怎么让英文大语言模型支持中文？（二）—— 继续预训练篇

来自： AiGC面试宝典

宁静致远

2023年09月29日 12:28

一、为什么需要进行继续预训练？

前面我们已经讲过怎么构建中文领域的tokenization：

接下来我们将介绍继续预训练。

我们**新增加了一些中文词汇到词表中，这些词汇是没有得到训练的，因此在进行指令微调之前我们要进行预训练。**预训练的方式一般都是相同的，简单来说，就是根据上一个字预测下一个字是什么。为了方便起见，我们这里直接使用IDEA-CCNL/Wenzhong2.0-GPT2-110M-BertTokenizer-chinese模型，并且tokenizer也是其自带的。

二、如何对 继续预训练 数据预处理？

同样的，我们使用的数据还是斗破苍穹小说数据。首先我们看看是怎么处理数据的， 数据位于data下，分别为corpus.txt和test_corpus.txt，每一行为一句或多句话。再看看数据预处理的部分，在test_dataset.py里面：

```
import os
import logging
import datasets
import transformers

from pprint import pprint
from itertools import chain
from datasets import load_dataset, concatenate_datasets
from transformers.testing_utils import CaptureLogger
from transformers import AutoTokenizer, LlamaTokenizer

tok_logger =
transformers.utils.logging.get_logger("transformers.tokenization_utils_base")

logger = logging.getLogger(__name__)

lm_datasets = []
files = ["data/test_corpus.txt"]
data_cache_dir = "./cache_data"
preprocessing_num_workers = 1

# tokenizer = AutoTokenizer.from_pretrained("hfl/chinese-bert-wwm-ext")
tokenizer = LlamaTokenizer.from_pretrained("ziqingyang/chinese-llama-lora-7b")
tokenizer = AutoTokenizer.from_pretrained("IDEA-CCNL/Wenzhong2.0-GPT2-110M-
BertTokenizer-chinese")

def print_dict(adict):
  for k,v in adict.items():
    print(k, v)
```

```
def tokenize_function(examples):
        with CaptureLogger(tok_logger) as cl:
            output = tokenizer(examples["text"])
        # clm input could be much much longer than block_size
        if "Token indices sequence length is longer than the" in cl.out:
            tok_logger.warning(
                "^^^^^^^^^^^^^^^^^^^^ Please ignore the warning above - this long input
will be chunked into smaller bits"
                " before being passed to the model."
            )
        return output

block_size = 128

# 将所有文本进行拼接
def group_texts(examples):
        # Concatenate all texts.
        concatenated_examples = {k: list(chain(*examples[k])) for k in
examples.keys()}
        total_length = len(concatenated_examples[list(examples.keys())[0]])
        # We drop the small remainder, we could add padding if the model supported
it instead of this drop, you can
        # customize this part to your needs.
        if total_length >= block_size:
            total_length = (total_length // block_size) * block_size
        # Split by chunks of max_len.
        result = {
            k: [t[i : i + block_size] for i in range(0, total_length, block_size)]
            for k, t in concatenated_examples.items()
        }
        result["labels"] = result["input_ids"].copy()
        return result

for idx, file in enumerate(files):
    data_file = file
    filename = ''.join(file.split(".")[:-1])

    cache_path = os.path.join(data_cache_dir, filename)
    os.makedirs(cache_path, exist_ok=True)
    try:
        processed_dataset = datasets.load_from_disk(cache_path,
keep_in_memory=False)
        print(f'training datasets-{filename} has been loaded from disk')
    except Exception:
        cache_dir = os.path.join(data_cache_dir, filename + "_text")
        os.makedirs(cache_dir, exist_ok=True)
```

```
        raw_dataset = load_dataset("text", data_files=data_file,
cache_dir=cache_dir, keep_in_memory=False)
        print_dict(raw_dataset["train"][0])
        # 直接进行tokenize，需要注意的是只需要在句子开头加上bos_token
        tokenized_dataset = raw_dataset.map(
            tokenize_function,
            batched=True,
            num_proc=preprocessing_num_workers,
            remove_columns="text",
            load_from_cache_file=True,
            keep_in_memory=False,
            cache_file_names={k: os.path.join(cache_dir, f'tokenized.arrow') for k
in raw_dataset},
            desc="Running tokenizer on dataset",
        )

        print_dict(tokenized_dataset["train"][0])

        grouped_datasets = tokenized_dataset.map(
            group_texts,
            batched=True,
            num_proc=preprocessing_num_workers,
            load_from_cache_file=True,
            keep_in_memory=False,
            cache_file_names={k: os.path.join(cache_dir, f'grouped.arrow') for k in
tokenized_dataset},
            desc=f"Grouping texts in chunks of {block_size}",
        )
        processed_dataset = grouped_datasets

        print_dict(processed_dataset["train"][0])
        processed_dataset.save_to_disk(cache_path)
    if idx == 0:
        lm_datasets = processed_dataset['train']
    else:
        assert lm_datasets.features.type ==
processed_dataset["train"].features.type
        lm_datasets = concatenate_datasets([lm_datasets,
processed_dataset["train"]])

lm_datasets = lm_datasets.train_test_split(test_size=0.1)

print_dict(lm_datasets["train"][0])
```

