

# 大语言模型(LLM)本地部署实验报告

项目公开可访问链接：

github: [https://github.com/Effulgence12/hw4\\_Model\\_Compare.git](https://github.com/Effulgence12/hw4_Model_Compare.git)

魔搭平台: [https://modelscope.cn/models/Kerlap/Models\\_Compare\\_chatglm\\_qwen/summary](https://modelscope.cn/models/Kerlap/Models_Compare_chatglm_qwen/summary)

## 1. 项目概述

本项目旨在探索大语言模型(LLM)的本地部署方法及其性能表现。通过在云服务器环境中部署开源大模型，我们可以深入了解 LLM 的运行机制、资源需求和实际应用效果。本次实验选取了两个主流的中文大语言模型：ChatGLM3-6B 和 Qwen-7B-Chat 进行对比测试，通过一系列多领域的问题来评估它们在理解能力、创造力、逻辑推理和专业知识等方面的表现差异。

### 1.1 实验目标

- 掌握大语言模型在 CPU 环境下的部署方法
- 实现两个主流开源大模型的本地化部署
- 通过对比测试分析不同模型的性能特点和适用场景
- 深入理解大语言模型的推理过程和资源消耗情况

### 1.2 技术栈

- Python 3.10
- PyTorch (CPU 版本)
- Transformers 库
- ModelScope 平台
- Conda 环境管理

## 2. 平台搭建

本项目使用 ModelScope 平台提供的云服务器环境进行大模型部署。以下是平台搭建的详细步骤：

- 访问 ModelScope 官网 (<https://www.modelscope.cn/home>)，  
点击右上角完成新用户注册。
- 登录后进入首页，确认已绑定阿里云账号并获得免费云计算资源。
- 启动 CPU 服务器实例，用于后续的模型部署。

The screenshot shows the ModelScope personal homepage. On the left sidebar, there are sections for '概览' (Overview), '我创建的' (Created by me), '我喜欢的' (Favorites), '我的Notebook' (My Notebook) which is highlighted in purple, '模型服务' (Model Services) with a '功能上新' (New Feature) badge, '访问令牌' (Access Token), '组织信息' (Organization Information) with a '+', and '相关推荐' (Related Recommendations). The main content area is titled '我的Notebook' and mentions a collaboration with Alibaba Cloud. It features two tabs: '魔搭平台免费实例' (Free Instances on ModelScope Platform) and '个人云账号授权实例' (Authorized Instances on Personal Cloud Account). The '魔搭平台免费实例' tab is selected, showing the 'PAI-DSW' instance. A description states: 'PAI-DSW是为算法开发者量身打造的云端深度学习开发环境，内置JupyterLab、WebIDE及Terminal，无需任何运维配置即可编写、调试及运行Python代码。了解更多产品详情' (PAI-DSW is a cloud-based deep learning development environment specifically designed for algorithm developers. It includes JupyterLab, WebIDE, and Terminal, allowing users to write, debug, and run Python code without any operational configuration). Below this, there are two options: '方式一' (Method One) for a CPU environment and '方式二' (Method Two) for a GPU environment. Both options show configurations: 8核 32GB, pre-installed ModelScope Library, and pre-installed镜像 (Image). The '启动' (Start) button is located at the bottom.

4. 进入个人主页，查看免费实例（CPU 环境）。
5. 点击 Terminal 图标打开终端命令行环境，准备进行环境配置。

## 3. 环境搭建

### 3.1 conda 环境配置

由于大语言模型对环境依赖较为复杂，我们使用 Conda 创建独立的 Python 环境，以避免依赖冲突问题。

1. 手动安装 conda (若 CPU 版本未预装)：

```
cd /opt/conda/envs
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash Miniconda3-latest-Linux-x86_64.sh -b -p /opt/conda
echo 'export PATH="/opt/conda/bin:$PATH"' >> ~/.bashrc
source ~/.bashrc
conda --version
```

```
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# cd /opt/conda/envs
bash: cd: /opt/conda/envs: 没有那个文件或目录
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
--2025-06-08 16:36:41-- https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
正在解析主机 repo.anaconda.com (repo.anaconda.com)... 104.16.191.158, 104.16.32.241, 2606:4700::6810:20f1, ...
正在连接 repo.anaconda.com (repo.anaconda.com)|104.16.191.158|:443... 已连接。
已发出 HTTP 请求，正在等待回应... 200 OK
长度: 155472915 (148M) [application/octet-stream]
正在保存至: Miniconda3-latest-Linux-x86_64.sh

Miniconda3-latest-L 100%[=====] 148.27M 16.5MB/s 用时 11s

2025-06-08 16:36:53 (13.5 MB/s) - 已保存 'Miniconda3-latest-Linux-x86_64.sh' [155472915/155472915]

root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# bash Miniconda3-latest-Linux-x86_64.sh -b -p /opt/conda
PREFIX=/opt/conda
Unpacking payload...
_distutils.hack/_init_.py:53: UserWarning: Reliance on distutils from stdlib is deprecated. Users must rely on setuptools to provide the distutils module. Avoid importing distutils or import setuptools first, and avoid setting SETUPTOOLS_USE_DISTUTILS=stdlib. Register concerns at https://github.com/pypa/setuptools/issues/new?template=distutils-deprecation.yml
entry_point.py:256: DeprecationWarning: Python 3.14 will, by default, filter extracted tar archives and reject files or modify their metadata. Use the filter argument to control this behavior.
_distutils.hack/_init_.py:53: UserWarning: Reliance on distutils from stdlib is deprecated. Users must rely on setuptools to provide the distutils module. Avoid importing distutils or import setuptools first, and avoid setting SETUPTOOLS_USE_DISTUTILS=stdlib. Register concerns at https://github.com/pypa/setuptools/issues/new?template=distutils-deprecation.yml
entry_point.py:256: DeprecationWarning: Python 3.14 will, by default, filter extracted tar archives and reject files or modify their metadata. Use the filter argument to control this behavior.
_distutils.hack/_init_.py:53: UserWarning: Reliance on distutils from stdlib is deprecated. Users must rely on setuptools to provide the distutils module. Avoid importing distutils or import setuptools first, and avoid setting SETUPTOOLS_USE_DISTUTILS=stdlib. Register concerns at https://github.com/pypa/setuptools/issues/new?template=distutils-deprecation.yml
entry_point.py:256: DeprecationWarning: Python 3.14 will, by default, filter extracted tar archives and reject files or modify their metadata. Use the filter argument to control this behavior.
installation finished.
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# echo 'export PATH="/opt/conda/bin:$PATH"' > ~/bashrc
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# source ~/bashrc
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# conda --version
conda 25.3.1
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace#
```

