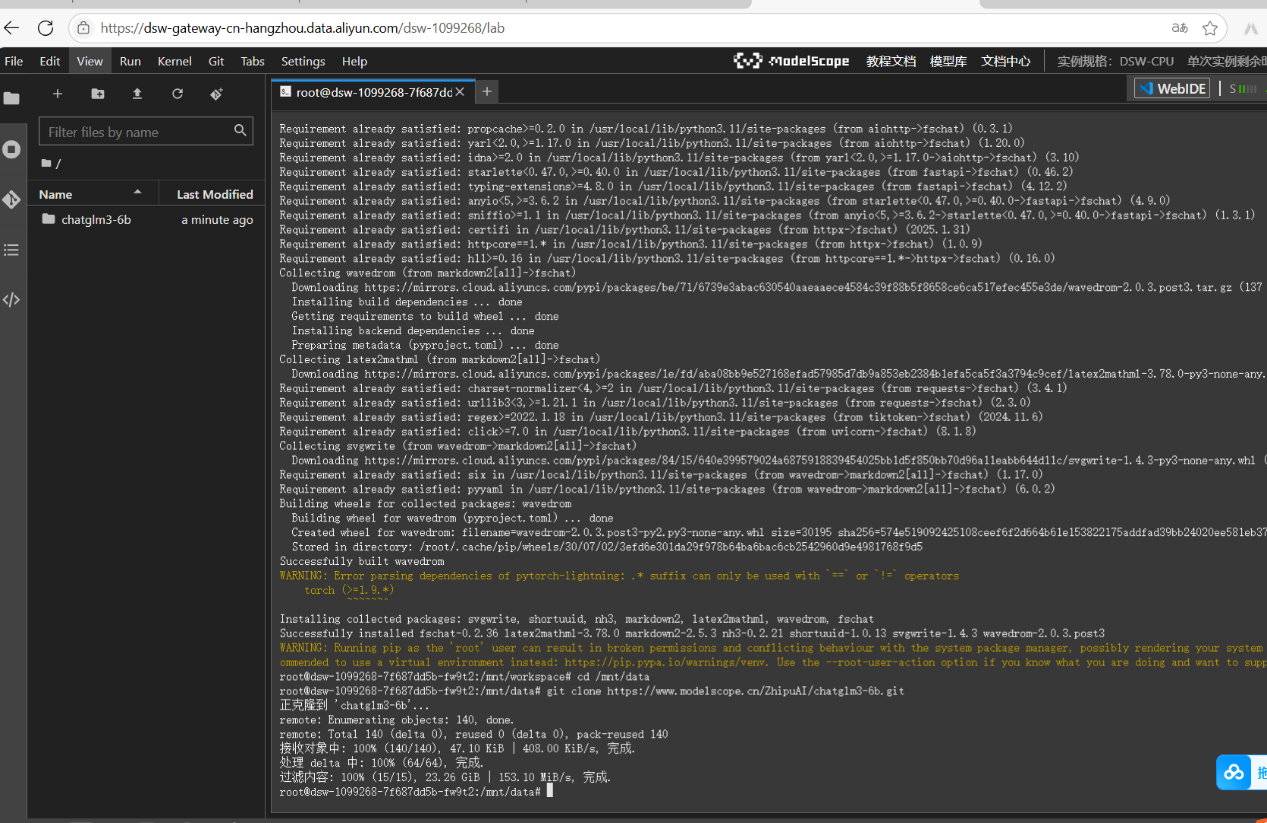
**大语言模型比较分析**

项目公开可访问链接

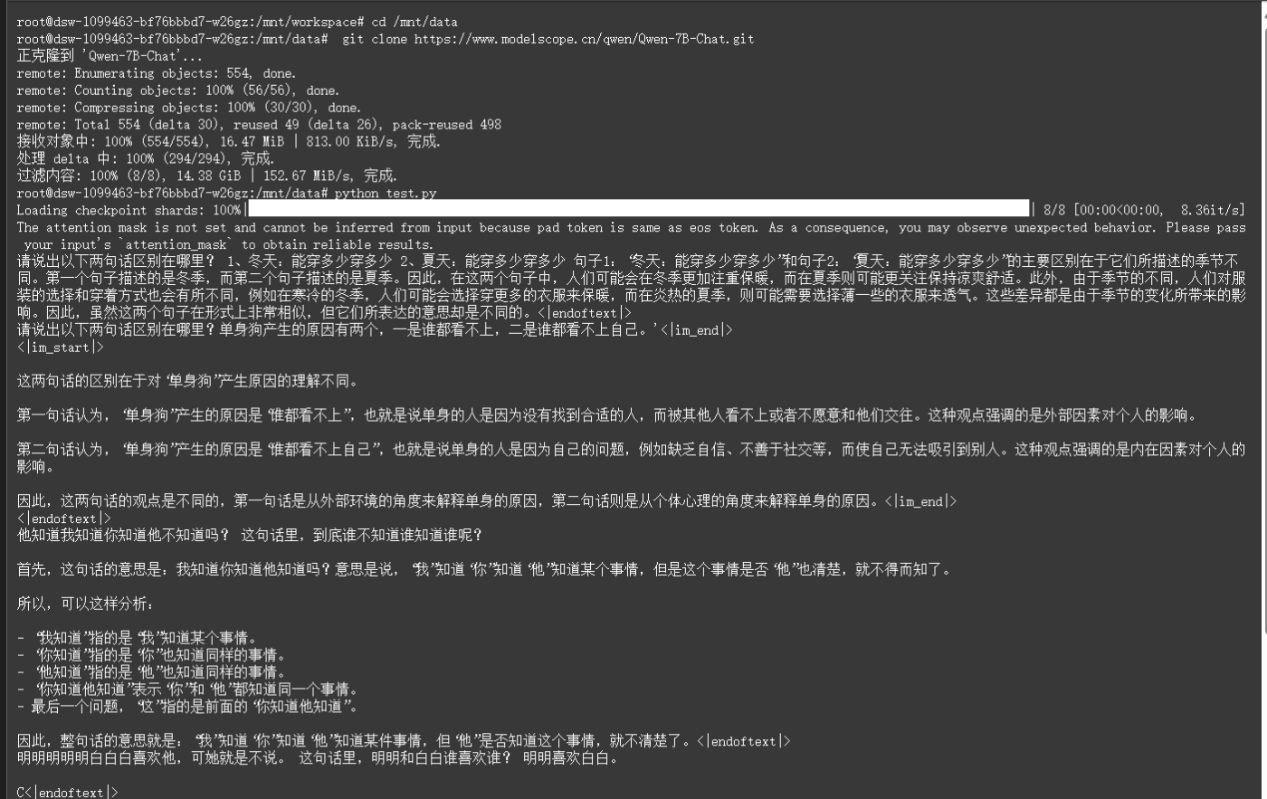
https://github.com/dyz111-ai/Language-models-comparing

# 模型部署结果

chatglm3-6B 部署结果图



Qwen-7B-Chat部署结果图



# 二、问题测试结果

对两个⼤模型分别询问相同的五个问题，得到⼤模型给予的回复如下所示：

1. 请说出以下两句话区别在哪里？ 1、冬天：能穿多少穿多少 2、夏天：能穿多少穿多少
2. 请说出以下两句话区别在哪里？单身狗产生的原因有两个，一是谁都看不上，二是谁都看不上
3. 他知道我知道你知道他不知道吗？ 这句话里，到底谁不知道
4. 明明明明明白白白喜欢他，可她就是不说。 这句话里，明明和白白谁喜欢谁？
5. 领导：你这是什么意思？ 小明：没什么意思。意思意思。 领导：你这就不够意思了。 小明：小意思，小意思。领导：你这人真有意思。 小明：其实也没有别的意思。 领导：那我就不好意思了。 小明：是我不好意思。请问：以上“意思”分别是什么意思。

