تولید زبان طبیعی با تعداد نمونه‌های محدود برای گفتگوی مبتنی بر وظیفه

محمد مصطفی رستم خانی - غزل زمانی نژاد - سید محمد عرفان موسوی منزه

۴۰۱۷۲۲۲۳۵ - ۴۰۱۷۲۲۲۴۴ - ۴۰۱۷۲۲۱۹۹

فهرست مطالب

آشنایی با SC-GPT

آموزش مدل SC-GPT در سه مرحله

1. پیش آموزش بر روی حجم زیادی از متن‌های ساده
2. پیش آموزش بر روی مقدار زیادی داده برچسب‌گذاری شده برای کنش-گفتگو (dialog-act):
   بر روی مجموعه دادگان Schema-guided Dialog corpus، MultiWOZ corpus، Frame corpusو Facebook Multilingual Dialog corpus
3. تنظیم دقیق در زمینه مورد نظر با تعداد محدودی داده

   بر روی مجموعه داده FEWSHOTWOZ

تنظیمات پروژه

راه‌اندازی تنظیمات اولیه برای اجرای پروژه

- نصب conda روی colab برای استفاده از پایتون ۳.۷و نصب بقیه پیش نیازها بدون conflict
- نصب پکیج‌ها و نیازمندی‌ها (در این مرحله تغییراتی در فایل requirement.txt اعمال شد)
- مرحله سخت و چالشی به دلیل هماهنگ نبودن نسخه بعضی از پکیج‌ها و همچنین موجود نبودن برخی پکیج‌ها (مانند پکیج transformers که از نسخه قدیمی استفاده شده بود)
- بعضی خطوط کد نیاز بود تا تغییر داده شوند زیرا کار نمی کردند.
- عوض کردن کد مربوط به generation زیرا دارای اشتباه مربوط به & بود.

```
!sed -i '19d' req.txt
# adding transformers==4.8.2 instead of 2.1.1
!sed -i '$ d' req.txt
!echo "transformers==4.8.2" >> req.txt
```

```
%pip install -r req.txt
```

اجرا و بررسی کدهای پروژه

گرفتن وزن های SC-GPT

- استفاده از مدل SC-GPT که بر روی حجم زیادی داده پیش آموزش داده شده

Fetch and unzip the checkpoint

```
wget https://bapengstorage.blob.core.windows.net/fileshare/scgpt.tar.gz
tar -xvf scgpt.tar.gz
```

- لود کردن وزن‌های مدل پیش آموخته بر روی داده برچسب‌گذاری شده از Hugging Face (به دلیل عمومی نبودن لینک checkpointموجود در ریپازیتوری)

```
# the above link is not public so we use hugging face SC-GPT model instead
!git lfs install
!git clone https://huggingface.co/metehergul/scgpt
```

تنظیم دقیق با نمونه‌های محدود بر روی دامنه‌های مختلف

مشخص کردن آرگومان‌های آموزش

```
!export CUDA_VISIBLE_DEVICES=0
!python train.py --output_dir=MODEL_SAVE_PATH \
                 --model_type=gpt2 \
                 --model_name_or_path='./scgpt' \
                 --do_train \
                 --do_eval \
                 --eval_data_file=data/restaurant/train.txt \
                 --per_gpu_train_batch_size 1 \
                 --num_train_epochs 20 \
                 --learning_rate 1e-5 \
                 --overwrite_cache \
                 --use_tokenize \
                 --train_data_file=data/restaurant/train.txt \
                 --overwrite_output_dir
```