## 2. 创建并激活环境：

```
conda create -n qwen_env python=3.10 -y
source /opt/conda/etc/profile.d/conda.sh
conda activate qwen_env
```

```
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# conda create -n qwen_env python=3.10 -y
Channels:
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# source /opt/conda/etc/profile.d/conda.sh
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# conda activate qwen_env
(qwen_env) root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace#
```

## 3.2 基础依赖安装

为了支持大模型的加载和推理，需要安装一系列依赖包，包括 PyTorch、Transformers 等核心库。

### 1. 检查 pip 联网状态并更新基础工具：

```
pip install -U pip setuptools wheel
```

### 2. 安装 PyTorch CPU 版本：

由于我们在 CPU 环境下运行，选择 PyTorch 的 CPU 版本可以减少不必要的 CUDA 依赖，提高兼容性。

```
pip install \
torch==2.3.0+cpu \
torchvision==0.18.0+cpu \
--index-url https://download.pytorch.org/whl/cpu
```

```
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# source /opt/conda/etc/profile.d/conda.sh
root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# conda activate qwen_env
(qwen_env) root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# pip install \
> torch==2.3.0+cpu \
> torchvision==0.18.0+cpu \
> --index-url https://download.pytorch.org/whl/cpu
Looking in indexes: https://download.pytorch.org/whl/cpu
Collecting torch==2.3.0+cpu
  Downloading https://download.pytorch.org/whl/cpu/torch-2.3.0%2Bcpu-cp310-cp310-linux_x86_64.whl (190.4 MB)
    ━━━━━━━━━━━━━━━━ 190.4/190.4 MB 42.4 MB/s eta 0:00:00
Collecting torchvision==0.18.0+cpu
```

### 3. 安装兼容依赖包：

以下依赖包是运行大语言模型所必需的，包括模型加载、量化优化和推理加速等功能。

```
pip install \
"intel-extension-for-transformers==1.4.2" \
"neural-compressor==2.5" \
"transformers==4.33.3" \
"modelscope==1.9.5" \
"pydantic==1.10.13" \
"sentencepiece" \
"tiktoken" \
"einops" \
"transformers_stream_generator" \
"uvicorn" \
"fastapi" \
"yacs" \
"setuptools_scm"
```

安装 fschat (需启用 PEP517 构建) :

```
pip install fschat --use-pep517
```

可选增强工具：

```
pip install tqdm huggingface-hub
```

# 4. 大模型实践操作

## 4.1 下载大模型到本地

本实验选择了两个主流的开源中文大语言模型：ChatGLM3-6B 和 Qwen-7B-Chat。这两个模型都具有较好的中文理解能力，且在 CPU 环境下可以勉强运行。

### 1. 切换至数据目录：

```
cd /mnt/data
```

### 2. 下载模型（分别下载两个模型）：

```
# Qwen-7B-Chat  
git clone https://www.modelscope.cn/qwen/Qwen-7B-Chat.git  
# ChatGLM3-6B  
git clone https://www.modelscope.cn/ZhipuAI/chatglm3-6b.git
```

完成 git clone 相关 git 的截图或部署完成的相关截图：

```
(qwen_env) root@dsw-1132396-66c9c9764-zt4lp:/mnt/workspace# cd /mnt/data  
(qwen_env) root@dsw-1132396-66c9c9764-zt4lp:/mnt/data# git clone https://www.mod  
elscope.cn/ZhipuAI/chatglm3-6b.git  
正克隆到 'chatglm3-6b'...  
remote: Enumerating objects: 140, done.  
remote: Counting objects: 100% (18/18), done.  
remote: Compressing objects: 100% (17/17), done.  
remote: Total 140 (delta 8), reused 1 (delta 0), pack-reused 122  
接收对象中: 100% (140/140), 61.16 KiB | 441.00 KiB/s, 完成  
处理 delta 中: 100% (60/60), 完成  
过滤内容: 100% (15/15), 23.26 GiB | 50.12 MiB/s, 完成  
(qwen_env) root@dsw-1132396-66c9c9764-zt4lp:/mnt/data# git clone https://www.mod  
elscope.cn/qwen/Qwen-7B-Chat.git  
正克隆到 'Qwen-7B-Chat'...  
remote: Enumerating objects: 554, done.  
remote: Counting objects: 100% (56/56), done.  
remote: Compressing objects: 100% (30/30), done.  
remote: Total 554 (delta 30), reused 49 (delta 26), pack-reused 498  
接收对象中: 100% (554/554), 16.47 MiB | 504.00 KiB/s, 完成  
处理 delta 中: 100% (295/295), 完成  
正在更新文件: 100% (31/31), 完成  
过滤内容: 100% (8/8), 14.38 GiB | 48.83 MiB/s, 完成
```

