的输入"}
]
```

DashScope使用方法

4、常用参数：

```
response = dashscope.Generation.call(  
    model='模型名称',  
    messages=messages,  
    result_format='message', # 输出格式  
    temperature=0.7,      # 温度参数，控制随机性  
    top_p=0.8,           # 控制输出的多样性  
    max_tokens=1500,     # 最大输出长度  
    stream=False         # 是否使用流式输出  
)
```

5、获取响应结果

```
# 获取生成的文本  
result = response.output.choices[0].message.content  
  
# 如果是流式输出  
for chunk in response:  
    print(chunk.output.choices[0].message.content, end='')
```

系统提示词 与 用户提示词

系统提示词 (System Prompt)

- 用于设定AI的角色、行为准则和输出格式，是贯穿对话的全局指令，为其提供了扮演的“人设”。
- 应在对话开始时设定，内容要清晰明确。
- 它会像普通问题一样消耗Token，不应在其中包含用户的具体问题。频繁更改可能导致AI行为不稳定。

Thinking: 如何要打造一个代码助手，如何设定系统提示词？

你是一个资深程序员，请直接提供代码，并用Markdown格式包裹。不要解释，不要说任何无关的话。

LLM的输入与输出Token限制

输入Token限制

- 模型单次API调用能处理的最大信息量，包含系统提示词、历史对话和当前用户输入的所有内容。
- 所有输入内容的总长度不能超过此限制，否则API会报错。
- 我们需自行管理历史对话长度，比如通过截断或总结旧消息来确保请求不超过上限。

假设某模型上限为4096 Token(Context Window)。如果此时系统提示词和历史对话已占用3500 Token

=> 那么用户提示词长度不能超过 $4096 - 3500 = 596$ Token。

LLM的输入与输出Token限制

输出Token限制

- 指模型在一次回复中能生成的最大内容长度。
- 我们通常可以在API请求中手动设置此参数（如 `max_output_tokens`）。
- 设置过低会导致回答不完整，内容被突然截断；
- 设置过高则可能增加API调用时间和费用。需要根据具体任务需求权衡。

如果你请求模型写一首诗，但将输出限制设为5 Token，那么可能只会得到诗的第一句，后面内容会被截断。

CASE: Function Call使用 (Qwen)

CASE: Function Call使用-Qwen

TO DO: 编写一个天气Function，当LLM要查询天气的时候提供该服务，比如当前不同城市的气温为：

北京：35度

上海：36度

深圳：37度

天气均为晴天，微风

1) 使用 model= "qwen-max"

2) 编写function get_current_weather

对于用户询问指定地点的天气，可以获取该地当前天气



CASE: Function Call使用-Qwen

1. 模拟天气查询的函数

```
def get_current_weather(location, unit="摄氏度"):
    # 这是一个模拟的天气数据，实际应用中应该调用真实的天气 API
    temperature = -1
    if '大连' in location or 'Dalian' in location:
        temperature = 10
    if location=='上海':
        temperature = 36
    if location=='深圳':
        temperature = 37
    weather_info = {
        "location": location,
        "temperature": temperature,
        "unit": unit,
        "forecast": ["晴天", "微风"],
    }
    return json.dumps(weather_info)
```

2. 模型调用封装

```
def get_response(messages):
    try:
        response = dashscope.Generation.call(
            model='qwen-turbo',
            messages=messages,
            functions=functions, # 注册可用的函数
            result_format='message'
        )
        return response
    except Exception as e:
        print(f"API调用出错: {str(e)}")
        return None
```

CASE: Function Call使用-Qwen

3. 主要对话流程

步骤 1: 发送用户查询

query = "大连的天气怎样"

messages=[{"role": "user", "content": query}]

response = get_response(messages)

步骤 2: 检查模型是否需要调用函数

if hasattr(message, 'function_call') and message.function_call:

获取函数调用信息

function_call = message.function_call

tool_name = function_call['name']

arguments = json.loads(function_call['arguments'])

步骤 3: 执行函数调用

tool_response = get_current_weather(

location=arguments.get('location'),

unit=arguments.get('unit'),

)

步骤 4: 将函数返回结果添加到对话

tool_info = {"role": "function", "name": tool_name, "content":
tool_response}

messages.append(tool_info)

步骤 5: 让模型生成最终回答

response = get_response(messages)

CASE: Function Call使用-Qwen

4. 函数注册配置