بررسی کد train.py
کلاس TextSeqDataset

```
89  class TextSeqDataset(Dataset):
90      def __init__(self, tokenizer, args, file_path='train', block_size=512, max_seq=80, seperator=' & '):
101             self.examples = []
102             self.labels = []
103             self.masks = []
104             with open(file_path, encoding="utf-8") as f:
105                 for line in f:
106                     line = line.strip()
107                     raw_str = line.lower()
108                     code_str = line.lower().split(seperator)[0] + seperator
109                     code_str = code_str.strip()
110                     if len(raw_str.split()) > max_seq -1:
111                         raw_str = ' '.join(raw_str.split()[:max_seq -1])
112                     raw_str += ' ' + tokenizer.eos_token
113                     if args.use_tokenize:
114                         tokenized_text = tokenizer.convert_tokens_to_ids(tokenizer.tokenize(raw_str))
115                         code_str_len =  len(tokenizer.convert_tokens_to_ids(tokenizer.tokenize(code_str)))
116                     else:
117                         tokenized_text = tokenizer.convert_tokens_to_ids(raw_str.split())
118                         code_str_len =  len(tokenizer.convert_tokens_to_ids(code_str.split()))
119
120                     label = [-1] *  max_seq
121                     label[:len(tokenized_text)] = tokenized_text
122                     mask = [1] *  max_seq
123
124
125                     if len(tokenized_text) < max_seq:
126                         mask[-(max_seq - len(tokenized_text)):] = [0] * (max_seq - len(tokenized_text))
127                         # label[code_str_len:len(tokenized_text)] = tokenized_text[code_str_len:]
128                         tokenized_text = tokenized_text + [0] * (max_seq - len(tokenized_text))
129                     else:
130                         tokenized_text = tokenized_text[:max_seq]
133                  self.examples.append(tokenized_text)
134                  self.masks.append(mask)
135                  self.labels.append(label)
146     def __len__(self):
147         return len(self.examples)
148
149     def __getitem__(self, item):
150         return torch.tensor(self.examples[item]), torch.tensor(self.masks[item]), torch.tensor(self.labels
```

قالب داده ورودی: $\mathcal{A}' = [\mathbf{I}\ (\ s_1 = v_1\ ,\ \cdots\ s_P = v_P\ )]$

انجام پیش پردازش:

- اسپلیت و توکنایز کردن داده
- پد کردن طول به اندازه max_seq

بررسی کد train.py
تابع train

```
270         model.zero_grad()
271         train_iterator = trange(int(args.num_train_epochs), desc="Epoch", disable=args.local_rank not in [-1, 0])
272         set_seed(args)  # Added here for reproducibility (even between python 2 and 3)
273         for e in train_iterator:
274 
275             # epoch_iterator = tqdm(train_dataloader, desc="Iteration", disable=args.local_rank not in [-1, 0])
276             for step, batch in enumerate(train_dataloader):
277                 # inputs, labels = mask_tokens(batch, tokenizer, args) if args.mlm else (batch, batch)
278                 logger.info(f"  PROGRESS: {float(global_step)/t_total*100}%")
279                 inputs, masks, labels = batch
280                 # import pdb
281                 # pdb.set_trace()
282                 inputs = inputs.to(args.device)
283                 # masks = masks.to(args.device)
284                 labels = labels.to(args.device)
285 
286                 model.train()
287                 outputs = model(inputs, masked_lm_labels=labels) if args.mlm else model(inputs, labels=labels)
288                 loss = outputs[0]  # model outputs are always tuple in transformers (see doc)
295                  if args.fp16:
296                      with amp.scale_loss(loss, optimizer) as scaled_loss:
297                          scaled_loss.backward()
298                  else:
299                      loss.backward()
300 
301                  tr_loss += loss.item()
302                  if (step + 1) % args.gradient_accumulation_steps == 0:
303                      if args.fp16:
304                          torch.nn.utils.clip_grad_norm_(amp.master_params(optimizer), args.max_grad_norm)
305                      else:
306                          torch.nn.utils.clip_grad_norm_(model.parameters(), args.max_grad_norm)
307                      optimizer.step()
308                      scheduler.step()  # Update learning rate schedule
309                      model.zero_grad()
310                      global_step += 1
```

استفاده از:

- بهینه‌ساز AdamW
- کلیپ کردن گرادیان
- scheduler with warmup برای نرخ یادگیری

بررسی کد train.py
تابع evaluate

```
374        model.eval()
375
376        for batch in tqdm(eval_dataloader, desc="Evaluating"):
377            # inputs, labels = mask_tokens(batch, tokenizer, args) if args.mlm else (batch, batch)
378
379            inputs, masks, labels = batch
380                # import pdb
381                # pdb.set_trace()
382            inputs = inputs.to(args.device)
383            masks = masks.to(args.device)
384            labels = labels.to(args.device)
385            # inputs = inputs.to(args.device)
386            # labels = labels.to(args.device)
387
388            with torch.no_grad():
389                outputs = model(inputs, masked_lm_labels=labels) if args.mlm else model(inputs, labels=labels)
390                lm_loss = outputs[0]
391                eval_loss += lm_loss.mean().item()
392            nb_eval_steps += 1
393
394        eval_loss = eval_loss / nb_eval_steps
395        perplexity = torch.exp(torch.tensor(eval_loss))
396
397        result = {
398            "perplexity": perplexity
399        }
```

- بررسی مدل بر روی داده ارزیابی (در اینجا مطابق با اجراهای مقاله اصلی، از همان داده آموزش استفاده شده)
- گزارش مقدار perplexity

تولید متن از روی داده آزمون

مشخص کردن آرگومان‌های تولید متن

```
!export CUDA_VISIBLE_DEVICES=0
!python generate.py --model_type=gpt2 \
                    --model_name_or_path=MODEL_SAVE_PATH \
                    --num_samples 5 \
                    --input_file=data/restaurant/test.txt \
                    --top_p 0.9 \
                    --output_file=results.json \
                    --length 80
```

بررسی کد generate.py
تابع main

```
139  def main():
204      fin = open(args.input_file)
205      inputs = [i.strip() for i in fin]
206      output_tests = []
207      for idx in range(0, len(inputs), 1):
229          lines = inputs[idx]
230          raw_text = lines.split(' & ')[0] + ' & '
231          if args.model_type in ["transfo-xl", "xlnet"]:
232              # Models with memory likes to have a long prompt for short inputs.
233              raw_text = (args.padding_text if args.padding_text else PADDING_TEXT) + raw_text
234
235          context_tokens = tokenizer.encode(raw_text, add_special_tokens=False)
236
237          if args.model_type == "ctrl":
238              if not any(context_tokens[0] == x for x in tokenizer.control_codes.values()):
239                  logger.info("WARNING! You are not starting your generation from a control code so
240          out = sample_sequence(
241              model=model,
242              context=context_tokens,
243              num_samples=args.num_samples,
244              length=args.length,
245              temperature=args.temperature,
246              top_k=args.top_k,
247              top_p=args.top_p,
248              repetition_penalty=args.repetition_penalty,
249              is_xlnet=bool(args.model_type == "xlnet"),
250              is_xlm_mlm=is_xlm_mlm,
251              xlm_mask_token=xlm_mask_token,
252              xlm_lang=xlm_lang,
253              device=args.device,
254          )
255          out = out[:, len(context_tokens):].tolist()
256          examples = []
257          for o in out:
258              text = tokenizer.decode(o, clean_up_tokenization_spaces=True)
259              text = text[: text.find(args.stop_token) if args.stop_token else None]
260              examples.append(text)
261
262          output_tests.append(examples)
263          # break
264          # if args.prompt:
265              # break
266      import json
267      json.dump(output_tests, open(args.output_file,'w'), indent=2)
268      return text
```

1. پیش‌پردازش داده آزمون
2. پیش‌بینی خروجی با دادن داده آزمون به عنوان ورودی مدل (با استفاده از تابع sample_sequence)
3. کدگشایی خروجی‌های مدل
4. ذخیره در فایل json