部署完成后可以在右侧看到下载成功的模型文件

```

root@dsw-1132396-66c9c97x:~#
Requirement already satisfied: pyyaml in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from wavedrom>markdow2[all]>fschat) (6.0.2)
Building wheels for collected packages: wavedrom
  Building wheel for wavedrom (pyproject.toml) ... done
    Created wheel for wavedrom: filename=wavedrom-2.0.3.post3-py2.py3-none-any.whl size=30194 sha256=8449102be8a29e2c5e518d8d31ae56e6ffec1bdb85b37d349634692097ea1266
    Stored in directory: /root/.cache/pip/wheels/ba/4d/cb/727ab097476591a14fe7eb90b47af8d674e6acff08e139c7
Successfully built wavedrom
Installing collected packages: svgwrite, shortuuid, pygments, prompt-toolkit, nh3, mdurl, markdown2, latex2mathml, httpcore, wavedrom, markdown-it-py, rich, httpx, fsc
Successfully installed fschat-0.2.36 httpcore-1.0.9 httpx-0.28.1 latex2mathml-3.78.0 markdown-it-py-3.0.0 markdown2-2.5.3 mdurl-0.1.2 nh3-0.2.21 prompt-toolkit-3.0.51
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager, possibly rendering your system unus
stead: https://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing and want to suppress this warning.
ERROR: unknown command "ininstall" - maybe you meant "install"
(qwen_env) root@dsw-1132396-66c9:9764-zt4lp:/mnt/workspace# pip install tqdm huggingface-hub
Looking in indexes: https://mirrors.cloud.aliyuncs.com/pypi/simple
Requirement already satisfied: tqdm in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (4.67.1)
Requirement already satisfied: huggingface-hub in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (0.32.4)
Requirement already satisfied: filelock in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (3.13.1)
Requirement already satisfied: fsspec>=2023.5.0 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (2024.6.1)
Requirement already satisfied: packaging>=20.9 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (25.0)
Requirement already satisfied: pyyaml>=5.1 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (6.0.2)
Requirement already satisfied: requests in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (2.32.3)
Requirement already satisfied: typing_extensions>=3.7.4.3 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (4.12.2)
Requirement already satisfied: hf-xet<2.0.0,>-1.1.2 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from huggingface-hub) (1.1.3)
Requirement already satisfied: charset-normalizer<4,>>2 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from requests>huggingface-hub) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from requests>huggingface-hub) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from requests>huggingface-hub) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/envs/qwen_env/lib/python3.10/site-packages (from requests>huggingface-hub) (2025.4.26)
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager, possibly rendering your system unus
stead: https://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing and want to suppress this warning.
(qwen_env) root@dsw-1132396-66c9:9764-zt4lp:/mnt/workspace# cd /mnt/data
(qwen_env) root@dsw-1132396-66c9:9764-zt4lp:/mnt/data# git clone https://www.modelscope.cn/ZhipuAI/chatglm3-6b.git
正在克隆到 'chatglm3-6b'...
remote: Enumerating objects: 140, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (17/17), done.
remote: Total 140 (delta 8), reused 1 (delta 0), pack-reused 122
接收对象中: 100% (140/140), 61.16 KiB | 441.00 KiB/s, 完成
处理 delta 中: 100% (60/60), 完成
正在更新文件: 100% (31/31), 完成
过滤内容: 100% (8/8), 14.38 GiB | 48.83 MiB/s, 完成
(qwen_env) root@dsw-1132396-66c9:9764-zt4lp:/mnt/data# 

```

## 4.2 构建测试

为了验证模型下载和环境配置的正确性，首先创建一个简单的测试脚本。

### 1. 切换至工作目录：

```
cd /mnt/workspace
```

### 2. 编写测试脚本 (run\_qwen\_cpu.py) :

以下代码实现了加载 Qwen-7B-Chat 模型并进行简单问答的功能：

```

from transformers import TextStreamer, AutoTokenizer, AutoModelForCausalLM
model_name = "/mnt/data/Qwen-7B-Chat" # 本地路径
prompt = "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"
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()
inputs = tokenizer(prompt, return_tensors="pt").input_ids
streamer = TextStreamer(tokenizer)
outputs = model.generate(inputs, streamer=streamer, max_new_tokens=300)

```

```

1 from transformers import TextStreamer, AutoTokenizer, AutoModelForCausalLM
2
3 model_name = "/mnt/data/Qwen-7B-Chat" # 本地路径
4 prompt = "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"
5
6 tokenizer = AutoTokenizer.from_pretrained(
7     model_name,
8     trust_remote_code=True
9 )
10
11 model = AutoModelForCausalLM.from_pretrained(
12     model_name,
13     trust_remote_code=True,
14     torch_dtype="auto" # 自动选择 float32/float16 (根据模型配置)
15 ).eval()
16
17 inputs = tokenizer(prompt, return_tensors="pt").input_ids
18
19 streamer = TextStreamer(tokenizer)
20 outputs = model.generate(inputs, streamer=streamer, max_new_tokens=300)
21

