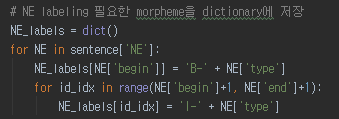
**자연어처리론 숙제 5**

Named Entity Recognition

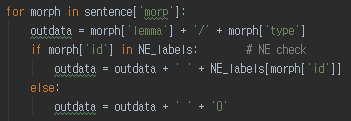
**숙제#1**

Json2CoNLL.py에 구현

json.load 함수를 통해 list([]) & dict({}) 형태로 변환하여 read



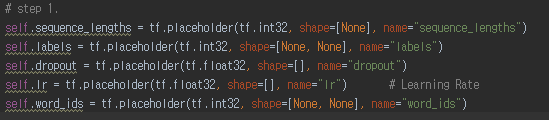
‘NE’ 카테고리의 label data를 NE\_labels에 dict형태로 저장. (형태소의 id를 key로 사용)



형태소가 NE\_labels dictionary에 존재할 경우는 labeling 수행. 없으면, ‘O’ 출력.

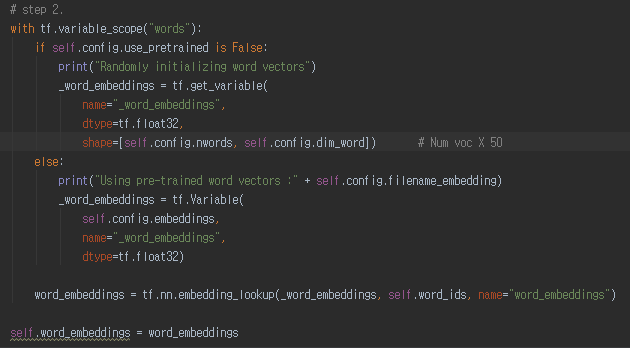
**숙제#2**

**Step 1.** Placeholder 생성



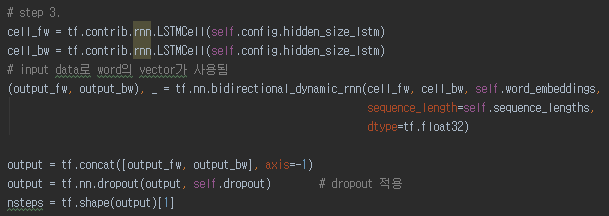
데이터 입력부가 되는 placeholder 생성. Shape를 None으로 선언함으로써, input data 개수와 문장의 길이에 가변적으로 학습이 가능하다.

**Step 2.** Word embedding 생성



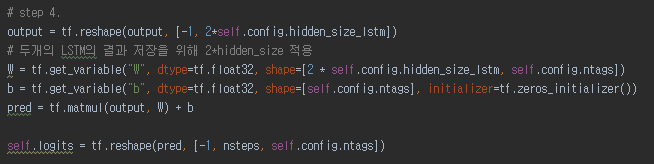
Use\_pretrained 가 True일 경우는 기존의 word2vector정보를 불러와서 저장, False일 경우는 random한 초기값을 활용하여, training을 수행한다. Vector size는 50이고 training data의 word 만큼의 vector가 존재한다. Embedding\_lookup은 word\_ids에 해당되는 vector를 찾는 용도로 사용된다.

**Step 3.** 모델 생성



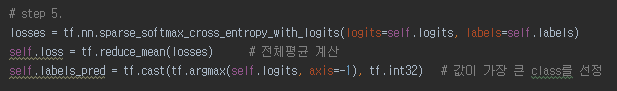
LSTM Cell두개로 Bi-LSTM cell을 만드는 과정으로, 앞서 저장된 word\_embeddings가 input이 된다. 각 LSTM의 출력결과는 concatenate하여 하나의 변수인 output에 저장한다.

**Step 4.** 출력 레이어 정의



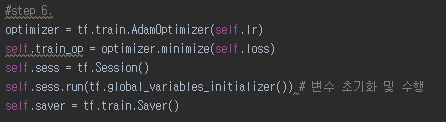
Hidden layer가 100인 LSTM의 결과가 합쳐져 있으므로, W는 LSTM layer size의 두배와 최종결과인 tag개수에 따라, (200x11)의 size를 갖는 vector가 된다. Bias는 출력 개수와 동일한 (11x1)의 size를 갖는다. 연산결과는 logits에 저장.

**Step 5.** Loss 계산



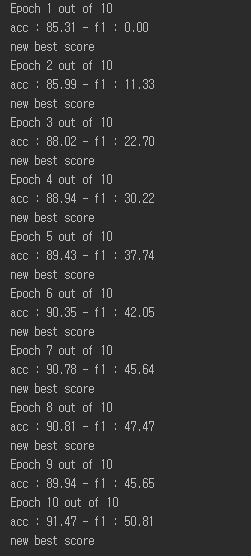
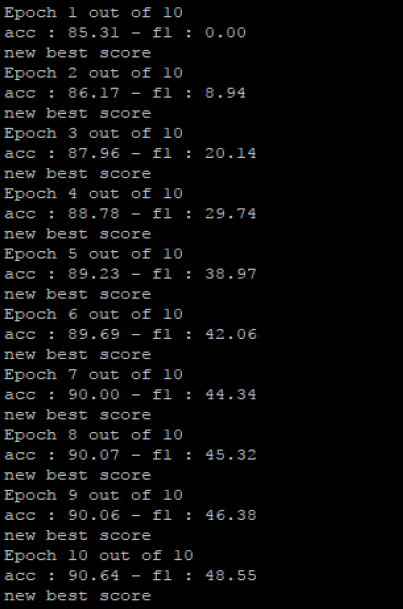
Network를 통과한 결과와 input으로 받은 label을 이용해, Cross entropy로 loss를 구한다. Reduce\_mean은 모든 결과 class들의 loss평균을 계산한다. 최종 class 결정은 argmax함수로 결정.

**Step 6.** Optimizer정의와 Session 초기화



learning에는 Adam optimizer를 사용하였고, self.loss를 최소화하는 방향으로 learning이 진행된다. Saver는 변수 저장 및 복구에 사용되는 class다.

**Train.py 실행 결과**



**Evaluate.py 실행 결과**

