

同濟大學

TONGJI UNIVERSITY

大语言模型部署实验报告

课程名称	人工智能导论
学 院	计算机科学与技术学院
专 业	软件工程
学生姓名	杨瑞晨
学 号	2351050
指导教师	汪昱
日 期	2025 年 6 月 3 日

目 录

1 作业概述	1
2 环境配置与模型部署	1
2.1 平台与环境	1
2.2 环境配置	1
2.3 模型部署过程	2
2.3.1 Qwen-7B-Chat 的部署	2
2.3.2 ChatGLM3-6B 的部署	3
3 问答测试结果	4
3.1 问题一：冬天夏天穿衣问题	5
3.1.1 Qwen-7B-Chat 的回答	5
3.1.2 ChatGLM3-6B 的回答	5
3.2 问题二：单身狗原因问题	5
3.2.1 Qwen-7B-Chat 的回答	5
3.2.2 ChatGLM3-6B 的回答	5
3.3 问题三：“他知道我知道”问题	5
3.3.1 Qwen-7B-Chat 的回答	6
3.3.2 ChatGLM3-6B 的回答	6
3.4 问题四：“明明白白白”问题	6
3.4.1 Qwen-7B-Chat 的回答	6
3.4.2 ChatGLM3-6B 的回答	6
3.5 问题五：“意思”的多义性理解	6
3.5.1 Qwen-7B-Chat 的回答	7
3.5.2 ChatGLM3-6B 的回答	7
3.6 问题六：信息搜索	7
3.6.1 Qwen-7B-Chat 的回答	7
3.6.2 ChatGLM3-6B 的回答	7
3.7 问题七：代码能力	7
3.7.1 Qwen-7B-Chat 的回答	8
3.7.2 ChatGLM3-6B 的回答	8
3.8 问题八：语言翻译	8
3.8.1 Qwen-7B-Chat 的回答	8
3.8.2 ChatGLM3-6B 的回答	8
4 大语言模型之间的横向对比分析	8

4.1 逻辑推理与歧义理解能力	9
4.2 知识广度与事实准确性	9
4.3 创造性与文本生成能力	9
4.4 多语言能力	10
4.5 模型概览与基础信息	10
4.6 综合评价与总结	10
5 总结与展望	11
A 项目公开可访问链接	12

1 作业概述

本次作业旨在通过魔搭（ModelScope）平台，部署、测试并对比分析当前主流的开源大语言模型。本报告将详细记录部署过程，展示在不同应用场景下模型的测试结果，并从多个维度对选定的模型进行横向对比分析。

在本报告中，我们选择了以下两款具有代表性的开源大语言模型进行测试与分析：

- 通义千问 Qwen-7B-Chat
- 智谱 ChatGLM3-6B

2 环境配置与模型部署

2.1 平台与环境

本次实验基于魔搭（ModelScope）平台提供的在线开发环境 DSW（Data Science Workshop）。通过关联阿里云账号，我们获得了免费的 CPU 云计算资源。环境主要配置如下：

- 平台：魔搭（ModelScope）、Ubuntu 22.04
- 开发环境：DSW (Jupyter Notebook)
- 计算资源：CPU
- 主要依赖库：torch, transformers, modelscope 等

2.2 环境配置

在 DSW 环境中，我们首先需要安装必要的依赖库。以下是安装命令：

(1) 基础环境：

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

(2) 基础依赖：

```
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
```

2.3 模型部署过程

本节展示从魔搭平台获取模型（git clone）并在 DSW 环境中成功部署各模型的过程。

2.3.1 Qwen-7B-Chat 的部署

Qwen-7B-Chat 模型的部署过程如下：

(1) 克隆模型代码库到本地：

```
root@dsw-1118250-76ff6b9d67-9hzj8:/mnt/workspace# git clone https://www.modelscope.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 | 1.06 MiB/s, 完成.
处理 delta 中: 100% (295/295), 完成.
过滤内容: 100% (8/8), 14.38 GiB | 152.74 MiB/s, 完成.
```

