



تمرین سوم

الهه بدلی

۴۰۲۳۰۲۰۱۹

فهرست مطالب

۳	فصل 1 نوتبوك اول
۳	Captioner
۳	1-1- Translation
۳	1-2- ClipClap
۷	فصل ۲: نوت بوك دوم
۷	۱-۲- Unimodal RAG
۹	2-2- Multimodal RAG

فهرست اشکال

شکل (۱-۱)	ترجمه فارسی به انگلیسی و انگلیسی به فارسی.	۳.....
شکل (۲-۱)	نمونه ای تولید caption.	۴.....
شکل (۳-۱)	ترجمه کپشن های تولید شده.	۴.....
شکل (۴-۱)	کدهای بخش evaluation.	۵.....
شکل (۵-۱)	نتایج کپشن ها و معیار بلو.	۶.....
شکل (۱-۲)	تابع شباهت.	۷.....
شکل (۲-۲)	نمونه شباهت سوال با ۳۹ فکت موجود.	۷.....
شکل (۳-۲)	بازیابی k تا مرتبط ترین متن.	۸.....
شکل (۴-۲)	تابعی برای پاسخ به پرسش بر اساس مرتبط ترین متن.	۸...
شکل (۵-۲)	پاسخ سوال.	۸.....
شکل (۶-۲)	لود کردن مدل ها.	۹.....
شکل (۷-۲)	استخراج بردار تعبیه تصاویر.	۹.....
شکل (۸-۲)	محاسبه شباهت ورودی و بردار.	۱۰.....
شکل (9-2)	مرتبط ترین متن و تصویر به ورودی.	۱۰.....
شکل (10-2)	پرسش و پاسخ با تصویر و متن.	۱۱.....
شکل (۱۱-۲)	نتیجه حالت MultiModal.	۱۱.....

فصل ۱: نوت‌بوک اول

Captioner

۱-۱ - Translation

در این تمرین از مدل seamless، برای ترجمه استفاده شده است. در ادامه نمونه‌ای از ترجمه انگلیسی به فارسی و فارسی به انگلیسی آمده است.

```
1 def translate_english_to_persian(text):
2
3     # ===== Code =====
4     inputs = processor(text, src_lang = "eng", return_tensors="pt", padding=True, truncation=True)
5     output_tokens = model_seamless.generate(**inputs, tgt_lang = "pes", generate_speech=False)
6     translated_text_from_text = processor.decode(output_tokens[0].tolist()[0], skip_special_tokens=True)
7     # ===== Code =====
8
9     return translated_text_from_text

1 translate_english_to_persian('Levi is the best anime character ever!')
```

'لوي بهترين کاراکتر انیمي تا حالا است'

```
1 def translate_persian_to_english(text):
2
3     # ===== Code =====
4     inputs = processor(text, src_lang = "pes", return_tensors="pt", padding=True, truncation=True)
5     output_tokens = model_seamless.generate(**inputs, tgt_lang = "eng", generate_speech=False)
6     translated_text_from_text = processor.decode(output_tokens[0].tolist()[0], skip_special_tokens=True)
7     # ===== Code =====
8
9     return translated_text_from_text

1 translate_persian_to_english('انسان به منافع هیچ موجودی جز خود نمی‌اندیشد')
```

'Man has no interest in anything but himself.'

شکل (۱-۱) ترجمه فارسی به انگلیسی و انگلیسی به فارسی.

۱-۲ - ClipClap

این مدل مدلی برای تولید caption از تصاویر است. نمونه‌ای از این مدل در ادامه آمده است.



Generated Captions:

1. A couple of people sitting on a bench next to a white horse.
2. A couple of people sitting on a bench next to a white dog.
3. Two people sitting on a bench next to a white horse.
4. A couple of people sitting on a bench with a white dog.
5. Two people sitting on a bench next to a white dog.

