Information Retrieval and Extraction HW3

312657008_ 江祐姍

1. What kind of pre-processing did you apply to the photo or dialogue text? Additionally, please discuss how different preprocessing methods affected the performance of the models?

分別採用了兩種方法做預處理

(1) PCA

```
# List to store image features
image_embeddings = []

# Extract features for all images
for image_path in image_paths:
    # Read and preprocess image
    image = Image.open(image_path).convert('RGB')
    image_tensor = transform(image).unsqueeze(0).to(device)  # Add batch dimension and move to GPU

    # Get image features
    with torch.no_grad():
        image_features = resnet_model(image_tensor).flatten().cpu().numpy()  # Move back to CPU for

    # Add image features to list
    image_embeddings.append(image_features)

# Convert image features to NumPy array
image_embeddings = np.array(image_embeddings)

# 3. Dimensionality reduction to match text embedding dimension
# We'll use PCA to reduce image features to 384 dimensions
pca = PCA(n_components=384)
image_embeddings_reduced = pca.fit_transform(image_embeddings)
```

這段代碼的目的是處理多張圖像，並將它們轉換為 384 維的嵌入向量（特徵）。這些嵌入向量可以用於與文本特徵進行比較或其他下游任務。在過程中，使用了以下幾個重要步驟：

預處理圖像（如縮放、裁剪、標準化）。

使用 ResNet 提取圖像的特徵。

使用 PCA 對特徵進行降維，以匹配文本特徵的維度

(2)open_clip

使用 open_clip.get_tokenizer 來將文本轉換為模型所需的 token 格式，這樣可以確保文本進入模型時的結構和形式是正確的。這種 tokenization 過程有助於提取出文本中的語言特徵，使得模型能夠理解文本的含義。

圖像預處理：適當的圖像預處理能顯著提升模型的性能，特別是圖像大小調整和標準化，這些步驟能夠幫助模型有效地提取圖像特徵。如果圖像預處理不當，模型可能無法學到有效的特徵，從而降低其準確率。

文本預處理：tokenization

使得文本能夠更精確地轉換為詞嵌入，進而提高模型對文本理解的效果。如果預處理過程中對文本的標準化處理不夠嚴格，模型可能會錯誤理解文本中的語言特徵，導致表現不佳。

然而，經由嘗試數次後，發現方法一二搭配著不同的方式以方法二的結果較好，因此改採用 open_clip 去做預測。

2. How did you align the photo and dialogue text in the same embedding space? Use pretrained model or train your own?

在這部分我透過 CLIP (Contrastive Language-Image Pretraining) 模型，並使用了 pretrain model。

```
model, _, preprocess = open_clip.create_model_and_transforms('ViT-B-32', pretrained='laion2b_s34b_b79k')
model = model.to(device)
tokenizer = open_clip.get_tokenizer('ViT-B-32')
```

```
model, _, preprocess = open_clip.create_model_and_transforms(
        'ViT-B-32',
        pretrained='datacomp_xl_s13b_b90k'
    )
model = model.cuda().eval()
tokenizer = open_clip.get_tokenizer('ViT-B-32')
```

```
model, _, preprocess = open_clip.create_model_and_transforms('ViT-B-32', pretrained='laion400m_e32')
model = model.to(device)
tokenizer = open_clip.get_tokenizer('ViT-B-32')
```

```
model, _, preprocess = open_clip.create_model_and_transforms('ViT-L-14', pretrained='laion2b_s32b_b82k')
model = model.cuda()
tokenizer = open_clip.get_tokenizer('ViT-L-14')
```

```
model, _, preprocess = open_clip.create_model_and_transforms('ViT-L-14', pretrained='openai')

model = model.to(device)
tokenizer = open_clip.get_tokenizer('ViT-L-14')
```

3.Please discuss based on your experimental results. How do you improve the performance of your model? (e.g. add a module or try different models and observing performance changes). What was the result?

在這部分如上方第二題的程式碼所示我使用了不同的 modek 及 pre-train 得到了不同的準確性

ViT-B-32 架構的模型（如 laion2b_s34b_b79k 和 datacomp_xl_s13b_b90k）的表現較好，尤其是 laion2b_s34b_b79k。

ViT-L-14 模型（laion2b_s32b_b82k）表現更強，因為它具有更多的參數和更強的特徵提取能力，從而在大規模資料集上獲得了較好的表現

ViT-B-32 模型（laion400m_e32）的表現稍弱，可能是因為它的訓練資料集較小，無法捕捉到更多樣化的語意。

最差的為 ViT-L-14 模型 (openai)。

這些結果表明，模型架構（如 ViT-B-32 vs ViT-L-14）和訓練資料集的大小對最終性能有顯著影響。