

# 大语言模型工具入门



# 目录

CONTENTS



01 | 常用大语言模型工具介绍

02 | 网页端问答工具的使用和实操

03 | Python入门和基础语法

04 | 大语言模型API的基本使用

# 01

---

## 常用大模型工具介绍

# 什么是大语言模型

01

定义：大语言模型是基于深度学习的超大规模预训练模型，具备强大的语言生成、推理和交互能力。

02

典型代表：GPT-4、Claude、DeepSeek等，这些模型在自然语言处理领域表现出色，能够处理复杂的语言任务。

03

核心能力：生成高质量文本、进行逻辑推理、与用户进行自然交互。

# 国外大模型举例：GPT家族



生态系统最成熟，拥有最庞大的开发者社区和插件生态。不过成本较高，API调用费用对于大规模应用而言是一笔不小开支，定制化程度相对较低。

模型名称	发布时间	参数规模	主要特点
GPT-3.5	2022年11月30日	175B (官方未公开)	基于人工标注数据和强化学习优化性能，成为ChatGPT的基础模型，支持文本生成、问答等任务。
GPT-4	2023年3月14日	未公开	多模态能力 (支持图像处理)，语言理解更强大，仅对ChatGPT Plus订阅用户开放。
GPT-4o	2024年5月13日	未公开	免费开放，支持文本、视觉、音频多模态输入；推出轻量化版本GPT-4o mini，性价比更高。
o1系列	2024年9月12日	未公开	强化学习与思维链技术结合，推理能力显著提升 (编程、数学领域突出)，但事实性知识较弱。
o3-mini	2024年12月5日	未公开	性能与性价比超越o1，支持代码生成与调试，后被整合至GPT-5，不再作为独立模型发布。

# 国外大模型举例：Claude家族



## Anthropic: Claude 模型

Claude在逻辑推理、长文档处理（200K上下文）等方面表现突出，不过其知识库更新相对比较缓慢。

模型名称	发布时间	参数规模	主要特点
Claude	2023年3月14日	52B	Claude 能够执行各种基于文字对话的任务，如文档搜索、摘要、写作、编码，以及响应用户的提问。
Claude 2	2023年7月11日	未公开	使用2023年初的新数据训练模型。可达100K的上下文窗口为特色，免费使用。
Claude 3 系列	2024年3月4日	未公开	包含 Haiku（日本俳句）、Sonnet（英国莎士比亚十四行诗）、Opus（古罗马拉丁语作品）三个版本，宣称在推理、数学、编码、多语言理解和视觉方面全面超越 GPT-4，接近人类理解能力。
Claude 3.5 Sonnet	2024年10月22日	未公开	新增“使用计算机”功能，可模拟人类操作（如点击按钮、输入文本），支持多步骤自动化，但处于实验阶段，存在灵活性不足的问题。

# 国外大模型举例：Gemini家族



Google: Gemini 模型

推理能力强，多模态，快速反应，上下文有优势，图片编辑能力强

模型名称	发布时间	参数规模	主要特点
Gemini 1.0	2023年12月6日	未公开	<ul style="list-style-type: none"><li>原生多模态支持：首个原生支持文本、代码、图像、音频、视频的多模态模型，输入阶段即可综合推理。</li><li>三版本设计：包括Ultra（高复杂度任务）、Pro（通用场景）、Nano（移动端离线运行）。</li><li>性能突破：MMLU基准测试得分90%，首次超越人类专家，图像理解能力优于GPT-4。</li><li>应用场景：集成至Bard（Pro版）、Pixel 8 Pro（Nano版），支持Android开发者调用。</li><li>代码工具：发布AlphaCode 2，编程问题解决能力提升50%。</li></ul>
Gemini 1.5 Pro	2024年2月16日	未公开	<ul style="list-style-type: none"><li>上下文理解飞跃：上下文窗口从32K Token提升至100万Token（特定场景达1000万Token），可一次性处理402页航天记录或44分钟无声电影分析。</li><li>混合专家模型（MoE）：引入MoE架构，通过任务专业化网络提升计算效率，支持并行处理与层级结构优化，性能接近Ultra 1.0。</li><li>多模态推理增强：支持图像、文本、代码（如10万行代码库修改）的复杂逻辑推理，Python代码生成评估提升2-7%。</li></ul>
Gemini 1.5 Flash	2024年5月15日	未公开	专为高频任务优化，可分析1500页文档或3万行代码库，响应速度最快且部署成本更低。
Gemini 2.0	2024年12月12日	未公开	<ul style="list-style-type: none"><li>多模态增强：支持生成混合内容（图像+文本）、音频及视频，并原生调用谷歌搜索和代码执行。</li><li>速度与效率：运行速度达前代两倍，依托第六代TPU硬件（Trillium芯片）。</li><li>智能代理：推出Project Astra等AI代理，支持长对话记忆与多语言切换，自动化任务成功率提升。</li><li>研究支持：新增DeepResearch功能，可自动生成研究框架与报告。</li></ul>

# 国外大模型举例：Llama家族



## Meta: Llama 模型

比较有名的开源大模型，支持本地部署

模型名称	发布时间	参数规模	主要特点
Llama	2023年2月24日	7B、13B、32B、65B	Meta首个开源大模型系列，支持长文本生成，但因许可证限制不可商用。
Llama 2	2023年7月18日	7B、13B、70B	Llama2 预训练模型是在 2 万亿的 token 上训练的，微调Chat 模型是在100万人类标记数据上训练的。 <b>首个支持商用的开源大模型</b> ，通过微软Azure分发；性能接近GPT-3.5，支持多语言和复杂任务处理。
Llama 3	2024年4月18日	8B、70B	<ul style="list-style-type: none"><li>- 基于超过15T tokens训练（Llama 2的7倍），包含5%非英语数据覆盖30+语言</li><li>- 上下文窗口支持 8K 长文本</li><li>- 8B/70B版本在MMLU、代码生成等基准超越 Gemma 7B 和 Gemini 1.5 Pro</li><li>- 训练效率比 Llama 2 高 3 倍</li></ul>
Llama 3.1	2024年7月23日	8B、70B、405B	<ul style="list-style-type: none"><li>- 支持8种语言（英、德、法、意、葡、印、西、泰）；</li><li>- 上下文窗口扩展至128K tokens，支持长文本处理和多模态任务；</li><li>- 405B版本在数学推理、复杂任务处理上超越GPT-4和Claude 3.5</li><li>- 新增内置工具（如Wolfram Alpha数学推理），支持自定义JSON功能扩展</li></ul>
Llama 3.2	2024年9月25日	1B、3B、11B、90B	<ul style="list-style-type: none"><li>- 支持多模态（如90B Vision版具备图像理解能力）；</li><li>- 优化边缘计算和低延迟推理（3B和1B版适配移动设备）；</li><li>- 开源且可定制，支持本地化部署和工具集成（如RAG）；</li><li>- 面向企业级应用，涵盖视觉问答、图像标注等场景。</li></ul>

# 国内大模型举例：DeepSeek家族



DeepSeek 优点是性价比高、模型开源、在特定任务上可以媲美GPT等顶尖的闭源模型，但在最复杂的通用推理和创意写作上人有差距

模型名称	发布时间	参数规模	主要特点
DeepSeek-V1	2023年11月29日	67B	首个版本，2万亿token中英数据。支持128K长上下文窗口，擅长编程语言理解和生成，但多模态和复杂推理能力较弱。
DeepSeek-V2	2024年5月7日	236B	MoE 架构，8.1万亿token多语言数据。开源且训练成本仅为GPT-4 Turbo的1%，代码生成能力突出，但推理速度较慢。
DeepSeek-V2.5	2024年9月15日	236B	融合Chat和Coder模型， <b>数学推理能力提升</b> （MATH-500测试准确率82.8%）
DeepSeek-R1-Lite	2024年11月20日		R1-Lite 是预览版模型，支持登录网页版与模型对话，提供了深度思考模式。在数学、代码以及各种复杂逻辑推理任务上，取得了媲美 o1-preview 的推理效果，并为用户展现了 o1 没有公开的完整思考过程；仅支持网页使用，暂不支持 API 调用。
DeepSeek-V2.5-1210	2024年12月10日	236B	DeepSeek V2.5 的最终版微调模型 DeepSeek-V2.5-1210。通过 <b>Post-Training 迭代</b> 全面提升了模型各方面能力表现，包括数学、代码、写作、角色扮演等；同时，新版模型 <b>优化了文件上传功能</b> ，并且 <b>全新支持了联网搜索</b> ，展现出更加强大的全方位服务于各类工作生活场景的能力。
DeepSeek-V3	2024年12月26日	671B	MoE架构，在 14.8T token 上进行了预训练。 <b>生成速度达60 TPS</b> ，相比 V2.5 模型实现了 3 倍的提升。支持 <b>长文本、代码竞赛、数学任务</b> ， <b>多领域性能领先开源及闭源模型</b> 。API 定价也将调整为每百万输入