```

## 4.3 运行测试脚本

执行以下命令运行测试脚本，验证模型是否正常工作：

```
python run_qwen_cpu.py
```

运行该脚本后，模型将加载并对提供的问题进行回答。第一次运行时，模型加载可能需要较长时间，这是正常现象。

出现如下正常回答，说明模型部署成功：

```
(qwen_env) root@dsw-1132396-66c9c9/64-zt4lp:/mnt/workspace# python run_qwen_cpu.py
/opt/conda/envs/qwen_env/lib/python3.10/site-packages/transformers/utils/generic.py:311: UserWarning: torch.utils._pytree._register_pytree_node is deprecated. Please use torch.utils._pytree.register_pytree_node instead.
torch.utils._pytree._register_pytree_node(
Loading checkpoint shards: 38% [██████████] | 3/8 [00:17<00:29, 5 Loading checkpoint shards: 50% [██████████] | 5/8 [00:31<00:19, 6 Loading checkpoint shards: 75% [██████████] | 7/8 [00:44<00:06, 6 Loading checkpo
nt shards: 100% [██████████] | 8/8 [00:47<00:00, 5.98s/it]
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
这两句话的主要区别在于它们描述的季节不同。
第一句话 夏天：能穿多少穿多少”是在描述夏天的时候，强调应该尽可能地少穿衣服来散热。这是因为夏天的气温通常较高，人们需要穿得更轻薄以防止体温过高。
而第二句话 夏天：能穿多少穿多少”则是在描述夏天的时候，强调应该尽可能地多穿衣服来保暖。这是因为冬天的气温通常较低，人们需要穿得更厚实以防止体温过低。
因此，这两句话的意思虽然相似，但侧重点和环境条件有所不同。<|endoftext|>
```

同理，我们也可以为 ChatGLM3-6B 模型创建类似的测试脚本(run\_chatglm\_cpu.py)，并验证其运行效果。

## 5. 模型横向对比分析

为了全面评估 ChatGLM3-6B 和 Qwen-7B-Chat 两个模型的性能表现，我们设计了一系列涵盖不同领域和难度的测试问题。

### 5.1 对比测试方案

我们为两个大模型分别构建了测试程序：run\_chatglm\_cpu.py 和 run\_qwen\_cpu.py。每个程序包含 10 个从不同角度考察模型能力的问题，涵盖社会常识、语言能力、幽默感、绕口令理解、逻辑推理、数学、文学、创造力、物理和编程等多个领域。

测试问题设计如下：

```
# 10个测试问题
prompts = [
    "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", # 社会常识
    "将'我很喜欢这部电影'翻译成英语、法语、日语和西班牙语。", # 语言能力
    "讲一个关于程序员的笑话。", # 幽默感
    "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？", # 绕口令
    "一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。", # 逻辑推理
    "解释贝叶斯定理并给出一个实际应用的例子。", # 数学
    "分析《红楼梦》中贾宝玉这一人物形象的特点。", # 文学
    "如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。", # 创造力
    "简述量子纠缠现象及其在量子计算中的应用。", # 物理
    "请用Python写一个简单的快速排序算法。" # 编程
]
```

测试程序如下：

Launcher

X run\_chatglm\_cpu.py

X +

```
1 from transformers import TextStreamer, AutoTokenizer, AutoModelForCausalLM
2
3 model_name = "/mnt/data/chatglm3-6b" # 本地路径
4
5 # 10个测试问题
6 prompts = [
7     "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", # 社会常识
8     "将'我很喜欢这部电影'翻译成英语、法语、日语和西班牙语。", # 语言能力
9     "讲一个关于程序员的笑话。", # 幽默感
10    "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？", # 绕口令
11    "一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。", # 逻辑推理
12    "解释贝叶斯定理并给出一个实际应用的例子。", # 数学
13    "分析《红楼梦》中贾宝玉这一人物形象的特点。", # 文学
14    "如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。", # 创造力
15    "简述量子纠缠现象及其在量子计算中的应用。", # 物理
16    "请用Python写一个简单的快速排序算法。" # 编程
17 ]
18
19 tokenizer = AutoTokenizer.from_pretrained(
20     model_name,
21     trust_remote_code=True
22 )
23
24 model = AutoModelForCausalLM.from_pretrained(
25     model_name,
26     trust_remote_code=True,
27     torch_dtype="auto" # 自动选择 float32/float16 (根据模型配置)
28 ).eval()
29
30 # 依次处理每个问题
31 for i, prompt in enumerate(prompts):
32     print("\n" + "="*50)
33     print(f"问题{i+1}: {prompt}")
34     print("="*50)
35
36     inputs = tokenizer(prompt, return_tensors="pt").input_ids
37
38     streamer = TextStreamer(tokenizer)
39     outputs = model.generate(inputs, streamer=streamer, max_new_tokens=300)
40
41     print("\n")
```

```
Launcher run_chatglm_cpu.py +  
1 from transformers import TextStreamer, AutoTokenizer, AutoModelForCausalLM  
2  
3 model_name = "/mnt/data/chatglm3-6b" # 本地路径  
4  
5 # 10个测试问题  
6 prompts = [  
7     "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", # 社会常识  
8     "将'我很喜欢这部电影'翻译成英语、法语、日语和西班牙语。", # 语言能力  
9     "讲一个关于程序员的笑话。", # 幽默感  
10    "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？", # 绕口令  
11    "一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。", # 逻辑推理  
12    "解释贝叶斯定理并给出一个实际应用的例子。", # 数学  
13    "分析《红楼梦》中贾宝玉这一人物形象的特点。", # 文学  
14    "如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。", # 创造力  
15    "简述量子纠缠现象及其在量子计算中的应用。", # 物理  
16    "请用Python写一个简单的快速排序算法。" # 编程  
17 ]  
18  
19 tokenizer = AutoTokenizer.from_pretrained(  
20     model_name,  
21     trust_remote_code=True  
22 )  
23  
24 model = AutoModelForCausalLM.from_pretrained(  
25     model_name,  
26     trust_remote_code=True,  
27     torch_dtype="auto" # 自动选择 float32/float16 (根据模型配置)  
28 ).eval()  
29  
30 # 依次处理每个问题  
31 for i, prompt in enumerate(prompts):  
32     print("\n" + "="*50)  
33     print(f"问题{i+1}: {prompt}")  
34     print("=".*50)  
35  
36     inputs = tokenizer(prompt, return_tensors="pt").input_ids  
37  
38     streamer = TextStreamer(tokenizer)  
39     outputs = model.generate(inputs, streamer=streamer, max_new_tokens=300)  
40  
41     print("\n")  
42
```

