

딥러닝 및 응용

text classification 개선

2015004475 김태훈

1. 구동 환경

OS: Windows 10 64비트

Language: Python 3.7

필요 패키지와 버전 정보는 requirements.txt에 기재됨.

2. 코드 설명

batch_data(): 들어온 인덱스만큼 배치 데이터를 떼어 반환.

get_vocabulary_size(): 학습 데이터 값 중 가장 큰 값 반환.

fit_in_vocabulary(): vocabulary 사이즈에 dev 데이터를 맞춤.

zero_pad(): 데이터 길이를 주어진 값만큼 0을 넣어 맞춤.

build_classifier():

데이터 embed, 지정된 EMBEDDING_DIM에 맞게.

GRUCell로 rnn 레이어 구성, rnn_outputs, states 반환.

hidden state 크기만큼의 인풋을 2개의 아웃풋(긍정/부정) 으로 classify 하는 Fully connected layer 구성. activation 함수는 softmax. sigmoid 사용 가능.

SEQUENCE_LENGTH: 데이터에서 읽을 단어 수. 길수록 뒷부분까지 읽게 됨.

EMBEDDING_DIM: 데이터 임베딩 크기 지정.

HIDDEN_SIZE: RNN 셀의 은닉상태(HIDDEN STATE) 벡터 사이즈.

BATCH_SIZE: 미니배치 사이즈.

NUM_EPOCHS: 학습 반복 수. 20~30 이후 큰 변화 없어 25로 설정.

learning_rate: 옵티마이저 러닝 레이트.

KEEP_PROB: 어텐션 메커니즘 적용 시 각 RNN 셀의 드롭아웃 레이트, 미사용.

ATTENTION_SIZE: 어텐션 메커니즘 어텐션 벡터 차원.

dev_num: 전체 데이터의 25% 사용.

x, y dev, train, onehot, 등 변수들 초기화.

batch_ph, target_ph: 배치데이터, 라벨데이터 플레이스 홀더.

y_pred, logits: build_classifier() 반환값 (hypothesis, logits) 저장할 변수.

loss: 소프트맥스 cross-entropy 함수 사용.

optimizer: Adam 옵티마이저 learning_rate 적용하여 loss 최소화 방향으로 학습.

세션 가동 후

1. 변수 초기화
2. 정해진 epoch 수(25 epoch) 만큼 반복
 - A. 전체 데이터를 가져와 인덱스 셔플.

B. batch 수 만큼

i. batch 데이터를 가져와서 학습 진행.

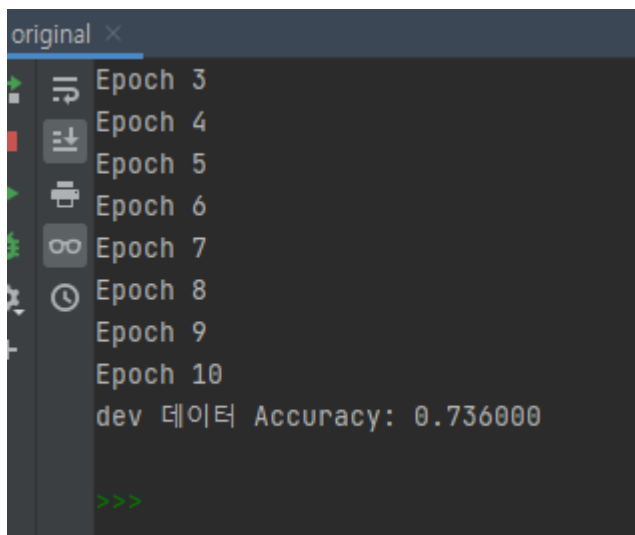
3. 학습된 모델 체크포인트 저장 후 불러와서

4. dev 셋을 이용해 정확도 측정.

5. 결과 파일 저장.

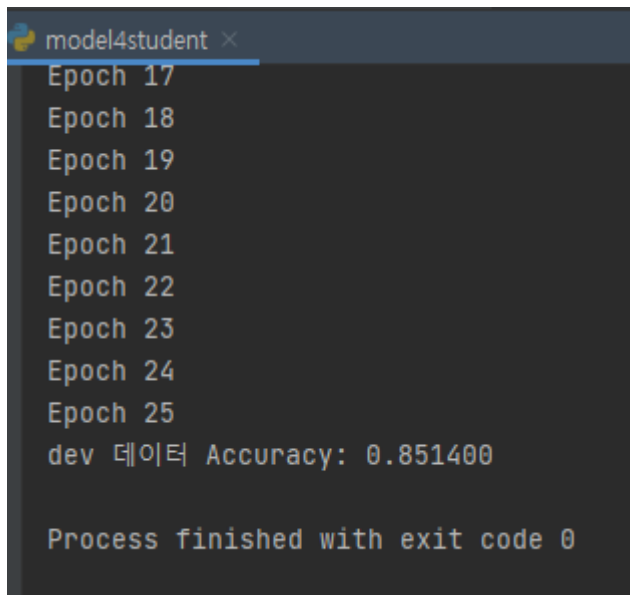
3. 실험 결과

오리지널 코드



```
original x
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
dev 데이터 Accuracy: 0.736000
>>>
```

개선 버전



```
model4student ×
Epoch 17
Epoch 18
Epoch 19
Epoch 20
Epoch 21
Epoch 22
Epoch 23
Epoch 24
Epoch 25
dev 데이터 Accuracy: 0.851400

Process finished with exit code 0
```

4. 결론

basic rnn cell보다 gru cell을 이용한 모델이 뒤로 갈수록 앞부분을 잊어버리는 정도가 크게 완화되어 더 나은 정확도를 보인다. 같은 조건에서 베이직 셀은 0.72, gru 셀은 0.74.

학습 데이터들의 평균 길이가 50보다 훨씬 큰 길이를 가져 개선 버전에서 이를 200으로 늘린 점이 정확도 향상에 영향이 있었다.

GRU 셀을 이용했을 때 시퀀스 길이 50, 100, 150, 200으로 실험 한 결과 각각 0.74, 0.79, 0.82, 0.85의 정확도를 보였다.

epoch이 20~30일 때와 100일 때 정확도는 약 0.85로 비슷했다.