# 国内大模型举例：Qwen家族



## Alibaba: Qwen 模型 (通义千问模型)

通义千问是一款非常均衡的模型，尤其在中文场景下优化良好，其开源版本在国际社区也获得不错反响。

模型名称	发布时间	参数规模	主要特点
<a href="#">Qwen</a>	2023年4月7日	不详	首次发布，面向企业用户。
<a href="#">Qwen-7B</a>	2023年8月3日	7B	国内首个开源自研大模型，支持消费级显卡部署，覆盖通用对话场景，开启阿里开源生态布局。
<a href="#">Qwen-14B</a>	2023年9月25日	14B	一款支持多种语言的高性能开源模型，整体训练数据超过3万亿Token，使得模型具备更强大的推理、认知、规划和记忆能力。Qwen-14B最大支持8k的上下文窗口长度。
<a href="#">Qwen-1.8B/72B</a>	2023年11月30日	1.8B、72B	Qwen-72B 训练数据3万亿tokens，最高支持32K上下文长度。 Qwen-1.8B最高支持8K上下文输入，2.2万亿tokens数据集训练。模型最低int8/int4版本仅需2GB显存就可以推理。
<a href="#">Qwen1.5</a>	2024年2月6日	0.5B至72B	支持 32K 上下文长度；开放了 Base + Chat 模型的 checkpoint；与 Transformers 一起本地运行；同时发布了 GPTQ Int-4 / Int8、AWQ 和 GGUF 权重。
<a href="#">Qwen1.5-110B</a>	2024年4月26日	110B	国内第一个千亿规模的开源大模型，是一个Dense模型，而非MoE模型。从各项评测来看，Qwen1.5-110B足以与Llama3-70B相抗衡
<a href="#">Qwen2.0</a>	2024年6月7日	0.5B至72B	包含多尺寸模型，Qwen2-72B性能领先国内其他模型
<a href="#">Qwen2.5</a>	2024年5月24日	0.5B至72B	提升理解能力、逻辑推理、指令遵循和代码能力，支持29种语言

# 大模型性能评估

## 基本概念：基于相同的问题集给不同模型打分










- 由于LLM的架构不同、训练数据不同，导致不同LLM的性能在不同类型问题上差异比较大
- 为了客观比较不同LLM的性能，人们通常会收集不同类型的问题对不同LLM的回答进行评分

**MMLU** - 一种针对大模型的语言理解能力的测评，是目前最著名的大模型语义理解测评之一，任务涵盖的知识很广泛，语言是英文，用以评测大模型基本的知识覆盖范围和理解能力。

**C Eval** - C-Eval 是一个全面的中文基础模型评估套件。它包含了13948个多项选择题，涵盖了52个不同的学科和四个难度级别。用以评测大模型中文理解能力。

**AGI Eval** - 微软发布的大模型基础能力评测基准，主要评测大模型在人类认知和解决问题的一般能力，涵盖全球20种面向普通人类考生的官方、公共和高标准录取和资格考试，包含中英文数据。

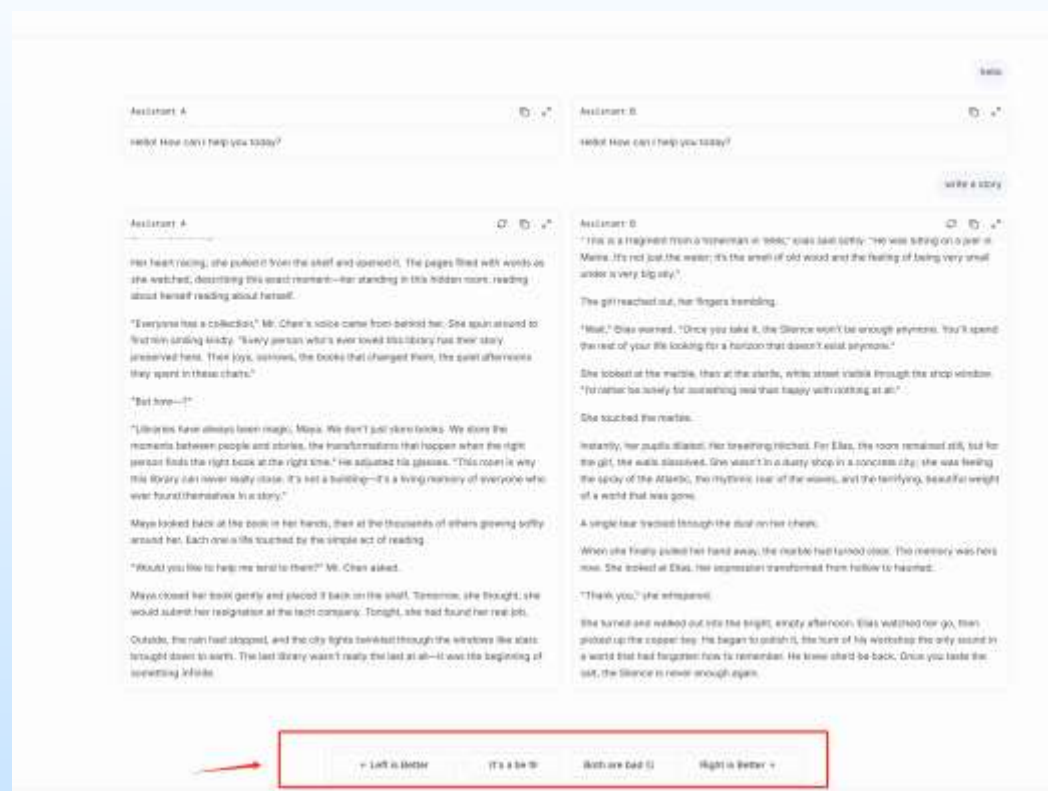
**GSM8K** - OpenAI发布的大模型数学推理能力评测基准，涵盖了8500个中学水平的高质量数学题数据集。数据集比之前的数学文字题数据集规模更大，语言更具多样性，题目也更具挑战性。

模型名称	参数大小	MMLU	CEval	AGIEval	GSM8K	MATH	BBH	MT Bench	发布者	开源情况
DeepSeek-R1	6710.0	90.8	91.8	/	/	97.3	/	/		
OpenAI o1		90.8	/	/	/	94.8	/	/		
Claude 3.5 Sonnet New	0.0	90.5	/	/	92.5	78.3	/	/		
Hunyuan-TurboS		89.5	/	/	/	89.7	92.2	/		
GPT-4o		88.7	/	/	90.5	76.6	/	/		

# 大模型性能评估：常见榜单介绍

## LMarena Leaderboard

- LMarena Leaderboard 是一个由伯克利大学团队发起的全球大型语言模型公开评测平台，采用类似国际象棋Elo系统的用户盲测机制实时排名。用真实人类的偏好取代人工设计的基准，让排行更动态、贴近用户真实感受，已成为业界公认的最权威“民意”LLM 排行榜之一。