بررسی کد generate.py
تابع sample_sequene

```
94  def sample_sequence(model, length, context, num_samples=1, temperature=1, top_k=0, top_p=0.0, repetition_penalty=1.0,
95                      is_xlnet=False, is_xlm_mlm=False, xlm_mask_token=None, xlm_lang=None, device='cpu'):
96      context = torch.tensor(context, dtype=torch.long, device=device)
97      context = context.unsqueeze(0).repeat(num_samples, 1)
98      generated = context
99      with torch.no_grad():
100         for _ in range(length):
101
102             inputs = {'input_ids': generated}
122             outputs = model(**inputs)  # Note: we could also use 'past' with GPT-2/Transfo-XL/XLNet/CTRL (cached hidden
123             next_token_logits = outputs[0][:, -1, :] / (temperature if temperature > 0 else 1.)
124
125             # repetition penalty from CTRL (https://arxiv.org/abs/1909.05858)
126             for i in range(num_samples):
127                 for _ in set(generated[i].tolist()):
128                     next_token_logits[i, _] /= repetition_penalty
129
130             filtered_logits = top_k_top_p_filtering(next_token_logits, top_k=top_k, top_p=top_p)
131             if temperature == 0: # greedy sampling:
132                 next_token = torch.argmax(filtered_logits, dim=-1).unsqueeze(-1)
133             else:
134                 next_token = torch.multinomial(F.softmax(filtered_logits, dim=-1), num_samples=1)
135             generated = torch.cat((generated, next_token), dim=1)
136     return generated
```

1. گرفتن خروجی از مدل
2. محاسبه logit های توکن بعدی
3. فیلتر کردن توکن‌ها بر اساس top k و top p
4. نمونه‌برداری حریصانه به شرط temperature = 0
5. در غیر این صورت، نمونه‌برداری بر اساس توزیع احتمالی

بررسی کد generate.py
تابع top_k_top_p_filtering

```python
def top_k_top_p_filtering(logits, top_k=0, top_p=0.0, filter_value=-float('Inf')):
    """ Filter a distribution of logits using top-k and/or nucleus (top-p) filtering
        Args:
            logits: logits distribution shape (batch size x vocabulary size)
            top_k > 0: keep only top k tokens with highest probability (top-k filtering).
            top_p > 0.0: keep the top tokens with cumulative probability >= top_p (nucleus filtering).
                Nucleus filtering is described in Holtzman et al. (http://arxiv.org/abs/1904.09751)
        From: https://gist.github.com/thomwolf/1a5a29f6962089e871b94cbd09daf317
    """
    top_k = min(top_k, logits.size(-1))  # Safety check
    if top_k > 0:
        # Remove all tokens with a probability less than the last token of the top-k
        indices_to_remove = logits < torch.topk(logits, top_k)[0][..., -1, None]
        logits[indices_to_remove] = filter_value

    if top_p > 0.0:
        sorted_logits, sorted_indices = torch.sort(logits, descending=True)
        cumulative_probs = torch.cumsum(F.softmax(sorted_logits, dim=-1), dim=-1)

        # Remove tokens with cumulative probability above the threshold
        sorted_indices_to_remove = cumulative_probs > top_p
        # Shift the indices to the right to keep also the first token above the threshold
        sorted_indices_to_remove[..., 1:] = sorted_indices_to_remove[..., :-1].clone()
        sorted_indices_to_remove[..., 0] = 0

        # scatter sorted tensors to original indexing
        indices_to_remove = sorted_indices_to_remove.scatter(dim=1, index=sorted_indices, src=sorted_indices_to_remove)
        logits[indices_to_remove] = filter_value
    return logits
```

اگر top k باشد:

نگه داشتن k توکن با بیشترین احتمال و حذف سایر

اگر top p باشد:

حذف توکن‌هایی که مقدار احتمال تجمعی کمتر از حد آستانه دارند

ارزیابی عملکرد

مشخص کردن آرگومان‌های ارزیابی عملکرد

```
1 import nltk
2 nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
True
```

```
1 !python evaluator.py --domain restaurant --target_file results.json
```

