

WAIC-黑客松：行情波动下的金融问答挑战赛

[数据描述与数据预处理](#)

[模型训练](#)

[如何推断](#)

[实验环境和结果](#)

写在最后

[联系作者](#)

[项目文件路径](#)

[其他说明文件](#)

WAIC-黑客松：行情波动下的金融问答挑战赛

数据描述与数据预处理

原始数据：

```
1  {
2      {
3          'content':...,
4          'summary':...,
5          'qdomain':...,
6          'adomain':...,
7          'retrieval':{
8              'search_news':[{'score':..., 'title':..., 'content':...},...],
9              'search_point':
10             [{'score':..., 'title':..., 'content':...},...],
11             'search_xueqiu':
12             [{'score':..., 'title':..., 'content':...},...],
13             'search_qa':[{'score':..., 'answer':..., 'query':...},...]
14         }
15     },
16     {...},
17     ...
18 }
```

将`retrieval.search_qa`中相关性最高的问题对应的答案拼接到原问题答案中，即

$$summary = summary + retrieval.search_qa[0].answer$$

注意：`retrieval.search_qa`中的问题按照相关性递减排列，因此第一个问题就是最相关的问题

代码实现：详见 `dataprep.py` 文件

```
1 | python dataprep.py
```

得到预处理后的文件 `train_.json`, `test_.json`, `valid_.json`.

模型训练

采用在中文语料上预训练过的bart作为基座模型，并通过huggingface作为接口。（详见 `train.py` 文件）

具体步骤如下：

1. 首先新建文件夹，并命名为 `datasets`，将上述预处理好数据文件 `train_.json`, `test_.json`, `valid_.json` 放入其中（文件夹命名必须需要与 `train.py` 中的文件路径对应，此处以 `dataset` 为例）
2. 首先将上文中预处理好的数据转化为huggingface内置的 `datasets` 类

```
1 python3 run_mybart.py --model_name_or_path fnlp-bart-base \  
2 --train_file ./dataset/train_.json \  
3 --validation_file ./dataset/valid_.json \  
4 --test_file ./dataset/test_.json \  
5 --output_dir output \  
6 --exp_name waic \  
7 --max_source_length 512 \  
8 --max_target_length 256 \  
9 --chinese_data True
```

参数的意义如下所示：

```
1 --model_name_or_path: 基座模型的名称（对应huggingface中model模块的命名），或者是本地  
   模型的保存路径  
2 --train_file      : 预处理后的训练数据  
3 --valid_file      :  
4 --test_file       :  
5 --output_dir      : 输出的文件夹，可随便指定  
6 --max_source_length : 最大输入长度（可截断）  
7 --max_target_length : 最大输出长度  
8 --chinese_data     : 是否使用中文数据
```

注意：如果缺少comet_ml，需要手动pip安装

实际运行结果：

```
[...]$ python train.py \  
> --model_name_or_path 'HuggingFaceModel/fnlp-bart-base \  
> --train_file ./datasets/train_.json \  
> --validation_file ./datasets/valid_.json \  
> --test_file ./datasets/test_.json \  
> --output_dir output \  
> --exp_name waic \  
> --max_source_length 512 --max_target_length 256 \  
> --chinese_data True
```