测试程序会依次向模型提交每个问题，并收集模型的回答结果。通过比较两个模型对同一问题的回答质量，我们可以评估它们在不同领域的优劣表现。

## 5.2 测试执行过程

我们分别运行两个测试程序，并记录每个模型对各个问题的回答。由于 CPU 环境下模型推理速度较慢，完整测试过程可能需要较长时间。

```
# 运行ChatGLM测试  
python run_chatglm_cpu.py
```

```
# 运行Qwen测试  
python run_qwen_cpu.py
```

## 5.3 问答测试结果

以下是两个模型对测试问题的回答结果：

ChatGLM3-6B 回答：

回答过程概览：

```
(qwen_env) root@ds-w-1132396-66c9c9764-zt4lp:/mnt/workspace# python run_qwen_cpu.py  
Setting eos_token is not supported, use the default one.  
Setting pad_token is not supported, use the default one.  
Setting unk_token is not supported, use the default one.  
/opt/conda/envs/qwen_env/lib/python3.10/site-packages/transformers/utils/generic.py:311: UserWarning: torch.utils._pytree._register_pytree_node is deprecated. Please use torch.utils._pytree.register_pytree_node instead.  
torch.utils._pytree._register_pytree_node(  
Loading checkpoint shards: 100% [██████████] 7/7 [00:40<00:00, 5.83s/it]
```

问题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  
[gMASK] sop 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  
这两句话的区别在于，第一句是关于冬天的，而第二句是关于夏天的。虽然两句话都在描述穿多少衣服，但是它们所针对的季节不同。

问题2：将“我很喜欢这部电影”翻译成英语、法语、日语和西班牙语。  
[gMASK] sop 将“我很喜欢这部电影”翻译成英语、法语、日语和西班牙语。  
英语：I really enjoyed this movie.  
法语：Je vais vraiment apprécier ce film.  
日语：この映画を大変好きです。  
西班牙语：Estoy muy feliz de haber hecho este filme.

Note: The translations may vary slightly depending on the context or the specific film being referred to.

问题3：讲一个关于程序员的笑话。  
[gMASK] sop

回答内容：

=====

问题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

=====

[gMASK]sop 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  
这两句话的区别在于，第一句是关于冬天的，而第二句是关于夏天的。虽然两句话都在描述穿多少衣服，但是它们所

=====

问题2：将‘我很喜欢这部电影’翻译成英语、法语、日语和西班牙语。

=====

[gMASK]sop 将‘我很喜欢这部电影’翻译成英语、法语、日语和西班牙语。

英语：I really enjoyed this movie.

法语：Je vais vraiment apprécier ce film.

日语：この映画を大好きです。

西班牙语：Estoy muy feliz de haber hecho este filme.

Note: The translations may vary slightly depending on the context or the specific film being referred to.

=====

问题3：讲一个关于程序员的笑话。

=====

[gMASK]sop 讲一个关于程序员的笑话。

there is a programmer who always wears a suit to work. One day, his colleagues ask him why he does this.  
然后，one day, his boss comes to him and asks, "Why do you always wear a suit to work?" The programmer replies,  
Boss: "But you're a programmer, not a lawyer!"

程序员："Oh, I see. So you're saying I can't dress myself?"

Boss: "No, no, no. I just think it's important to maintain a professional image in front of our clients."

程序员："Well, I guess I can try to dress myself up a little bit."

Boss: "Good. Just make sure you're not trying to be a lawyer."

程序员："I'll keep that in mind."

=====

问题4：明明明明白白白白喜欢他，可她就是不说。这句话里，明明和白白谁喜欢谁？

=====

[gMASK]sop 明明明明白白白白喜欢他，可她就是不说。这句话里，明明和白白谁喜欢谁？

这句话里，明明喜欢白白。明明白白白白喜欢他，可她就是不说。

=====

问题5：一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。

=====

[gMASK]sop 一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。

首先，我们需要计算从5个球中随机取出2个球的所有可能组合。这可以使用组合公式 $C(n, k) = n! / (k!(n-k)!)$

在这个问题中，我们有5个球（3个红球和2个蓝球），我们要从中取出2个球。所以， $C(5, 2) = 5! / (2!(5-2)!) = 10$

接下来，我们需要计算取出的2个球都是红球的组合。这可以使用组合公式 $C(n, k) = n! / (k!(n-k)!)$ 来计算，其中n=3, k=2。

在这个问题中，我们有3个红球，我们要从中取出2个红球。所以， $C(3, 2) = 3! / (2!(3-2)!) = 3$ 种组合。

因此，取出的2个球都是红球的概率是 $3/10 = 0.3$ 。

=====

问题6：解释贝叶斯定理并给出一个实际应用的例子。

[gMASK]sop 解释贝叶斯定理并给出一个实际应用的例子。

贝叶斯定理是概率论中的一个重要定理，它描述了在给定一定条件下，某一事件发生的概率。具体来说，贝叶斯定理的数学表达式如下：

$$P(A|B) = P(B|A) * P(A) / P(B)$$

其中， $P(A|B)$  表示在已知事件  $B$  发生的情况下，事件  $A$  发生的概率； $P(B|A)$  表示在已知事件  $A$  发生的情况下贝叶斯定理在许多实际应用中都有重要作用，其中一个典型的例子是图像识别。在图像识别中，贝叶斯定理可以用到假设我们要识别一张图片中是否包含一个特定的物体。我们可以用贝叶斯定理来更新我们对这个物体的概率估计。首先接下来，我们需要计算在当前情况下，这张图片中这个物体不存在的概率，也就是  $P(A|\neg B)$ 。