بررسی کد evaluator.py

- استفاده از DataReader برای خواندن فایل مربوط به خروجی مدل
- استفاده از تابع GentScorer برای به دست آوردن BLEUScore و ERR
- ذخیره dialog act ها و جملات متناظر در دیکشنری da2sents
- محاسبه معیارهای مورد نیاز و چاپ آنها

```
bleuModel   = gentscorer.scoreSBLEU(parallel_corpus)
bleuHDC     = gentscorer.scoreSBLEU(hdc_corpus)
print ('############################################')
print ('BLEU SCORE & SLOT ERROR on GENERATED SENTENCES')
print ('############################################')
print ('Metric       :\tBLEU\tT.ERR\tA.ERR')
print ('HDC          :\t%.4f\t%2.2f%%\t%2.2f%%'% (bleuHDC,0.0,0.0))
print ('Ref          :\t%.4f\t%2.2f%%\t%2.2f%%'% (1.0,
       100*refcnts[1]/refcnts[0],100*refcnts[2]/refcnts[0]))
print ('--------------------------------------------')
print ('This Model   :\t%.4f\t%2.2f%%\t%2.2f%%'% (bleuModel,
       100*gencnts[1]/gencnts[0],100*gencnts[2]/gencnts[0]))

print(f'FIELNAME: {target_file}, BLEU: {bleuModel}, ERR:{100*gencnts[1]/gencnts[0]}')
```

آزمایش‌های انجام شده و تغییرات در ابرپارامترها

انجام آزمایش با ابرپارامترهای گوناگون بر روی تمامی دامنه‌ها

1. آزمایش اول

- Learning rate = 1e-5
- Training epochs = 5
- Decoding strategy: top_p = 0.9

2. آزمایش دوم

- Learning rate = 1e-5
- Training epochs = 20
- Decoding strategy: top_p = 0.9

برای دستیابی به نتایج مقاله، در آزمایش دوم ابرپارامترها مطابق با لینک تنظیم شدند.

نتایج به دست آمده

نتایج به دست آمده و مقایسه با نتایج مقاله

نتایج مقاله:

Model	Restaurant		Laptop		Hotel		TV		Attraction		Train		Taxi	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
SC-GPT	**38.08**	**3.89**	**32.73**	**3.39**	**38.25**	**2.75**	**32.95**	**3.38**	**20.69**	**12.72**	**17.21**	**7.74**	**19.70**	**3.57**

نتایج پیاده سازی:

Exp	Restaurant		Laptop		Hotel		TV		Attraction		Train		Taxi	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
1	**21.62**	**3.29**	**23.86**	**5.86**	**29.38**	**8.24**	**19.34**	**5.61**	**9.13**	**12.91**	**7.96**	**9.22**	**8.24**	**4.36**
2	**34.37**	**3.29**	**27.45**	**5.51**	**33.49**	**2.74**	**25.75**	**4.63**	**15.81**	**12.73**	**12.73**	**8.28**	**13.84**	**3.70**

چالش‌ها

چالش‌ها

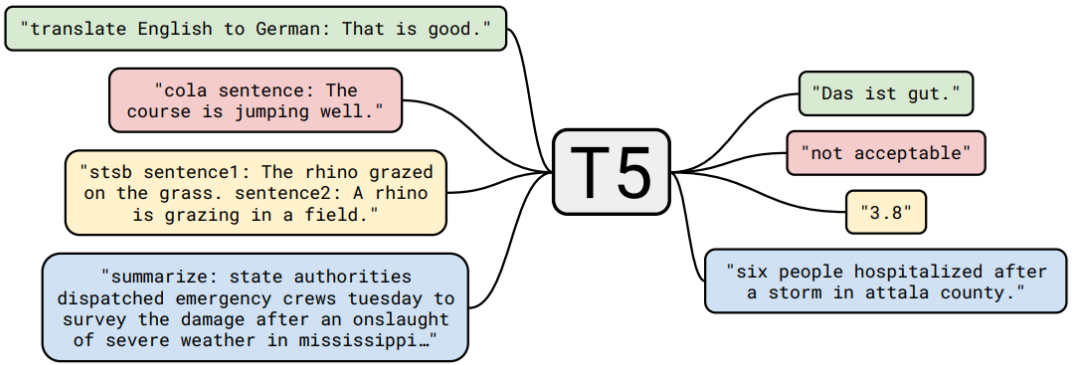
- بزرگترین چالش مربوط به راه‌اندازی تنظیمات اولیه پروژه بود:
  - عدم همخوانی نسخه‌های پکیج‌ها با آخرین نسخه پایتون
  - نبود برخی پکیج‌ها (مثلا absl)
  - نصب conda و ساختنenvironment روی google colab
  - عدم همخوانی پکیج transformers با کد های موجود
- عدم دسترسی به وزن‌های مدل پیش آموخته
- عدم توانایی استفاده از مدل موجود در Hugging Face به دلیل اشتباه در فایل config در model_type
- برخی اشتباهات موجود در پیش پردازش داده ها برای دادن به مدل