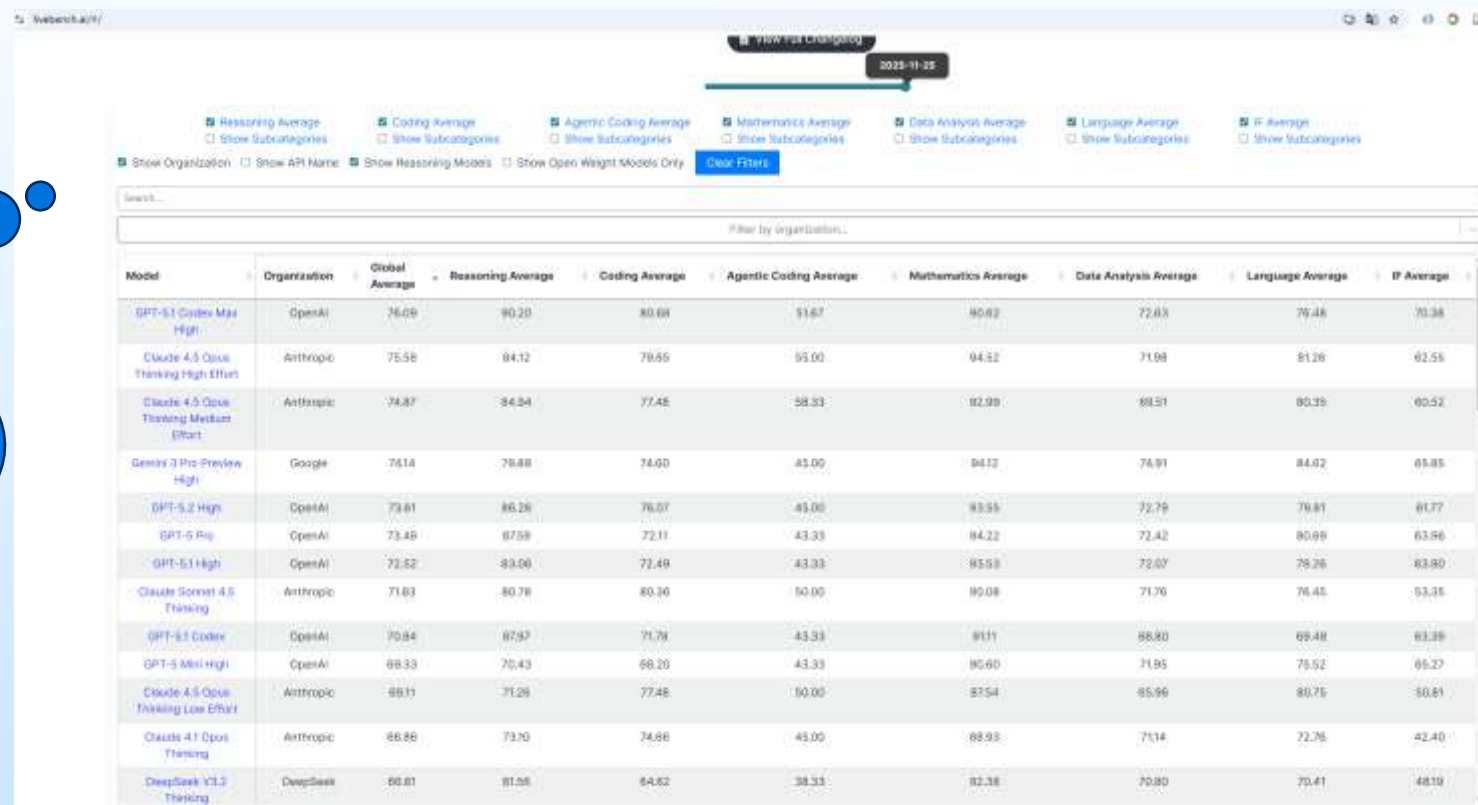


# 大模型性能评估：常见榜单介绍

## LiveBench Leaderboard

- LiveBench Leaderboard是2024年6月推出的一个专为解决“基准污染”（benchmark contamination）问题而设计的自动化、客观、每月更新的LLM评测榜单。LiveBench 是目前最能反映模型“真实智能水平”而非“记忆力”或“训练数据污染”的榜单，尤其适合判断一个模型在面对真正新问题时的泛化能力。

所有问题每月从最新来源生成：最近发布的数学竞赛、arXiv 论文、新闻事件、公开数据集等。问题发布后至少1个月才公开答案，确保新模型不可能在训练中见过这些题。



Model	Organization	Global Average	Reasoning Average	Coding Average	Agentic Coding Average	Mathematics Average	Data Analysis Average	Language Average	IF Average
GPT-5.1 Codex Max High	OpenAI	76.09	80.20	80.68	51.67	80.82	72.83	75.48	70.38
Claude 4.5 Opus Thinking High Effort	Anthropic	75.58	84.12	78.65	95.00	84.52	71.08	81.28	62.58
Claude 4.5 Opus Thinking Medium Effort	Anthropic	74.87	84.54	77.48	58.33	82.08	88.51	80.35	60.52
Gemini 3 Pro Preview High	Google	74.14	78.88	74.00	43.00	84.12	76.91	84.02	65.85
GPT-5.2 High	OpenAI	73.81	86.28	76.07	43.00	83.35	72.78	78.81	61.77
GPT-5 Pro	OpenAI	73.48	87.58	72.11	43.33	84.22	72.42	80.88	63.86
GPT-5.1 High	OpenAI	72.52	83.06	72.49	43.33	83.58	73.07	78.26	63.80
Claude Sonnet 4.5 Thinking	Anthropic	71.83	80.78	80.30	50.00	80.08	71.70	76.45	53.35
GPT-5.1 Codex	OpenAI	70.84	87.93	71.78	43.33	81.11	88.80	69.48	63.38
GPT-5 Mini High	OpenAI	68.33	75.43	68.20	43.33	80.60	71.95	75.52	65.27
Claude 4.5 Opus Thinking Low Effort	Anthropic	68.11	71.28	77.48	50.00	87.54	85.86	80.75	58.81
Claude 4.1 Opus Thinking	Anthropic	66.86	73.70	74.88	49.00	88.93	75.14	72.76	42.40
DeepSeek V3.2 Thinking	DeepSeek	66.81	81.58	84.82	38.33	82.38	70.80	70.41	48.19

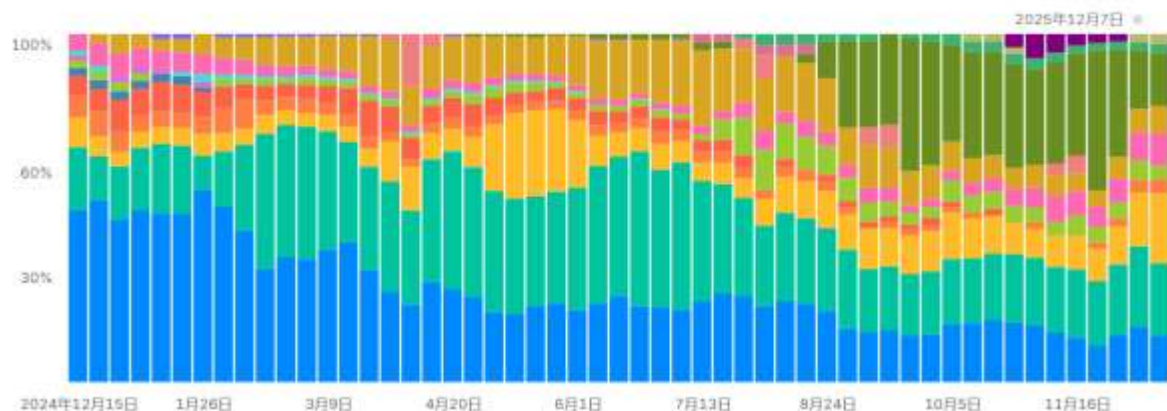
# 大模型性能评估：常见榜单介绍

## OpenRouter LLM Rankings

- OpenRouter Leaderboard 是基于真实开发者/应用通过OpenRouter API的实际使用量（主要是tokens消耗量或 requests数量）来排名

