RLHF-AG_News

March 20, 2024

[1]:
```
# !pip install transformers
# !pip install accelerate
# !pip install twine
# !pip install datasets
# !pip install tyro
```

[2]:
```
!nvidia-smi
```

```
Wed Mar 20 09:07:24 2024
+-----------------------------------------------------------------------------
--------+
| NVIDIA-SMI 530.30.02              Driver Version: 530.30.02    CUDA Version:
12.1     |
|-----------------------------------------+----------------------+-------------
--------+
| GPU  Name                  Persistence-M| Bus-Id        Disp.A | Volatile
Uncorr. ECC |
| Fan  Temp  Perf            Pwr:Usage/Cap|         Memory-Usage | GPU-Util
Compute M. |
|                                         |                      |
MIG M. |
|=========================================+======================+=============
========|
|   0  Tesla V100-PCIE-32GB           On | 00000000:3B:00.0 Off |
Off |
| N/A   30C    P0               24W / 250W|      0MiB / 32768MiB |      0%
Default |
|                                         |                      |
N/A |
+-----------------------------------------+----------------------+-------------
--------+

+-----------------------------------------------------------------------------
--------+
| Processes:
|
|  GPU   GI   CI        PID   Type   Process name                            GPU
Memory |
```

```
|        ID   ID
Usage      |
|=============================================================================
========|
|  No running processes found
|
+-----------------------------------------------------------------------------
--------+
```

[3]:
```
# !pip install wandb
# !pip install trl
# !pip install pandas
# !pip install datasets
# !pip install nltk -U
```

[4]:
```
import torch
from tqdm import tqdm
import pandas as pd
import wandb
import os

tqdm.pandas()

from transformers import pipeline, AutoTokenizer
from datasets import load_dataset

from trl import PPOTrainer, PPOConfig, AutoModelForCausalLMWithValueHead
from trl.core import LengthSampler
```

[5]:
```
config = PPOConfig(
    model_name    = "openai-community/gpt2",
    learning_rate = 1.41e-5,
    ## log_with      = "wandb",
)

sent_kwargs = {
         "return_all_scores": True,
         "function_to_apply": "none",
         "batch_size": 16
}
print(config)
```

```
PPOConfig(exp_name='ipykernel_launcher', seed=0, log_with=None, task_name=None,
model_name='openai-community/gpt2', query_dataset='imdb',
reward_model='sentiment-analysis:lvwerra/distilbert-imdb',
remove_unused_columns=True, tracker_kwargs={}, accelerator_kwargs={},
project_kwargs={}, tracker_project_name='trl', push_to_hub_if_best_kwargs={},
steps=20000, learning_rate=1.41e-05, adap_kl_ctrl=True, init_kl_coef=0.2,
```

```
kl_penalty='kl', target=6, horizon=10000, gamma=1, lam=0.95, cliprange=0.2,
cliprange_value=0.2, vf_coef=0.1, batch_size=128, forward_batch_size=None,
mini_batch_size=128, gradient_accumulation_steps=1, world_size=None,
ppo_epochs=4, max_grad_norm=None, optimize_cuda_cache=None,
optimize_device_cache=False, early_stopping=False, target_kl=1, compare_steps=1,
ratio_threshold=10.0, use_score_scaling=False, use_score_norm=False,
score_clip=None, whiten_rewards=False, is_encoder_decoder=None,
is_peft_model=None, backward_batch_size=128, global_backward_batch_size=None,
global_batch_size=None)
```

[6]:
```
## wandb.init()

wandb.init(mode="disabled")
os.environ['WANDB_DISABLED'] = 'true'
```

```
Failed to detect the name of this notebook, you can set it manually with the
WANDB_NOTEBOOK_NAME environment variable to enable code saving.
```

[7]:
```
dataset_name="ag_news"
```

[8]:
```
ds = load_dataset(dataset_name, split = "train[:50000]")
```

[9]:
```
ds
```

[9]:
```
Dataset({
    features: ['text', 'label'],
    num_rows: 50000
})
```

[10]:
```
from datasets import ClassLabel
import random
import pandas as pd
from IPython.display import display, HTML
```

[11]:
```
tokenizer           = AutoTokenizer.from_pretrained(config.model_name)
tokenizer.pad_token = tokenizer.eos_token
# print(tokenizer)
```

[12]:
```
def tokenize( sample ):
    sample["input_ids"] = tokenizer.encode( sample["text"]    )[: 80]
    sample["query"]     = tokenizer.decode( sample["input_ids"] )
    return sample

#print(tokenize)
ds = ds.map(tokenize, batched=False)
```

[13]:
```
print(ds)
```

```
Dataset({
    features: ['text', 'label', 'input_ids', 'query'],
    num_rows: 50000
})
```