تلاش برای بهبود نتایج

ایده برای بهبود

مقاله اصلی از معماری مربوط به GPT-2 به عنوان معماری پایه برای SC-GPT استفاده کرده بود و آن را بر روی 400k داده، پیش آموزش داده بود.

ما از مدلT5 برای معماری پایه استفاده کردیم و آن را بر روی 100k از داده های موجود پیش آموزش داده و مدل خود را بر روی Hugging Face آپلود کردیم.



تنظیم دقیق T5

سپس مدل به دست آمده را بر روی domain های موجود در FEWSHOTWOZ تنظیم دقیق کرده و آنها را نیز در Hugging Face آپلود کردیم.

- Restaurant
- Laptop
- Hotel
- TV
- Attraction
- Train
- Taxi

بررسی کد T5

```
from transformers import DataCollatorWithPadding

def tokenize_func(examples):
    input_encodings = tokenizer(
        examples['text'], max_length=MAX_INPUT_LEN,
        padding='max_length', truncation=True
    )

    output_encodings = tokenizer(
        examples['output'], max_length=MAX_OUTPUT_LEN,
        padding='max_length', truncation=True
    )

    encodings = {
        'input_ids': input_encodings['input_ids'],
        'attention_mask': input_encodings['attention_mask'],
        'labels': output_encodings['input_ids']
    }

    return encodings

tds = ds.map(tokenize_func, batched=True)
print(tds)
data_collator = DataCollatorWithPadding(tokenizer=tokenizer)
```

- توکنایز کردن ورودی و خروجی متناظر
- استفاده از DataCollatorWithPadding برای استفاده از Trainer مربوط به Hugging Face

بررسی کد T5 pre-training

- استفاده از Trainerمربوط به Hugging Face برای آموزشT5 بر روی مجموعه داده

```python
from transformers import TrainingArguments, Trainer, PushToHubCallback

# Set the training arguments
training_args = TrainingArguments(
    f'models', True,
    num_train_epochs=NUM_EPOCHS, auto_find_batch_size=True,

    evaluation_strategy='steps', eval_steps=EVAL_STEPS, logging_steps=EVAL_STEPS,
    save_strategy='steps', save_steps=SAVE_STEPS, save_total_limit=5,

    #push_to_hub=True, push_to_hub_model_id=MODEL_ID,
)

# Create the trainer
trainer = Trainer(
    model,
    training_args,
    train_dataset=tds['train'],
    eval_dataset=tds['test'],
    data_collator=data_collator,
    tokenizer=tokenizer,
)

# Fine-tune the model
try:
    print('training from scratch')
    trainer.train(resume_from_checkpoint=False)
except:
    print('continout training')
    trainer.train(resume_from_checkpoint=True)
```

بررسی کد T5 fine-tuning

```
from transformers import TrainingArguments, Trainer

for split in splits:
    print(f'---- Fine Tuning on {split.capitalize()} Tasks ----')
    model_id = f'T5-Task-Dialogue-FineTuned-{split.capitalize()}-{NUM_EPOCHS}-{LR}'
    set_train = tds[f'{split}.train']
    set_test = tds[f'{split}.test']
    !rm -r models

    model = AutoModelForSeq2SeqLM.from_pretrained("ErfanMoosaviMonazzah/T5-Task-Dialogue-Pretrained")

    # Set the training arguments
    training_args = TrainingArguments(
        f'models', True,
        num_train_epochs=NUM_EPOCHS, auto_find_batch_size=True,

        do_eval=False, #evaluation_strategy='epoch',
        #logging_strategy='epoch',
        save_strategy='no',
        learning_rate=LR,
    )

    # Create the trainer
    trainer = Trainer(
        model,
        training_args,
        train_dataset=set_train,
        eval_dataset=set_test,
        data_collator=data_collator,
        tokenizer=tokenizer,
    )

    trainer.train()
    model.push_to_hub(model_id)
```