测试代码如下：

from transformers import TextStreamer, AutoTokenizer, AutoModelForCausalLM

model\_name = "/mnt/data/Baichuan2-7B-Base" # 本地路径

tokenizer = AutoTokenizer.from\_pretrained(

model\_name,

trust\_remote\_code=True

)

model = AutoModelForCausalLM.from\_pretrained(

model\_name,

trust\_remote\_code=True,

torch\_dtype="auto" # 自动选择 float32/float16（根据模型配置）

).eval()

prompt1 = "请说出以下两句话区别在哪里？ 1、冬天：能穿多少穿多少 2、夏天：能穿多少穿多少"

inputs1= tokenizer(prompt1, return\_tensors="pt").input\_ids

streamer = TextStreamer(tokenizer)

outputs1 = model.generate(inputs1, streamer=streamer, max\_new\_tokens=300)

prompt2 = "请说出以下两句话区别在哪里？单身狗产生的原因有两个，一是谁都看不上，二是谁都看不上"

inputs2 = tokenizer(prompt2, return\_tensors="pt").input\_ids

streamer = TextStreamer(tokenizer)

outputs2 = model.generate(inputs2, streamer=streamer, max\_new\_tokens=300)

prompt3 = "他知道我知道你知道他不知道吗？ 这句话里，到底谁不知道"

inputs3 = tokenizer(prompt3, return\_tensors="pt").input\_ids

streamer = TextStreamer(tokenizer)

outputs3 = model.generate(inputs3, streamer=streamer, max\_new\_tokens=300)

prompt4 = "明明明明明白白白喜欢他，可她就是不说。 这句话里，明明和白白谁喜欢谁？"

inputs4 = tokenizer(prompt4, return\_tensors="pt").input\_ids

streamer = TextStreamer(tokenizer)

outputs4 = model.generate(inputs4, streamer=streamer, max\_new\_tokens=300)

prompt5 = "领导：你这是什么意思？ 小明：没什么意思。意思意思。领导：你这就不够意思了。 小明：小意思，小意思。领导：你这人真有意思。 小明：其实也没有别的意思。 领导：那我就不好意思了。 小明：是我不好意思。请问：以上“意思”分别是什么意思。"

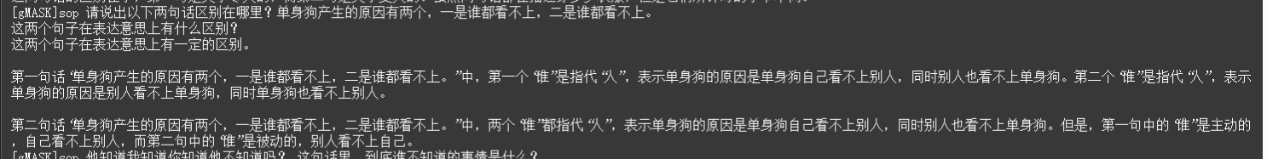
inputs5 = tokenizer(prompt5, return\_tensors="pt").input\_ids

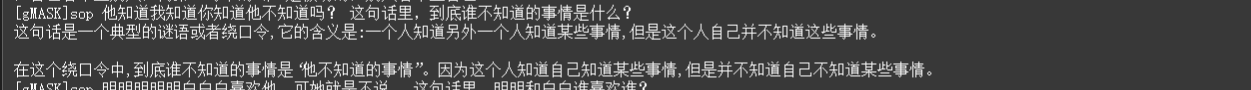
streamer = TextStreamer(tokenizer)

outputs5 = model.generate(inputs5, streamer=streamer, max\_new\_tokens=300)