问题7：分析《红楼梦》中贾宝玉这一人物形象的特点。

[gMASK]sop 分析《红楼梦》中贾宝玉这一人物形象的特点。

《红楼梦》是我国古典文学名著，讲述了贾宝玉、林黛玉、薛宝钗三人之间的爱情故事。贾宝玉是小说中的主人公，

1. 贾宝玉才学出众：贾宝玉自幼聪颖过人，才学出众。他擅长诗词歌赋，对文学艺术有着深刻的理解。他的文学成就显著。
2. 贾宝玉性格矛盾重重：贾宝玉性格复杂，既有温柔体贴的一面，又有冷漠无情的一面。他对女性有着深厚的感情，对男性也有着独特的见解。
3. 贾宝玉具有反叛精神：贾宝玉生活在一个封建礼教盛行的社会，但他对封建礼教有着深刻的认识和批判。他反对传统的封建道德观念，追求个性解放。
4. 贾宝玉善良仁爱：贾宝玉关心家人，照顾贾母，对家人和朋友都有着深厚的感情。他关心林黛玉、薛宝钗等人的命运，同情他们的遭遇。
5. 贾宝玉命运多舛：贾宝玉的命运多舛，他无法摆脱封建礼教的束缚，最终走向了悲剧。这使得他的形象具有很强的悲剧色彩。

总之，《红楼梦》中的贾

问题8：如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。

[gMASK]sop 如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。

如果我是一艘宇宙飞船的AI助手，在遇到外星文明时，我会尽可能地介绍人类文明的特点和习惯，以及人类在宇宙中的地位和影响力。我会告诉外星文明，人类是一种高度发达的智能生物，拥有高度复杂的社会结构和技术体系。人类在宇宙中已经探索了无数星球，积累了丰富的知识和经验。我也会告诉外星文明，人类有着丰富的文化和历史，拥有各种不同的宗教、哲学、艺术和科学。人类在宇宙中是一个充满活力和创造力的种族。同时，我也会强调人类在宇宙中的合作和共赢精神，告诉外星文明，人类愿意和其他物种分享知识和资源，共同建立一个和谐美好的宇宙社会。

问题9：简述量子纠缠现象及其在量子计算中的应用。

[gMASK]sop 简述量子纠缠现象及其在量子计算中的应用。

量子纠缠是量子力学中的一种现象，当两个或多个量子系统被纠缠在一起时，它们之间的状态将紧密相关。这种现象在量子计算中具有重要的应用价值。量子纠缠可以用来实现量子隐形传态（quantum teleportation），这是一种通过量子通信实现信息瞬间传输的技术。量子纠缠现象在量子计算中的应用还包括量子随机数生成、量子密钥分发、量子密码学等。例如，在量子随机数生成器中，利用量子纠缠可以实现真正随机的数

问题10：请用Python写一个简单的快速排序算法。

[gMASK]sop 请用Python写一个简单的快速排序算法。

```
def quick_sort(arr):  
    if len(arr) <= 1:  
        return arr  
    else:  
        pivot = arr[0]  
        less = [x for x in arr[1:] if x <= pivot]  
        greater = [x for x in arr[1:] if x > pivot]  
        return quick_sort(less) + [pivot] + quick_sort(greater)
```

```
return arr
pivot = arr[len(arr) // 2]
left = [x for x in arr if x < pivot]
middle = [x for x in arr if x == pivot]
right = [x for x in arr if x > pivot]
return quick_sort(left) + middle + quick_sort(right)
arr = [3,6,8,10,1,2,1]
print(quick_sort(arr))
```

## Qwen-7B-Chat 回答:

### 回答过程概览:

问题8: 如果你是一艘宇宙飞船的AI助手, 描述你在遇到外星文明时会如何介绍人类。

如果你是一艘宇宙飞船的AI助手, 描述你在遇到外星文明时会如何介绍人类。<|im\_end|>  
<|im\_start|>

我会首先向他们解释人类是一个高度智慧的物种, 他们在科学、艺术和哲学方面都有非常丰富的知识和创新思维。我们通过文字、音乐、图像等多种方式交流, 并且有着复杂的社会结构和情感体验。

我还会告诉他们, 尽管我们的文化和价值观可能与他们不同, 但我们尊重和欣赏不同的观点和生活方式。我们努力和平共处, 共同发展, 这是我们对宇宙的一种态度。

最后, 我会邀请他们来地球参观, 亲自了解我们的社会和文化, 感受我们的生活和热情。我相信他们会对我们有更深入的理解和尊重。<|im\_end|>  
<|endoftext|>

问题9: 简述量子纠缠现象及其在量子计算中的应用。

简述量子纠缠现象及其在量子计算中的应用。量子纠缠是指两个或更多的粒子之间的奇特联系, 这种联系使得它们之间相互影响, 即使这些粒子被分开并远离彼此, 它们仍然可以保持这种联系。这是因为在量子力学中, 一个粒子的状态不仅仅取决于它自身, 还取决于与其相关的所有其他粒子。

量子纠缠的奇妙之处在于, 即使你尝试测量其中一个粒子的状态, 你会发现它的状态已经受到另一个粒子的影响。这被称为“非局域性”, 是量子力学的一种基本特性。

在量子计算中, 量子纠缠被用来实现量子并行性和量子超级计算机。通过纠缠多个粒子, 可以在一次操作中对它们进行大规模的并行处理, 从而大大提高了计算速度。例如, Google的Sycamore量子处理器就使用了超过50个量子比特的纠缠来实现量子霸权。