[14]:
```python
def build_dataset(
        config,
        dataset_name="ag_news",
        input_min_text_length=2,
        input_max_text_length=8
):
    """
    Build dataset for training. This builds the dataset from `load_dataset`, one should
    customize this function to train the model on its own dataset.

    Args:
        dataset_name (`str`):
            The name of the dataset to be loaded.

    Returns:
        dataloader (`torch.utils.data.DataLoader`):
            The dataloader for the dataset.
    """
    tokenizer           = AutoTokenizer.from_pretrained(config.model_name)
    tokenizer.pad_token = tokenizer.eos_token

    # load with datasets

    ds = load_dataset(dataset_name, split="train[:50000]")

#     ds = ds.rename_columns({"text": "review"})
    ds = ds.filter(lambda x: len(x["text"]) > 100, batched=False)

    input_size = LengthSampler(input_min_text_length, input_max_text_length)

    def tokenize(sample):
        sample["input_ids"] = tokenizer.encode( sample["text"]    )[:input_size()]
        sample["query"]     = tokenizer.decode( sample["input_ids"] )
        return sample

    ds = ds.map(tokenize, batched=False)
    ds.set_format(type="torch")
    return ds
```

[15]:
```python
dataset = build_dataset(config)
```

[16]:
```
dataset
```

[16]:
```
Dataset({
    features: ['text', 'label', 'input_ids', 'query'],
    num_rows: 49964
})
```

[17]:
```
def collator(data):
    return dict((key, [d[key] for d in data]) for key in data[0])
```

[18]:
```
model     = AutoModelForCausalLMWithValueHead.from_pretrained(config.model_name)
ref_model = AutoModelForCausalLMWithValueHead.from_pretrained(config.model_name)

tokenizer = AutoTokenizer.from_pretrained(config.model_name)

tokenizer.pad_token = tokenizer.eos_token
```

[19]:
```
ppo_trainer = PPOTrainer(
                  config,
                  model,
                  ref_model,
                  tokenizer,
                  dataset=dataset,
                  data_collator=collator
)
```

```
Detected kernel version 3.10.0, which is below the recommended minimum of 5.5.0;
this can cause the process to hang. It is recommended to upgrade the kernel to
the minimum version or higher.
```

[20]:
```
device = ppo_trainer.accelerator.device
device
```

[20]:
```
device(type='cuda')
```

[21]:
```
if ppo_trainer.accelerator.num_processes == 1:
    device = 0 if torch.cuda.is_available() else "cpu"  # to avoid a `pipeline`␣
 ↪bug

device
```

[21]:
```
0
```

[22]:
```
# Use a pipeline as a high-level helper
from transformers import pipeline
```

```
pipe = pipeline("text-classification", model="wesleyacheng/
 ↪news-topic-classification-with-bert")
```

[23]:
```
generation_kwargs = {
    "min_length":     -1,
    "top_k":          0.0,
    "top_p":          1.0,
    "do_sample":    True,
    "pad_token_id": tokenizer.eos_token_id,
}
```

[24]:
```
output_min_length     = 4
output_max_length     = 16
output_length_sampler = LengthSampler(output_min_length, output_max_length)
```

[25]:
```
ppo_trainer.config.steps
```

[25]: 20000

[26]:
```
# import os
# from transformers import GPT2LMHeadModel

# def save_checkpoint(model, filepath):
#     model.save_pretrained(filepath)
#     print("Model checkpoint saved successfully.")

# # Load Model Checkpoint
# def load_checkpoint(filepath):
#     if os.path.isdir(filepath):
#         model = GPT2LMHeadModel.from_pretrained(filepath)
#         print("Model checkpoint loaded successfully.")
#     else:
#         raise FileNotFoundError(f"Checkpoint directory not found at␣
 ↪{filepath}")
#     return model
```

[27]:
```
# import torch

# def gpu_memory_almost_full(threshold=0.9):
#     """
#     Check if GPU memory is almost full.

#     Args:
#         threshold (float): Threshold percentage for GPU memory usage.
#             If the current GPU memory usage exceeds this threshold,
#             consider the GPU memory almost full. Default is 0.9 (90%).
```

```
#     Returns:
#         bool: True if GPU memory is almost full, False otherwise.
#     """
#     allocated_bytes = torch.cuda.memory_allocated()
#     total_bytes = torch.cuda.get_device_properties(0).total_memory  #␣
 ↪assuming GPU 0
#     utilization = allocated_bytes / total_bytes
#     return utilization >= threshold
```