输出

```
text 又一次上架了，这次比上次还激动，甚至激动到了上传了章节却不知道发出来的地步。
```

```
input_ids [21134, 1348, 671, 3613, 677, 3373, 749, 8024, 6821, 3613, 3683, 677,
3613, 6820, 4080, 1220, 8024, 4493, 5635, 4080, 1220, 1168, 749, 677, 837, 749,
4995, 5688, 1316, 679, 4761, 6887, 1355, 1139, 3341, 4638, 1765, 3635, 511, 21133]
token_type_ids [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
attention_mask [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
input_ids [21134, 1348, 671, 3613, 677, 3373, 749, 8024, 6821, 3613, 3683, 677,
3613, 6820, 4080, 1220, 8024, 4493, 5635, 4080, 1220, 1168, 749, 677, 837, 749,
4995, 5688, 1316, 679, 4761, 6887, 1355, 1139, 3341, 4638, 1765, 3635, 511, 21133,
21134, 2219, 2217, 8024, 1068, 754, 3173, 741, 8024, 677, 3373, 1184, 2768, 5327,
1962, 2533, 3300, 763, 1139, 725, 1759, 6486, 4638, 2692, 3160, 8024, 2190, 754,
6821, 819, 1331, 4798, 4638, 2768, 5327, 8024, 1759, 6486, 2552, 7027, 6820, 4696,
3300, 1126, 1146, 2684, 2607, 680, 2558, 2559, 8024, 6006, 6432, 3295, 5307, 3300,
782, 6432, 1759, 6486, 3221, 1170, 1139, 3341, 4638, 3144, 2945, 8024, 2190, 754,
6821, 763, 4522, 6241, 8024, 2769, 738, 2400, 3313, 1922, 6814, 1762, 2692, 8024,
1166, 4638, 2769, 679]
token_type_ids [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
attention_mask [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
labels [21134, 1348, 671, 3613, 677, 3373, 749, 8024, 6821, 3613, 3683, 677, 3613,
6820, 4080, 1220, 8024, 4493, 5635, 4080, 1220, 1168, 749, 677, 837, 749, 4995,
5688, 1316, 679, 4761, 6887, 1355, 1139, 3341, 4638, 1765, 3635, 511, 21133, 21134,
2219, 2217, 8024, 1068, 754, 3173, 741, 8024, 677, 3373, 1184, 2768, 5327, 1962,
2533, 3300, 763, 1139, 725, 1759, 6486, 4638, 2692, 3160, 8024, 2190, 754, 6821,
819, 1331, 4798, 4638, 2768, 5327, 8024, 1759, 6486, 2552, 7027, 6820, 4696, 3300,
1126, 1146, 2684, 2607, 680, 2558, 2559, 8024, 6006, 6432, 3295, 5307, 3300, 782,