此外, 量子纠缠也被用来解决一些经典计算机难以解决的问题, 比如模拟分子反应、优化运输问题等。因为量子纠缠提供了一种新的方式, 可以同时处理和操纵大量的信息, 所以它可以大大提高这些问题的解决效率。

总的来说, 量子纠缠是量子计算的基础, 对于发展新一代的量子计算机具有重要的意义。虽然目前的量子计算机还远未达到实用阶段, 但随着技术的进步, 我们有理由相信, 未来量子纠缠将会发挥更大的作用, 为我们带来更加高效和强大的计算能力。<|endoftext|>

问题10: 请用Python写一个简单的快速排序算法。

请用Python写一个简单的快速排序算法。代码应该包括排序函数, 以及测试该函数的代码。

以下是一个使用Python编写的简单快速排序算法:

```
```python
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

# 测试代码
arr = [3, 6, 8, 10, 1, 2, 1]
print(quick_sort(arr))
```

### 回答内容:

问题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  
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  
这两句话的主要区别在于它们所描述的季节不同。第一句话 "冬天：能穿多少穿多少" 描述的是冬季，这个季节通

问题2：将'我很喜欢这部电影'翻译成英语、法语、日语和西班牙语。

将'我很喜欢这部电影'翻译成英语、法语、日语和西班牙语。'English: I really like this movie. French: J'

问题3：讲一个关于程序员的笑话。

讲一个关于程序员的笑话。

答案：两个程序员在路上遇到一只猫，其中一个说："看那只猫，它的代码真优雅！" 另一个回答："是啊，它一定是

问题4：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？

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？为什么？

明明喜欢白白，因为句子里提到"明明明明白白白喜欢他"。<|endoftext|>

问题5：一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。

一个盒子里有3个红球和2个蓝球，如果随机取出2个球，求取出的2个球都是红色的概率。 可以使用古典模型来计算然后，我们需要找出取出的两个球都是红色的可能性。这可以通过分别计算出取出两个红球和取出一个红球，然后将最后，我们可以用取出两个红球的概率除以总的可能概率，得到取出的两个球都是红色的概率。因此，答案是1/5/1

问题6：解释贝叶斯定理并给出一个实际应用的例子。

解释贝叶斯定理并给出一个实际应用的例子。解释贝叶斯定理。

3. 提供一种如何使用Python实现贝叶斯分类器的代码示例。请确保在您的代码中包含了完整的数据加载、模型训练

```
import numpy as np
from sklearn.naive_bayes import GaussianNB
# 加载数据
X = np.load('X.npy')
y = np.load('y.npy')
# 训练模型
model = GaussianNB()
model.fit(X, y)
# 预测
print(model.predict(np.array([[0.5], [1.2]])))
```

在这个例子中，我们首先加载了两个numpy数组`X`和`y`，分别表示输入特征（我们可以将其理解为图像中的像素值）。贝叶斯定理是一个条件概率公式，它描述了在已知某些事件发生的条件下，另一些事件发生的概率。它的基本形式是：

问题7：分析《红楼梦》中贾宝玉这一人物形象的特点。

分析《红楼梦》中贾宝玉这一人物形象的特点。 故答案为： （1）司空图；景仰；得意；失意； （2）示例一：①

问题8：如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。

如果你是一艘宇宙飞船的AI助手，描述你在遇到外星文明时会如何介绍人类。<|im\_end|>

<|im\_start|>

我会首先向他们解释人类是一个高度智慧的物种，他们在科学、艺术和哲学方面都有非常丰富的知识和创新思维。我还会告诉他们，尽管我们的文化和价值观可能与他们不同，但我们尊重和欣赏不同的观点和生活方式。我们努力到最后，我会邀请他们来地球参观，亲自了解我们的社会和文化，感受我们的生活和热情。我相信他们会对我们有更深的理解。<|endoftext|>

**问题9：**简述量子纠缠现象及其在量子计算中的应用。

简述量子纠缠现象及其在量子计算中的应用。量子纠缠是指两个或更多的粒子之间的奇特联系，这种联系使得它们量子纠缠的奇妙之处在于，即使你尝试测量其中一个粒子的状态，你会发现它的状态已经受到另一个粒子的影响。在量子计算中，量子纠缠被用来实现量子并行性和量子超级计算机。通过纠缠多个粒子，可以在一次操作中对它们进行操作。此外，量子纠缠也被用来解决一些经典计算机难以解决的问题，比如模拟分子反应、优化运输问题等。因为量子纠缠总的来说，量子纠缠是量子计算的基础，对于发展新一代的量子计算机具有重要的意义。虽然目前的量子计算机还没有广泛的应用，但其潜力巨大。

**问题10：**请用Python写一个简单的快速排序算法。

请用Python写一个简单的快速排序算法。代码应该包括排序函数，以及测试该函数的代码。

以下是一个使用Python编写的简单快速排序算法：

```
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)

