پروژه پایانی درس یادگیری عمیق

کورش تقی‌پور پاسدار

۱۲ بهمن ۱۴۰۳

فهرست مطالب

۱ Preprocess

در این بخش، به import کردن کتابخانه‌های لازم و همچنین دانلود دیتاست Flickr8k و import کردن ترجمه‌ای که قبلا انجام داده‌ایم می‌پردازیم.

۱.۱ Import Libraries

```
[ ]  import torch
     import torch.nn as nn
     import clip
     import requests
     import os
     import cv2
     import glob
     import zipfile
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt

     from statistics import mean
     from PIL import Image
     from tqdm import tqdm
     from torchvision.transforms import ToPILImage
```

۲.۱ Preprocess Flickr8k Dataset

۱.۲.۱ Download and Unzip Dataset

در ابتدا تابعی برای دانلود و ذخیره دیتاست تعریف می‌کنیم.

۲.۲.۱ Read and Save Captions

سپس به خواندن و استخراج کپشن‌ها از این دیتاست می‌پردازیم.

```
[ ]  captions_flickr = []
     with open("datasets/flickr/captions.txt", 'r') as file:
         for line in file:
             img_name, caption = line.strip().split(',', 1)
             captions_flickr.append((img_name, caption))
     captions_flickr = captions_flickr[1:]
```

```
[ ]  captions = pd.DataFrame(captions_flickr, columns=['image', 'caption_en'])
     captions['id'] = range(len(captions))

     captions.head()
```

	image	caption_en	id
0	1000268201_693b08cb0e.jpg	A child in a pink dress is climbing up a set o...	0
1	1000268201_693b08cb0e.jpg	A girl going into a wooden building .	1
2	1000268201_693b08cb0e.jpg	A little girl climbing into a wooden playhouse .	2
3	1000268201_693b08cb0e.jpg	A little girl climbing the stairs to her playh...	3
4	1000268201_693b08cb0e.jpg	A little girl in a pink dress going into a woo...	4

Extract Translated Captions and Merge ۳.۲.۱

سپس کپشن‌های فارسی ترجمه شده از قبل را از فایل‌ها خوانده و با کپشن‌های انگلیسی در یک دیتافریم [1] قرار می‌دهیم.

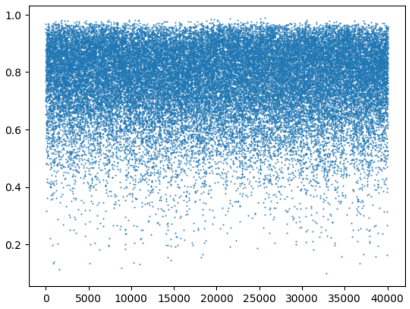
```
[ ]  files_path = '/content/drive/MyDrive/'

     fa_captions = []
     for i in range(1,7):
         data = pd.read_excel(''.join([files_path, f'flickr8k_translated_captions{i}.xlsx']))
         fa_captions.append(data)

     merged_df = pd.concat(fa_captions, axis=0).dropna(subset=['caption_fa'])
     merged_df = merged_df.drop_duplicates(subset = ['image','caption_en'])
     merged_caps = pd.merge(captions, merged_df, on=['image','caption_en'],how='inner')
     merged_caps.head()
```

	image	caption_en	id	caption_fa
0	1000268201_693b08cb0e.jpg	A child in a pink dress is climbing up a set o...	0	...کودکی با لباس صورتی در حال بالا رفتن از مجموعه
1	1000268201_693b08cb0e.jpg	A girl going into a wooden building .	1	.دختری که وارد یک ساختمان چوبی می شود
2	1000268201_693b08cb0e.jpg	A little girl climbing into a wooden playhouse .	2	...دختر بچه ای در حال بالا رفتن از یک خانه بازی چ
3	1000268201_693b08cb0e.jpg	A little girl climbing the stairs to her playh...	3	...دختر بچه ای که از پله ها به سمت خانه بازی اش ب
4	1000268201_693b08cb0e.jpg	A little girl in a pink dress going into a woo...	4	...دختر بچه ای با لباس صورتی که وارد یک کابین چوب

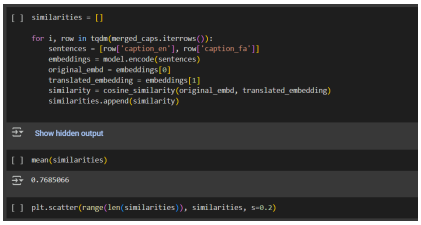
Check Translation Validity ۴.۲.۱

سپس با استفاده از یک مدل هوش مصنوعی Sentence Transformer به مقایسه و بررسی کیفیت ترجمه می‌پردازیم.