6432, 1759, 6486, 3221, 1170, 1139, 3341, 4638, 3144, 2945, 8024, 2190, 754, 6821,
763, 4522, 6241, 8024, 2769, 738, 2400, 3313, 1922, 6814, 1762, 2692, 8024, 1166,
4638, 2769, 679]
input_ids [21134, 1348, 671, 3613, 677, 3373, 749, 8024, 6821, 3613, 3683, 677,
3613, 6820, 4080, 1220, 8024, 4493, 5635, 4080, 1220, 1168, 749, 677, 837, 749,
4995, 5688, 1316, 679, 4761, 6887, 1355, 1139, 3341, 4638, 1765, 3635, 511, 21133,
21134, 2219, 2217, 8024, 1068, 754, 3173, 741, 8024, 677, 3373, 1184, 2768, 5327,
1962, 2533, 3300, 763, 1139, 725, 1759, 6486, 4638, 2692, 3160, 8024, 2190, 754,
6821, 819, 1331, 4798, 4638, 2768, 5327, 8024, 1759, 6486, 2552, 7027, 6820, 4696,
3300, 1126, 1146, 2684, 2607, 680, 2558, 2559, 8024, 6006, 6432, 3295, 5307, 3300,
782, 6432, 1759, 6486, 3221, 1170, 1139, 3341, 4638, 3144, 2945, 8024, 2190, 754,
6821, 763, 4522, 6241, 8024, 2769, 738, 2400, 3313, 1922, 6814, 1762, 2692, 8024,
1166, 4638, 2769, 679]
```

```
token_type_ids [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
attention_mask [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
labels [21134, 1348, 671, 3613, 677, 3373, 749, 8024, 6821, 3613, 3683, 677, 3613,
6820, 4080, 1220, 8024, 4493, 5635, 4080, 1220, 1168, 749, 677, 837, 749, 4995,
5688, 1316, 679, 4761, 6887, 1355, 1139, 3341, 4638, 1765, 3635, 511, 21133, 21134,
2219, 2217, 8024, 1068, 754, 3173, 741, 8024, 677, 3373, 1184, 2768, 5327, 1962,
2533, 3300, 763, 1139, 725, 1759, 6486, 4638, 2692, 3160, 8024, 2190, 754, 6821,
819, 1331, 4798, 4638, 2768, 5327, 8024, 1759, 6486, 2552, 7027, 6820, 4696, 3300,
1126, 1146, 2684, 2607, 680, 2558, 2559, 8024, 6006, 6432, 3295, 5307, 3300, 782,
6432, 1759, 6486, 3221, 1170, 1139, 3341, 4638, 3144, 2945, 8024, 2190, 754, 6821,
763, 4522, 6241, 8024, 2769, 738, 2400, 3313, 1922, 6814, 1762, 2692, 8024, 1166,
4638, 2769, 679]
```