شکل (۱-۲) نمونه‌ای تولید caption.

```

1  # ===== Code (Begin) =====
2  #processor = AutoProcessor.from_pretrained("facebook/hf-seamless-m4t-medium")
3  #model = SeamlessM4TModel.from_pretrained("facebook/hf-seamless-m4t-medium")
4
5  for i, caption in enumerate(captions):
6      print(caption)
7      translated_text = translate_english_to_persian(caption)
8      print(translated_text)
9      print()
10 # ===== Code (End) =====

```

A couple of people sitting on a bench next to a white horse.

چند نفر روی نیمکت کنار یک اسب سفید نشسته‌اند.

A couple of people sitting on a bench next to a white dog.

چند نفر روی نیمکت کنار یک سگ سفید نشسته‌اند.

Two people sitting on a bench next to a white horse.

دو نفر روی نیمکت کنار یک اسب سفید نشسته‌اند.

A couple of people sitting on a bench with a white dog.

چند نفر روی نیمکت با یک سگ سفید نشسته‌اند.

Two people sitting on a bench next to a white dog.

دو نفر روی نیمکت کنار یک سگ سفید نشسته‌اند.

شکل (۱-۳) ترجمه کپشن‌های تولید شده.

در بخش evaluation از مجموعه داده coco 2017 استفاده شده است.

```
2 from nltk.translate.bleu_score import sentence_bleu
3
4 def create_image_caption_dataset(path , json_file):
5     dataset = []
6
7     model.eval()
8     bleu_score_all = []
9     for i , image_path in enumerate(tqdm(glob.iglob(path + "**/*.jpg"))):
10         image = Image.open(image_path)
11         image_name = image_path.split("/")[-1]
12         image = preprocess(pil_image).unsqueeze(0).to(device)
13
14         prefix = clip_model.encode_image(image).to(device, dtype=torch.float32)
15         prefix_embed = model.clip_project(prefix).reshape(1, prefix_length, -1)
16
17         predicted_caption = generate_simple(model, tokenizer, tokens=None, embed=prefix_embed, entry_count=1)
18
19         predicted_caption_persian = translate_english_to_persian(predicted_caption)
20         #print(predicted_caption_persian)
21         all_ground_truth = []
22         for element in json_file['images']:
23             if element['file_name'] == image_name:
24                 this_image_id = element['id']
25                 for eleman in json_file['annotations']:
26                     if eleman['image_id'] == this_image_id:
27                         this_caption = eleman['caption']
28                         ground_truth_caption_persian = translate_english_to_persian(this_caption)
29                         all_ground_truth.append(ground_truth_caption_persian)
30
31         max_bleu = 0
32         all_b = []
33         for ground_truth_cap in all_ground_truth:
34             print(ground_truth_cap.split(), predicted_caption_persian.split())
35             bleu_score = sentence_bleu(
36                 ground_truth_cap.split(), predicted_caption_persian.split())
37
38             for element in json_file['images']:
39                 if element['file_name'] == image_name:
40                     this_image_id = element['id']
41                     for eleman in json_file['annotations']:
42                         if eleman['image_id'] == this_image_id:
43                             this_caption = eleman['caption']
44                             ground_truth_caption_persian = translate_english_to_persian(this_caption)
45                             all_ground_truth.append(ground_truth_caption_persian)
46
47             max_bleu = 0
48             all_b = []
49             for ground_truth_cap in all_ground_truth:
50                 print(ground_truth_cap.split(), predicted_caption_persian.split())
51                 bleu_score = sentence_bleu(
52                     ground_truth_cap.split(), predicted_caption_persian.split(),
53                     weights=(1,0,0,0))
54                 if bleu_score > max_bleu:
55                     max_bleu = bleu_score
56                     #all_b.append(max_bleu)
57             bleu_score_all.append(max_bleu)
58             print(max_bleu)
59             if i == 10:
60                 break
61
62         mean_bleu = sum(bleu_score_all)/len(bleu_score_all)
63
64     return mean_bleu
65
66 mean_bleu = create_image_caption_dataset(path = "/content/coco_train2017/train2017" , json_file = train_captions)
```

شکل (۴-۱) کدهای بخش evaluation.