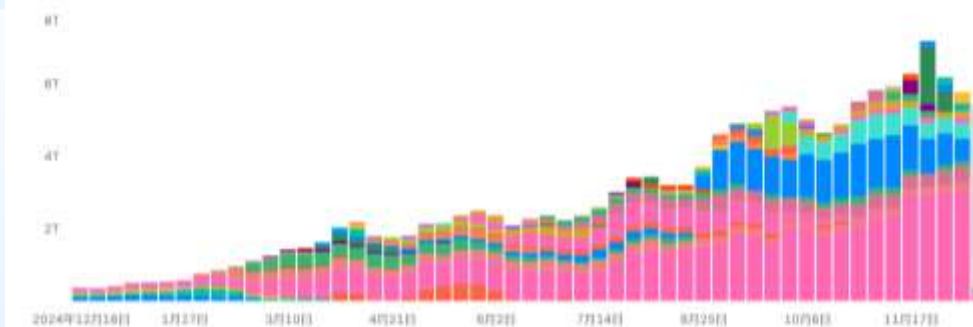
### Market Share

Compare OpenRouter token share by model author



### Leaderboard

Token usage across models on OpenRouter

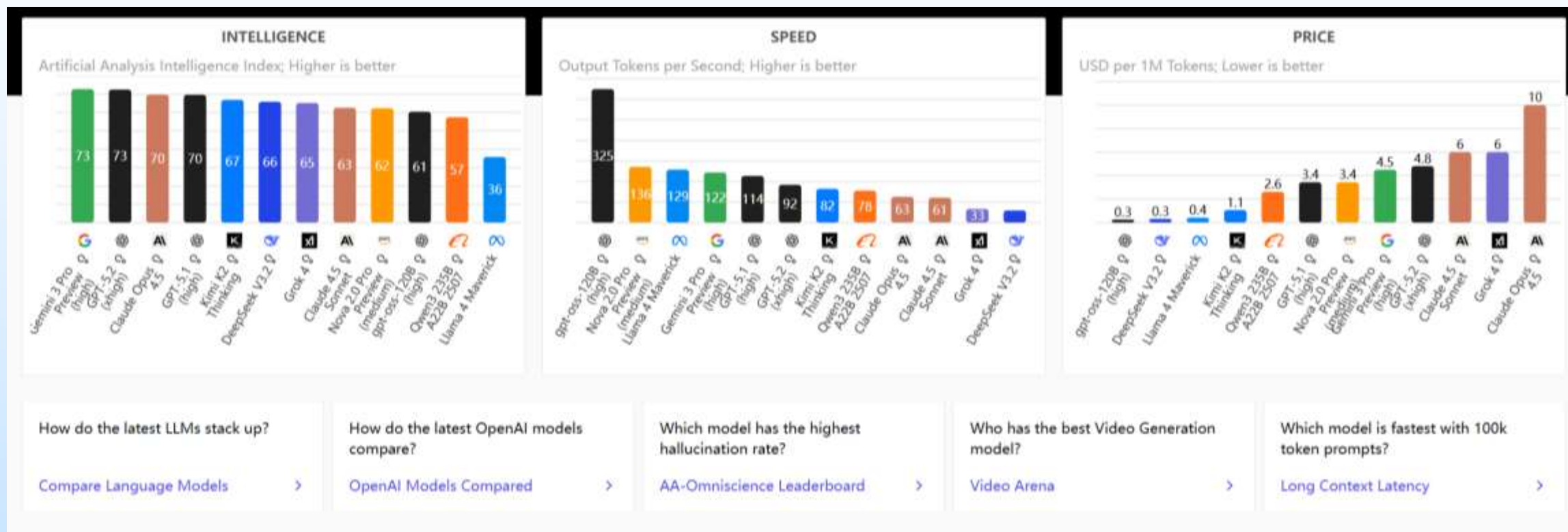


1.	Grok Code Fast 1 by x.ai	8180 tokens ↓30%	11.	Grok 4.1 Fast by x.ai	1330 tokens ↑8%
2.	Gemini 2.5 Flash by google	4540 tokens ↑14%	12.	DeepSeek V3.2 by deepseek	1270 tokens ↑71%
3.	Claude Sonnet 4.5 by anthropic	4318 tokens ↑0%	13.	MiniMax M2 by minimax	1258 tokens ↓13%
4.	gpt-oss-120b by openai	3200 tokens ↑120%	14.	DeepSeek V3 0324 by deepseek	1060 tokens ↓4%
5.	Claude Opus 4.5 by anthropic	2320 tokens ↓2%	15.	Claude Haiku 4.5 by anthropic	1018 tokens ↑11%
6.	Gemini 2.0 Flash by google	1880 tokens ↑14%	16.	GLM 4.6 by z.ai	98.50 tokens ↑0%
7.	Gemini 2.5 Flash Lite by google	1690 tokens ↑22%	17.	KAT-Coder-Pro V1 (free) by katcode	84.50 tokens ↑24%
8.	Grok 4 Fast by x.ai	1600 tokens ↓1%	18.	DeepSeek R1T2 Chimera (free) by tngtech	83.40 tokens ↑8%
9.	Gemini 2.5 Pro by google	1550 tokens ↓4%	19.	Claude Sonnet 4 by anthropic	82.00 tokens ↓11%
10.	Gemini 3 Pro Preview by google	1500 tokens ↓8%	20.	GPT-5 Mini by openai	80.50 tokens ↓2%

# 大模型性能评估：常见榜单介绍

## Artificial Analysis Leaderboard

- Artificial Analysis Leaderboard 由独立第三方团队 Artificial Analysis 维护，它不像 LMArena 那样靠人类投票，也不像 LiveBench 那样只测纯能力，而是把质量（Quality）、速度（Speed）、价格（Price）、延迟（Latency）、上下文长度 等所有开发者最关心的维度统一测评并可视化。