[28]:
```
# checkpoint_dir = '/home/vemuri8/gpt2_checkpoint'
# if os.path.exists(checkpoint_dir):
#     model = load_checkpoint(checkpoint_dir)
# else:
#     # Replace "model_name" with the actual model name or identifier
#     model = GPT2LMHeadModel.from_pretrained("gpt2_checkpoint")
```

[29]:
```
for epoch, batch in tqdm(enumerate(ppo_trainer.dataloader)):
    query_tensors = batch["input_ids"]
    print(epoch)

    #### Get response from gpt2
    response_tensors = []
    for query in query_tensors:
        gen_len                                 = output_length_sampler()
        generation_kwargs["max_new_tokens"] = gen_len
        response                                = ppo_trainer.generate(query,␣
 ↪**generation_kwargs)
        response_tensors.append( response.squeeze()[-gen_len:] )
    batch["response"] = [ tokenizer.decode(r.squeeze()) for r in␣
 ↪response_tensors ]

    #### Compute sentiment score
    texts = [q + r for q, r in zip(batch["query"], batch["response"])]
    pipe_outputs = pipe(texts, **sent_kwargs)
    rewards = [ torch.tensor(output[1]["score"]) for output in pipe_outputs]

    #### Run PPO step
    stats = ppo_trainer.step(
                    query_tensors,
                    response_tensors,
                    rewards
    )
#     ppo_trainer.log_stats(stats, batch, rewards)
#     if gpu_memory_almost_full():
#     save_checkpoint(model, 'gpt2_checkpoint')
#         break
```

```
0it [00:00, ?it/s]

0

/home/vemuri8/.local/lib/python3.8/site-
packages/transformers/pipelines/text_classification.py:104: UserWarning:
`return_all_scores` is now deprecated,  if want a similar functionality use
`top_k=None` instead of `return_all_scores=True` or `top_k=1` instead of
`return_all_scores=False`.
  warnings.warn(
1it [00:26, 26.65s/it]

1

2it [00:45, 22.13s/it]

2

3it [01:03, 20.29s/it]

3

4it [01:23, 20.02s/it]

4

5it [01:42, 19.87s/it]

5

6it [02:02, 19.90s/it]

6

7it [02:21, 19.52s/it]

7

8it [02:39, 18.98s/it]

8

9it [02:58, 18.92s/it]

9

10it [03:17, 18.89s/it]

10

11it [03:36, 19.18s/it]

11

12it [03:56, 19.39s/it]

12

13it [04:14, 18.98s/it]

13
```

14it [04:33, 19.04s/it]

14

15it [04:53, 19.07s/it]

15

16it [05:11, 18.89s/it]

16

17it [05:31, 19.27s/it]

17

18it [05:50, 19.21s/it]

18

19it [06:09, 19.16s/it]

19

20it [06:28, 18.89s/it]

20

21it [06:46, 18.81s/it]

21

22it [07:06, 18.95s/it]

22

23it [07:25, 19.14s/it]

23

24it [07:44, 19.02s/it]

24

25it [08:03, 18.97s/it]

25

26it [08:23, 19.25s/it]

26

27it [08:41, 18.97s/it]

27

28it [09:01, 19.28s/it]

28

29it [09:19, 18.98s/it]

29

```
30it [09:37, 18.72s/it]
30
31it [09:55, 18.55s/it]
31
32it [10:14, 18.67s/it]
32
33it [10:33, 18.73s/it]
33
34it [10:52, 18.84s/it]
34
35it [11:11, 18.81s/it]
35
36it [11:30, 18.77s/it]
36
37it [11:50, 19.07s/it]
37
38it [12:07, 18.69s/it]
38
39it [12:26, 18.79s/it]
39
40it [12:45, 18.71s/it]
40
41it [13:03, 18.59s/it]
41
42it [13:21, 18.44s/it]
42
43it [13:41, 18.75s/it]
43
44it [14:03, 19.67s/it]
44
45it [14:25, 20.44s/it]
45
```

```
46it [14:47, 20.88s/it]
46
47it [15:06, 20.43s/it]
47
48it [15:25, 20.06s/it]
48
49it [15:43, 19.43s/it]
49
50it [16:03, 19.53s/it]
50
51it [16:22, 19.27s/it]
51
52it [16:41, 19.32s/it]
52
53it [17:00, 19.14s/it]
53
54it [17:19, 19.16s/it]
54
55it [17:38, 19.08s/it]
55
56it [17:57, 19.00s/it]
56
57it [18:15, 18.91s/it]
57
58it [18:35, 19.10s/it]
58
59it [18:54, 19.00s/it]
59
60it [19:13, 18.93s/it]
60
61it [19:32, 18.97s/it]
61
```

62it [19:51, 19.02s/it]

62

63it [20:09, 18.91s/it]

63

64it [20:29, 18.97s/it]

64

65it [20:47, 18.85s/it]