- استفاده از داده های FEWSHOTWOZ برای fine-tuning
- لود کردن مدل پیش آموخته در قسمت قبل
- استفاده از trainer برای آموزش مدل
- ذخیره مدل تنظیم دقیق شده بر روی Hugging Face

بررسی کد T5
T5_Generate

```python
from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
import json

for split in splits:
    print(f'---- Generating for {split.capitalize()} Tasks ----')
    inputs = ds[f'{split}.test']['text']
    model_id = f'ErfanMoosaviMonazzah/T5-Task-Dialogue-FineTuned-{split.capitalize()}-{NUM_EPOCHS}-{LR}'

    model = AutoModelForSeq2SeqLM.from_pretrained(model_id)

    si = 0
    while si < len(inputs):
        if len(inputs) - si <= 300:
            ei = len(inputs)
        else:
            ei = si + 300

        input_ids = tokenizer.batch_encode_plus(inputs[si:ei], return_tensors="pt", padding=True)["input_ids"]
        output_ids = model.generate(input_ids, top_p=P, do_sample=True, num_return_sequences=K)

        print("output_ids generated")

        outputs = []
        tmp_l = []
        for i, output_id in enumerate(output_ids):
            if i % K == 0:
                tmp_l = []

            t_out = ' & ' + tokenizer.decode(output_id, skip_special_tokens=True)+'|endoftext|'
            t_out = t_out + (LENGTH - len(t_out)) * '!'
            tmp_l.append(t_out)
            if i%K == K-1:
                outputs.append(tmp_l)

        with open(f'resutls-{model_id.split("/")[1]}-{si//300}.json', 'w') as jf:
            json.dump(outputs, jf, indent = 4)

        si = ei
```

- خواندن داده های مربوط به هر دامنه
- لود کردن مدل تنظیم دقیق شده بر روی همان دامنه
- شکاندن داده های مربوط به هر دامنه به بازه های 300 تایی برای جلوگیری از پر شدن RAM
- تولید چند جمله خروجی مدل برای هر داده تست
- ذخیره داده های تولید شده در فایل json برای evaluate

بررسی کد T5
evaluate

استفاده از فایل های json تولید شده در قسمت قبل برای ارزیابی

```
!python evaluator.py --domain restaurant --target_file Restaurant.json
```

```
##############################################
BLEU SCORE & SLOT ERROR on GENERATED SENTENCES
##############################################
Metric       :  BLEU    T.ERR   A.ERR
HDC          :  1.0000  0.00%   0.00%
Ref          :  1.0000  0.60%   0.30%
----------------------------------------------
This Model   :  0.0152  27.25%  22.46%
FIELNAME: Restaurant.json, BLEU: 0.015168269840996305, ERR:27.24550898203593
```

نتایج T5

علت‌های احتمالی نتایج ضعیف T5:

- تعداد epoch های کم برای pretrain
- تعداد داده‌های کمتر از مدل اصلی برای pretrain
- استفاده از condition ممکن است باعث افزایش کارایی مدل شود
- استفاده از K کوچک برای تولید جمله
- تمایل مدل به تولید جمله های طولانی تر از حداکثر قابل قبول مدل

Model	Restaurant		Laptop		Hotel		TV		Attraction		Train		Taxi	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
T5-100k	**1.52**	**27.25**	**3.32**	**40.36**	**1.58**	**36.81**	**2.18**	**32.63**	**2.05**	**35.63**	**2.57**	**37.19**	**2.17**	**32.27**

با تشکر از توجه شما

کد ها و نتایج در https://github.com/mohammadmostafarostamkhani/SC-GPT موجود است.

مدل ها و مجموعه داده در https://huggingface.co/ErfanMoosaviMonazzah موجود است.