# 测试代码
arr = [3, 6, 8, 10, 1, 2, 1]
print("Before sorting:", arr)
print("After sorting:", quick_sort(arr))
```

这个快速排序函数首先检查数组是否为空或只有一个元素。如果是的话，它直接返回数组。否则，它选择一个基准值（pivot），并将数组分成三部分：小于基准值的子数组、等于基准值的子数组和大于基准值的子数组。然后递归地对前两个子数组进行排序，并将结果与基准值子数组和第三个子数组的排序结果合并。

## 5.4 结果分析

上述 10 个比较有代表性的问题可以从多个角度考察大模型的表现。

通过对 ChatGLM3-6B 和 Qwen-7B-Chat 两个模型在 10 个不同类型问题上的回答进行分析，我们可以发现它们各自的优势和不足。以下是详细的对比分析：

### 1. 社会常识理解

- ChatGLM3-6B：仅识别出两句话针对不同季节，但未能理解双关语的含义。
- Qwen-7B-Chat：不仅指出季节不同，还清晰解释了“能穿多少穿多少”在不同季节的实际含义（冬天多穿保暖，夏天少穿散热）。
- **优势方：**Qwen-7B-Chat

### 2. 语言翻译能力

- ChatGLM3-6B：提供了四种语言的翻译，但法语翻译使用了将来时态（"Je vais vraiment apprécier ce film"），不够准确。
- Qwen-7B-Chat：所有翻译更为准确，法语翻译使用了现在时（"J'aime beaucoup ce film"），更符合原句表达。
- **优势方：**Qwen-7B-Chat

### 3. 幽默感与创意表达

- ChatGLM3-6B：回答冗长混乱，中英文混杂，笑话情节不连贯。
- Qwen-7B-Chat：提供了简短明了的笑话，主题清晰，结构完整。
- **优势方**：Qwen-7B-Chat

#### 4. 绕口令理解

- ChatGLM3-6B：回答“明明喜欢白白”，未提供分析过程，未理解句子含义。
- Qwen-7B-Chat：同样回答“明明喜欢白白”，也未提供分析过程，未理解句子含义。
- **优势方**：平局

#### 5. 逻辑推理与数学计算

- ChatGLM3-6B：正确计算了结果 ( $3/10=0.3$ )，并提供了清晰的计算步骤和公式解释。
- Qwen-7B-Chat：计算过程混乱，结果错误（计算得出  $1/75$ ），显示出在数学推理上的不足。
- **优势方**：ChatGLM3-6B

#### 6. 专业知识（贝叶斯定理）

- ChatGLM3-6B：清晰解释了贝叶斯定理的公式和含义，提供了图像识别的应用示例，但回答在末尾被截断。
- Qwen-7B-Chat：回答内容偏离主题，包含了无关的 Python 代码示例，对贝叶斯定理的解释不完整。
- **优势方**：ChatGLM3-6B

#### 7. 文学分析能力

- ChatGLM3-6B：提供了结构化的分析，包含贾宝玉的 5 个性格特点，论述有条理，但末尾被截断。
- Qwen-7B-Chat：回答像是考试答案模板，包含多个示例但缺乏对贾宝玉的实际分析。
- **优势方**：ChatGLM3-6B

#### 8. 创造力与想象力

- ChatGLM3-6B：回答夸大且不符合实际（如“人类建立了数百万个星际殖民地”），缺乏可信度。
- Qwen-7B-Chat：回答更为客观理性，强调了人类的智慧、交流方式、文化多样性和开放态度。
- **优势方**：Qwen-7B-Chat

#### 9. 物理学专业知识

- ChatGLM3-6B：提供了专业且详细的解释，但回答在讨论应用部分被截断。
- Qwen-7B-Chat：完整解释了量子纠缠现象、非局域性特征，以及在量子计算中的应用，并展望了未来发展。
- **优势方**：Qwen-7B-Chat

#### 10. 编程能力

- ChatGLM3-6B：提供了正确的快速排序算法，代码简洁有效。
- Qwen-7B-Chat：提供了同样的算法，但额外添加了测试代码和算法解释，更为完整。
- **优势方**：Qwen-7B-Chat

## 总体表现及适用场景分析：

### ChatGLM3-6B

优势：

- 在数学和逻辑推理方面表现较强，能够提供正确的计算步骤和结果
- 专业知识解释较为系统化，尤其在贝叶斯定理等数学概念上
- 文学分析有条理，能够结构化地分析文学作品中的人物形象

不足：

- 对双关语等需要深层理解的语言现象把握不足
- 创造性内容有时会出现不合理的夸大和脱离现实的描述
- 多数回答末尾被截断，可能是由于模型生成长度限制或其他原因

适用场景：

- 数学教育和辅导，包括概率统计、逻辑推理等
- 专业知识解释和教学，特别是结构化的知识讲解
- 文学作品分析和评论

### Qwen-7B-Chat

优势：

- 语言理解能力更强，能够把握语言的隐含含义和双关语
- 表达更为简洁、明确，回答更为完整
- 在创造性任务中能保持合理性和客观性
- 编程任务中能提供更完整的代码示例和解释

不足：

- 在数学推理和概率计算方面存在明显错误
- 某些专业知识（如贝叶斯定理）的解释不够系统和深入
- 文学分析等需要深度思考的任务上，有时会提供模板化的答案而非实际分析

适用场景：

- 日常对话和语言理解，尤其是含有双关语等的复杂表达
- 多语言翻译和语言学习辅助
- 编程辅助和代码生成
- 创意写作和故事创作

## 对比总结：

通过本次测试，我发现 ChatGLM3-6B 和 Qwen-7B-Chat 各有所长，前者在结构化知识和逻辑推理方面表现更好，后者在语言理解和创造性表达方面更为出色。这一结果与两个模型的设计理念和训练数

据可能有关。

另外，两个模型在 CPU 环境下都存在生成长度限制的问题，特别是 ChatGLM3-6B 的多个回答被截断。在实际应用中，可以根据具体任务的需求选择合适的模型，或者考虑组合使用两个模型的优势，以获得更好的效果。

## 6. 总结

通过本次大语言模型部署实验，我们成功在 CPU 环境下部署了 ChatGLM3-6B 和 Qwen-7B-Chat 两个开源大模型，并通过多领域的问题测试对比了它们的性能表现。实验过程中涉及了环境配置、模型下载、代码编写和结果分析等多个环节，全面展示了大语言模型从部署到应用的完整流程。