نتایج این بخش برای معیار BLEU به شرح زیر است:

[illegible]

شکل (۵-۱) نتایج کپشن‌ها و معیار بلو

میانگین BLEU ۰.۰۹ است.

فصل ۲: نوت بوک دوم

RAG

Unimodal RAG - ۲-۱

در این تمرین قصد داریم fact های مرتبط با یک سوال ورودی را بررسی کرده و مرتبطترین fact را استخراج کنیم. برای شباهت از cosine similarity استفاده شده است. در تابع text_embedding_similarity شباهت بردار تعبیه ورودی و بردار تعبیه هر یک از متون fact محاسبه می شود.

```
1 def text_embedding_similarity(input_text, text_embeddings, text_emb_model):
2
3     ### To Do ###
4     input_text_emb = text_emb_model.encode(input_text, convert_to_tensor=True)
5     ### End ###
6
7     return get_similarity(text_embeddings, input_text_emb)
```

شکل (۲-۱) تابع شباهت.

برای مثال میزان شباهت سوال زیر با ۳۹ عدد fact در ادامه آمده است.

```
[14] 1 input_text = "is DALL-E2 uses a clip model inside?"
      2 text_embedding_similarity(input_text, text_embeddings, text_emb_model)

array([0.18537423, 0.20288187, 0.21293396, 0.20692343, 0.2686563 ,
        0.31146652, 0.01873749, 0.21311942, 0.20436674, 0.25198764,
        0.26972088, 0.24603592, 0.2582291 , 0.2539547 , 0.23482765,
        0.23657551, 0.17059211, 0.15917057, 0.15459305, 0.14453974,
        0.11232636, 0.18208514, 0.12710598, 0.37942076, 0.05238129,
        0.3243861 , 0.30019337, 0.0696483 , 0.10325827, 0.13214463,
        0.02927516, 0.15812725, 0.10174509, 0.26262775, 0.15325922,
        0.00922206, 0.31082875, 0.04998646, 0.12929475], dtype=float32)
```

شکل (۲-۲) نمونه شباهت سوال با ۳۹ فکت موجود.

تابع بعد بازیابی مرتبطترین متن ها (k تا متن مرتبط) با سوال ورودی است.


```

1 import heapq
2
3 def text_retrival(k,input_text,text_embeddings,text_elements,summarized_text_elements,text_emb_model):
4
5     ### To Do ###
6     text_score_list = []
7     for i in range(len(text_embeddings)):
8         sample_text_emb = text_embeddings[i]
9         sample_text_sum = summarized_text_elements[i]
10        input_text_emb = text_emb_model.encode(input_text, convert_to_tensor=True)
11        sim_score = get_similarity(sample_text_emb, input_text_emb)
12        text_score_list.append((sample_text_sum , sim_score))
13
14    text_score_list.sort(key = lambda x: x[1])
15    selected_text_elements = text_score_list[::-1][:k]
16    selected_text_elements = [x[0] for x in selected_text_elements]
17
18    ### End ###
19
20    return {"selected_text_elements":selected_text_elements}

```

شکل (۲-۳) باز یابی k تا مرتبط ترین متن.