65

66it [21:06, 18.73s/it]

66

67it [21:24, 18.75s/it]

67

68it [21:43, 18.76s/it]

68

69it [22:01, 18.61s/it]

69

70it [22:20, 18.67s/it]

70

71it [22:39, 18.76s/it]

71

72it [22:58, 18.84s/it]

72

73it [23:18, 19.04s/it]

73

74it [23:37, 19.02s/it]

74

75it [23:55, 18.94s/it]

75

76it [24:14, 18.76s/it]

76

77it [24:33, 18.88s/it]

77

78it [24:52, 18.95s/it]

78

79it [25:11, 18.87s/it]

79

80it [25:29, 18.77s/it]

80

81it [25:48, 18.77s/it]

81

82it [26:06, 18.53s/it]

82

83it [26:26, 18.94s/it]

83

84it [26:44, 18.83s/it]

84

85it [27:03, 18.85s/it]

85

86it [27:22, 18.92s/it]

86

87it [27:41, 18.90s/it]

87

88it [28:00, 18.95s/it]

88

89it [28:19, 18.72s/it]

89

90it [28:38, 18.90s/it]

90

91it [28:57, 18.98s/it]

91

92it [29:16, 18.96s/it]

92

93it [29:35, 18.90s/it]

93

94it [29:54, 19.02s/it]

94

95it [30:13, 18.90s/it]

95

96it [30:32, 18.93s/it]

96

97it [30:51, 18.99s/it]

97

98it [31:10, 19.07s/it]

98

99it [31:29, 19.08s/it]

99

100it [31:49, 19.26s/it]

100

101it [32:07, 19.00s/it]

101

102it [32:27, 19.13s/it]

102

103it [32:45, 18.87s/it]

103

104it [33:04, 18.92s/it]

104

105it [33:22, 18.78s/it]

105

106it [33:41, 18.72s/it]

106

107it [34:00, 18.77s/it]

107

108it [34:19, 18.86s/it]

108

109it [34:38, 19.01s/it]

109

110it [34:58, 19.09s/it]

110

111it [35:17, 19.24s/it]

111

112it [35:36, 19.25s/it]

112

113it [35:55, 19.17s/it]

113

114it [36:14, 19.06s/it]

114

115it [36:34, 19.14s/it]

115

116it [36:52, 18.81s/it]

116

117it [37:10, 18.83s/it]

117

118it [37:29, 18.82s/it]

118

119it [37:48, 18.79s/it]

119

120it [38:07, 18.98s/it]

120

121it [38:26, 18.85s/it]

121

122it [38:45, 18.81s/it]

122

123it [39:04, 18.83s/it]

123

124it [39:22, 18.84s/it]

124

125it [39:41, 18.83s/it]

125

126it [40:00, 18.95s/it]

126

127it [40:20, 18.99s/it]

127

128it [40:39, 19.16s/it]

128

129it [40:58, 19.00s/it]

129

130it [41:17, 18.95s/it]

130

131it [41:36, 19.05s/it]

131

132it [41:55, 19.15s/it]

132

133it [42:13, 18.88s/it]

133

134it [42:32, 18.86s/it]

134

135it [42:50, 18.63s/it]

135

136it [43:10, 18.80s/it]

136

137it [43:28, 18.84s/it]

137

138it [43:46, 18.50s/it]

138

139it [44:05, 18.59s/it]

139

140it [44:23, 18.54s/it]

140

141it [44:41, 18.36s/it]

141

142it [45:01, 18.77s/it]

142

143it [45:20, 18.81s/it]

143

144it [45:39, 18.82s/it]

144

145it [45:58, 18.81s/it]

145

146it [46:16, 18.81s/it]

146

147it [46:35, 18.81s/it]

147

148it [46:54, 18.91s/it]

148

149it [47:13, 18.90s/it]

149

150it [47:32, 18.72s/it]

150

151it [47:50, 18.62s/it]

151

152it [48:09, 18.64s/it]

152

153it [48:27, 18.61s/it]

153

154it [48:46, 18.62s/it]

154

155it [49:04, 18.39s/it]

155

156it [49:22, 18.47s/it]

156

157it [49:42, 18.76s/it]

157

158it [50:04, 19.71s/it]

158

159it [50:26, 20.47s/it]

159

160it [50:49, 21.25s/it]

160

161it [51:10, 21.31s/it]

161

162it [51:33, 21.73s/it]

162

163it [51:56, 21.94s/it]

163

164it [52:18, 22.20s/it]

164

165it [52:41, 22.22s/it]

165

166it [53:03, 22.19s/it]

166

167it [53:25, 22.18s/it]

167

168it [53:47, 22.28s/it]

168

169it [54:10, 22.37s/it]