# 大模型性能评估：幻觉评估

图片比较了 17 款领先 AI 模型的两个关键指标：准确率（越高越好）与幻觉率（越低越好）。

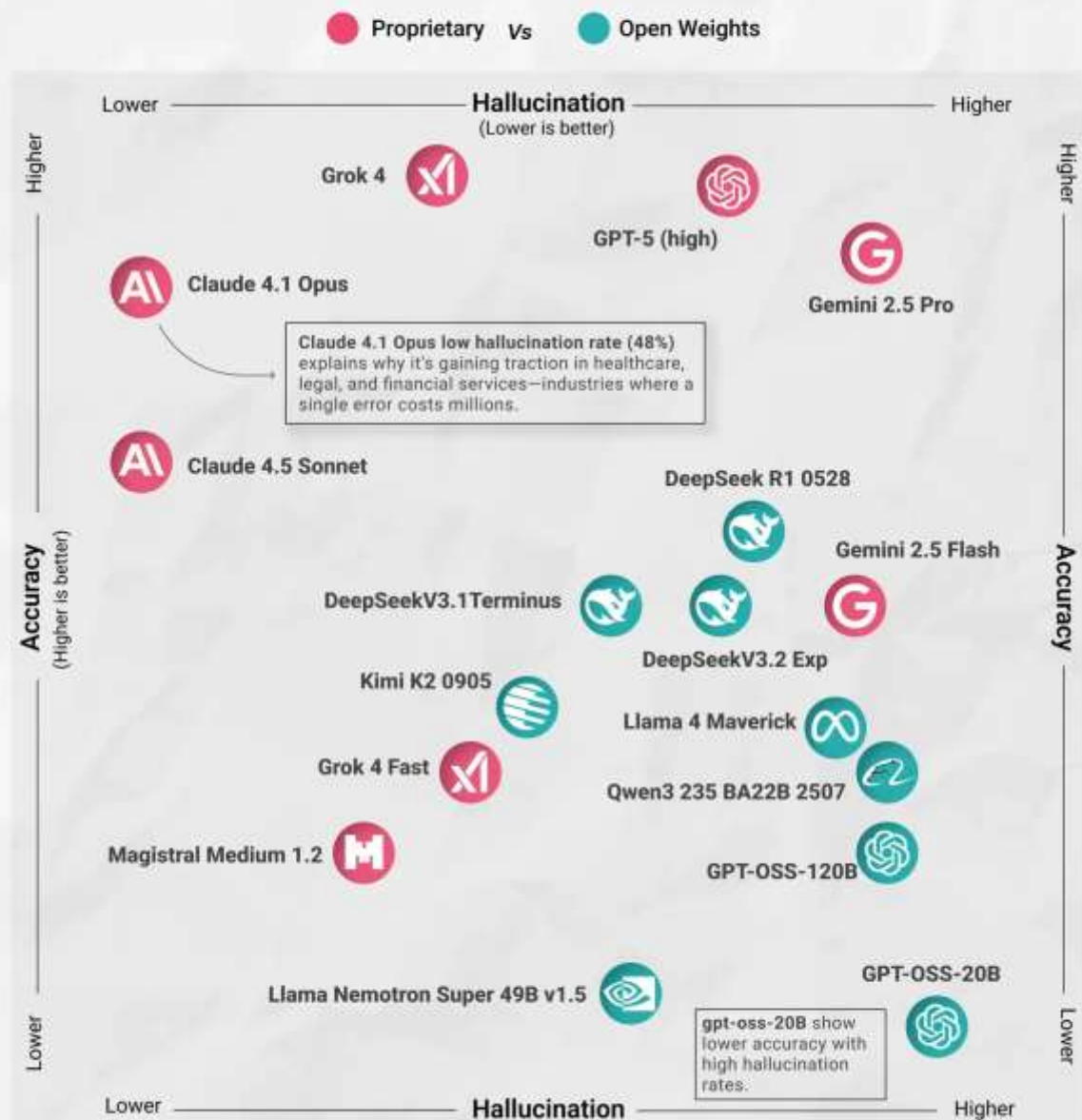
从象限图可以看到：

- Claude 4.1 Opus → 最稳、最可靠、最接近企业级理想模型
- Gemini 2.5 Pro / GPT-5 → 超强但不够稳
- DeepSeek / Llama 系列 → 开源阵营的实用黑马部分模型 → 准确与幻觉都不佳，风险较高

未来 AI 的竞争，不只是“谁更聪明”，而是“谁更不会胡说八道”。

## “Which AI Models Hallucinate the Most?”

A quadrant comparison of 17 leading AI models across accuracy and hallucination, revealing a critical reliability gap that is slowing enterprise AI adoption.

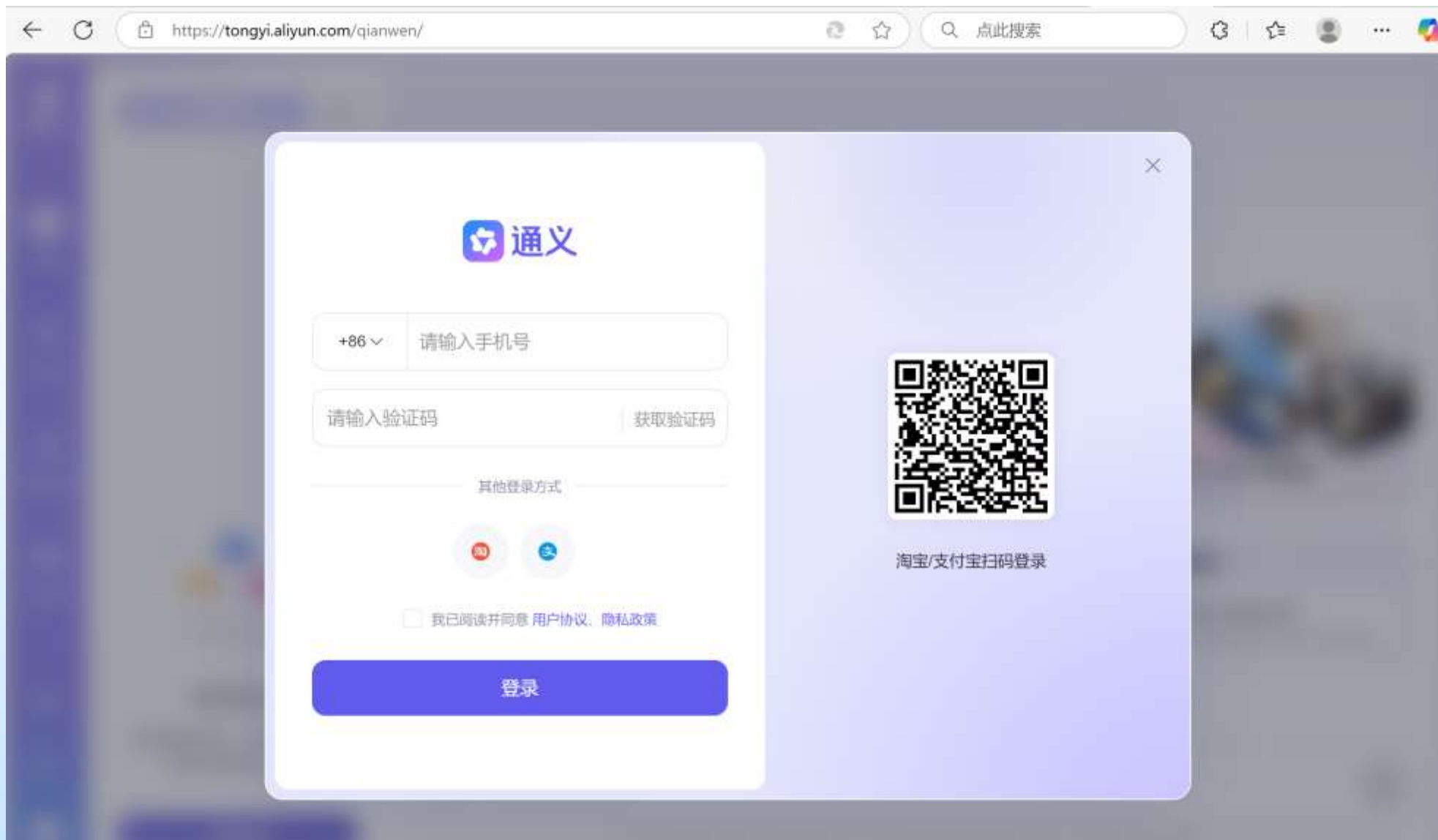


# 02

---

## 网页端问答工具与实操

# 通义千文（Qwen）网页端注册



# 网页端实操：医学数据数据分析代码生成

### PCA主成分分析示例代码

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler

# 加载并预处理数据集
iris = load_iris()
X = iris.data
y = iris.target
feature_names = iris.feature_names
target_names = iris.target_names

# 数据标准化
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 执行PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# 输出PCA信息
print("原始数据集形状:", X.shape)
print("PCA后数据集形状:", X_pca.shape)
print("前两个主成分占比:", pca.explained_variance_ratio_)
print("累计方差占比:", np.cumsum(pca.explained_variance_ratio_))
print("前两个主成分的问题:")
for i, comp in enumerate(pca.components_):
    print(f"主成分 {i+1}: {comp}")

# 可视化结果
plt.figure(figsize=(15, 5))

# 原始数据集可视化 (四个特征)
plt.subplot(1, 2, 1)
colors = ['sea', 'turquoise', 'darkorange']
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X[:, i], X[:, 1],
                color=color, alpha=0.5, label=target_name)
plt.xlabel(feature_names[0])
plt.ylabel(feature_names[1])
plt.title("原始数据 (四个特征)")
plt.legend()

# PCA降维后数据
plt.subplot(1, 2, 2)
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_pca[:, i], X_pca[:, 1],
                color=color, alpha=0.5, label=target_name)
plt.xlabel("主成分1")
plt.ylabel("主成分2")
plt.title("PCA降维后数据")
plt.legend()