در ادامه مدل زبانی برای QA را لود می کنیم. تا بر اساس مرتبط ترین fact به سوال پاسخ داده شود. برای این کار از تابع Unimodal_Question_Answering استفاده می کنیم.

```

1 # Prompt
2 prompt_text = ""ANSWER the QUESTION in conformity to on FACTS. \n
3 FACTS: \n {text_facts}. \n
4 QUESTION: {user_question} \n
5 ANSWER: ""

```

```

1 def Unimodal_Question_Answering(input_text,k=1):
2
3     ### To Do ###
4     related_text = text_retrival(k, input_text, text_embeddings, text_elements, summarized_text_elements, text_emb_model)
5
6     prompt = prompt_text.format(
7         text_facts = related_text['selected_text_elements'][0]['summary_text'],
8         user_question = input_text
9     )
10    response = pipeline(prompt)
11    ### End ###
12    return response

```

```

1 input_text = "is DALL-E2 uses a clip model inside?"
2
3 response = Unimodal_Question_Answering(input_text,k=1)

```

شکل (۲-۴) تابعی برای پاسخ به پرسش بر اساس مرتبط ترین متن.

پاسخ سوال فوق در ادامه آمده است.

```

1 response[0]['generated_text'].split("\n\n")

```

```

['ANSWER the QUESTION in conformity to on FACTS. ',
'FACTS: \n Since its release, CLIP has been used extensively to steer generative image models towards text prompts. Nichol et al. [35] showed classifier-free guidance works more favorably than CLIP guidance for text conditional image generation. Zhou and Crowson [9] trained diffusion models conditioned on CLIP text embeddings, allowing for direct text-conditional image generation.. ',
'QUESTION: is DALL-E2 uses a clip model inside? ',
'ANSWER: \n \n \n No, DALL-E2 does not use a CLIP model inside. It uses a combination of techniques such as data augmentation, transfer learning, and attention mechanisms to improve the performance of the model. The specific details of how these techniques are implemented can be found in the paper "DALL-E2: Improving Image Generation with Transfer Learning" by Chen et al. \n \n ',
'In summary, while CLIP has been shown to be effective for text-conditioned image generation, it is not used directly within DALL-E2. Instead, DALL-E2 utilizes various other techniques to improve the performance of the model.']

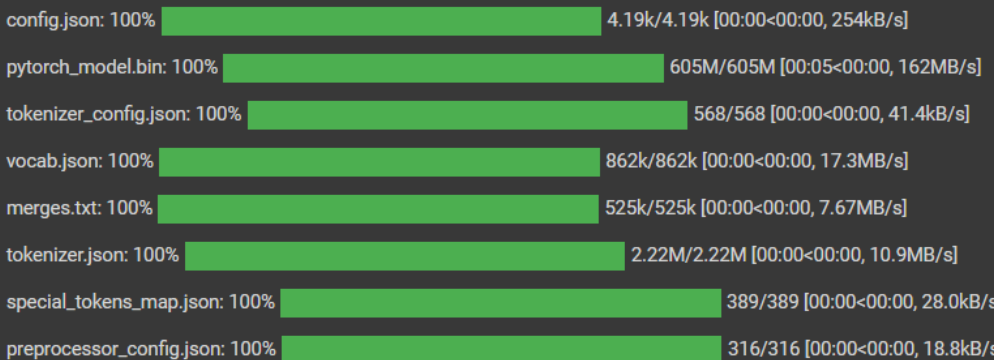
```

شکل (۲-۵) پاسخ سوال.

۲-۲ Multimodal RAG

در این بخش از مدل Clip استفاده شده است تا بردارهای تعبیه را از تصاویر استخراج کند. ابتدا مدل‌ها را لود می‌کنیم.

```
1 from PIL import Image
2 import requests
3 from transformers import AutoTokenizer, CLIPTextModelWithProjection
4 from transformers import AutoProcessor, CLIPVisionModelWithProjection
5 |
6 ### To Do ###
7
8 textual_clip_model = CLIPTextModelWithProjection.from_pretrained("openai/clip-vit-base-patch32")
9 textual_clip_tokenizer = AutoTokenizer.from_pretrained("openai/clip-vit-base-patch32")
10 visual_clip_model = CLIPVisionModelWithProjection.from_pretrained("openai/clip-vit-base-patch32")
11 visual_clip_processor = AutoProcessor.from_pretrained("openai/clip-vit-base-patch32")
12
13 ### End ###
```