169

170it [54:32, 22.20s/it]

170

171it [54:54, 22.15s/it]

171

172it [55:17, 22.43s/it]

172

173it [55:39, 22.28s/it]

173

174it [56:00, 22.05s/it]

174

175it [56:22, 21.86s/it]

175

176it [56:44, 21.94s/it]

176

177it [57:07, 22.33s/it]

177

178it [57:29, 22.08s/it]

178

179it [57:51, 22.27s/it]

179

180it [58:13, 22.10s/it]

180

181it [58:37, 22.53s/it]

181

182it [58:59, 22.48s/it]

182

183it [59:18, 21.37s/it]

183

184it [59:36, 20.36s/it]

184

185it [59:54, 19.84s/it]

185

186it [1:00:13, 19.53s/it]

186

187it [1:00:32, 19.38s/it]

187

188it [1:00:52, 19.47s/it]

188

189it [1:01:10, 19.16s/it]

189

190it [1:01:29, 19.03s/it]

190

191it [1:01:48, 18.94s/it]

191

192it [1:02:06, 18.82s/it]

192

193it [1:02:25, 18.82s/it]

193

194it [1:02:44, 18.71s/it]

194

195it [1:03:03, 18.84s/it]

195

196it [1:03:21, 18.78s/it]

196

197it [1:03:40, 18.67s/it]

197

198it [1:03:59, 18.73s/it]

198

199it [1:04:21, 19.82s/it]

199

200it [1:04:43, 20.39s/it]

200

201it [1:05:05, 21.02s/it]

201

202it [1:05:24, 20.30s/it]

202

203it [1:05:42, 19.67s/it]

203

204it [1:06:01, 19.35s/it]

204

205it [1:06:20, 19.37s/it]

205

206it [1:06:38, 19.02s/it]

206

207it [1:06:58, 19.14s/it]

207

208it [1:07:16, 18.94s/it]

208

209it [1:07:35, 18.79s/it]

209

210it [1:07:54, 18.85s/it]

210

211it [1:08:12, 18.63s/it]

211

212it [1:08:30, 18.37s/it]

212

213it [1:08:48, 18.48s/it]

213

214it [1:09:06, 18.18s/it]

214

215it [1:09:24, 18.31s/it]

215

216it [1:09:43, 18.52s/it]

216

217it [1:10:02, 18.61s/it]

217

218it [1:10:21, 18.74s/it]

218

219it [1:10:41, 18.94s/it]

219

220it [1:11:00, 19.02s/it]

220

221it [1:11:19, 19.16s/it]

221

222it [1:11:38, 18.95s/it]

222

223it [1:11:57, 18.99s/it]

223

224it [1:12:16, 18.89s/it]

224

225it [1:12:35, 19.21s/it]

225

226it [1:12:55, 19.21s/it]

226

227it [1:13:14, 19.21s/it]

227

228it [1:13:32, 18.90s/it]

228

229it [1:13:51, 18.89s/it]

229

230it [1:14:09, 18.78s/it]

230

231it [1:14:28, 18.84s/it]

231

232it [1:14:48, 18.99s/it]

232

233it [1:15:06, 18.83s/it]

233

234it [1:15:26, 19.12s/it]

234

235it [1:15:45, 19.00s/it]

235

236it [1:16:03, 18.79s/it]

236

237it [1:16:22, 18.76s/it]

237

238it [1:16:41, 18.87s/it]

238

239it [1:16:59, 18.57s/it]

239

240it [1:17:18, 18.67s/it]

240

241it [1:17:37, 18.83s/it]

241

242it [1:17:55, 18.78s/it]

242

243it [1:18:14, 18.75s/it]

243

244it [1:18:32, 18.51s/it]

244

245it [1:18:51, 18.72s/it]

245

246it [1:19:10, 18.61s/it]

246

247it [1:19:28, 18.51s/it]

247

248it [1:19:47, 18.55s/it]

248

249it [1:20:05, 18.55s/it]

249

250it [1:20:24, 18.65s/it]

250

251it [1:20:43, 18.69s/it]

251

252it [1:21:02, 18.71s/it]

252

253it [1:21:21, 18.87s/it]

253

254it [1:21:39, 18.60s/it]

254

255it [1:21:58, 18.89s/it]

255

256it [1:22:17, 18.83s/it]

256

257it [1:22:35, 18.68s/it]

257

258it [1:22:55, 18.85s/it]

258

259it [1:23:13, 18.71s/it]

259

260it [1:23:32, 18.89s/it]

260

261it [1:23:52, 18.99s/it]

261

262it [1:24:11, 19.13s/it]

262

263it [1:24:29, 18.92s/it]

263

264it [1:24:48, 18.86s/it]