[1] DataFrame



(د) رسم بصری هر یک از مقادیر تطابق



(ج) بررسی کیفیت ترجمه داده‌ها و نمایش میانگین تطابق

۲ Define Model

حال در اینجا به تعریف مدل می‌پردازیم. برای این کار، از مدل CLIP استفاده کرده‌ایم و به تعریف دقیق‌تر، یک مدل pretrain شده را استفاده می‌کنیم که سعی می‌کنیم با Fine-tune کردن آن، برای تسک مورد نظر آموزش دهیم.

```
[ ] from transformers import CLIPProcessor, CLIPModel

[ ] import os
    os.environ['HF_TOKEN'] = # Inser your HF_TOKEN

    from huggingface_hub import login
    login(token=os.getenv('HF_TOKEN'))

[ ] model = CLIPModel.from_pretrained("openai/clip-vit-base-patch32")
    processor = CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")
```

۳ Define Dataset

در اینجا، دیتاست خود را تعریف می‌کنیم. منظور از تعریف دیتاست، تعریف **کلاس** [۲] دیتاست است تا داده‌ها را از آن بگیریم.

[2] Class

```python
class ImageCaptionDataSet(Dataset):
    def __init__(self, dataset, path, processor):
        self.dataset = dataset
        self.processor = processor
        self.image_folder_path = path

    def __len__(self):
        return len(self.dataset)

    def __getitem__(self, idx):
        item = self.dataset.iloc[idx]
        caption = item['caption_fa']
        image = Image.open(''.join([self.image_folder_path, item['image']]))
        inputs = self.processor(text=[caption],
                                images=image,
                                return_tensors="pt",
                                padding=True,
                                truncation=True,
                                max_length=77)
        input_ids = inputs['input_ids'].squeeze(0)
        attention_mask = inputs['attention_mask'].squeeze(0)
        max_text_len = 77
        input_ids_padded = torch.nn.functional.pad(input_ids,
         (0, max_text_len - input_ids.size(0)),
                                                   value=processor.tokenizer.pad_token_id)
        attention_mask_padded = torch.nn.functional.pad(attention_mask, (0, max_text_len - attention_mask.size(0)))
        pixel_values = inputs['pixel_values'].squeeze(0)
        return {
            'input_ids': input_ids_padded,
            'attention_mask': attention_mask_padded,
            'pixel_values': pixel_values
        }
```

سپس داده‌های آموزشی و تست را با استفاده از train test split از کتابخانه sklearn جدا می‌کنیم.

```python
[ ] x_train, x_test,y_train, y_test = train_test_split(merged_caps[['caption_fa', 'id']],
                                      merged_caps[['image', 'id']],
                                      random_state=42,
                                      test_size=0.2,
                                      shuffle=True)

    path = '/content/datasets/flickr/Images/'
```

```python
[ ] train_dataset = ImageCaptionDataSet(pd.merge(x_train, y_train,on='id',how='inner' ), path, processor)
    test_dataset = ImageCaptionDataSet(pd.merge(x_test, y_test, on='id',how='inner'), path, processor)

    train_dataloader = DataLoader(train_dataset, batch_size=32, shuffle=True, num_workers=2)
    test_dataloader = DataLoader(test_dataset, batch_size=32, shuffle=True, num_workers=2)
```

۴ Fine-Tune

حال به بخش Fine-Tune مدل می‌رسیم.

۱.۴ Freeze Some Layers

با توجه به اینکه این یک شبکه‌ی از پیش آموزش دیده است، پس توانایی برقراری ارتباط بین تصویر و متن را دارد. لازم است که بخش‌های استخراج ویژگی از تصویر و همچنین برقراری ارتباط بین ویژگی‌های تصویر و متن ثابت مانده و تنها استخراج ویژگی از متن تغییر کند. به همین دلیل، به فریز کردن بخش بزرگی از شبکه می‌پردازیم.

```
[ ] for param in model.parameters():
        param.requires_grad = False

    for param in model.text_model.embeddings.token_embedding.parameters():
        param.requires_grad = True

[ ] optimizer = torch.optim.Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=1e-2)
    loss_fn = nn.CrossEntropyLoss()
    device = "cuda" if torch.cuda.is_available() else "cpu"

[ ] device
```

همچنین از بهینه‌ساز Adam و تابع CrossEntropyLoss برای محاسبه خطا استفاده می‌کنیم. سپس به تعریف تابع train می‌پردازیم.

سپس به آموزش مدل می‌پردازیم.

```
[ ] def save_model(model):
        torch.save(model.state_dict(), "/content/drive/MyDrive/model_parameters.pth")

[ ] train_loss1, val_loss1 = train_model(model, train_dataloader, test_dataloader, optimizer, loss_fn, 10, device)
    optimizer = torch.optim.Adam(filter(lambda p: p.requires_grad, model.parameters()), lr=1e-3)
    loss_fn = nn.CrossEntropyLoss()
    device = "cuda" if torch.cuda.is_available() else "cpu"
    train_loss2, val_loss2 = train_model(model, train_dataloader, test_dataloader, optimizer, loss_fn, 10, device)
    save_model(model)
```

۵ Import Evaluation Data

حال در اینجا به ویدیوهای تست، استخراج فریم‌ها و encode کردن آنها به آرایه‌های تنسور [۳] می‌پردازیم.