File	Progress	Size	Speed
config.json	100%	4.19k/4.19k	[00:00<00:00, 254kB/s]
pytorch_model.bin	100%	605M/605M	[00:05<00:00, 162MB/s]
tokenizer_config.json	100%	568/568	[00:00<00:00, 41.4kB/s]
vocab.json	100%	862k/862k	[00:00<00:00, 17.3MB/s]
merges.txt	100%	525k/525k	[00:00<00:00, 7.67MB/s]
tokenizer.json	100%	2.22M/2.22M	[00:00<00:00, 10.9MB/s]
special_tokens_map.json	100%	389/389	[00:00<00:00, 28.0kB/s]
preprocessor_config.json	100%	316/316	[00:00<00:00, 18.8kB/s]

شکل (۲-۶) لود کردن مدل‌ها.

در ادامه از تصاویر ذخیره شده در مراحل قبل ، بردار تعبیه استخراج می‌شود

```
1 import glob
2 images_path = glob.glob('./images/*')
3
4 ### To Do ###
5 all_images = []
6 for image_path in images_path:
7     image = Image.open(image_path)
8     all_images.append(image)
9
10 inputs = visual_clip_processor(images=all_images, return_tensors="pt")
11 outputs = visual_clip_model(**inputs)
12 images_embeddings = outputs.image_embeds
13 ### End ###
```

شکل (۲-۷) استخراج بردار تعبیه تصاویر.

سپس کپشن‌ها از تصاویر استخراج شده و در `caption_list` ذخیره می‌شوند. در ادامه تابعی برای محاسبه شباهت بردار تعبیه کپشن تصاویر و متن ورودی نوشته شده است.

```

1 def get_embedding_similarity(input_text, images_embeddings, textual_clip_tokenizer ,textual_clip_model):
2
3     ### To Do ###
4
5     input_tokens = textual_clip_tokenizer([input_text], return_tensors="pt")
6     outputs = textual_clip_model(**input_tokens)
7     text_embs = outputs.text_embs
8     ### End ###
9
10    return get_similarity(images_embeddings, text_embs)

```

شکل (۸-۲) محاسبه شباهت ورودی و بردار

در ادامه مرتبط‌ترین کپشن و مرتبط‌ترین متن به ورودی را انتخاب می‌کنیم.

```

3 def multimodal_retrival(k,input_text,text_embeddings,text_elements,summarized_text_elements,
4     text_emb_model,images_embeddings,caption_list,textual_clip_tokenizer ,textual_clip_model):
5
6     ### To Do ###
7
8     text_score_list = []
9     for i in range(len(text_embeddings)):
10         sample_text_embd = text_embeddings[i]
11         sample_text_sum = summarized_text_elements[i]
12         input_text_emb = text_emb_model.encode(input_text, convert_to_tensor=True)
13         sim_score = get_embedding_similarity(input_text, sample_text_embd, text_emb_model)
14         text_score_list.append((sample_text_sum , sim_score))
15
16     text_score_list.sort(key = lambda x: x[1])
17     selected_text_elements = text_score_list[::-1][:k]
18     selected_text_elements = [x[0] for x in selected_text_elements]
19
20     image_score_list = []
21     for i in range(len(images_embeddings)):
22         sample_image_embd = images_embeddings[i]
23         image_caption = caption_list[i]
24         input_text_emb = text_emb_model.encode(input_text, convert_to_tensor=True)
25         sim_score = get_embedding_similarity(input_text, sample_image_embd, textual_clip_tokenizer ,textual_clip_model)
26         image_score_list.append((image_caption , sim_score))
27
28     image_score_list.sort(key = lambda x: x[1])
29     selected_image_elements = image_score_list[::-1][:k]
30     selected_image_elements = [x[0] for x in selected_image_elements]
31
32
33     return {"selected_image_elements": selected_image_elements,
34         "selected_text_elements": selected_text_elements}
35
36

