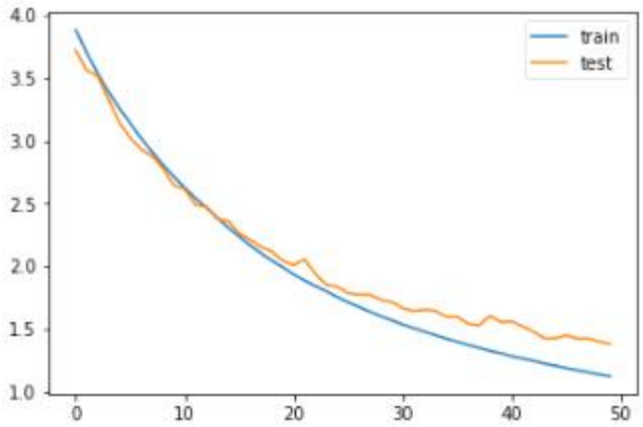
데이터 캡스톤 디자인 13주차 과제 진행내역

2014103929 김찬규

1) 데이터셋 크기를 늘려서 LSTM-ATTN 모델 튜닝

- 표본이 되는 데이터셋의 크기를 3만쌍까지 늘려서 똑같은 구조에 주입해 보았습니다. 이전과는 달리 loss와 val_loss가 여러 epoch를 거치며 굉장히 잘 수렴했습니다만, 실제 예측 결과를 확인해보니 overfiting이 생겼다는 것을 알게 되었습니다.

```
Epoch 47/50
93/93 [==============================] - 1081s 12s/step - loss: 1.1706 - val_loss: 1.4229
Epoch 48/50
93/93 [==============================] - 1075s 12s/step - loss: 1.1549 - val_loss: 1.4242
Epoch 49/50
93/93 [==============================] - 1075s 12s/step - loss: 1.1365 - val_loss: 1.4002
Epoch 50/50
93/93 [==============================] - 1068s 11s/step - loss: 1.1227 - val_loss: 1.3801
```



원문 : american airline whose passenger rating actually better american airline wa ranked many month back airline well southwest added fewer plane seat time number passenger plane gate american airline wa even better airline ontime
실제 요약문 : hawaiian airline land ontime performance airline quality ranking report look largest us airline expressjet american airline worst ontime performance virgin america best baggage handling southwest lowest complaint rate
예측 요약문 : bee minister replaces torres fund pianist

- loss는 이전에 비해 훨씬 줄였음에도 불구하고 예측 요약문이 예전보다 오히려 더 안 좋아졌습니다. 이전에는 그래도 원문과 상관관계를 가진 텍스트가 출력되었는데 지금은 전혀 관계가 없는 단어들이 출력됩니다.

2) 워드 임베딩 glove 활용

- 그 동안은 케라스에서 제공하는 API를 사용하여 원-핫-인코딩 방식의 워드 임베딩을 사용하고 있었습니다. 원-핫-인코딩의 경우 단어들 간의 관계를 알 수 없다는 문제가 있으므로 glove라는 사전 훈련된 워드 임베딩을 활용해 보았습니다.

```
Model: "glove_seq2attn"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 850)]        0
__________________________________________________________________________________________________
embedding (Embedding)           (None, 850, 300)     19200000    input_1[0][0]
__________________________________________________________________________________________________
enc_lstm1 (LSTM)                [(None, 850, 128), ( 219648      embedding[0][0]
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
enc_lstm2 (LSTM)                [(None, 850, 128), ( 131584      enc_lstm1[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 300)    19200000    input_2[0][0]
__________________________________________________________________________________________________
enc_lstm3 (LSTM)                [(None, 850, 128), ( 131584      enc_lstm2[0][0]
__________________________________________________________________________________________________
dec_lstm (LSTM)                 [(None, None, 128),  219648      embedding_1[0][0]
                                                                 enc_lstm3[0][1]
                                                                 enc_lstm3[0][2]
__________________________________________________________________________________________________
attention_layer (AttentionLayer ((None, None, 128),  32896       enc_lstm3[0][0]
                                                                 dec_lstm[0][0]
__________________________________________________________________________________________________
concat_layer (Concatenate)      (None, None, 256)    0           dec_lstm[0][0]
                                                                 attention_layer[0][0]
__________________________________________________________________________________________________
time_distributed (TimeDistribut (None, None, 64000)  16448000    concat_layer[0][0]
==================================================================================================
Total params: 55,583,360
Trainable params: 17,183,360
Non-trainable params: 38,400,000
__________________________________________________________________________________________________
```

```
Epoch 1/5
75/75 [==============================] - 816s 11s/step - loss: 5.4220 - accuracy: 0.3364 - recall: 1.2618e-04 - precision: 3.0304e-04 - f1score: 1.7793e-04 - val_loss: 5.3432 -
Epoch 2/5
75/75 [==============================] - 790s 11s/step - loss: 5.2766 - accuracy: 0.3379 - recall: 3.2731e-04 - precision: 7.8017e-04 - f1score: 4.6050e-04 - val_loss: 5.2516 -
Epoch 3/5
75/75 [==============================] - 819s 11s/step - loss: 5.1671 - accuracy: 0.3393 - recall: 5.9115e-04 - precision: 0.0014 - f1score: 8.2864e-04 - val_loss: 5.1772 - va
Epoch 4/5
75/75 [==============================] - 816s 11s/step - loss: 5.0639 - accuracy: 0.3406 - recall: 9.0882e-04 - precision: 0.0021 - f1score: 0.0013 - val_loss: 5.1189 - val_ac
Epoch 5/5
75/75 [==============================] - 814s 11s/step - loss: 4.9676 - accuracy: 0.3422 - recall: 0.0012 - precision: 0.0029 - f1score: 0.0017 - val_loss: 5.0510 - val_accura
```

- epoch를 좀 더 늘리면 좀 더 좋은 결과를 얻을 수 있을 것 같습니다.

2) transformer 모델

- scaled_dot_product_attention, multi-head-attention 등을 직접 함수로 구현하는데서 문제에 봉착해 있었는데, 대부분의 코드들이 모두 거의 똑같은 코드를 사용하는 것을 발견하고 저도 이를 사용하기로 하였습니다.
- 제 코드와 비교해보니 마스킹 값을 매우 작은 값으로 초기화하는 작업이 필요했는데 그 과정을 빼먹어서 제대로 작동하지 않았던 것 같습니다.
- resource exhausted 문제에 봉착해 있습니다. 이전에는 배치 사이즈를 줄이면 해결되었었는데, 지금은 배치 사이즈를 줄여도 런타임이 다운되고 에러가 계속 발생합니

다. RAM을 늘리라고 하는데 20GB의 램으로도 부족하면 어떻게 해야할지 고민입니다.

3) 결과보고서 초안 작성 중

- 과제를 처음 시작할 때 생각했던 만큼의 성과에 못 닿은 수준이고 실패에 더 가깝지만, 이제 뭔가 더 해보려 하는 것은 과욕이라 생각이 들어 지금까지 수행하였던 활동들을 바탕으로 결과보고서를 작성 중에 있습니다. 과제를 하며 만나고 해결했던 시행착오들을 세세히 적고 분석하는 방법으로 작성하려 합니다.