인공지능프로젝트 Final Project Report

2017313008 김태은

1. Implementation

A. Dataset Preprocessing

ImgCaptionDataset

(image, caption) pair를 만들기 위해 ImgCaptionDataset이라는 커스텀 데이터셋을 만든다.

먼저 __init__에서는 image 경로, caption 경로, transform을, tokenize_func을 인자로 받고 데이터셋 전처리를 한다. Img2caption은 각 image의 파일명을 key 값으로 하고 그에 상응하는 caption들의 list를 value 값으로 가지는 dictionary이다. image 마다 caption의 개수가 다른데, 각 (image, caption) pair를 batch size 만큼 불러서 training 할 때 문제가 발생하므로 max_captions_num을 설정하여 __getitem__에서 image 마다 같은 개수의 caption을 가지도록 조정해준다. caption_vocab 는 Vocabulary() class를 이용하여 caption의 vocabulary를 구축한다. captions.json 파일에는 각 이미지의 파일명과 caption이 모두 존재하므로 순회하면서 img2caption에 (image, caption) pair를 저장한다. 이 때 caption은 여러 개가 될 수 있으므로 list 형태로 들어간다. max_seq_length 은 caption들 중 가장 긴 길이를 가리키며, caption을 추론할 때 종료조건으로서 활용할 수 있다.

Vocabulary

Vocabulary class는 caption들에 대한 vocabulary를 구축한다. 총 4가지 고정 토큰을 사용하여 문장에서 시작, 끝, 패딩, 보류 상태를 확인할 수 있게 했다. word2index는 key값이 word, value값이 index인 dictionary이고, index2word는 key값이 index, value값이 word인 dictionary이다. 여기서 frequencies를 만들어 어떤 단어가 threshold이상 출현하였을 때만 vocabulary 사전(word2index, index2word)에 등록될 수 있도록 했다. 또한 새로운 caption이 add_sentence에 들어왔을 때, sentence를 tokenize하여 add_word를 호출하면 frequency등을 확인한 후 사전에 추가한다. 여기서 caption들 중 가장 긴 길이를 업데이트하여 이후 caption inference 시에 종료 조건으로 활용한다.

caption vocabulary class는 파일로 저장하여 test.py 에서 활용할 수 있도록 한다.

dataset 생성 이후 train/test 과정에서 batch size만큼씩 가져오기 위해 trainloader, validloader를 생성한다. 이 때 전체 이미지 데이터셋에서 10%를 validation set으로 한다.

B. Model Construction

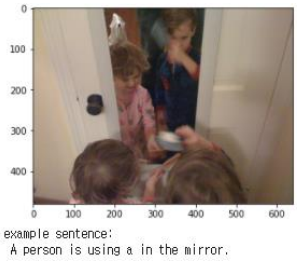
Img2Cap 모델은 encoder(CNN) – decoder(LSTM) 으로 이루어져있다. encoder 에서는 pretrained CNN 모델으로 resnet152 를 사용하였다. 이후 embedding 을 수행하여 encoder 의 output vector 차원을 embed_size=256 으로 맞춘다. Encoder 를 통해 이미지의 특징 추출이 이루어진다. 이후 decoder 에서는

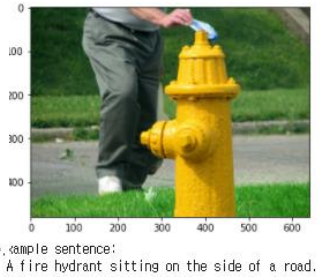
embedding 된 caption 과 image feature vector 를 합치고, lstm 을 수행한다. 마지막으로 decoder 의 output vector 차원을 단어 개수로 맞춰준다. 학습이 끝난 후, image 에 대한 caption inference 과정에서는 Img2Cap 모델의 make_caption 함수를 사용한다. 이 함수는 image 에 대해 encoding 을 수행해 특징 추출을 한 후, 이 vector 를 decoder 의 lstm 에 넣는다. 이후에는 계속해서 lstm 의 결과로 나온 vector 에서 max 값을 가지는 index 를 찾아 결과 list 에 append 하고, 이 index 를 다시 lstm 의 input 으로 넣어서 다음 단어를 예측한다. Caption 의 최대길이만큼 이 과정을 반복한다. 이 때, [UNK] token 이 예측된다면 무시하고, [EOS] token 이 등장하면 추론 과정을 멈춘다. 마지막으로 vocabulary.index2word 를 사용하여 index 를 단어로 바꿔준 후 이 리스트를 반환한다.

C. Training

training은 N_EPOCHS만큼 수행하며 각 epoch 마다 train, validation을 수행한다. 각 image마다 caption의 수가 여러 개이므로 captions.transpose(0, 1)을 통해 [batch_size, caption_num, vocab_size] 를 [caption_num, batch_size, vocab_size]로 바꾼 후, caption_num만큼 반복문을 돌며 각 caption들이 모두 학습될 수 있게 했다. Validation 과정에서도 동일하다. Best model은 valid_loss가 이제껏 나왔던 loss들보다 작을 경우에 저장된다. 또한 valid_loss를 metric으로 하는 scheduler를 이용해 learning rate를 조정한다. 마지막으로 patience 변수를 통해 early stopping을 수행한다.

주어진 이미지 데이터셋을 (test: 0.05, valid: 0.1, train: 0.85)비율로 나누어 모델을 테스트해보았다.