具体是:

1. 先使用tokenizer()得到相关的输入，需要注意的是可能会在文本前后添加特殊的标记，比如bos_token_id和eos_token_id，针对于不同模型的tokneizer可能会不太一样。这里在unput_ids前后添加了21134和21133两个标记。
2. 然后将所有文本的input_ids、attention_mask, token_type_ids各自拼接起来（展开后拼接，不是二维数组之间的拼接），再设定一个最大长度block_size，这样得到最终的输入。

三、如何 构建模型?

在test_model.py里面我们可以初步使用预训练的模型看看效果:

```
from transformers import BertTokenizer,GPT2LMHeadModel, AutoModelForCausalLM
hf_model_path = 'IDEA-CCNL/Wenzhong2.0-GPT2-110M-BertTokenizer-chinese'
tokenizer = BertTokenizer.from_pretrained(hf_model_path)
# model = GPT2LMHeadModel.from_pretrained(hf_model_path)
model = AutoModelForCausalLM.from_pretrained(hf_model_path)


def generate_word_level(input_text,n_return=5,max_length=128,top_p=0.9):
    inputs =
tokenizer(input_text,return_tensors='pt',add_special_tokens=False).to(model.device)
    gen = model.generate(
                            inputs=inputs['input_ids'],
                            max_length=max_length,
                            do_sample=True,
                            top_p=top_p,
                            eos_token_id=21133,
                            pad_token_id=0,
                            num_return_sequences=n_return)
```

```
    sentences = tokenizer.batch_decode(gen)
    for idx,sentence in enumerate(sentences):
        print(f'sentence {idx}: {sentence}')
        print('*'*20)
    return gen
# 西湖的景色
outputs = generate_word_level('西湖的景色',n_return=5,max_length=128)
print(outputs)
```

输出

```
sentence 0: 西湖的景色很美丽，古代有个名叫：西湖的湖南和江南的一段。湖面上有一座小小的湖泊，有一片湖泊和一座小岛，有一处小的小镇。在西湖里，每个人都是在湖边，你可以在小小湖里畅游。西湖上是古代建筑，但湖水不多。西湖上是一座水库，古代有个名叫：西湖的湖南和江南的一段。湖
********************
sentence 1: 西湖的景色美不胜数。近日，位于湖北省湖北省石家庄市的石家庄旅游风景区被命名为"湖北省国家级森林公园"。园内有一座石屋，位于石屋与石屋的对面，总面积3.2平方公里，其中一座石屋，由石屋和石屋组成，一栋大型石屋由石屋组成，三栋石屋由石屋组成。石屋主要是一座石屋
********************
sentence 2: 西湖的景色在古城、小镇和城郊中，有大片的湖泊，是古典中的佳肴，湖水清澈，湖中有一大块鱼，在湖水里散发着浓郁的清香。湖水中，有各种颜色的鱼、蟹、贝壳类的水产品。湖边有的池塘也有的水果摊位，可供上千家店。在湖中央的湖中央有三个小水塘，水塘长约三丈，两端长，塘底
********************
sentence 3: 西湖的景色也很漂亮，可以说是城市的象征，而且还有小小的山洞，看到了，我们在西湖的中心也很近，所以也没有停止，西湖的风景很秀美，我们也不愿意停留在这样的地方。西湖是世界上最美的湖泊，也是最令人羡慕的旅游区，西湖的美丽不容小视，是我们心中最美的风景。西湖在西湖
********************
sentence 4: 西湖的景色是如此独特，那水碧草如黛，池水清新，一池青湖，游人可以品一小池花。""好景如画，山清水秀，碧草如茵，池清潭秀。"黄湖"是西湖的"绿色湖"。西湖的景色是如此独特，那水碧草如黛，池水清新，一池青湖，游人可以品一小池花。""好景如画，山清水秀，碧草如茵
********************
```

接下来是使用该模型针对我们自己的数据进行继续预训练了。需要注意的几个地方：

1. 如果是我们自己定义的tokenizer，需要将模型的嵌入层和lm_head层的词表数目进行重新设置：

```
model_vocab_size = model.get_output_embeddings().weight.size(0)
model.resize_token_embeddings(len(tokenizer))
```

2. 这里我们使用参数有效微调方法lora进行微调，我们需要设置额外保存的参数：transformer.wte,lm_head。这个可以通过find_lora_names.py里面获得。