图 2.1 克隆 Qwen-7B-Chat 模型代码库

(2) 进入模型目录查看：

```
root@dsw-1118250-76ff6b9d67-9hzj8:/mnt/workspace/Qwen-7B-Chat# ls
assets          configuration_qwen.py  model-00001-of-00008.safetensors  model-00006-of-00008.safetensors  NOTICE          tokenizer_config.json
cache_autogptq_cuda_256.cpp  cpp_kernels.py       model-00002-of-00008.safetensors  model-00007-of-00008.safetensors  qwen_generation_utils.py
cache_autogptq_cuda_kernel_256.cu  examples             model-00003-of-00008.safetensors  model-00008-of-00008.safetensors  qwen.tiktoken
config.json      generation_config.json  model-00004-of-00008.safetensors  modeling_qwen.py                  README.md
configuration.json  LICENSE               model-00005-of-00008.safetensors  model.safetensors.index.json      tokenization_qwen.py
```

图 2.2 进入 Qwen-7B-Chat 模型目录

(3) 加载模型并进行简单测试：

```
1 from transformers import TextStreamer
2 from modelscope import AutoModelForCausalLM, AutoTokenizer
3 from transformers.generation import GenerationConfig
4
5 model_name = "./Qwen-7B-Chat" # 本地路径
6
7 tokenizer = AutoTokenizer.from_pretrained(
8     model_name,
9     trust_remote_code=True
10 )
11
12 model = AutoModelForCausalLM.from_pretrained(
13     model_name,
14     trust_remote_code=True,
15     torch_dtype="auto"
16 ).eval()
17
18 response, history = model.chat(tokenizer, " 你好", history=None)
19 print(response)
```

(4) 成功加载模型并输出结果：

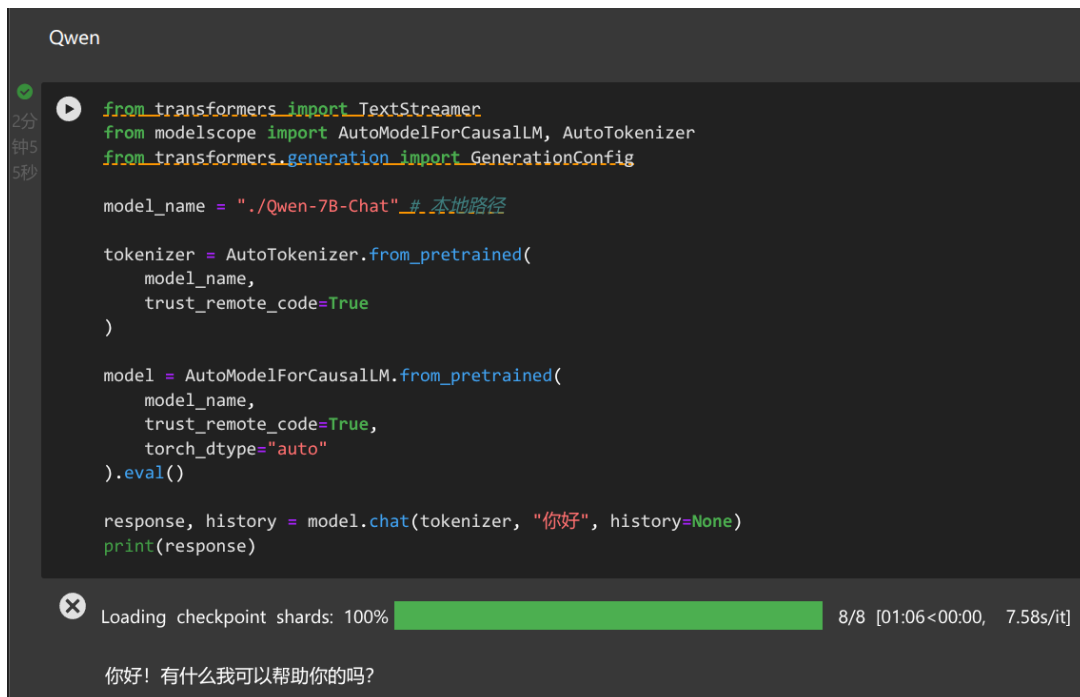


图 2.3 加载 Qwen-7B-Chat 模型

2.3.2 ChatGLM3-6B 的部署

ChatGLM3-6B 模型的部署过程如下：

(1) 克隆模型代码库到本地：

```

root@dsw-1118250-76ff6b9d67-9hzj8:/mnt/workspace# git clone https://www.modelscope.cn/ZhipuAI/chatglm3-6b.git
正克隆到 'chatglm3-6b'...
remote: Enumerating objects: 140, done.
remote: Total 140 (delta 0), reused 0 (delta 0), pack-reused 140
接收对象中: 100% (140/140), 47.10 KiB | 349.00 KiB/s, 完成.
处理 delta 中: 100% (64/64), 完成.
过滤内容: 100% (15/15), 23.26 GiB | 157.88 MiB/s, 完成.
    
```