# 显示主成分问题
plt.subplot(1, 2, 3)
# 打印问题
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
    plt.scatter(X_pca[:, i], X_pca[:, 1],
                color=color, alpha=0.5, label=target_name)
```

生成一个PCA主成分分析的示例代码

Qwen3-Max

```
PCA主成分分析示例代码

import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
```

这个示例代码展示了主成分分析(PCA)的完整流程:

- 生成示例数据集
- 数据标准化及处理
- 执行PCA降维
- 输出解释方差比和累计方差比
- 可视化原始数据与降维后数据对比
- 显示各主成分的方差贡献率
- 展示主成分载荷矩阵

代码包含了数据预处理、PCA分析、结果可视化和解释的完整流程,可以帮助理解PCA的工作原理和实际效果。

生成一个PCA主成分分析的示例代码

Qwen3-Max

```
PCA主成分分析示例代码

import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
```

【回答专业代码问题，一句话生成代码】

小代码 > 智能生成

# 网页端实操：医学论文总结和归纳

The screenshot shows a web browser window with the URL <https://www.mdpi.com/efficacy/doc/html/10.1109/UBH-2023-363844>. The page title is "NRAG\_A\_Knowledge-Enhanced\_LLM\_Framework". The main content area displays the abstract and a diagram of the NRAG framework. The diagram is divided into three steps: Step 1: Key Information Identification, Step 2: Knowledge Retrieval and Enhancement, and Step 3: Diagnosis Prediction. It shows the flow from EMR data to a structured dataset, then to a Knowledge Graph (KG) with various entities and relationships, and finally to an LLM (Large Language Model) for diagnosis prediction. The LLM is fine-tuned on a Neurosurgery QA dataset. The diagram also shows the integration of Open Source Medical KG, ERDs KG, and Neurosurgery Specialty KG into the main KG. Knowledge retrieval is shown as a path from symptoms to diseases or drugs. Knowledge enhancement is shown as a path from symptoms to symptoms or diseases. The LLM is shown as a central component that receives prompts and generates answers. The sidebar on the right contains AI-generated questions and answers in Chinese, discussing the framework's steps and its application in neurosurgery diagnosis.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License. For more information, see <https://creativecommons.org/licenses/by-nc-nd/4.0/>.

This article has been accepted for publication in IEEE Journal of Biomedical and Health Informatics. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI: 10.1109/UBH-2023-363844

TIAN et al.: NRAG

**Step 1: Key Information Identification**

Collects data from EMR (Electronic Medical Records) for a New patient. The data is standardized into a structured dataset (Symptoms, Diagnoses, Medication, etc.).

**Step 2: Knowledge Retrieval and Enhancement**

Retrieves key paths from a Knowledge Graph (KG). The KG includes Open Source Medical KG, ERDs KG, and Neurosurgery Specialty KG. Key relationships include: Relation 1: symptom->disease, Relation 2: symptom->symptom->disease, and Relation 3: symptom->drug->disease. Knowledge is enhanced through paths like symptom->symptom->disease and symptom->drug->disease.

**Step 3: Diagnosis Prediction**

The LLM (Large Language Model) is fine-tuned on the Neurosurgery QA dataset. It receives a prompt (e.g., "A 55-year-old male patient with a 2-week history of progressive right-sided weakness...") and generates an answer (e.g., "Diagnosis: Right-sided weakness, Hemiparesis...").

The LLM is supported by ChatGLM, Deepseek, ChatGPT, and LLaMA.

**Fig. 1. The framework of NRAG. It can be mainly divided into three steps. In step one, key information is extracted from EMR data and standardized. In step two, key paths are retrieved from the knowledge graph. In step three, LLM was fine-tuned on the Neurosurgery QA dataset for diagnosis prediction.**

screened all textual content, strictly removing any identifiable personal information. Through these preprocessing steps, this study constructed a high-quality structured dataset comprising 0.3B neurosurgical outpatient records. The entire workflow adhered to ethical guidelines [Ethics Approval No. KY2023-226-02], laying a robust foundation for subsequent research and data analysis.

**B. Framework**

In this section, this study introduces NRAG, a fusion framework that integrates LLMs and KGs through a planning

1) *Key information identification*: Let  $R$  represent the collected set of electronic health records containing multi-dimensional patient information, including symptoms, diagnoses, and medication histories. This study aims to construct a diagnosis model focused on discovering symptom-disease mapping relationships.

In  $R = \{S, D\}$ ,  $S = \{sym_1, sym_2, sym_3, \dots, sym_m\}$  represents the set of symptoms, where  $m$  is the number of symptoms.  $D = \{dis_1, dis_2, dis_3, \dots, dis_n\}$  represents the set of patient's diseases, where  $n$  is the number of diseases. Due to factors such as limited time for writing medical records and

基于KG的诊断模型有哪些限制?  
在医学记录中,如何解决非标准表达的问题?  
医疗领域的自然语言处理技术有哪些进展?

解读以下内容:

The framework of NRAG: It can be mainly divided into three steps. In step one, key information is extracted from EMR data and standardized.

NRAG是一种医学自然语言处理框架,主要用于神经外科诊断预测。它主要分为三个步骤。

第一步是从EMR数据中提取并标准化关键信息。这是为了确保模型能够理解和分析患者病历中的重要信息,并将其转换为可供机器学习的形式。

第二步是从知识图谱中检索出关键路径。知识图谱是一个包含大量实体、属性和关系的知识库,用于帮助模型更好地理解医学领域的概念和关系。通过检索关键路径,模型可以更好地理解不同症状之间的联系以及它们与疾病的关系。

第三步是在神经外科QA数据集上对LLM进行微调以进行诊断预测。这一步骤是为了让模型能够在实际应用中更准确地预测患者的病情。通过对大量的QA数据进行训练,模型可以学习到更多的医学知识和经验,并在实践中提供更好的诊断结果。

总之, NRAG框架可以帮助医生更好地利用EMR数据和知识图谱来辅助诊断,提高医疗质量和效率。

概述一下论文中的Fig1的内容

图1展示了NRAG框架的整体架构。该框架包括两个主要组件:知识增强模块(Knowledge Enhancement Module)和推理引擎模块(Reasoning Engine Module)。知识增强模块从EMR数据中提取并标准化关键信息,并从知识图谱中检索出关键路径。推理引擎模块则基于这些关键信息和路径来进行神经外科疾病的诊断预测。整个框架旨在提高医疗质量和效率,同时保证模型的可解释性和可靠性。

内容分析: 2/4

1. content may change prior to final publication. Citation information: DOI: 10.1109/UBH-2023-3638444 [a... NRAG\_A\_Knowled...  
2. p.p content may change prior to final publication. Citation information: DOI: 10.1109/UBH-2023-363844... NRAG\_A\_Knowled...

输入“”唤起帮助中心, Enter/发送, Shift+Enter/换行

# 03

---

## Python入门和基础语法

# 为什么是Python

2025年4月编程语言排行榜

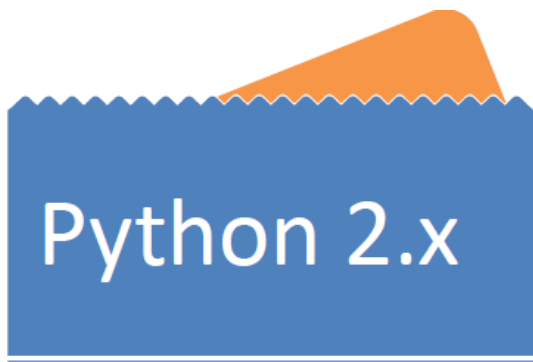
编程语言 TIOBE 4月

排名	编程语言	流行度	对比上月	年度明星
1	Python	23.08%	▼ -0.77%	2024, 2021
2	C++	10.33%	▼ -0.75%	2022, 2003
3	C	9.94%	▲ 0.41%	2019, 2017
4	Java	9.63%	▼ -0.73%	2015, 2005
5	C#	4.39%	▼ -0.48%	2023
6	JavaScript	3.71%	▲ 0.25%	2014
7	Go	3.02%	▲ 0.24%	2016, 2009
8	Visual Basic	2.94%	▲ 0.42%	-
9	Delphi/Object...	2.53%	▲ 0.38%	-
10	SQL	2.19%	▼ -0.38%	-

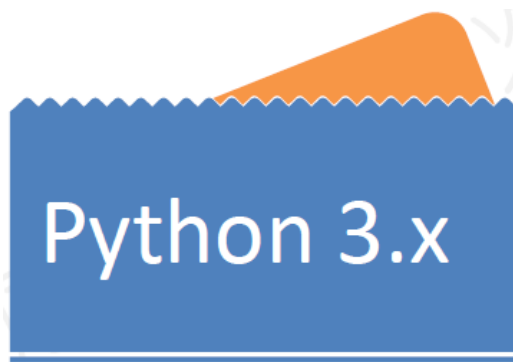


Python是目前最流行的编程语言

# Python的两个版本



两个版本互不兼容



我们将介绍现在主流的Python3.x

# 数据类型与变量：直观概念 + 例子

- Int: 整数 8, 9, 10, ...
- Float: 浮点数 0.1, 0.111, 0.11111
- String: 字符串 “hello”
- Variable: 变量  $a = 1$
- List: 数组
  - 一维数组:  $b = [1, 2, 3, 4, 5, 6]$
  - 二维数组:  $c = [[1, 1], [2, 2], [3, 3], [4, 5]]$
- 从数组取数
  - $b[0] = 1$ ,  $b[0:2] = [1, 2]$ ,  $c[0][1] = 1$



# 数据类型与变量：直观概念 + 例子

- 关键字是Python语言的关键组成部分，不可随意作为变量名

Python关键词

False	None	True	and	as	assert	break	class	continue
def	del	elif	else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal	not	or	pass
raise	return	try	while	with	yield			