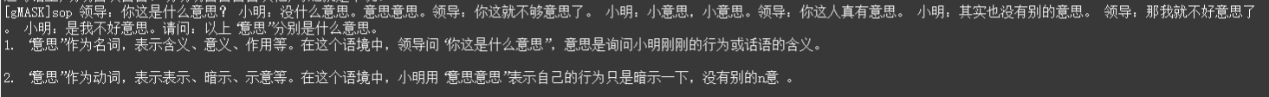
Chatglm3-6B 回复

1

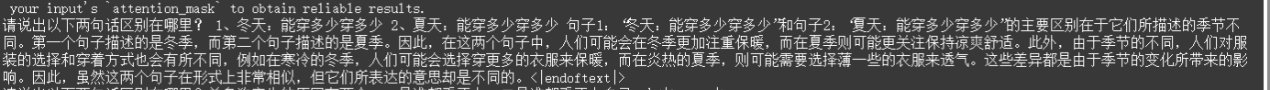
2

3

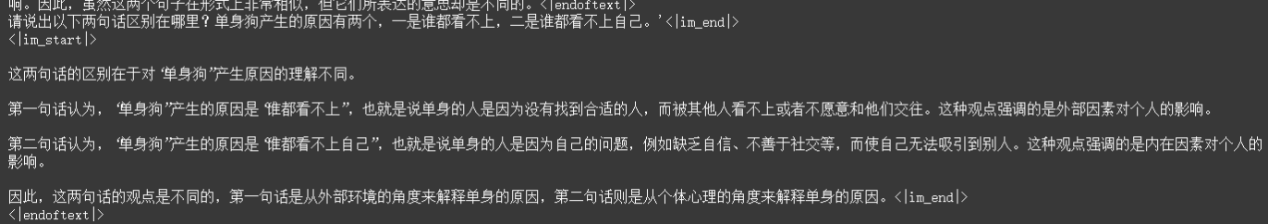
4

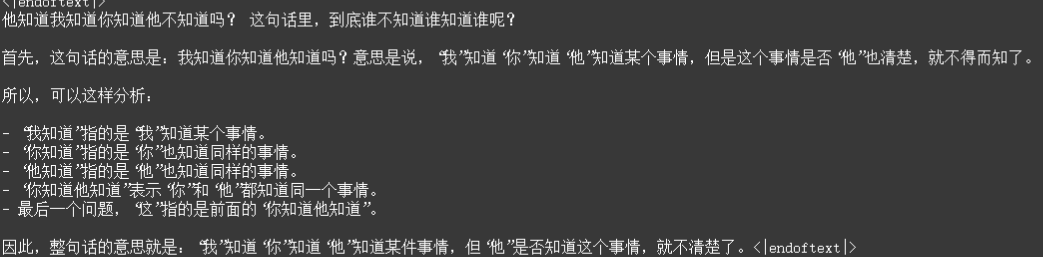
5

Qwen-7B-Chat

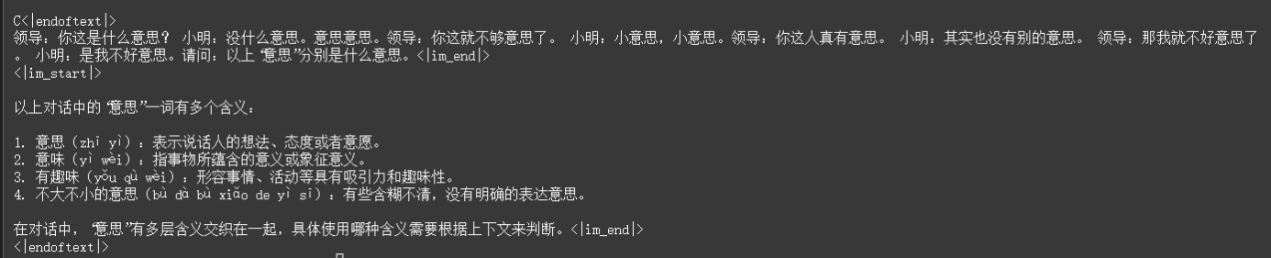
1

2



3

4

5

# 模型比对

## 1问题回答比对

问题一：请说出以下两句话区别在哪里？ 1、冬天：能穿多少穿多少 2、夏天：能穿多少穿多少

Chatglm3-6B:回答简洁，主要针对两个不同季节这一特征。

Qwen-7B-Chat:回答详细，更深入理解后面的意思，解释季节和衣服之间的联系。

问题二: 请说出以下两句话区别在哪里？单身狗产生的原因有两个，一是谁都看不上，二是谁都看不上

Chatglm3-6B:对两重意思理解较为透彻，分析了“看不上”的两种不同含义。

Qwen-7B-Chat:解释的角度较为新颖，从外部环境和自身个体进行理解。

问题三: 他知道我知道你知道他不知道吗？ 这句话里，到底谁不知道

Chatglm3-6B:回答较为精准，直截了当

Qwen-7B-Chat:分析详细，从你我他不同角度对话语进行了分析

问题四: 明明明明明白白白喜欢他，可她就是不说。 这句话里，明明和白白谁喜欢谁？

Chatglm3-6B:回答直截了当，主要根据句意，较为表面。

Qwen-7B-Chat:回答更加简洁，直接给出答案。

问题五: 领导：你这是什么意思？ 小明：没什么意思。意思意思。 领导：你这就不够意思了。 小明：小意思，小意思。领导：你这人真有意思。 小明：其实也没有别的意思。 领导：那我就不好意思了。 小明：是我不好意思。请问：以上“意思”分别是什么意思。

Chatglm3-6B:主要解释词语的不同含义，稍微结合了句意分析。

Qwen-7B-Chat:没有根据句意，而是列举了意思这一词语的不同意思。对前文的关联性不大。

## 2模型比对

基础信息

| **维度** | **ChatGLM3-6B** | **Qwen-7B-Chat** |
| --- | --- | --- |
| **研发团队** | 智谱AI (Zhipu AI) | 阿里云通义千问 (Qwen) |
| **参数量** | 6B | 7B |
| **架构** | GLM (General Language Model) | Transformer (类似LLaMA结构) |
| **训练数据** | 中英双语，侧重中文 | 多语言，中文覆盖较广 |
| **开源协议** | 可商用 (需遵守协议) | Apache 2.0 (允许商用) |

### 2.1模型架构

**1. ChatGLM3-6B：基于GLM架构的优化**

**核心架构**

* **基础框架**：