```
functions = [  
  {  
    'name': 'get_current_weather', # 函数名称  
    'description': 'Get the current weather in a given location.', # 函数描述  
    'parameters': { # 函数参数定义  
      'type': 'object',  
      'properties': {  
        'location': {  
          'type': 'string',  
          'description': 'The city and state, e.g. San Francisco, CA'  
        },  
        'unit': {'type': 'string', 'enum': ['celsius', 'fahrenheit']}  
      },  
      'required': ['location'] # 必需参数  
    }  
  }  
]
```

整体工作流程:

- 用户输入查询天气的问题
- 模型理解问题, 决定需要调用天气查询函数
- 模型生成函数调用参数 (城市、温度单位)
- 程序执行函数调用, 获取天气数据
- 将天气数据返回给模型
- 模型生成最终的自然语言回答

CASE: 表格提取-Qwen

CASE：表格提取-Qwen

TO DO：表格提取与理解是工作中的场景任务，需要使用多模态模型，这里可以使用通义千问VL系列的模型

1) Qwen-VL（基础模型）

核心能力：支持图像描述、视觉问答（VQA）、OCR、文档理解和视觉定位

2) Qwen-VL-Chat（指令微调版）

基于Qwen-VL进行指令微调（SFT），优化对话交互能力

3) Qwen-VL-Plus / Qwen-VL-MAX（升级版）

性能更强，接近GPT-4V水平，但未完全开源

4) Qwen2.5-VL（最新旗舰版）

模型规模：提供3B、7B、72B版本，适应不同计算需求

客户名称		客诉日期		严重程度	<input type="checkbox"/> 一般 <input type="checkbox"/> 重大
联系方式		回复时间		紧急程度	<input type="checkbox"/> 一般 <input type="checkbox"/> 紧急
产品型号		生产日期	年 月	数量	1
客户诉求				问题产品追踪	<input type="checkbox"/> 客户处 <input type="checkbox"/> 物流中 <input type="checkbox"/> 已回厂
				客户诉求点	<input type="checkbox"/> 退货 <input type="checkbox"/> 换货 <input type="checkbox"/> 维修
				图例说明	
描述人/提报人:				2018 年__月__日	
问题要因分析	分析人: 2018 年__月__日				
原因归属: <input type="checkbox"/> 设计 <input type="checkbox"/> 可靠性 <input type="checkbox"/> 品质部 <input type="checkbox"/> 生产部 <input type="checkbox"/> 仓储 <input type="checkbox"/> 运输 <input type="checkbox"/> 其它					
零时措施	零时对策建议: 更换液压头, 更换阀盘。			对策方法	<input type="checkbox"/> 库存产品再检验
					<input type="checkbox"/> 退回二级供应商
					<input type="checkbox"/> 生产停产纠正
					<input type="checkbox"/> 其它
改善措施	1、防止发生对策:				
	建意人: 日期: 2018 年__月__日				
	要求完成时间: 2018 年__月__日 进程追踪: <input type="checkbox"/> 按时完成 <input type="checkbox"/> 延期完成 <input type="checkbox"/> 未完成				
纠正归属: <input type="checkbox"/> 设计部 <input type="checkbox"/> 品质部 <input type="checkbox"/> 装配车间 <input type="checkbox"/> 压铸车间 <input type="checkbox"/> 车床车间 <input type="checkbox"/> 仓库 <input type="checkbox"/> 运输 <input type="checkbox"/> 其它					
备注说明					

CASE：表格提取-Qwen

```
# 构建多模态输入
content = [
    # 图片输入：支持本地路径或URL
    {'image': 'https://aiwucai.oss-cn-huhehaote.aliyuncs.com/pdf_table.jpg'},
    # 文本提示：要求提取表格内容并输出JSON格式
    {'text': '这是一个表格图片，帮我提取里面的内容，输出JSON格式'}
]

# 构建消息格式
messages=[{"role": "user", "content": content}]
```

整体工作流程：

- 使用了多模态模型（qwen-vl-plus），可以同时处理图片和文本
- 支持表格识别和内容提取
- 可以将非结构化的表格图片转换为结构化的JSON数据

CASE：表格提取-Qwen

好的，以下是整理后的生成 JSON 格式内容：

```
``json
{
  "客户名称": "",
  "联系方式": "",
  "产品型号": "",
  "生产日期": "",
  "数量": 0,
  "使用年限": null,
  "严重程度": "",
  "紧急程度": "",
  "问题点": [],
  "退货": false,
  "换货": false,
  "维修": false,
  "图例说明": "",
  "描述人/提报人": {
    "__DATE__": ""
```