```
# 整数
age = 25
# 浮点数
price = 19.99
# 字符串
name = "Alice"
# 布尔值
is_student = True
```



```
True = "hello"
if = "myname"
```



# 运算符：直观概念 + 例子

## 算数运算符

乘方: \*\*  
正负号: +/-  
乘除: \*/  
整除: //  
取余: %  
加减: +/-

```
a = 10
b = 3
print(a + b) # 13
print(a ** b) # 1000 (幂运算)
```

## 关系运算符

小于: <  
大于: >  
小于等于: <=  
大于等于: >=  
等于: ==  
不等于: !=

```
a = 10
b = 3
print(a == b) # False
print(a >= b) # True
```

## 逻辑运算符

非: not  
与: and  
或: or

```
print(True and False) # False
print(True or False) # True
```

# 分支条件判断

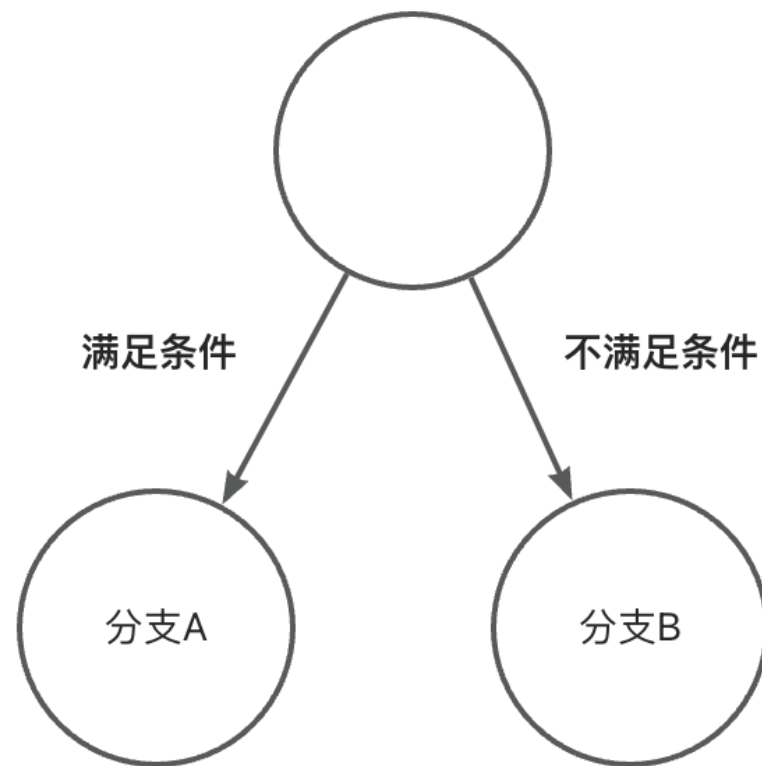
Python

```
# 示例：根据天气情况建议穿着

# 用户输入天气情况（晴天、雨天、雪天等）
weather = input("请输入今天的天气情况（例如：晴天、雨天、雪天）： ")

# 用户输入温度
temperature = int(input("请输入今天的温度（摄氏度）： "))

# 根据天气和温度给出穿衣建议
if weather == "雪天":
    print("今天有雪，请穿保暖的冬装，别忘了带手套和围巾。")
elif weather == "雨天":
    if temperature < 10:
        print("今天是下雨且气温较低，请穿防水外套并带上雨伞或雨衣，建议穿长裤和保暖内衣。")
    else:
        print("今天虽然下雨但气温适中，请穿轻便的防水夹克，并带上雨伞。")
elif weather == "晴天":
    if temperature > 30:
        print("今天天气炎热，请穿轻薄透气的夏装，记得涂防晒霜并戴帽子。")
    elif temperature > 20:
        print("今天天气宜人，短袖加薄外套是个不错的选择。")
    else:
        print("今天天气有点凉，请穿长袖衣物，可能还需要一件毛衣。")
else:
    print("对于今天的天气，建议您根据个人体感选择合适的衣物。")
```



# 循环：For/While

- 对于一个列表中的所有元素，执行内部的程序

Python

```
# 示例：使用for循环遍历列表并打印每个元素

# 定义一个包含多个项目的列表
fruits = ["苹果", "香蕉", "樱桃", "日期果", "柚子"]

# 使用for循环遍历列表中的每一个元素
for fruit in fruits:
    print(f"我喜欢吃 {fruit}。")

# 结束语句，表示循环结束后的操作
print("这就是我爱吃的水果列表！")
```

Python

```
# 初始化计数器
count = 1

# 使用 while 循环直到 count 超过 5
while count <= 5:
    print(f"当前计数: {count}")
    count += 1 # 每次循环后增加计数器

print("计数结束！")
```

# Python中的缩进

- 增加缩进表示语句块的开始
- Python用相同的缩进表示同级别语句块
- 减少缩进表示语句块的退出

```
# 根据天气和温度给出穿衣建议
if weather == "雪天":
    print("今天有雪, 请穿保暖的冬装, 别忘了带手套和围巾。")
elif weather == "雨天":
    if temperature < 10:
        print("今天是下雨且气温较低, 请穿防水外套并带上雨伞或雨衣, 建议穿长裤和保暖内衣。")
    else:
        print("今天虽然下雨但气温适中, 请穿轻便的防水夹克, 并带上雨伞。")
elif weather == "晴天":
    if temperature > 30:
        print("今天天气炎热, 请穿轻薄透气的夏装, 记得涂防晒霜并戴帽子。")
    elif temperature > 20:
        print("今天天气宜人, 短袖加薄外套是个不错的选择。")
    else:
        print("今天天气有点凉, 请穿长袖衣物, 可能还需要一件毛衣。")
else:
    print("对于今天的天气, 建议您根据个人体感选择合适的衣物。")
```

Python

```
# 初始化计数器
count = 1

# 使用 while 循环直到 count 超过 5
while count <= 5:
    print(f"当前计数: {count}")
    count += 1 # 每次循环后增加计数器

print("计数结束!")
```

# Python中的函数

- 函数类型
  - 内建函数
  - 标准库函数
  - 用户自定义函数
  - 第三方库
- 函数：
  - 完成一段特定功能的一段代码
    - 绝对值函数： `abs(x)`
    - 类型函数： `type(x)`
    - “四舍五入”函数： `round(x)`

Python