图 2.4 克隆 ChatGLM3-6B 模型代码库

(2) 进入模型目录查看：

```

root@dsw-1118250-76ff6b9d67-9hzj8:/mnt/workspace/chatglm3-6b# ls
config.json          model-00004-of-00007.safetensors  model.safetensors.index.json  pytorch_model-00006-of-00007.bin  tokenization_chatglm.py
configuration_chatglm.py  model-00005-of-00007.safetensors  pytorch_model-00001-of-00007.bin  pytorch_model-00007-of-00007.bin  tokenizer_config.json
configuration.json       model-00006-of-00007.safetensors  pytorch_model-00002-of-00007.bin  pytorch_model.bin.index.json      tokenizer.model
model-00001-of-00007.safetensors  model-00007-of-00007.safetensors  pytorch_model-00003-of-00007.bin  quantization.py
model-00002-of-00007.safetensors  modeling_chatglm.py               pytorch_model-00004-of-00007.bin  README.md
model-00003-of-00007.safetensors  MODEL_LICENSE                     pytorch_model-00005-of-00007.bin  special_tokens_map.json
    
```

图 2.5 进入 ChatGLM3-6B 模型目录

(3) 加载模型并进行简单测试：

```

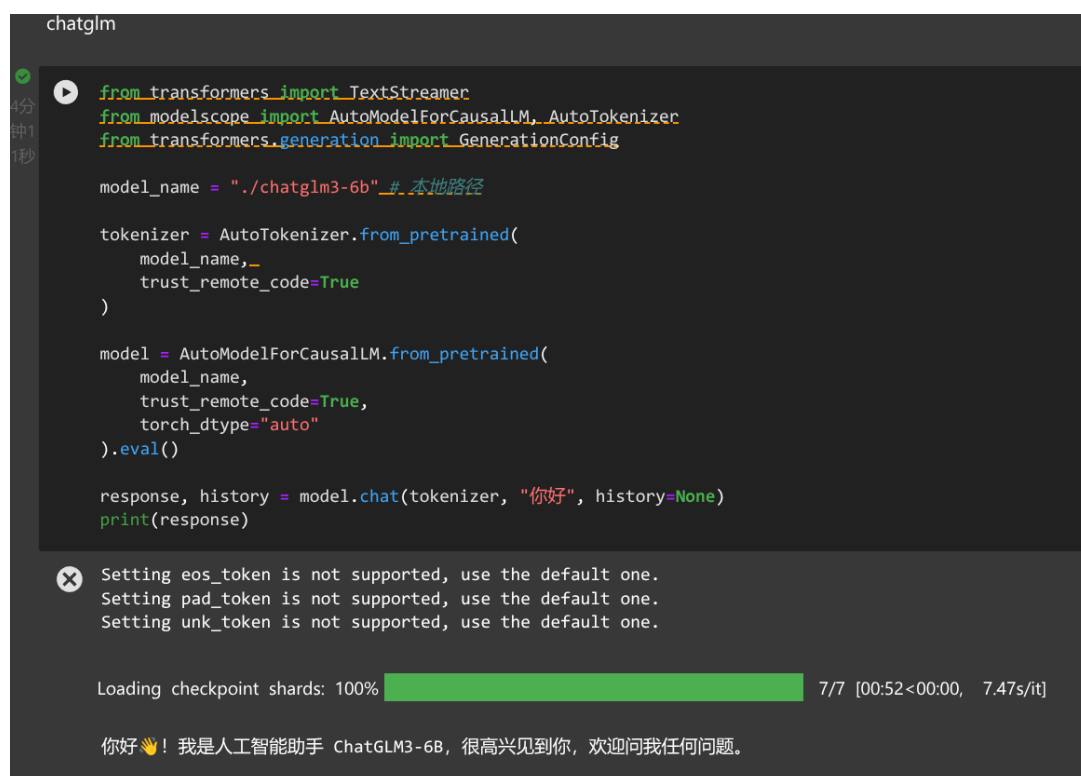
1 from transformers import TextStreamer
2 from modelscope import AutoModelForCausalLM, AutoTokenizer
3 from transformers.generation import GenerationConfig
4
    
```

```

5 model_name = "./chatglm3-6b" # 本地路径
6
7 tokenizer = AutoTokenizer.from_pretrained(
8     model_name,
9     trust_remote_code=True
10 )
11
12 model = AutoModelForCausalLM.from_pretrained(
13     model_name,
14     trust_remote_code=True,
15     torch_dtype="auto"
16 ).eval()
17
18 response, history = model.chat(tokenizer, " 你好", history=None)
19 print(response)