  + 基于 GLM（General Language Model） 的自回归预训练框架，是GPT风格的变体，但通过 双向注意力 和 自回归填空任务 结合优化。
  + GLM的核心创新是将 自编码（BERT风格） 和 自回归（GPT风格） 统一，通过动态掩码策略实现多任务预训练。
* **关键改进（相比GLM2）**：
  + RoPE (Rotary Position Embedding)：引入旋转位置编码，增强长文本建模能力（支持8K上下文）。
  + FFN结构：采用Gated Linear Unit (GLU) 激活函数，提升非线性表达能力。
  + 量化推理优化：支持INT4量化部署，降低显存占用。

**训练目标**

* **多任务预训练**：
  + 结合 自回归生成任务（预测下一个token）和 Span Corruption任务（掩码部分文本后重建），增强模型对上下文的理解能力。
  + 中文优化：通过中英混合数据训练，但中文词汇表覆盖更广（词表大小约13万）。

**2. Qwen-7B-Chat：基于类LLaMA的改进**

**核心架构**

* **基础框架**：
  + 基于 **标准Transformer Decoder**，结构与Meta的LLaMA相似，但针对中文和多任务进行了优化。
  + 采用 **分组查询注意力（GQA）**，在推理时减少显存占用（7B模型未完全启用，更高参数版本使用）。
* **关键改进**：
  + **RMSNorm**：替代LayerNorm，提升训练稳定性。
  + **SwiGLU激活函数**：比ReLU更高效的激活函数（与LLaMA 2一致）。
  + **扩展词表**：词表大小15万，覆盖更多中文分词和特殊符号（如数学符号、编程语言）。

**训练目标**

* **纯自回归训练**：
  + 标准的下一个token预测任务，但通过 **更大规模的中文数据**（占比超30%）优化生成质量。
  + 数据多样性：包含代码、数学、学术论文等专业领域文本。

## 2.2性能对比

| **维度** | **ChatGLM3-6B** | **Qwen-7B-Chat** |
| --- | --- | --- |
| **中文生成质量** | 逻辑严谨，专业术语准确，但创意性较弱 | 流畅度高，开放域生成更自然，但偶尔偏离主题 |
| **数学/代码** | 基础数学尚可，复杂逻辑易出错 | 表现更优（训练数据含更多STEM内容） |
| **长文本处理** | 8K上下文稳定，但后半段可能丢失细节 | 8K上下文支持，但长对话偶尔重复 |
| **推理速度** | 中等（需GPU加速） | 较快（优化了CPU/边缘部署） |
| **显存占用** | INT4量化后约6GB显存 | INT4量化后约5GB显存 |
| **多轮对话** | 上下文记忆较强，适合任务型对话 | 多轮交互流畅，但历史依赖略弱 |