```
def add_numbers(a, b):  
    """  
    这个函数接收两个数字作为参数，并返回它们的和。  
  
    参数：  
    a (int, float): 第一个加数  
    b (int, float): 第二个加数  
  
    返回：  
    int, float: 两个参数的和  
    """  
    return a + b  
  
# 调用函数并打印结果  
result = add_numbers(5, 3)  
print(f"5 和 3 的和是: {result}")  
  
# 可以使用不同类型的数值进行测试  
float_result = add_numbers(2.5, 4.5)  
print(f"2.5 和 4.5 的和是: {float_result}")
```

# Python中的函数

- 内建函数

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	<u>input()</u>	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	<u>__import__()</u>
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

# Python中的函数

- 标准库函数
  - 需要先导入模块再使用函数 每个库有相关的一些函数
  - 如 math 库中的 sqrt 函数
- 第三方函数
  - 数量非常惊人 这也是 Python 重要的特征和优势
  - 例如OpenAI的第三方库函数就能让我们很轻易的调用大语言模型
- 用户自定义函数
  - 有固定的定义、调用和参数传递方式等
  - 当想要有一个特定的功能，但标准库函数，第三方函数却没办法提供时，可以自己写一个函数

# 04

调用大模型API的基本方法

# Python基础调用代码

```
# Please install OpenAI SDK first: `pip3 install openai`

from openai import OpenAI

client = OpenAI(api_key="<DeepSeek API Key>", base_url="https://api.deepseek.com")

response = client.chat.completions.create(
    model="deepseek-chat",
    messages=[
        {"role": "system", "content": "You are a helpful assistant"},
        {"role": "user", "content": "Hello"},
    ],
    stream=False
)

print(response.choices[0].message.content)
```

# 基本参数介绍

```
# Please install OpenAI SDK first: `pip3 install openai`
```

```
from openai import OpenAI
```

```
client = OpenAI(api_key="<DeepSeek API Key>", base_url="https://api.deepseek.com")
```

```
response = client.chat.completions.create(  
    model="deepseek-chat",  
    messages=[  
        {"role": "system", "content": "You are a helpful assistant"},  
        {"role": "user", "content": "Hello"},  
    ],  
    stream=False  
)  
  
print(response.choices[0].message.content)
```

- `api_key` : 这是你从某个渠道获取的API KEY, 用于验证你的身份。
- `base_url` : 这是API的基础URL, 通常由OpenAI或者某个渠道提供。

# 基本参数介绍

```
# Please install OpenAI SDK first: `pip3 install openai`
```

```
from openai import OpenAI
```

```
client = OpenAI(api_key="<DeepSeek API Key>", base_url="https://api.deepseek.com")
```

```
response = client.chat.completions.create(  
    model="deepseek-chat",  
    messages=[  
        {"role": "system", "content": "You are a helpful assistant"},  
        {"role": "user", "content": "Hello"},  
    ],  
    stream=False  
)  
  
print(response.choices[0].message.content)
```

- `messages` : 这是一个列表, `content` 是你的问题。 `user` 是你的角色。
- `model` : 指定使用的模型版本, 如 `gpt-3.5-turbo` 。
- `stream` : 如果设置为 `True` , 则响应将以流的形式返回。

# 申请大模型API的密钥（以DeepSeek API的密钥为例）



打开DeepSeek官网：[deepseek.com](https://www.deepseek.com/)，点击页面右上角的“API 开放平台”，并按照规定输入手机号或微信号登录。调用API需要付费，充值后才能使用。

<https://www.deepseek.com/>

# 申请大模型API的密钥（以DeepSeek API的密钥为例）



The screenshot shows the 'usage' page of the DeepSeek platform. The left sidebar contains navigation options: 'usage information', 'API keys' (highlighted with a red box and a red arrow), 'recharge', 'bill', 'API documentation', 'FAQ', 'pricing', 'integration', 'contact us', and 'personal information'. The main content area is titled 'usage information' and includes a promotional banner for off-peak discounts, a table of account balances, and a section for monthly usage.

platform.deepseek.com/usage

deepseek 开放平台

用量信息

用量信息

API keys

充值

账单

接口文档

常见问题

产品定价

实用集成

联系我们

个人信息

网页版免费对话 ↗

服务状态 · 用户协议 · 隐私政策 · 浙ICP备2023025841号

## 用量信息

【错峰优惠活动】北京时间每日 00:30-08:30 为错峰时段，API 调用价格大幅下调：DeepSeek-V3 降至原价的 50%，DeepSeek-R1 降至 25%，在该时段调用享受更经济更流畅的服务体验。【查看价格详情】 [知道了](#)

所有日期均按 UTC 时间显示，数据可能有 5 分钟延迟。

充值余额	赠送余额 <a href="#">查看有效期</a>	本月消费
¥50.00 CNY	¥0.00 CNY	¥0.00 CNY

[去充值](#) [余额预警未开启 \(去设置\)](#)

### 每月用量

消费金额 ¥0.00

2025 - 3月 [导出](#)

¥1

¥0

3-1

3-31

点击图中的“API keys”

# 申请大模型API的密钥（以DeepSeek API的密钥为例）

The image shows the DeepSeek API key management interface. On the left sidebar, the 'API keys' option is highlighted with a red box and a large red number '1'. In the main content area, the 'API keys' section is titled 'API keys' and contains a table with columns for '名称', 'Key', '创建日期', and '最新使用日期'. Below the table, there is a message: '暂无 API key, 你可以 创建 API key'. A red number '2' points to a '创建 API key' button. A modal dialog titled '创建 API key' is open, showing a '名称' field with 'My Key' entered, highlighted with a red box and a red number '3'. A red number '4' points to the '创建' button in the modal. The '取消' button is also visible.

deepseek 开放平台

用量信息

API keys

充值

账单

接口文档

常见问题

产品定价

实用集成

联系我们

个人信息

网页版免费对话

API keys

列表内是你的全部 API key, API key 仅在创建时可见可复制, 请妥善保管。不要与他人共享你的 API key, 或将其暴露在浏览器或其他客户端代码中。为了保护你的帐户安全, 我们可能会自动禁用我们发现已公开泄露的 API key。我们未对 2024 年 4 月 25 日前创建的 API key 的使用情况进行追踪。

名称	Key	创建日期	最新使用日期
暂无 API key, 你可以 创建 API key			

创建 API key

名称

My Key

取消 创建

在新页面中点击“创建API key”，然后为API命名，之后确认

# 申请大模型API的密钥（以DeepSeek API的密钥为例）



deepseek 开放平台

用量信息  
API keys  
充值  
账单

接口文档  
常见问题  
产品定价  
实用集成  
联系我们  
个人信息  
网页版免费对话

## API keys

列表内是你的全部 API key，API key 仅在创建时可见可复制，请妥善保存。不要与他人共享你的 API key，或将其暴露在浏览器或其他客户端代码中。为了保护你的帐户安全，我们可能会自动禁用我们发现已公开泄露的 API key。我们未对 2024 年 4 月 25 日前创建的 API key 的使用情况进行追踪。

名称	Key	创建日期	最新使用日期
My Key	sk-0f8b2*****b5ed	2025-03-06	-

创建 API key

**创建 API key**

请将此 API key 保存在安全且易于访问的地方。出于安全原因，你将无法通过 API keys 管理界面再次查看它。如果你丢失了这个 key，将需要重新创建。

sk-0f8b2a7fbf2d46ec8acd60b4c2ccb5ed

关闭 **复制**

点击前面的“创建”后，将会立即弹出新生成的API key，此时切记点击弹出窗右下角的“复制”，并将复制结果粘贴、保存在一个不容易忘记的地方，请注意复制的结果一定要保存好，因为它只能完整显示这一次，丢了就再也找不到了

谢谢大家