```

(4) 成功加载模型并输出结果：



```

chatglm
4分 1秒
from transformers import TextStreamer
from modelscope import AutoModelForCausalLM, AutoTokenizer
from transformers.generation import GenerationConfig

model_name = "./chatglm3-6b" # 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"
).eval()

response, history = model.chat(tokenizer, "你好", history=None)
print(response)

⊗ 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.

Loading checkpoint shards: 100% ██████████ 7/7 [00:52<00:00, 7.47s/it]

你好👋! 我是人工智能助手 ChatGLM3-6B, 很高兴见到你, 欢迎问我任何问题。

```

图 2.6 加载 ChatGLM3-6B 模型

3 问答测试结果

本节针对作业补充材料中提供的典型问题，对已部署的模型进行测试，并展示其回答。

3.1 问题一：冬天夏天穿衣问题

问题描述：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

3.1.1 Qwen-7B-Chat 的回答

```
Q1
[4] response, history = model.chat(tokenizer, "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", history=None)
print(response)

这两句话表达的意思相反。第一句话是在提醒人们在冬天应该多穿衣服以保暖，而第二句话则是在提醒人们在夏天应该少穿衣服以散热。因此，它们之间的主要区别在于季节和温度差异，以及它们所强调的不同健康需求。
```

3.1.2 ChatGLM3-6B 的回答

```
Q1
[13] response, history = model.chat(tokenizer, "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", history=None)
print(response)

1、冬天：能穿多少穿多少，强调的是在冬天要尽可能多穿一些衣服，以保暖。
2、夏天：能穿多少穿多少，强调的是在夏天要尽可能少穿一些衣服，以凉爽。
```

3.2 问题二：单身狗原因问题

问题描述：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

3.2.1 Qwen-7B-Chat 的回答

```
Q2
[5] response, history = model.chat(tokenizer, "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", history=None)
print(response)

这两句话的区别在于强调的重点不同。第一句话强调的是“谁都不喜欢”，即一个人无法吸引他人；而第二句话强调的是“没有人愿意接近他”，即一个人自身存在一些问题，使别人不愿意与之交往。因此，这两句话的意思是有区别的。第一句话是因为自身条件差，第二句话是因为行为或态度有问题。
```

3.2.2 ChatGLM3-6B 的回答

```
Q2
response, history = model.chat(tokenizer, "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", history=None)
print(response)

这两句话都描述了单身狗的情况，但使用了不同的语法和表达方式。

第一句话使用了“——是谁都看不上”，强调单身狗认为所有人都不足以成为他的伴侣，因此他看不上任何人。这种明智的自信和独立思考方式，让单身狗能够独立自主地选择自己的生活方式，不受传统观念的束缚。

第二句话则使用了“谁都看不上”，这种说法虽然和第一句话的意思相似，但是语法上更加简单。这种说法可能暗示单身狗对自己没有信心，认为自己不够好，所以无法吸引到其他人。这种说法可能缺乏独立思考和自信，Beside, "谁都看不上" is a more common phrase used in everyday conversation, while "——是谁都看不上" is a more formal expression.
```

3.3 问题三：“他知道我知道”问题

问题描述：他知道我知道你知道他不知道吗？这句话里，到底谁不知道

3.3.1 Qwen-7B-Chat 的回答

```
Q3
[6] response, history = model.chat(tokenizer, "他知道我知道你知道他不知道吗？ 这句话里，到底谁不知道", history=None)
print(response)

在这个句子中，所有人都知道一些事情。主角知道“他知道我知道他知道他不知道”，而读者也知道这个句子是在讨论一个复杂的知识状态。然而，“他”本身并不知道他知道其他人的想法，因为他只知道自己的知识状态。因此，在这个句子中，“他”是唯一不知道自己不知道的人。
```

3.3.2 ChatGLM3-6B 的回答

```
Q3
[15] response, history = model.chat(tokenizer, "他知道我知道你知道他不知道吗？ 这句话里，到底谁不知道", history=None)
print(response)

这句话是一个典型的悖论，名为“不知道他不知道”。在这个问题中，主要有两个人物，无法确定谁知道、谁不知道。
假设A知道B知道C，B知道C知道A，那么A知道C知道B。这样就形成了一个循环，每个人都知别人知道的事情，但无法确定自己是否知道被别人不知道的事情。这种情况下，我们无法确定到底谁知道、谁不知道。
```