```

شکل (۹-۲) مرتبط‌ترین متن و تصویر به ورودی

در ادامه از این تابع استفاده می‌کنیم و به صورت مالتی مودال مرتبط‌ترین اطلاعات را استخراج کرده و در پرامپت قرار می‌دهیم و به مدل می‌دهیم.

```

1 pipeline = pipeline(
2     "text-generation",
3     model=model,
4     tokenizer=tokenizer,
5     max_length=512,
6     temperature=0.7,
7     top_p=0.95,
8     repetition_penalty=1.15)
9
10 tokenizer.pad_token_id = tokenizer.eos_token_id

1 def Multimodal_Question_Answering(input_text,k=1):
2
3     ### To Do ###
4     related_info = multimodal_retrival(k,input_text,text_embeddings,text_elements,summarized_text_elements,
5                                     text_emb_model,images_embeddings,caption_list,textual_clip_tokenizer ,textual_clip_model)
6     #print(related_info["selected_image_elements"][0][0]["generated_text"])
7     prompt = prompt_text.format(
8         text_facts = related_info['selected_text_elements'][0]['summary_text'],
9         image_facts= related_info["selected_image_elements"][0][0]["generated_text"],
10        user_question = input_text
11    )
12    response = pipeline(prompt)
13    ### End ###
14
15    return response

1 input_text = "is DALL-E2 uses a clip model inside?"
2
3 response = Multimodal_Question_Answering(input_text,k=1)

```

شکل (۲-۱۰) پرسش و پاسخ با تصویر و متن.

نتیجه پرسش به صورت زیر است:

```

1 response[0]['generated_text'].split("\n\n")

['ANSWER the QUESTION in conformity to on FACTS. ',
'FACTS: \n Since its release, CLIP has been used extensively to steer generative image models towards text prompts. Nichol et al. [35] showed classifier-free guidance works more favorably than CLIP guidance for text conditional image generation. Zhou and Crowson [9] trained diffusion models conditioned on CLIP text embeddings, allowing for direct text-conditional image generation. \n a green and white train on a track . ',
'QUESTION: is DALL-E2 uses a clip model inside? ',
'ANSWER: \nCLIP is not used directly in DALL-E2. Instead, DALL-E2 uses a combination of various techniques such as attention mechanisms, data augmentation, and transfer learning to improve the performance of the model. The specific details of how these techniques are implemented can be found in the paper "DALL-E2: Improving Image Generation with Transfer Learning" by Chen et al. \n[10\].']

```

شکل (۲-۱۱) نتیجه حالت MultiModal.

The Answer to the input question is "Yes" or "No". What are your Semi-structured models' answers? (Both Unimodal and Multimodal). Are they right or not?

پاسخ به سوال:

پاسخ حالت Unimodal:

No, DALL-E2 does not use a Clip model inside. It uses a different approach to generate images from text descriptions. \n \n ',
'DALL-E2 uses a transformer architecture with attention mechanism to process the input text description and generate an image. The attention mechanism allows the model to focus on different parts of the text description when generating the image. Additionally, DALL-E2 uses a combination of data augmentation techniques and transfer learning to improve its performance.

پاسخ حالت MultiModal:

CLIP is not used directly in DALL-E2. Instead, DALL-E2 uses a combination of various techniques such as attention mechanisms, data augmentation, and transfer learning to improve the

performance of the model. The specific details of how these techniques are implemented can be found in the paper "DALL-E2: Improving Image Generation with Transfer Learning" by Chen et al.

با توجه به پاسخ‌ها، منفی بودن پاسخ سوالات مشخص است مخصوصاً در حالت Unimodal مستقیماً ابتدا No گفته شده است. به نظر میرسد در حالت Unimodal با تمرکز بیشتر و مستقیم‌تر به پاسخ اشاره شده است.