[3] Tensor Array

```python
def video_frame_extract_get_fps(video_path, frames_dir):
    os.makedirs(frames_dir, exist_ok=True)

    cap = cv2.VideoCapture(video_path)
    fps = int(cap.get(cv2.CAP_PROP_FPS))
    step = 1 if fps < 2 else int(fps / 2)
    frame_count = 0
    frame_step = 0
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        frame_filename = os.path.join(frames_dir, f"frame_{frame_count:04d}.jpg")
        frame_step += 1
        if frame_step < step:
            continue

        cv2.imwrite(frame_filename, frame)
        frame_step = 0
        frame_count +=1

    cap.release()
```

(ز) تابع استخراج فریم‌های یک ویدیو

```python
def extract_frames_from_videos(videos_folder, output_folder):
    os.makedirs(output_folder, exist_ok=True)
    video_files = glob.glob(os.path.join(videos_folder, '*.mp4'))

    for video_file in video_files:
        video_name = os.path.splitext(os.path.basename(video_file))[0]
        video_frames_folder = os.path.join(output_folder, video_name)
        os.makedirs(video_frames_folder, exist_ok=True)
        video_frame_extract_get_fps(video_file, video_frames_folder)
    print(f"Extracted frames from {len(video_files)} videos.")
```

(ح) تابعی برای استخراج فریم‌های تمام ویدیوهای داخل یک پوشه

```python
def encode_frames(frames_dir):
    # List to store encoded frames
    encoded_frames = []

    # Iterate over each frame image
    for frame_file in sorted(os.listdir(frames_dir)):
        if frame_file.endswith('.jpg'):
            frame_path = os.path.join(frames_dir, frame_file)
            image = Image.open(frame_path)

            # Process the image to the required format
            inputs = processor(images=image, return_tensors="pt")

            # Get the pixel values
            pixel_values = inputs['pixel_values'].squeeze(0)  # Remove batch dimension
            encoded_frames.append(pixel_values)

    # Stack the encoded frames into a single tensor
    encoded_frames_tensor = torch.stack(encoded_frames)

    return encoded_frames_tensor
```

(ط) تابعی برای encode کردن ویدیوها و تبدیل آنها به یک Tensor

```python
def preprocess_videos(video_folder, output_folder, frame_folder):
    frame_list = []
    extract_frames_from_videos(video_folder, output_folder)
    os.makedirs(frame_folder, exist_ok=True)
    for _, folder in tqdm(enumerate(os.listdir(output_folder))):

        frames = encode_frames(''.join([output_folder, folder]))
        torch.save(frames, f"{frame_folder}{folder}.pt")
```

(ی) تابعی برای استخراج tensor های تمام ویدیوها. این تابع دربردارنده تابع‌های قبلی است.

```python
def load_frame_tensor(frame_folder):
    frame_array = []
    for frame in sorted(os.listdir(frame_folder)):
        frame_path = ''.join([frame_folder, frame])
        frame_array.append(torch.load(frame_path))
    return torch.concat(frame_array)
```

(ک) تابعی بری لود کردن tensor ها

۶ Text-to-Image Retrieval

حال به تابع یافتن فریم ویدیو براساس متن ورودی می‌پردازیم. در این تابع براساس متن ورودی و آرایه‌ای که در قبل از ویدیوها ساختیم، بهترین k فریم را پیدا می‌کنیم. این مقدار قابل تغییر است.

```python
def text_image_retrieval(model, processor, text, frames_tensor, top_k=5, device="cpu"):
    model = model.to(device)
    model.eval()
    frames_tensor = frames_tensor.to(device)
    text_inputs = processor(text=[text], return_tensors="pt", padding=True,
                            truncation=True, max_length=77).to(device)
    text_features = model.get_text_features(**text_inputs)
    frame_features = model.get_image_features(pixel_values=frames_tensor)

    text_features = text_features / text_features.norm(dim=-1, keepdim=True)
    frame_features = frame_features / frame_features.norm(dim=-1, keepdim=True)

    # Similarities
    similarities = torch.matmul(text_features, frame_features.T)

    top_k_indices = similarities.topk(top_k, dim=1).indices.squeeze(0).tolist()

    for i, idx in enumerate(top_k_indices):

        most_similar_frame = frames_tensor[idx].cpu()

        mean = torch.tensor([0.485, 0.456, 0.406])
        std = torch.tensor([0.229, 0.224, 0.225])

        # Convert the tensor back to a PIL image
        denormalized_pixel_values = most_similar_frame * std[:, None, None] + + mean[:, None, None]
        denormalized_pixel_values = torch.clamp(denormalized_pixel_values, 0, 1)
        image = ToPILImage()(denormalized_pixel_values)

        plt.figure()
        plt.imshow(image)
        plt.title(f"Top {i+1} Similar Frame.")
        plt.axis('off')  # Hide axes for better visualization
    plt.show()
```