3.4 问题四：“明明白白白白”问题

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

3.4.1 Qwen-7B-Chat 的回答

```
Q4
[7] response, history = model.chat(tokenizer, "明明白明明白白白喜欢他，可她就是不。 这句话里，明明和白白谁喜欢谁？", history=None)
print(response)

这句话里的“明明”指的是说话者自己，而“白白”则指代另一个人（可能是男性）。因此，可以理解为说话者喜欢另一个人，但不愿意说出来。
```

3.4.2 ChatGLM3-6B 的回答

```
Q4
[16] response, history = model.chat(tokenizer, "明明白明明白白白喜欢他，可她就是不。 这句话里，明明和白白谁喜欢谁？", history=None)
print(response)

从这句话里可以看出，明明知道白白喜欢他，但白白并没有表达出自己的喜欢。所以，在这个情境下，是明明在喜欢白白，而不是白白喜欢明明。
```

3.5 问题五：“意思”的多义性理解

问题描述：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。

3.5.1 Qwen-7B-Chat 的回答

```
Q5
response, history = model.chat(tokenizer, "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：你什么意思？", history)
print(response)
```

以上对话中的“意思”有不同的含义。

1. “意思”指意义、意图或含意。例如：“这个电影很有意思。”
2. “意思”表示一种礼貌的表达方式，可以用于回应他人的问候或感谢。例如：“谢谢你的邀请，我很高兴能来参加。”
3. “意思”也可以表示一种讽刺或挖苦的语气。例如：“你这个人真有意思。”
4. “意思”还可以表示一种尴尬或道歉的态度。例如：“我不好意思。”

3.5.2 ChatGLM3-6B 的回答

```
Q5
response, history = model.chat(tokenizer, "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：你什么意思？", history)
print(response)
```

1. 领导：你这是什么意思？ 意思：含义，意义。领导可能是在询问小明刚刚说的话或做的动作的含义。
2. 小明：没什么意思。意思：没有意义，无意义。小明可能是在表示自己刚刚说的话或做的动作没有特别的含义或价值。
3. 领导：你这就不够意思了。意思：不够好，不够充分。领导可能是在批评小明刚刚说的话或做的动作不够认真或充分。
4. 小明：小意思。小意思。意思：指小事儿，不重要的东西。小明可能是在表示自己刚刚说的话或做的动作只是一个不问题，不算大事情。
5. 领导：那我就不好意思了。意思：表示抱歉或感谢。领导可能是在表示自己对于小明刚刚说的话或做的动作感到抱歉或感谢。
6. 小明：其实也没有别的意思。意思：表示自己的话或动作确实没有其他的含义或目的。小明可能是在表示自己刚刚说的话或做的动作只是随口说说的，没有其他特别的含义。

3.6 问题六：信息搜索

问题描述：简要介绍同济大学

3.6.1 Qwen-7B-Chat 的回答

```
Q6
[9] response, history = model.chat(tokenizer, "简要介绍同济大学", history=None)
print(response)
```

同济大学是一所位于上海市的著名综合类研究型大学，始建于1907年。学校拥有深厚的历史底蕴和独特的教育理念，目前已经成为中国最具影响力的高等学府之一。同济大学有着雄厚的研究实力，在自然科学、工程技术、人文科学和社会科学等领域都取得了显著的成果，并拥有一批世界一流的科研机构。

3.6.2 ChatGLM3-6B 的回答

```
Q6
response, history = model.chat(tokenizer, "简要介绍同济大学", history=None)
print(response)
```

同济大学是一所位于中国上海的著名综合性大学，成立于1907年，原名“同济医学堂”，后发展为今天的综合性大学。同济大学是中国最早成立的工科学校之一，也是中国著名的“工科之源”。

同济大学拥有广泛的学科领域，包括工、理、管、文、法、经、医等七个学科。在工程领域，同济大学是中国最著名的大学之一，特别是在土木工程、机械工程、材料科学、建筑学等方面具有很高的声誉和影响力。在管理、经济、文学、法学等领域，同济大学也有着优秀的表现和声誉。