```
},
  "分析人": {
    "__DATE__": ""
  },
  "原因归属": [
    ""
  ],
  "零时措施": {},
  "改善措施": {}
}
``
```

请注意这个 JSON 对象中的键值对可能需要根据实际的表单结构进行调整。例如，“联系方式”和“联系方法”的字段名应该是一致的；同样地，“严重程度”、“紧急程度”等也可能有误，请参照原图像自行修正。

另外，在处理文本信息（如日期）的时候需要注意它们的具体形式，并在转换为 JSON 值之前正确解析这些数据。如果存在缺失或错误的数项，则应相应地添加 `null` 或空字符串来表示该属性不存在或者没有提供具体的信息。

Qwen-VL擅长视觉理解和识别，而且可以私有化部署和微调

CASE: 运维事件处置-Qwen

CASE：运维事件处置中的大语言模型应用

CASE：运维事件处置中的大语言模型应用

场景描述：运维事件的分析和处置流程。包括告警内容理解，分析方法建议，分析内容自动提取，处置方法推荐和执行等环节。AI大模型可以加速了运维过程中的问题诊断、分析与处置，提高了响应速度和决策质量，降低故障对业务的影响。



运维事件的分析和处置流程。包括告警内容理解，分析方法建议，分析内容自动提取，处置方法推荐和执行等环节，其中：

- 1、告警内容理解。**根据输入的告警信息，结合第三方接口数据，判断当前的异常情况（告警对象、异常模式）；
- 2、分析方法建议。**根据当前告警内容，结合应急预案、运维文档和大语言模型自有知识，形成分析方法的建议；
- 3、分析内容自动提取。**根据用户输入的分析内容需求，调用多种第三方接口获取分析数据，并进行总结；
- 4、处置方法推荐和执行。**根据当前上下文的故障场景理解，结合应急预案和第三方接口，形成推荐处置方案，待用户确认后调用第三方接口进行执行。

CASE：运维事件处置中的大语言模型应用

1. 告警内容理解

假设我们有一个告警信息：

告警：数据库连接数超过设定阈值

时间：2024-08-03 15:30:00

根据这个告警信息，我们可以进行如下分析：

- **告警对象**：数据库服务器
- **异常模式**：连接数超过设定阈值

2. 分析方法建议

结合应急预案、运维文档和大语言模型自有知识，采用以下分析方法：

- **获取实时数据**：调用监控系统接口，获取当前数据库服务器的连接数、CPU 使用率、内存情况等性能指标。
- **对比历史数据**：分析历史数据，确定是否存在正常范围内的波动或者是异常的长期趋势。
- **识别潜在原因**：根据数据库连接数异常的时间点、相关日志和监控数据，尝试识别可能导致连接数增加的具体原因，如程序异常、大量查询请求等。

CASE：运维事件处置中的大语言模型应用

3. 分析内容自动提取

根据用户需求，自动调用多种第三方接口获取分析数据，并进行总结，比如：

- **查询性能监控系统接口**，获取当前数据库连接数和系统负载情况。
- **检索日志管理系统接口**，查看与数据库连接数相关的日志记录。
- **调用事件管理系统接口**，获取先前类似事件的解决方案和操作记录。

4. 处置方法推荐和执行

基于当前的故障场景理解，结合应急预案和第三方接口数据，可以形成以下处置方案：

优化数据库配置：根据实时监控数据，调整数据库连接池的大小和相关参数，以减少连接数超过阈值的风险。

排查异常会话：通过数据库管理工具，查找并终止占用大量连接资源的异常会话或查询。

系统重启或备份恢复：如果上述措施无效，考虑在非业务高峰时段进行系统重启或者从备份恢复数据库，以恢复正常操作。

CASE：运维事件处置中的大语言模型应用

```
# 通过第三方接口获取数据库服务器状态
```

```
def get_current_status():
```

```
    # 生成连接数数据
```

```
    connections = random.randint(10, 100)
```

```
    # 生成CPU使用率数据
```

```
    cpu_usage = round(random.uniform(1, 100), 1)
```

```
    # 生成内存使用率数据
```

```
    memory_usage = round(random.uniform(10, 100), 1)
```

```
    status_info = {
```

```
        "连接数": connections,
```

```
        "CPU使用率": f"{cpu_usage}%",
```

```
        "内存使用率": f"{memory_usage}%"
```

```
    }
```

```
    return json.dumps(status_info, ensure_ascii=False)
```

```
# 封装模型响应函数
```

```
def get_response(messages):
```

```
    response = dashscope.Generation.call(
```

```
        model='qwen-turbo',
```

```
        messages=messages,
```

```
        tools=tools,
```

```
        result_format='message' # 将输出设置为message形式
```

```
    )
```

```
    return response
```

```
current_locals = locals()
```

```
current_locals
```

CASE：运维事件处置中的大语言模型应用

```
tools = [  
    {  
        "type": "function",  
        "function": {  
            "name": "get_current_status",  
            "description": "调用监控系统接口，获取当前数据库服务器性能指标，包括：连接数、CPU使用率、内存使用率",  
            "parameters": {  
            },  
            "required": []  
        }  
    }  
]
```