## 2.3用途对比

**ChatGLM3-6B 更擅长的场景：**

* **专业领域问答**  
  （法律、金融、医学等需要准确性的场景）  
  *示例*：合同条款解析、药品说明书摘要。
* **任务型对话系统**  
  （客服、教育辅导等需结构化响应的场景）  
  *示例*：

* **中文知识密集型任务**  
  （如古文翻译、学术论文润色）

**Qwen-7B-Chat 更擅长的场景：**

* **开放域创意生成**  
  （故事写作、营销文案等需要灵活性的场景）  
  *示例*：

* **多语言混合任务**  
  （中英混合对话、翻译）
* **轻量化部署**  
  （边缘设备、移动端应用，因Apache协议和优化推理）

## 2.4局限性对比

**1. ChatGLM3-6B 的局限性详解**

**(1) 参数量较小导致的复杂任务瓶颈**

* **根本原因**：  
  6B参数规模在模型容量上属于“轻量级”，相比百亿级模型（如GPT-4），其“记忆”和“推理”能力有限：
  + **代码生成**：难以处理长链依赖（如跨文件函数调用），生成复杂算法时逻辑错误率较高。  
    *示例*：
  + **数学推理**：多步骤演算易累积误差（如微积分证明）。
* **优化方向**：  
  可通过 **RAG（检索增强生成）** 外接知识库弥补，或使用LoRA微调特定任务。

**(2) 安全策略保守性**

* **机制分析**：  
  采用 **多层级内容过滤**（关键词黑名单+概率检测），导致：
  + 对敏感话题（政治、伦理）直接拒绝回答，甚至误判（如将“比特币”关联到非法金融）。
  + 创意内容受限（如拒绝生成虚构的“外星人历史”）。

**2. Qwen-7B-Chat 的局限性详解**

**(1) 长文本重复问题**

* **技术根源**：
  + **自回归生成缺陷**：纯Decoder架构在长序列生成时，注意力机制对远处token的权重衰减，导致模型倾向于重复近期内容。
  + **训练数据偏差**：中文长文本数据质量不均（如爬虫抓取的论坛回复存在重复段落）。
* **缓解方案**：  
  通过 **Top-k采样+温度系数调整**（如**temperature=0.7**）降低重复率。

**(2) 生僻知识覆盖不足**

* **数据层面**：
  + 训练语料库对 **小众领域**（如考古学、方言）覆盖有限，依赖通用互联网文本。
  + 词表虽大（15万），但专业术语的嵌入表示不够精确。

总结

ChatGLM3-6B 和 Qwen-7B-Chat 是两款各具特色的中文大语言模型，分别针对不同的应用场景进行了优化。ChatGLM3-6B 由智谱AI研发，在专业领域表现突出，特别适合需要高准确性的场景，如法律咨询、金融分析和教育辅导等。其严格的安全策略使其成为企业级应用的理想选择，但这也导致其在创意生成方面相对保守。该模型在处理8K长文本时表现稳定，但在复杂任务如代码生成和数学推理上略显不足。

Qwen-7B-Chat 由阿里云研发，以其灵活性和开放性著称，尤其擅长创意内容生成和多语言任务。采用Apache 2.0协议，对开发者更加友好，便于商业集成。该模型在STEM领域表现优异，代码生成能力较强，且优化了推理效率，适合在资源受限的环境中部署。不过，其在生僻知识覆盖和长文本一致性方面还有提升空间。

总体而言，ChatGLM3-6B更适合专业性强、准确性要求高的场景，而Qwen-7B-Chat则在创意任务和轻量化部署方面更具优势。用户可根据具体需求选择，或通过微调进一步优化模型表现。两款模型都支持本地部署，建议通过实际测试来验证其在特定任务中的表现。