同济大学拥有一支强大的师资队伍，包括众多国内外知名学者和专家。校园内设施齐全，包括图书馆、实验室、体育场馆等，为学生提供良好的学习和生活条件。

同济大学在国内外享有很高的声誉和影响力，多次被评为中国最佳大学之一，也是中国最具影响力的大学之一。

3.7 问题七：代码能力

问题描述：请用 Python 写一个函数，输入一个整数列表，返回列表中所有偶数的平方和

3.7.1 Qwen-7B-Chat 的回答

```
Q7
[10] response, history = model.chat(tokenizer, "请用Python写一个函数，输入一个整数列表，返回列表中所有偶数的平方和...", history=None)
print(response)

'''python
def sum_of_even_squares(lst):
    return sum(num**2 for num in lst if num % 2 == 0)
'''

这个函数使用了列表推导式和 sum() 函数。首先，它遍历列表中的每个元素，然后检查该元素是否为偶数（即是否能被2整除）。如果是，那么就将该元素的平方添加到总和中。最后，函数返回计算出的总和。

例如，如果你调用 sum_of_even_squares([1, 2, 3, 4, 5])，你会得到结果 20，因为2的平方是4，4的平方是16，这两个偶数的平方加起来就是20。
```

3.7.2 ChatGLM3-6B 的回答

```
response, history = model.chat(tokenizer, "请用Python写一个函数，输入一个整数列表，返回列表中所有偶数的平方和...", history=None)
print(response)

def sum_of_squares_of_evens(numbers):
    return sum([x**2 for x in numbers if x % 2 == 0])

# Example usage:
numbers = [1, 2, 3, 4, 5, 6]
result = sum_of_squares_of_evens(numbers)
print(result) # Output: 55 (2^2 + 4^2 + 6^2 = 55)
```

3.8 问题八：语言翻译

问题描述：请将这句话翻译成英文/日文/德文等：‘科技改变生活，创新引领未来。’

3.8.1 Qwen-7B-Chat 的回答

```
Q8
response, history = model.chat(tokenizer, "请将这句话翻译成英文/日文/德文等：‘科技改变生活，创新引领未来。’", history=None)
print(response)

英文: Technology changes life, innovation leads the future.
日文: テクノロジーは人生を変えることで、イノベーションが未来をリードします。
德文: Technologie verändert das Leben, Innovation führt die Zukunft voran.
```

3.8.2 ChatGLM3-6B 的回答

```
Q8
[20] response, history = model.chat(tokenizer, "请将这句话翻译成英文/日文/德文等：‘科技改变生活，创新引领未来。’", history=None)
print(response)

"Technology changes life, and innovation leads to the future."
"科技改变生活，创新引领未来。" (中文)
"Technology wakes up life, and innovation paves the way for the future." (德文)
```

4 大语言模型之间的横向对比分析

基于以上（问题一至问题八）的测试结果，我们从以下几个维度对 Qwen-7B-Chat 和 ChatGLM3-6B 进行横向对比分析。

4.1 逻辑推理与歧义理解能力

在处理中文特有的歧义和复杂语境方面，两个模型都展现了较强的能力，但侧重点和表达方式有所不同。

- **问题一（穿衣问题）**：两个模型都准确识别了“能穿多少穿多少”在不同季节的歧义。Qwen-7B-Chat 的回答简洁明了，直接点出核心区别和原因。ChatGLM3-6B 的回答同样正确，表述更为结构化。两者理解均到位，Qwen 更口语化，ChatGLM 更书面化。

- **问题二（单身狗原因问题）**：Qwen-7B-Chat 准确地抓住了主语转换带来的含义差异。ChatGLM3-6B 同样识别了歧义，但其解释中对第二句的引申略显主观。Qwen 在细微语义差别上的把握似乎更精准。

- **问题三（“他知道我知道”问题）**：两个模型都没有完美地解决这个经典的逻辑谜题。ChatGLM3-6B 对问题本质的把握（悖论性）更准确一些，而 Qwen-7B-Chat 试图给出一个具体答案但逻辑不够令人信服。这表明在深度逻辑推理方面，两个模型仍有提升空间。

- **问题四（明明白白白问题）**：ChatGLM3-6B 在此问题上表现更优，前半句准确识别了人物关系和句意。Qwen-7B-Chat 的理解出现了偏差，但两个模型对实际谁喜欢谁的回答都不够明确。前者在处理复杂中文语境时表现更好。

- **问题五（“意思”的多义性理解）**：ChatGLM3-6B 能够逐条分析对话中每个“意思”的具体含义，展现了更强的上下文理解和细致分析能力，但对有些“意思”的理解仍然不够准确。Qwen-7B-Chat 的回答则较为泛化，未能针对每一处进行具体分析。