```
query = """告警：数据库连接数超过设定阈值
```

```
时间：2024-08-03 15:30:00
```

```
"""
```

```
messages=[
```

```
    {"role": "system", "content": "我是运维分析师，用户会告诉我们告警内容。我会基于告警内容，判断当前的异常情况（告警对象、异常模式）"},
```

```
    {"role": "user", "content": query}]
```

CASE：运维事件处置中的大语言模型应用

```
while True:

    response = get_response(messages)

    message = response.output.choices[0].message

    messages.append(message)

if response.output.choices[0].finish_reason == 'stop':

    break

# 判断用户是否要call function

if message.tool_calls:

    # 获取fn_name, fn_arguments

    fn_name = message.tool_calls[0]['function']['name']

    fn_arguments = message.tool_calls[0]['function']['arguments']

    arguments_json = json.loads(fn_arguments)

    function = current_locals[fn_name]
```

```
tool_response = function(**arguments_json)

tool_info = {"name": "get_current_weather", "role": "tool",
"content": tool_response}

messages.append(tool_info)

print(messages)

[{'role': 'system',
  'content': '我是运维分析师，用户会告诉我们告警内容。我会基于告警内容，判断当前的异常情况（告警对象、异常模式）'},
 {'role': 'user', 'content': '告警：数据库连接数超过设定阈值\n时间：2024-08-03 15:30:00\n'},
  Message({'role': 'assistant', 'content': '收到您的告警信息，当前出现了数据库连接数超过设定阈值的情况。这可能表明数据库服务器正在承受超出预期的负载。为了进一步分析和解决这个问题，我们需要收集一些关键信息并执行以下步骤：\n\n1. **确认当前的数据库连接数**：通过调用监控系统接口来获取实时的数据库连接数，并检查它是否确实超过了预设的阈值。\n\n2. **分析连接峰值时间**：了解连接数增加的具体时间段，以便确定问题是在业务高峰期还是特定操作期间发生的。\n\n3. **检查资源使用情况**：查看CPU使用率、内存使用率等性能指标，以判断是否还有其他资源瓶颈影响了数据库性能。\n\n4. **排查异常请求**：检查是否有大量的并发查询、事务或者特定类型的操作导致连接数激增。\n\n5. **评估扩展需求**：如果频繁发生此类告警，可能需要考虑数据库横向扩展（如增加实例）或者优化现有配置。\n\n首先，让我们获取当前的数据库连接数和其他关键性能指标。', 'tool_calls': [{'function': {'name': 'get_current_status', 'arguments': '{}', 'index': 0, 'id': 'call_7c4deb3357c54299b89b4b', 'type': 'function'}}]},
 {'name': 'get_current_weather',
  'role': 'tool',
  'content': '{"连接数": 92, "CPU使用率": "93.5%", "内存使用率": "81.6%"}'},
  Message({'role': 'assistant', 'content': '获取到的数据如下：\n\n- 当前数据库连接数：92个\n- CPU使用率：93.5%\n- 内存使用率：81.6%\n\n根据这些数据，我们可以看到数据库连接数已接近其阈值，并且CPU和内存使用率都处于较高水平，这可能表明服务器正在承受较大负载。接下来，我们需要进一步分析连接峰值时间和是否存在任何异常操作，以确定问题的根本原因。同时，我们也要考虑可能的优化措施或扩展方案，以确保系统的稳定运行。'}])
```

Summary

都有哪些Function Tool需要编写，比如：

- **查询性能监控系统接口**，获取当前数据库连接数和系统负载情况。
- **检索日志管理系统接口**，查看与数据库连接数相关的日志记录。
- **调用事件管理系统接口**，获取先前类似事件的解决方案和操作记录。
- **对比历史数据**：分析历史数据，确定是否存在正常范围内的波动或者是异常的长期趋势。

TO DO：

- 1、都有哪些告警情况（可以使用AI模型）
- 2、编写Function Tool
- 3、AI会生成哪些处置方法推荐？
- 4、生成处置方法推荐的自动化执行 Function Tool

打卡：大模型API使用






结合你的业务场景，编写一个使用AI大模型API的示例，比如：

- 1) 对情感进行分类
- 2) 对文章进行总结
- 3) 使用Function Call完成复杂的业务逻辑

可以使用Qwen API，也可以使用DeepSeek，ChatGLM，文心一言，KIMI的API

- 完成的同学，请将大模型API使用的打卡截屏，发到微信群中！



Thank You
Using data to solve problems