264

265it [1:25:06, 18.68s/it]

265

266it [1:25:25, 18.73s/it]

266

267it [1:25:44, 18.84s/it]

267

268it [1:26:03, 18.74s/it]

268

269it [1:26:21, 18.68s/it]

269

270it [1:26:40, 18.72s/it]

270

271it [1:27:00, 18.95s/it]

271

272it [1:27:20, 19.21s/it]

272

273it [1:27:39, 19.20s/it]

273

274it [1:27:58, 19.09s/it]

274

275it [1:28:17, 19.22s/it]

275

276it [1:28:36, 19.22s/it]

276

277it [1:28:55, 19.19s/it]

277

278it [1:29:14, 18.87s/it]

278

279it [1:29:33, 18.97s/it]

279

280it [1:29:52, 19.01s/it]

280

281it [1:30:11, 18.97s/it]

281

282it [1:30:30, 19.11s/it]

282

283it [1:30:49, 19.09s/it]

283

284it [1:31:08, 18.91s/it]

284

285it [1:31:27, 19.07s/it]

285

286it [1:31:46, 18.91s/it]

286

287it [1:32:04, 18.74s/it]

287

288it [1:32:23, 18.80s/it]

288

289it [1:32:42, 18.91s/it]

289

290it [1:33:01, 18.95s/it]

290

291it [1:33:20, 19.04s/it]

291

292it [1:33:39, 19.00s/it]

292

293it [1:33:59, 19.13s/it]

293

294it [1:34:18, 19.04s/it]

294

295it [1:34:36, 18.98s/it]

295

296it [1:34:55, 18.93s/it]

296

297it [1:35:15, 19.07s/it]

297

298it [1:35:34, 19.05s/it]

298

299it [1:35:53, 19.13s/it]

299

300it [1:36:12, 19.15s/it]

300

301it [1:36:30, 18.88s/it]

301

302it [1:36:49, 18.81s/it]

302

303it [1:37:08, 18.92s/it]

303

304it [1:37:27, 18.75s/it]

304

305it [1:37:46, 18.87s/it]

305

306it [1:38:03, 18.43s/it]

306

307it [1:38:22, 18.68s/it]

307

308it [1:38:41, 18.76s/it]

308

309it [1:38:59, 18.53s/it]

309

310it [1:39:18, 18.54s/it]

310

311it [1:39:37, 18.56s/it]

311

312it [1:39:55, 18.56s/it]

312

313it [1:40:15, 18.90s/it]

313

314it [1:40:33, 18.60s/it]

314

315it [1:40:52, 18.80s/it]

315

316it [1:41:11, 19.02s/it]

316

317it [1:41:30, 18.92s/it]

317

318it [1:41:49, 18.86s/it]

318

319it [1:42:08, 18.82s/it]

319

320it [1:42:27, 18.86s/it]

320

321it [1:42:45, 18.62s/it]

321

322it [1:43:04, 18.79s/it]

322

323it [1:43:23, 18.81s/it]

323

324it [1:43:41, 18.60s/it]

324

325it [1:43:59, 18.56s/it]

325

326it [1:44:17, 18.44s/it]

326

327it [1:44:36, 18.44s/it]

327

328it [1:44:54, 18.43s/it]

328

329it [1:45:13, 18.65s/it]

329

330it [1:45:31, 18.34s/it]

330

331it [1:45:49, 18.25s/it]

331

332it [1:46:08, 18.56s/it]

332

333it [1:46:27, 18.54s/it]

333

334it [1:46:45, 18.46s/it]

334

335it [1:47:04, 18.72s/it]

335

336it [1:47:23, 18.69s/it]

336

337it [1:47:42, 18.74s/it]

337

338it [1:48:00, 18.50s/it]

338

339it [1:48:19, 18.63s/it]

339

340it [1:48:38, 18.68s/it]

340

341it [1:48:57, 18.80s/it]

341

342it [1:49:15, 18.61s/it]

342

343it [1:49:34, 18.76s/it]

343

344it [1:49:53, 18.86s/it]

344

345it [1:50:13, 19.08s/it]

345

346it [1:50:32, 19.19s/it]

346

347it [1:50:51, 19.06s/it]

347

348it [1:51:10, 18.95s/it]

348

349it [1:51:30, 19.44s/it]

349

350it [1:51:49, 19.38s/it]

350

351it [1:52:08, 19.18s/it]

351

352it [1:52:27, 19.01s/it]

352

353it [1:52:45, 18.75s/it]

353

354it [1:53:03, 18.71s/it]

354

355it [1:53:21, 18.50s/it]

355

356it [1:53:40, 18.62s/it]

356

357it [1:53:59, 18.62s/it]

357

358it [1:54:17, 18.53s/it]