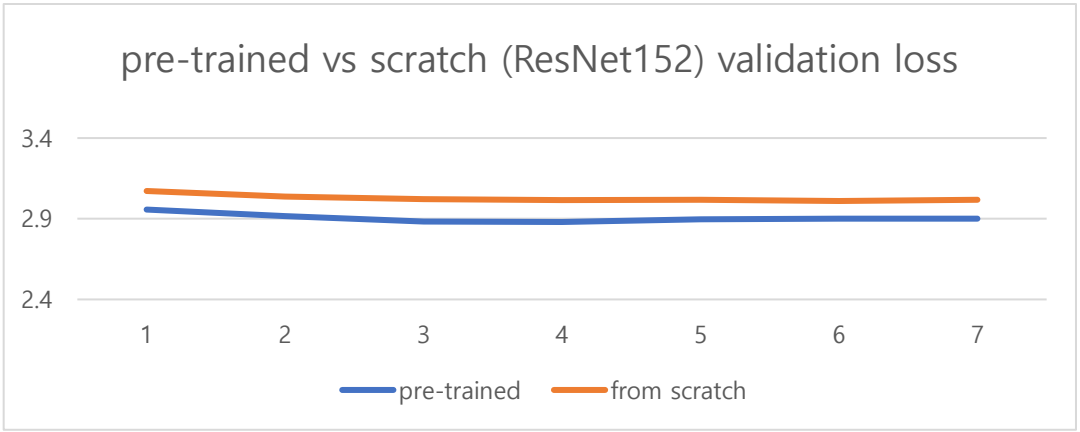




Image에 대한 caption 추론이 잘 되는 것을 확인할 수 있다.

Pre-trained vs from scratch 성능 비교

models.resnet152(pretrained=True) & models.resnet152(pretrained=False)



Pre-trained 버전이 validation loss가 더 낮게 나오는 것을 확인할 수 있다. 따라서 pre-trained의 성능이 조금 더 좋다.

2. What I have tried, studied

1. 각 이미지마다 여러 개의 caption이 있다. 따라서 각 이미지를 이 caption의 개수만큼 반복해서 학습한다. caption을 하나만 골라서 학습하는 방식이나, 모든 caption을 하나의 vector로 만들어서 학습시키는 방식을 시도해보았으나 각 이미지를 caption 개수만큼 반복해서 학습하는 경우가 가장 좋은 성능을 보였다.

2. Dataset을 만들 때, caption을 tokenize한 결과로 나온 단어들을 바로 사전에 등록하지 않고, 그 단어가 threshold=5 이상 등장할 때까지 보류한다. 출현횟수가 threshold에 도달한 경우, 단어를 사전에 등록한다. 만약 caption을 tokenized vector로 바꾸는 과정에서 아직 threshold에 도달하지 못해 보류된 단어가 등장한다면 이것을 [UNK] unknown token으로 표시해준다.

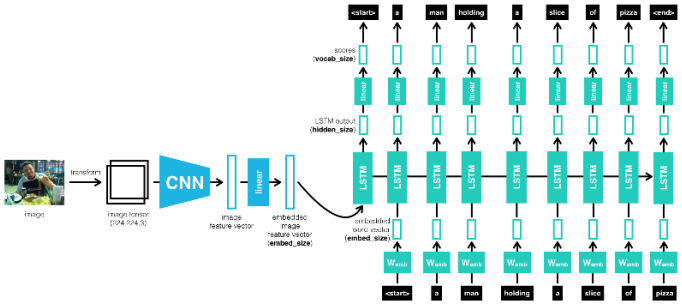
3. Early stopping, learning rate scheduler를 사용하였다.

   optim.lr_scheduler.ReduceLROnPlateau: 이 scheduler 는 학습이 잘 되고 있는지 아닌지에 따라 동적으로 learning rate 을 변화시킬 수 있다. 보통 validation set 의 loss 를 인자로 주어서 사전에 지정한 epoch 동안 loss 가 줄어들지 않으면 learning rate 를 감소시키는 방식이다.

   torch.optim.lr_scheduler.ReduceLROnPlateau(*optimizer*, *mode='min'*, *factor=0.1*, *patience=10*, *threshold=0.0001*, *threshold_mode='rel'*, *cooldown=0*, *min_lr=0*, *eps=1e-08*, *verbose=False*)

   ```
   factor (float) - Factor by which the learning rate will be reduced. new_lr = lr * factor.
   Default: 0.1.
   patience (int) - Number of epochs with no improvement after which learning rate will be reduced.
   For example, if patience = 2, then we will ignore the first 2 epochs with no improvement, and
   will only decrease the LR after the 3rd epoch if the loss still hasn't improved then. Default:
   10.
   ```

4. Image captioning (CNN-RNN architecture)

   입력 이미지는 CNN에 의해 특징 추출이 되며, CNN의 출력을 RNN의 입력에 연결하여 이미지를 설명하는 텍스트를 생성할 수 있다.

   Encoder(CNN)

   CNN은 이미지 내용을 smaller feature vector로 인코딩한다. 이 feature vector는 이미지에 대한 정보를 담고 있으며, RNN에 대한 초기 입력으로 사용된다.

   Decoder(RNN)

   RNN은 feature vector를 디코딩하여 단어 시퀀스로 변환한다.