

chatglm-4模型微调

作者：韦旭

开发方案：MindSpore+MindNLP下的chatglm-4模型微调（指令微调-文本分类）
对比Pytorch+NPU

1.硬件

1.1 MindSpore框架硬件测试基于启智社区：

NPU: 1*Ascend-D910B(显存: 64GB), CPU: 24,内存: 192GB

1.2 Pytorch+GPU:

GPU: 1*A100(显存: 40GB), CPU: 8, 内存: 50GB

2.模型与数据集

2.1模型

使用**GLM-4-9B-Chat** ([THUDM/glm-4-9b-chat · Hugging Face](https://huggingface.co/THUDM/glm-4-9b-chat))

启智社区url: [FoundationModel/THUDM - THUDM - OpenI - 启智AI开源社区提供普惠算力!](#)

2.2数据集

本次finetune使用的是**zh_cls_fudan-news**数据集，该数据集主要被用于训练文本分类模型。

zh_cls_fudan-news由几千条数据，每条数据包含text、category、output三列：

- text 是训练语料，内容是书籍或新闻的文本内容
- category 是text的多个备选类型组成的列表
- output 则是text唯一真实的类型

3.测试脚本

先前NPU上的测试脚本基于MindTorch转化，不过效果不佳，因此后续的测试脚本直接基于MindNLP编写（见附）

4.对比评估损失

4.1 基于GPU+Pytorch上的结果对比基于NPU+MindSpore上的结果，由于数据过多，本表格仅节选

Step	Loss of GPU+Pytorch	Loss of NPU+MindSpore
10	1.253500	3.384100
20	0.384100	0.277700
30	0.051100	0.104400
40	0.04400	0.025700

5.结论

在NPU环境下，MindSpore框架能够充分利用Ascend芯片的硬件优势，实现高效的模型训练和优化。

另附1: MindSpore加载ChatGLM4遇到的Bug（附解决，原创）

<https://blog.csdn.net/shee33/article/details/146317381>

另附2: MindNLP-ChatGLM4调试脚本

```
import os
import json
import pandas as pd
import numpy as np
import mindspore
from mindspore import nn, Tensor
from mindspore.common.initializer import Normal
from mindspore.train import Model, LossMonitor, TimeMonitor, Callback
from mindspore.train.callback import CheckpointConfig, ModelCheckpoint
from mindspore.nn import AdamWeightDecay
from mindspore.dataset import GeneratorDataset
```

```

from mindnlp.transformers import AutoTokenizer, AutoModelForCausalLM
from mindspore import context
from mindnlp.core import no_grad

# 设置上下文，使用Ascend设备
context.set_context(device_target="Ascend", device_id=0)

# 输出结果必须保存在该目录
output_path = "./output"

def dataset_jsonl_transfer(origin_path, new_path):
    messages = []

    # 读取旧的JSONL文件
    with open(origin_path, "r") as file:
        for line in file:
            # 解析每一行的json数据
            data = json.loads(line)
            context_data = data["text"]
            category = data["category"]
            label = data["output"]
            message = {
                "instruction": "你是一个文本分类领域的专家，你会接收到一段文本和几个潜在的分类选项，请输出文本内容的正确类型",
                "input": f"文本:{context_data},类型选型:{category}",
                "output": label,
            }
            messages.append(message)

    with open(new_path, "w", encoding="utf-8") as file:
        for message in messages:
            file.write(json.dumps(message, ensure_ascii=False) + "\n")

def process_func(example, tokenizer):
    MAX_LENGTH = 384
    input_ids, attention_mask, labels = [], [], []
    instruction = tokenizer(
        f"<|system|>\n你是一个文本分类领域的专家，你会接收到一段文本和几个