3. 原始chinsee-llama-alpaca使用lora保存参数有问题，这里进行了修改并只保存一份lora权重。

4. 使用test_pretrained_model.py的时候也要记得先对vocab_size进行重新设置。

```
    $ torchrun --nnodes 1 --nproc_per_node 1 run_clm_pt_with_peft.py --deepspeed
ds_zero2_no_offload.json --model_name_or_path IDEA-CCNL/Wenzhong2.0-GPT2-110M-
BertTokenizer-chinese --tokenizer_name_or_path IDEA-CCNL/Wenzhong2.0-GPT2-110M-
BertTokenizer-chinese --dataset_dir data --data_cache_dir temp_data_cache_dir --
validation_split_percentage 0.001 --per_device_train_batch_size 32 --
per_device_eval_batch_size 16 --do_train --seed $RANDOM --fp16 --max_steps 2500 --
lr_scheduler_type cosine --learning_rate 2e-4 --warmup_ratio 0.05 --weight_decay
0.01 --logging_strategy steps --logging_steps 10 --save_strategy steps --
save_total_limit 3 --save_steps 50 --gradient_accumulation_steps 1 --
preprocessing_num_workers 8 --block_size 512 --output_dir output_dir --
overwrite_output_dir --ddp_timeout 30000 --logging_first_step True --lora_rank 8 --
lora_alpha 32 --trainable c_attn --modules_to_save transformer.wte,lm_head --
lora_dropout 0.05 --torch_dtype float16 --gradient_checkpointing --
ddp_find_unused_parameters False
```

即:

```
torchrun --nnodes 1 --nproc_per_node 1 run_clm_pt_with_peft.py \
--deepspeed ds_zero2_no_offload.json \
--model_name_or_path IDEA-CCNL/Wenzhong2.0-GPT2-110M-BertTokenizer-chinese \
--tokenizer_name_or_path IDEA-CCNL/Wenzhong2.0-GPT2-110M-BertTokenizer-chinese \
--dataset_dir data \
--data_cache_dir temp_data_cache_dir \
--validation_split_percentage 0.001 \
--per_device_train_batch_size 32 \
--per_device_eval_batch_size 16 \
--do_train --seed $RANDOM \
--fp16 \
--max_steps 2500 \
--lr_scheduler_type cosine \
--learning_rate 2e-4 \
--warmup_ratio 0.05 \
--weight_decay 0.01 \
--logging_strategy steps \
--logging_steps 10 \
--save_strategy steps \
--save_total_limit 3 \
--save_steps 50 \
--gradient_accumulation_steps 1 \
--preprocessing_num_workers 8 \
--block_size 512 \
--output_dir output_dir \
--overwrite_output_dir \
--ddp_timeout 30000 \
--logging_first_step True \
--lora_rank 8 \
```

```
--lora_alpha 32 \
--trainable c_attn \
--modules_to_save transformer.wte,lm_head \
--lora_dropout 0.05 \
--torch_dtype float16 \
--gradient_checkpointing \
--ddp_find_unused_parameters False
```

由于使用了seepspeed中ZeRo，占用的显存会更小。

四、如何 使用模型?

最后我们可以这么使用模型，在test_pretrained_model.py中：

```
import os
import torch
from transformers import BertTokenizer,GPT2LMHeadModel, AutoModelForCausalLM
from peft import PeftModel
hf_model_path = 'IDEA-CCNL/Wenzhong2.0-GPT2-110M-BertTokenizer-chinese'
tokenizer = BertTokenizer.from_pretrained(hf_model_path)
# model = GPT2LMHeadModel.from_pretrained(hf_model_path)
model = AutoModelForCausalLM.from_pretrained(hf_model_path)

model_vocab_size = model.get_output_embeddings().weight.size(0)
model.resize_token_embeddings(len(tokenizer))

model = PeftModel.from_pretrained(model, os.path.join("output_dir",
"adapter_model"), torch_dtype=torch.float32)
model.cuda()
model.eval()

def generate_word_level(input_text,n_return=5,max_length=128,top_p=0.9):
    inputs =
tokenizer(input_text,return_tensors='pt',add_special_tokens=False).to(model.device)
    gen = model.generate(
                            inputs=inputs['input_ids'],
                            max_length=max_length,
                            do_sample=True,
                            top_p=top_p,
                            eos_token_id=21133,
                            pad_token_id=0,
                            num_return_sequences=n_return)

    sentences = tokenizer.batch_decode(gen)
    for idx,sentence in enumerate(sentences):
        print(f'sentence {idx}: {sentence}')
        print('*'*20)
    return gen
```

```
outputs = generate_word_level('眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎',n_return=5,max_length=128)
print(outputs)
```