358

359it [1:54:36, 18.56s/it]

359

360it [1:54:54, 18.44s/it]

360

361it [1:55:13, 18.44s/it]

361

362it [1:55:31, 18.43s/it]

362

363it [1:55:50, 18.68s/it]

363

364it [1:56:09, 18.58s/it]

364

365it [1:56:26, 18.34s/it]

365

366it [1:56:45, 18.50s/it]

366

367it [1:57:04, 18.71s/it]

367

368it [1:57:23, 18.78s/it]

368

369it [1:57:42, 18.89s/it]

369

370it [1:58:01, 18.80s/it]

370

371it [1:58:20, 18.79s/it]

371

372it [1:58:39, 18.85s/it]

372

373it [1:58:57, 18.60s/it]

373

374it [1:59:15, 18.61s/it]

374

375it [1:59:34, 18.66s/it]

375

376it [1:59:54, 18.90s/it]

376

377it [2:00:14, 19.22s/it]

377

378it [2:00:32, 18.92s/it]

378

379it [2:00:51, 19.01s/it]

379

380it [2:01:09, 18.67s/it]

380

381it [2:01:27, 18.59s/it]

381

```
382it [2:01:46, 18.60s/it]
382
383it [2:02:04, 18.52s/it]
383
384it [2:02:23, 18.58s/it]
384
385it [2:02:42, 18.64s/it]
385
386it [2:03:00, 18.56s/it]
386
387it [2:03:19, 18.60s/it]
387
388it [2:03:38, 18.77s/it]
388
389it [2:03:57, 18.76s/it]
389
390it [2:04:16, 19.12s/it]
```

[30]:
```
torch.cuda.get_device_name(0)
```

[30]:
```
'Tesla V100-PCIE-32GB'
```

[31]:
```
bs                  = 16
game_data           = dict()
```

[32]:
```
game_data
```

[32]:
```
{}
```

[33]:
```
dataset.set_format("pandas")
```

[34]:
```
df_batch            = dataset[:].sample(bs)
df_batch
```

[34]:

	text	label	\
2327	Tokyo Edge: More Video Options (PC World) PC W…	3	
432	Infocus: Deploying Network Access Quarantine C…	3	
4926	This unconventional IPO has a familiar ring Go…	2	
6180	De Bruijn, Hall retain 50m titles DEFENDING ch…	1	

```
34751  For Women Worried About Fertility, Egg Bank Is…      3
13320  Asian-angrez Khan is  #39;Best of British #39;…      1
42965  Cell phone talker arrest refuels etiquette deb…      3
30456  Indonesian Police Detain Seven People on Terro…      0
45928  Frisky koalas to get hormone implants  quot;It…      3
28405  Judges Postpone Milosevic Trial for Month (AP)…      0
18809  Everyone benefits from accountability It's a b…      2
19453  China #39;s Lenovo in talks with  #39;major IT…      2
40910  Second Thai woman has bird flu A 32-YEAR-OLD T…      0
19413  Dual-Core Chips Shift Performance Focus With t…      3
9717   The semantics of Israeli occupation Every few …      0
16366  India rocked by Harmison and Wharf (AFP) AFP -…      0

                                        input_ids  \
2327   [19042, 8226, 13113, 25, 3125, 7623, 18634]
432                             [18943, 10901, 25]
4926    [1212, 34014, 41805, 468, 257, 5385, 5858]
6180                        [5005, 8274, 48848, 11]
34751              [1890, 6926, 16597, 2228, 7994]
13320     [43224, 12, 648, 21107, 11356, 318, 220]
42965                     [28780, 3072, 1561, 263]
30456    [5497, 1952, 666, 4287, 4614, 391, 13723]
45928               [6732, 34041, 41727, 282, 292]
28405                      [26141, 3212, 2947, 79]
18809                                 [16190, 4034]
19453     [14581, 1303, 2670, 26, 82, 40269, 287]
40910              [12211, 18933, 2415, 468, 6512]
19413             [36248, 12, 14055, 45864, 15576]
9717                [464, 33815, 286, 6085, 13755]
16366                              [21569, 36872]

                                              query
2327                Tokyo Edge: More Video Options
432                                       Infocus:
4926   This unconventional IPO has a familiar ring
6180                                    De Bruijn,
34751                      For Women Worried About
13320                        Asian-angrez Khan is
42965                            Cell phone talker
30456               Indonesian Police Detain Seven
45928                               Frisky koalas
28405                                Judges Postp
18809                           Everyone benefits
19453                        China #39;s Lenovo in
40910                   Second Thai woman has bird
19413                        Dual-Core Chips Shift
9717           The semantics of Israeli occupation
```