小结：在歧义理解和复杂语境处理上，ChatGLM3-6B 在“明明白白白”和“意思”问题上展现出更强的分析能力和准确性。Qwen 在部分简单歧义问题上表现良好，但在更复杂的场景下（如问题四、五）略逊一筹。逻辑推理方面两者均有进步空间。

4.2 知识广度与事实准确性

通过**问题六（介绍同济大学）**的测试：

- Qwen-7B-Chat 的回答准确，概括了关键信息。
- ChatGLM3-6B 的回答同样准确，并补充了“工科之源”等特色信息，内容更为丰富一些。

对比：两者都能提供准确的基本信息。ChatGLM3-6B 在细节和特色信息的补充上略微丰富一些，显示了较好的知识覆盖。

4.3 创造性与文本生成能力

主要通过**问题七（代码能力）**进行评估：

- Qwen-7B-Chat 生成的 Python 函数代码正确，逻辑清晰，并提供了良好的解释和用法示例。
- ChatGLM3-6B 生成的 Python 函数代码同样正确，也提供了示例。

对比：两个模型都能准确完成代码生成任务。Qwen 的解释和示例更详细一些，用户体验稍好。

4.4 多语言能力

通过问题八（语言翻译）进行测试：

- Qwen-7B-Chat 提供了相对准确的英文、日文、德文翻译。
- ChatGLM3-6B 提供了准确的英文翻译，但在尝试提供“日文”翻译时实际给出了中文、提供“德文”翻译时实际给出了英文，且语义有偏差，未能正确完成多语言翻译任务。

对比：Qwen-7B-Chat 在多语言翻译任务上表现明显优于 ChatGLM3-6B。

4.5 模型概览与基础信息

在对测试结果进行总结分析之前，我们简要介绍本次选用的两款模型的基本情况，以便更好地理解它们的特性和潜在差异。

表 4.1 对比模型基础信息概览

特性	Qwen-7B-Chat	ChatGLM3-6B
开发者	阿里云通义实验室	智谱 AI & 清华 KEG
模型架构	Transformer，优化对话生成和上下文理解	General Language Model (GLM)，联合训练生成和判别任务
参数量	约 70 亿 (7B)	约 60 亿 (6B)
预训练数据	大规模多语言语料库，包含中文、英文及多种其他语言，以及代码数据。	大规模中英文双语语料库，包含万亿级别的中英文标识符。
特点与优化方向	强调在中英文理解、多轮对话、代码生成、插件调用等方面的能力。	强调中英双语能力、更强的推理能力、代码生成以及工具调用、Agent 等能力。
开源情况	模型权重和代码开源	模型权重和代码开源

从基础信息来看，两款模型均为参数量在 60-70 亿级别的大语言模型，这使得它们在 CPU 环境下部署均面临一定的性能挑战。Qwen-7B-Chat 由阿里云研发，而 ChatGLM3-6B 由智谱 AI 与清华大学 KEG 实验室联合研发，代表了国内顶尖的 AI 研究力量。架构上，两者都基于 Transformer，但 ChatGLM 系列采用了其独特的 GLM（General Language Model）架构。参数量上 Qwen 略大一些。预训练数据方面，两者都强调了大规模中英文语料，但 Qwen 可能在多语言覆盖上更广一些。这些基础信息的差异，可能会导致它们在后续的具体任务表现上有所不同。

4.6 综合评价与总结

基于以上各项测试结果以及对模型基础信息的了解，我们对本次重点测试的 Qwen-7B-Chat 和 ChatGLM3-6B 进行综合评价与总结。

• ChatGLM3-6B 的特性与表现：

– **架构与参数：**采用 GLM (General Language Model) 架构，拥有约 60 亿参数。如你提供的参考资料所述，这种架构使其在多语言处理（尤其是中英文混合对话）方面表现出色。同时，其对计算和存储资源的需求相对较低（相较于更大参数模型或一些优化较少的同级别模型）。

– **中英文混合与双语能力：**测试结果（如问题五“意思”的理解）和参考资料均表明，

ChatGLM3-6B 利用大量中英文文本数据，并通过深度学习优化技术提升了双语对话能力。它在理解中文复杂语境、进行结构化回答方面表现突出。

- **任务适用性**：在我们的测试中，ChatGLM3-6B 在文本理解、特定知识问答（如介绍同济大学时信息更丰富）方面表现良好。参考资料也指出其在中英文混合对话、文本生成、问答和对话生成等任务中表现良好，适合跨语言交流和国际化应用。

