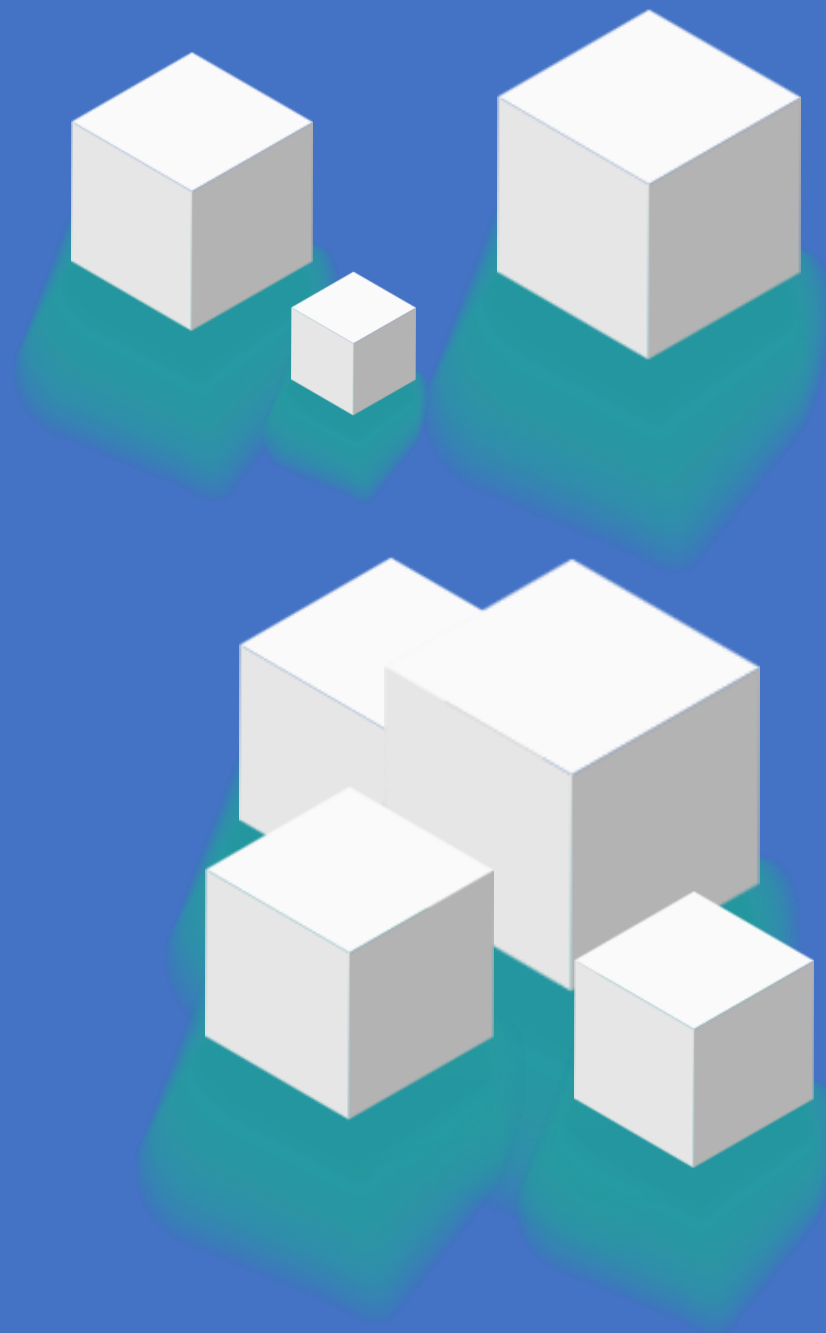
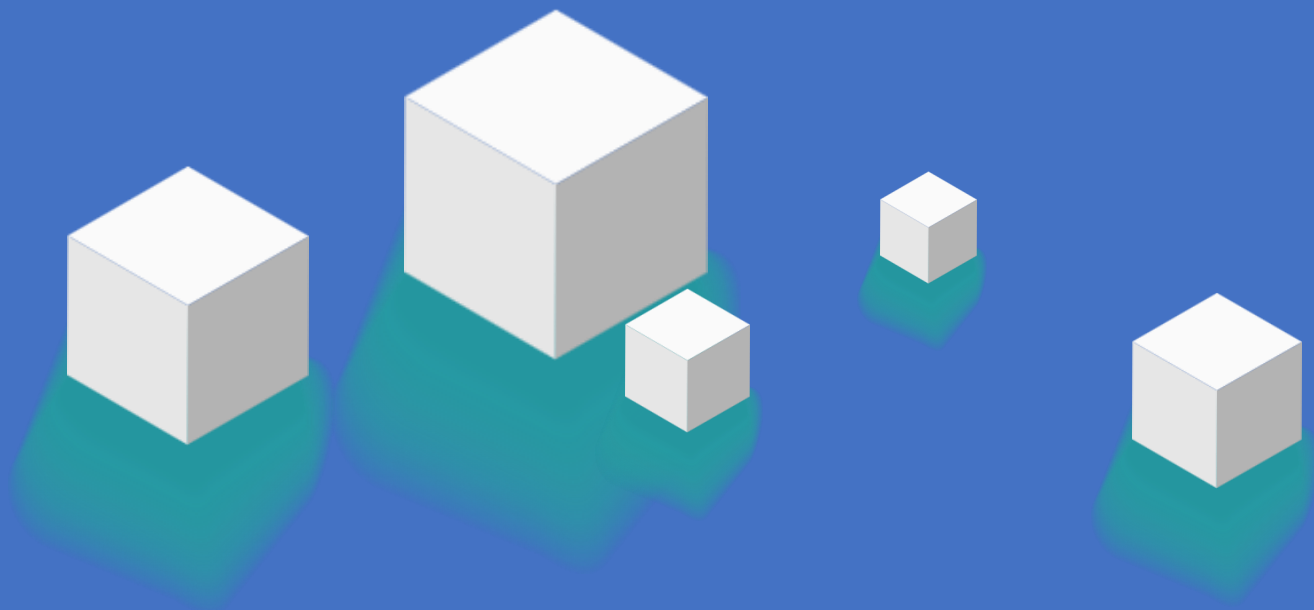