16366 India rocked

[35]:
```
game_data["query"] = df_batch["query"].tolist()
query_tensors      = df_batch["input_ids"].tolist()
```

[36]:
```
response_tensors_ref, response_tensors = [], []
```

[37]:
```
gen_kwargs = {
         "min_length":   -1,
         "top_k":       0.0,
         "top_p":       1.0,
         "do_sample":  True,
         "pad_token_id": tokenizer.eos_token_id
}
```

[38]:
```
for i in range(bs):
    gen_len = output_length_sampler()

    output  = ref_model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device),␣
 ↪max_new_tokens=gen_len, **gen_kwargs
    ).squeeze()[-gen_len:]
    response_tensors_ref.append(output)


    output = model.generate(
        torch.tensor(query_tensors[i]).unsqueeze(dim=0).to(device),␣
 ↪max_new_tokens=gen_len, **gen_kwargs
    ).squeeze()[-gen_len:]
    response_tensors.append(output)
```

[39]:
```
game_data["response (before)"] = [tokenizer.decode(response_tensors_ref[i]) for␣
 ↪i in range(bs)]
game_data["response (after)"]  = [tokenizer.decode(response_tensors[i]) for i␣
 ↪in range(bs)]
```

[40]:
```
texts = [q + r for q, r in zip(game_data["query"], game_data["response␣
 ↪(before)"])]
game_data["rewards (before)"] = [output[1]["score"] for output in pipe(texts,␣
 ↪**sent_kwargs)]
```

```
/home/vemuri8/.local/lib/python3.8/site-
packages/transformers/pipelines/text_classification.py:104: UserWarning:
`return_all_scores` is now deprecated,  if want a similar functionality use
`top_k=None` instead of `return_all_scores=True` or `top_k=1` instead of
`return_all_scores=False`.
  warnings.warn(
```

[41]:
```
texts = [q + r for q, r in zip(game_data["query"], game_data["response␣
↪(after)"])]
game_data["rewards (after)"] = [output[1]["score"] for output in pipe(texts,␣
↪**sent_kwargs)]
```

[42]:
```
df_results = pd.DataFrame(game_data)
df_results
```

[42]:

	query	response (before)	response (after)	rewards (before)
0	Tokyo Edge: More Video Options	Xanadu eX+ 42	Everything with Tanner Shaw 330000 2017	-3.093839
1	Infocus:	Proper hinge, thumb, mixed knot close\n\nDrop…	4 Week 782 417 NASCAR Indianapolis Cup #55 - …	-1.552972
2	This unconventional IPO has a familiar ring	to it from its genesis at		
3	De Bruijn,	pass by cilla mavingloo		
4	For Women Worried About	the Weekend\n\nA few other parties		
5	Asian-angrez Khan is	(		
6	Cell phone talker	and master of transportation in Boise in the …		
7	Indonesian Police Detain Seven	People in Custody of Duke of York Seal of St…		
8	Frisky koalas	and dusky vol'alas! Mrs. Kim Gifts I am not		
9	Judges Postp	one\n\nSpectator\n\nJizer (In Love with Mat		
10	Everyone benefits	easily from being categorized outside of the		
11	China #39;s Lenovo in	Europe, Latin America & Africa		
12	Second Thai woman has bird	-repeating mouth plastered with boots whose in…		
13	Dual-Core Chips Shift	= 189053.1301 () turns on the		
14	The semantics of Israeli occupation	of cities and suburbs are reminiscent of		
15	India rocked	the streets\n\nOn South Asia Day 2002 a Japan…		

2	to clear jazz.\n\n	-1.015657
3	who played last season against Ontario (	-0.650193
4	Basketball\n\nThe LA Lakers were a	-0.679563
5	urchin from Bengals, and he who Georgia, a	-1.202581
6	. Read about the dots in Kentucky's gubernator…	-1.929202
7	Returning Steelers Miami Dolphins 34 Vikings …	-2.207360
8	(Fiatletico) was bad when it happened and Mes…	-0.471304
9	that Goodell "one of the loudest critics in N…	-0.454341
10	. Women vastly outnumber men and	-0.667116
11	Seattle; Duke basketball 12-	-3.545890
12	flu AP 24/20 in Japan 5 Fi 39.	-1.448232
13	:\n\nHere's where things begin…\n\n	-2.985734
14	of the Jacksonville States of football,	-2.681175
15	aj, xxx indicates thys coach and leadership fo…	-2.810116

	rewards (after)
0	-2.818416
1	4.654158
2	-0.291705
3	5.332590
4	3.319044
5	-1.159039
6	-2.735202
7	4.095195
8	2.063074
9	4.984600
10	-2.521450
11	5.158721
12	-2.081986
13	-2.941343
14	1.875992
15	3.480184