处理之后

```
All the weights of MyBart were initialized from the model checkpoint at /data/shizhengliang-slurm/HuggingFaceModel/fnl-p-bart-base.
If your task is similar to the task the model of the checkpoint was trained on, you can already use MyBart for predictions without fu
rther training.
08/27/2022 09:17:52 - INFO - dataset_maker - ***** Making Dataset *****
Using custom data configuration default
Downloading and preparing dataset json/default-300a9f724eb172e6 (download: Unknown size, generated: Unknown size, post-processed: Unk
nown size, total: Unknown size) to /data/shizhengliang-slurm/.cache/huggingface/datasets/json/default-300a9f724eb172e6/0.0.0/70d89ed4
db1394f028c651589fcab6d6b28dddcabbe39d3b21b4d41f9a708514...
HF google storage unreachable. Downloading and preparing it from source
Dataset json downloaded and prepared to /data/shizhengliang-slurm/.cache/huggingface/datasets/json/default-300a9f724eb172e6/0.0.0/70d
89ed4db1394f028c651589fcab6d6b28dddcabbe39d3b21b4d41f9a708514. Subsequent calls will reuse this data.
Using eos_token, but it is not set yet.
100%|██████████████████████████████████████████████████████████████████████████| 101/101 [00:32<00:00, 3.09ba/s]
100%|██████████████████████████████████████████████████████████████████████████| 2/2 [00:00<00:00, 4.82ba/s]
100%|██████████████████████████████████████████████████████████████████████████| 3/3 [00:00<00:00, 17.45ba/s]
08/27/2022 09:18:41 - INFO - dataset_maker - saving dataset
08/27/2022 09:18:47 - INFO - dataset_maker - ***** Dataset Finish ./datasets/ *****
```

上述处理后的数据默认被保留在 `datasets` 文件夹中。项目文件中已经完成这一步，可以直接从下述的第三步开始执行

3. 再次运行 train.py 文件，开始训练模型，对应命令行

```
1 python train.py \
2 --model_name_or_path /data/shizhengliang-slurm/HuggingFaceModel/fnlp-bart-
base/ \
3 --save_dataset_path ./datasets \
4 --log_root ./logs \
5 --exp_name waic \
6 --do_train \
7 --eval_steps 200 \
8 --evaluation steps \
9 --predict_with_generate True \
10 --output_dir model \
11 --save_steps 100 \
12 --save_total_limit 200 \
13 --num_train_epochs 5 \
14 --per_device_train_batch_size 16 \
15 --gradient_accumulation_steps 32 \
16 --chinese_data True
```

参数的意义如下所示:

1	--model_name_or_path:	基座模型的名称（对应huggingface中model模块的命名），或者是本地模型的保存路径
2	--save_dataset_path	第2步中转化成datasets类之后对应的文件夹(源代码中默认为./datasets/)
3	--output_dir	输出的文件夹，可随便指定
4	--max_source_length	最大输入长度（可截断）
5	--max_target_length	最大输出长度
6	--chinese_data	是否使用中文数据
7	--per_device_train_batch_size	每一个GPU设备上的batch size大小
8	--gradient_accumulation_steps	梯度累计的步数
9	--save_steps	每训练多少步保留checkpoint
10	--num_train_epochs	一共训练多少轮
11	--evaluation	评估的方式（可选参数 epoch :每一轮评估一次，step:每间隔指定步数评估一次）
12	--eval_step	每间隔指定步数评估一次
13	--do_train	是否训练
14	--do_eval	是否验证
15	--log_root	设置程序运行时checkpoint以及各种输出文件保存的文件夹

```
16 --predict_with_generate : 是否解码（将词表中的编码转化为汉字）
```

实际效果

```
([...])$ python train.py \
> --model_name_or_path /HuggingFaceModel/fnlp-bart-base/ \
> --save_dataset_path ./datasets \
> --log_root ./logs \
> --exp_name waic \
> --do_train \
> --do_eval \
> --eval_steps 200 \
> --evaluation_steps \
> --predict_with_generate True \
> --output_dir model \
> --save_steps 100 \
> --save_total_limit 200 \
> --num_train_epochs 5 \
> --per_device_train_batch_size 16 \
> --gradient_accumulation_steps 32 \
> --chinese_data True
```

正常训练

```
**** Running training ****
Num examples = 100001
Num Epochs = 5
Instantaneous batch size per device = 16
Total train batch size (w. parallel, distributed & accumulation) = 1024
Gradient Accumulation steps = 32
Total optimization steps = 1024
0%|          | 0/100001 [00:00<?, ?it/s]
Warning: /anaconda3/envs/.../lib/python3.8/site-packages/torch/nn/parallel/_functions.py:61: UserWarning: Was asked
to gather along dimension 0, but all input tensors were scalars; will instead unsqueeze and return a vector.
warnings.warn('Was asked to gather along dimension 0, but all
{'loss': 4.0123, 'learning_rate': 4.8969072164948454e-05, 'epoch': 0.1}
{'loss': 3.1113, 'learning_rate': 4.793814432989691e-05, 'epoch': 0.2}
5%|          | 22/100001 [05:08< 12.93s/it]
```