大模型API使用



>> 今天的学习目标

大模型API使用

- 全球AI发展现状
- CASE-情感分析-Qwen
- CASE-Function Call-Qwen
- CASE-表格提取-Qwen
- CASE-运维事件处置-Qwen

全球AI发展现状

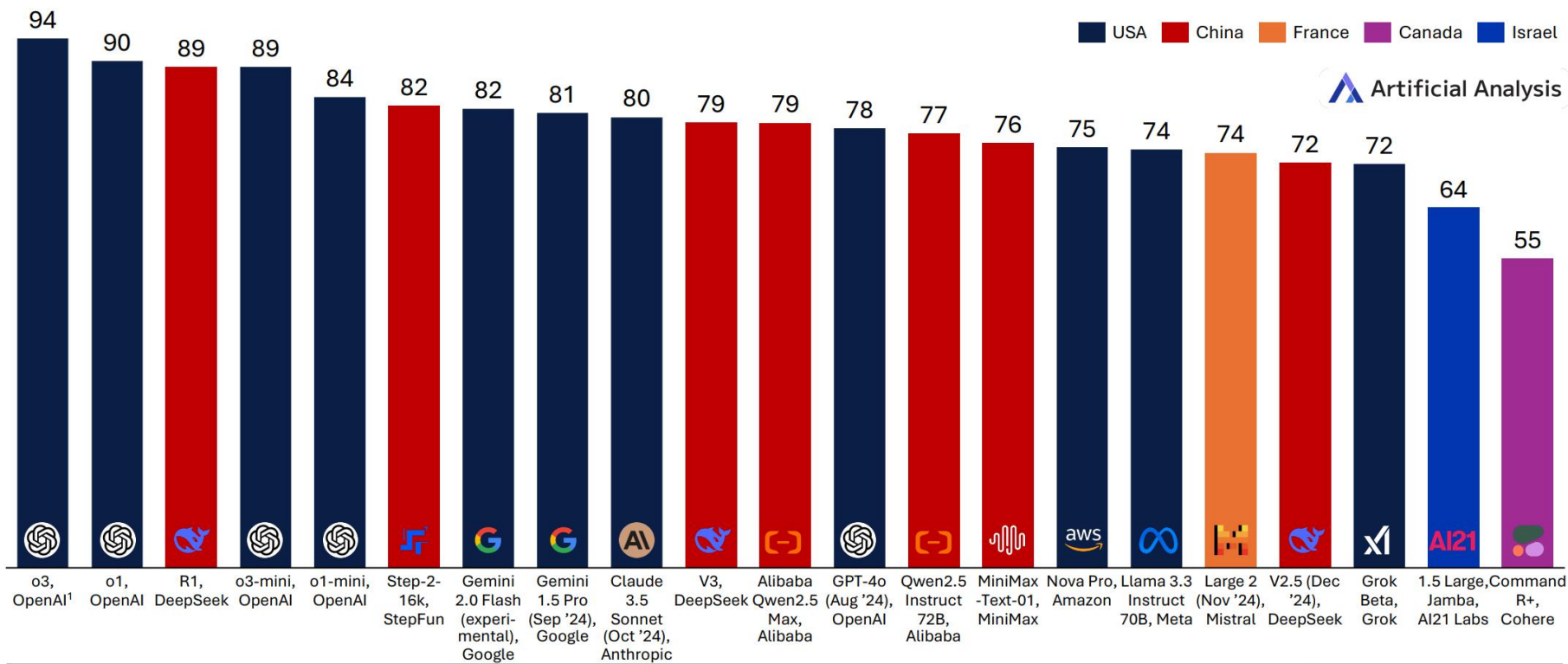
LANGUAGE MODEL COUNTRY OF ORIGIN



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

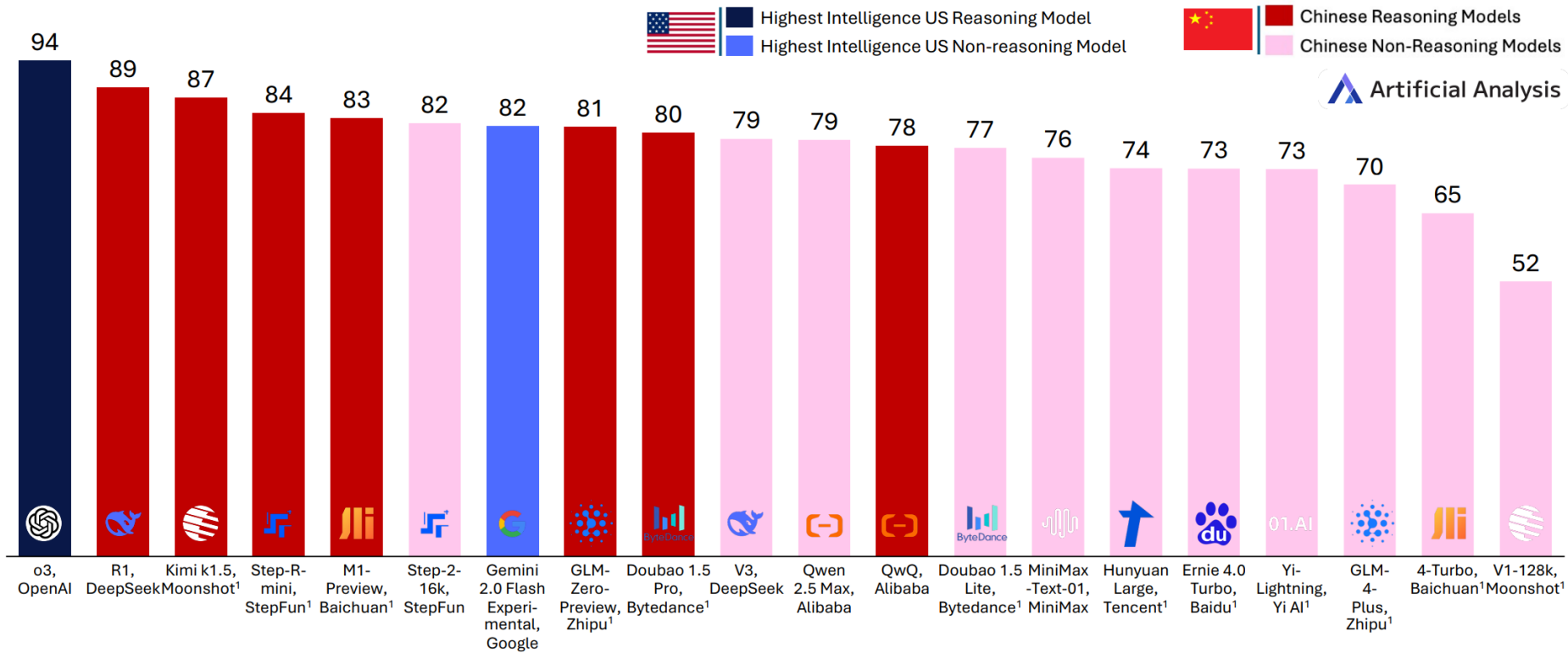
LANGUAGE MODEL COUNTRY OF ORIGIN



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 3.7 Sonnet、Gemini 2.0 Flash。
- **中国**：DeepSeek（如R1、V3）、阿里巴巴（如Qwen2.5）、Moonshot等公司快速追赶，部分模型（如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, Qwen2.5 Max (Intelligence: 79) 。
- **百度:** 文心一言 (Ernie 4.0 Turbo, Intelligence: 78) 。
- **腾讯:** 混元大模型 (Hunyuan Large, Intelligence: 74) 。
- **字节跳动:** 豆包 (Doubao 1.5 Pro, Intelligence: 80) 。
- **华为:** 盘古5.0 (Pangu 5.0 Large) 。

初创公司:

- **DeepSeek:** R1 (Intelligence: 89) 、 V3 (Intelligence: 79) , 开源模型表现优异。
- **Moonshot:** Kimi K1.5 (Intelligence: 87) , **专注长上下文窗口。**
- **MiniMax:** Text-01 (Intelligence: 76) , 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'

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='')
```

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-turbo"

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": {}
}
```

```
},
 "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“联系方式”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据项，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使用方案发到微信群中，有积分哦！



Thank You
Using data to solve problems