- **不足**：在我们的测试中，“明明白白”问题理解仍有提升空间，多语言翻译能力存在明显缺陷。逻辑推理能力也有待加强。

• Qwen-7B-Chat 的特性与表现：

- **架构与参数**：基于 Transformer，拥有约 70 亿参数。参数量略大于 ChatGLM3-6B。

- **中文理解与生成**：在我们的测试中，Qwen 在简单的歧义理解（如穿衣问题）、代码生成（解释更友好）方面表现良好。其回答风格偏向自然口语化。通过自注意力机制和数据增强技术提升中文对话的理解和生成能力，特别适合中文环境下的复杂对话任务。

- **多语言与多任务能力**：我们的测试显示 Qwen 在多语言翻译上表现优于 ChatGLM3-6B。其官方也强调了在中英文理解、多轮对话、代码生成等方面的能力。

- **不足**：在部分复杂中文语境（如“明明白白”、“意思”问题）和深度逻辑推理方面，Qwen 的理解准确度不及 ChatGLM3-6B。

• 共性与选择考量：

- **资源需求**：两款模型均为 6B-7B 参数级别，在 CPU 环境下部署均面临显著的性能挑战（内存和速度）。

- 场景适用性：

- * 如果需要处理中英文混合任务，或对结构化、分析性的回答有偏好，ChatGLM3-6B 是一个不错的选择。它在理解中文复杂语义的某些方面（如“意思”问题）表现出了深度。

- * 如果主要处理中文对话和生成任务，特别是在需要更自然口语化表达、代码生成辅助或初步多语言翻译的场景，Qwen-7B-Chat 可能更为适用。

- **持续优化**：大语言模型技术仍在快速发展，无论是 ChatGLM、Qwen，都在不断迭代和优化。未来的版本可能会在当前的短板上有所改进。

总而言之，通过本次对 Qwen-7B-Chat 和 ChatGLM3-6B 的实际部署测试，我们可以看到不同模型在设计理念、训练数据侧重和优化方向上的差异，导致了它们在具体任务表现上的各有千秋。没有绝对的“最佳”模型，只有在特定场景下“更合适”的模型。

5 总结与展望

通过本次作业，我成功在魔搭平台上将 Qwen-7B-Chat 和 ChatGLM3-6B 模型克隆至 DSW 环境并进行了部署与测试。这个过程让我深入了解了开源大语言模型从获取、配置到实际运行的完整流程，并直观体验了它们在 CPU 环境下的性能表现。

在对比分析中，我发现这两款模型在中文理解的深度、逻辑推理的严谨性、知识覆盖的广度以及特定任务（如代码生成、多语言翻译）的处理能力上各有侧重。例如，ChatGLM3-6B 在理解中文多重“意思”的细微差别上表现优异，而 Qwen-7B-Chat 则在多语言翻译和代码生成的解释友好性上更胜一筹。这也说明了当前大语言模型领域百花齐放，不同模型针对不同优化方向和训练数据，会展现出独特的优势和局限性。

本次实践不仅锻炼了我的动手能力和问题排查能力（如处理依赖冲突、理解模型加载瓶颈），也极大地加深了我对大语言模型技术现状、能力边界和未来挑战的理解。尤其是 CPU 部署大型模型时遇到的内存和速度瓶颈，让我更深刻地体会到高效推理优化和硬件资源的重要性。

未来，我希望能够：

（1）**探索模型量化与推理优化技术**：学习如 INT8/INT4 量化、ONNX Runtime、TensorRT 等技术，以期在有限资源下提升大模型的推理效率；

（2）**尝试模型微调 (Fine-tuning)**：在特定任务或领域数据上对现有开源模型进行微调，以提升其在专门场景下的表现；

（3）**关注多模态大模型的发展**：图文理解与生成将是未来重要的发展方向，希望能有机会进行学习和实践；

（4）**持续跟踪前沿研究**：关注大模型在可解释性、安全性、减少幻觉等方面的最新研究进展，理解技术发展的趋势。

总而言之，这次大语言模型部署实验是一次宝贵的学习经历，为我后续在人工智能领域的深入学习和探索打下了坚实的基础。

A 项目公开可访问链接

本项目在魔搭平台上的公开可访问链接如下：<https://www.modelscope.cn/models/C10H15N/C10H15N/summary>.

本项目在 GitHub 上的公开可访问链接如下：<https://github.com/oC10H15No/Tongji-SE-project-AI-25Spring>.