output

```
sentence 0: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 淡 淡 的 道 。 <|endoftext|> [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
******************
sentence 1: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 一 怔 。 手 掌 猛 然 一 僵 。 手 指 一 扯 。 旋 即 在 房 门 内 停 留 。 旋 即 一 口 鲜 血 喷 涌 而 出 。 <|endoftext|>
******************
sentence 2: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 顿 时 愣 了 愣 。 他 这 是 何 人 ? 怎 能 知 道 这 位 灰 袍 老 者 出 手 啊 ? <|endoftext|> [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
******************
sentence 3: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 心 中 有 着 什 么 感 触 ? <|endoftext|> [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
******************
sentence 4: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 微 皱 着 眉 头 。 转 过 身 。 轻 声 道 : “ 柳 翎 。 是 你 的 人 ? ” <|endoftext|> [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD] [PAD]
******************
```

对于没有经过继续预训练的模型结果:

```
sentence 0: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎, 男, 1964 年 生, 河 北 齐 齐 哈 尔 市 人 。 1979 年 毕 业 于 武 汉 工 学 院 中 文 系, 1988 年 毕 业 于 中 国 人 民 大 学 中 文 系, 历 任 中 国 人 民 大 学 高 级 教 师 、 教 育 部 大 学 文 学 系 主 任, 中 国 语 言 文 学 会 理 事, 中 国 人 民 大 学 历 史 学 会 副 会 长, 中 国 作 家 协 会 员, 中 国 作 家 协 会 会
******************
sentence 1: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 的 脸 庞 在 不 同 时 期 会 发 出 来 , 这 样 的 眉 目 和 眉 目 能 够 很 容 易 的 在 一 起 , 能 够 让 人 看 得 见 的 就 是 这 样 的 眉 目 。 那 一 对 情 侣 还 是 非 常 喜 欢 的 , 不 过 他 们 的 交 往 方 式 也 是 各 种 多 样 的 , 最 后 的 交 往 方 式 就 是 让 所 有 的 人 都 看 到 了 自 己 的 内 心 。 他 们 俩 是 非 常 相
******************
sentence 2: 眼 角 斜 瞥 着 柳 翎 那 略 微 有 些 阴 沉 的 脸 庞 。 萧 炎 眼 睛 看 向 柳 翎, 眼 眸 里 满 是 伤 痕 。 “ 天 边 来 客 。 ” 柳 翎 那 无 情 的 目 光 中 透 着 几 分 冷 漠 的 微 笑 。 “ 没 有 你 的 名 字, 你 只 是 名 字 。 ” 柳 翎 在 柳 翎 眼 前 一 怔, 无 意 中 却 看 出 了 柳 翎 已 经 在 想 要 离 开 了 。 柳 翎 说 这 些 东 西 有 的 是 一 次 次 的 意 外, 她 还 是 有 意 的,
******************
```

sentence 3: 眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎的脸上只有几分阴沉，但却能够带着微微的怜惜之心。萧炎眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎眼角

****

sentence 4: 眼角斜瞥着柳翎那略微有些阴沉的脸庞。萧炎已经是年轻貌美的人，在某处留下的是无尽的光影。她的微笑也在耳畔闪烁着光影。他不断地伸出手指，他在他的微笑中轻松地走着，而柳翎却始终沉默。他已经是个女孩子，在某处也许你听不见。他轻轻地接过他的手，轻轻地说道：〞没有人听

****

模型确实得到了有效的训练。