```

```

潜在的分类选项，请输出文本内容的正确类型<|endoftext|>\n<|user|>\n{example['input']}<|endoftext|>\n<|assistant|>\n",
    add_special_tokens=False,
)
response = tokenizer(f"{example['output']}", add_special_tokens=False)
input_ids = instruction["input_ids"] + response["input_ids"] + [tokenizer.pad_token_id]
attention_mask = (
    instruction["attention_mask"] + response["attention_mask"] + [1]
)
labels = [-100] * len(instruction["input_ids"]) + response["input_ids"] + [tokenizer.pad_token_id]
if len(input_ids) > MAX_LENGTH: # 做一个截断
    input_ids = input_ids[:MAX_LENGTH]
    attention_mask = attention_mask[:MAX_LENGTH]
    labels = labels[:MAX_LENGTH]
return {"input_ids": input_ids, "attention_mask": attention_mask, "labels": labels}

# 加载模型和分词器
model_name = "ZhipuAI/glm-4-9b-chat"
tokenizer = AutoTokenizer.from_pretrained(model_name, mirror='modelscope')
model = AutoModelForCausalLM.from_pretrained(
    model_name,
    mirror='modelscope',
    ms_dtype=mindspore.float16
)
train_dataset_path = "train.jsonl"
test_dataset_path = "test.jsonl"

train_jsonl_new_path = "new_train.jsonl"
test_jsonl_new_path = "new_test.jsonl"

if not os.path.exists(train_jsonl_new_path):
    dataset_jsonl_transfer(train_dataset_path, train_jsonl_new_path)
if not os.path.exists(test_jsonl_new_path):
    dataset_jsonl_new_path(test_dataset_path, test_jsonl_new_path)

```

```

# 得到训练集
train_df = pd.read_json(train_jsonl_new_path, lines=True)
train_df = train_df[:1000]

# 数据预处理
train_data = train_df.to_dict(orient='records')
processed_train_data = []
for example in train_data:
    processed_example = process_func(example, tokenizer)
    processed_train_data.append(processed_example)

# 转换为适合 MindSpore 的格式
def data_generator(dataset):
    for item in dataset:
        yield (
            np.array(item["input_ids"], dtype=np.int32),
            np.array(item["attention_mask"], dtype=np.int32),
            np.array(item["labels"], dtype=np.int32)
        )

train_dataset = GeneratorDataset(source=lambda: data_generator(processed_train_data),
                                column_names=["input_ids", "attention_mask", "labels"])

def forward_fn(input_ids, attention_mask, labels):
    outputs = model(input_ids=input_ids, attention_mask=attention_mask, labels=labels)
    loss = outputs.loss
    return loss

params_dict = model.parameters_dict()
trainable_params = [param for param in params_dict.values() if param.requires_grad]

```

```

optimizer = AdamWeightDecay(unique_params, learning_rate=3e-4)

train_net = mindspore.nn.TrainOneStepCell(nn.WithLossCell(model, nn.CrossEntropyLoss()), optimizer)

class EvalCallback(Callback):
    def __init__(self, model, dataset, tokenizer):
        super(EvalCallback, self).__init__()
        self.model = model
        self.dataset = dataset
        self.tokenizer = tokenizer

    def epoch_end(self, run_context):
        cb_params = run_context.original_args()
        eval_results = self.model.eval(self.dataset)
        print(f"Epoch {cb_params.cur_epoch_num} eval results: {eval_results}")

# 创建模型

model = Model(model, loss_fn=nn.CrossEntropyLoss(), optimizer=optimizer, metrics={'loss'})

config_ck = CheckpointConfig(save_checkpoint_steps=50, keep_checkpoint_max=5)
ckptpoint_cb = ModelCheckpoint(prefix="glm4-9b", directory=output_path, config=config_ck)

model.train(epoch_size=1, train_dataset=train_dataset, callbacks=[TimeMonitor(), LossMonitor(), ckptpoint_cb])

test_df = pd.read_json(test_jsonl_new_path, lines=True)[:10]
test_data = test_df.to_dict(orient='records')

processed_test_data = []
for example in test_data:
    processed_example = process_func(example, tokenizer)

```

```
processed_test_data.append(processed_example)

test_dataset = GeneratorDataset(source=lambda: data_generator(processed_test_data),
                                column_names=["input_ids", "attention_mask", "labels"])

eval_results = model.eval(test_dataset)
print(f"评估结果: {eval_results}")
```