如何推断

在模型的训练过程中会保存checkpoint，可以使用如下命令在测试集上进行推断。（详见 inference.py 文件）

```
1 python inference.py \
2 --model_name_or_path ./logs/seq2seqv4/waic/model/checkpoint-350 \
3 --log_root ./logs \
4 --save_dataset_path ./datasets \
5 --exp_name waic \
6 --predict_with_generate True \
7 --output_dir model
```

```
1 ./logs/seq2seqv4/waic/model/checkpoint-350 为训练过程中保存的checkpoint
```

实验环境和结果

本实验所采用的硬件环境如下

设备号	设备类型	设备容量
0	TITAN RTX	24220MB
1	TITAN RTX	24220MB
2	TITAN RTX	24220MB
3	TITAN RTX	24220MB

经过多次测试，模型最优性能为：

Score	Blue	Blue1	Blue2	Blue3	Blue4
1.14	1.14	6.55	1.27	0.53	0.38

c-rough1	c-rough2	c-roughLsum	c-roughL	meteor
24.00	11.49	19.06	19.23	0.22

写在最后

联系作者

如果对本项目有任何疑问，请联系作者

- 1

QQ : 1172159897
- 2

Email : 1172159897@qq.com ; shizhl@mail.sdu.edu.cn

项目文件路径

- 1

WAIC
- 2

├─ BestCheckpoint # 最优checkpoint，可以通过inference.py文件，
将--model_name_or_path设置为对应路径进行推断
- 3

│ ├─ config.json
- 4

│ ├─ optimizer.pt
- 5

│ ├─ pytorch_model.bin
- 6

│ ├─ scheduler.pt
- 7

│ ├─ special_tokens_map.json
- 8

│ ├─ tokenizer_config.json
- 9

│ ├─ trainer_state.json
- 10

│ ├─ training_args.bin
- 11

│ └─ vocab.txt
- 12

├─ datasets # 最终处理好的数据集，可以直接用于train.py的训练
(详见上述模型训练的第3步)
- 13

│ ├─ dataset_dict.json # 数据集映射文件（详见上述模型训练第2步）
- 14

│ ├─ test # 测试集
- 15

│ │ ├─ cache-7c5e3e9ecb704a31.arrow
- 16

│ │ ├─ dataset_info.json
- 17

│ │ └─ state.json

```
18 | | └─ test_.json          # 预处理后的测试集（取出来无用字段）
19 | | └─ test.txt
20 | | └─ train              # 训练集
21 | |   └─ cache-befd04cbc1089214.arrow
22 | |   └─ dataset_info.json
23 | |     └─ state.json
24 | | └─ train_.json
25 | | └─ validation         # 验证集
26 | |   └─ cache-d126d4c4bb3bd269.arrow
27 | |   └─ dataset_info.json
28 | |     └─ state.json
29 |   └─ valid_.json
30 | └─ magic_bart2.py        # 模型骨干结构
31 | └─ requirements.txt      # 环境，如果缺少comet_ml，可额外手动pip安装
32 | └─ evaluation.py        # 计算BLUE，ROUGH等指标
33 | └─ train.py             # 模型训练文件
34 | └─ inference.py         # 模型推断文件
35 | └─ args.py              # 命令行参数文件
36 | └─ dataprep.py          # 构造检索增强的数据集
37 | └─ dataset_maker.py     # 构造huggingface的datasets类
38 | └─ list.txt             # 项目路径
```

其他说明文件

```
1 | └─ README.md            # 项目说明（本文件）
2 | └─ README.pdf          # 项目说明PDF版
3 | └─ 技术报告.pdf        # 技术报告
4 | └─ output_checkpoint_375 .csv # 输出文件样例
```