Bi-directional LSTM for Named Entity Recognition

서강대학교
자연어처리연구실
김주애

Named Entity Recognition (NER)

개체명 인식(NER) 이란?

- 텍스트에서 개체명을 인식하고, 인식한 개체명들을 미리 정의된 category로 classification 하는 task
- Information Extraction (IE)의 핵심 task

In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell–Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

Tag colours:
LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

company product
Samsung to launch Galaxy S6 in March

Protein DNA RNA Chemical Disease

Glucocorticoid receptors in peripheral blood lymphocytes of patients with bronchial asthma. Quantitation of glucocorticoid receptors (GCR) and the study of their affinity for glucocorticosteroids (GCS) were made in peripheral blood lymphocytes of bronchial asthma (BA) patients in consideration of GCR treatment and serum levels of endogenous cortiso.

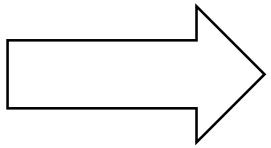
Sequential Labeling

- **Sequential한 형태의 입력에 대해 각 sequence마다 label을 지정**
  - Ex) POS tagging

She	sells	seashells	on	the	seashore
PRP	VBZ	NNS	IN	DT	NN

  - Ex) NER

Jobs	is	a	CEO	of	Apple	Inc
B-PER	O	O	O	O	B-ORG	I-ORG

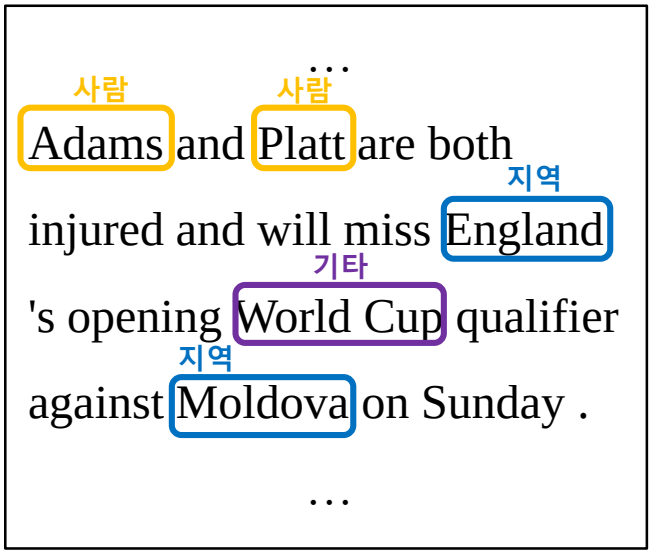


Jobs : 사람
Apple Inc : 기관/단체

Sequential Labeling

Data format

- CoNLL format



입력 sequence	자질		정답 label
Adams	NNP	I-NP	B-PER
and	CC	I-NP	O
Platt	NNP	I-NP	B-PER
are	VBP	I-VP	O
both	DT	I-NP	O
injured	JJ	I-NP	O
and	CC	O	O
will	MD	I-VP	O
miss	VB	I-VP	O
England	NNP	I-NP	B-LOC
's	POS	B-NP	O
opening	NN	I-NP	O
World	NNP	I-NP	B-MISC
Cup	NNP	I-NP	I-MISC
qualifier	NN	I-NP	O
against	IN	I-PP	O
Moldova	NNP	I-NP	B-LOC
on	IN	I-PP	O
Sunday	NNP	I-NP	O
.	.	O	O

*다른 문장 사이는 뉴라인("\n")으로 구분 되어 있음

Sequential Labeling

숙제 1

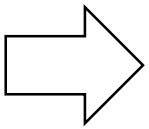
- 2016lipexpo_NERcorpus_*.json 파일을 sequential labeling 하기 용이한 형태인 CoNLL format으로 변환한다.

형태소 분석

어절 분석

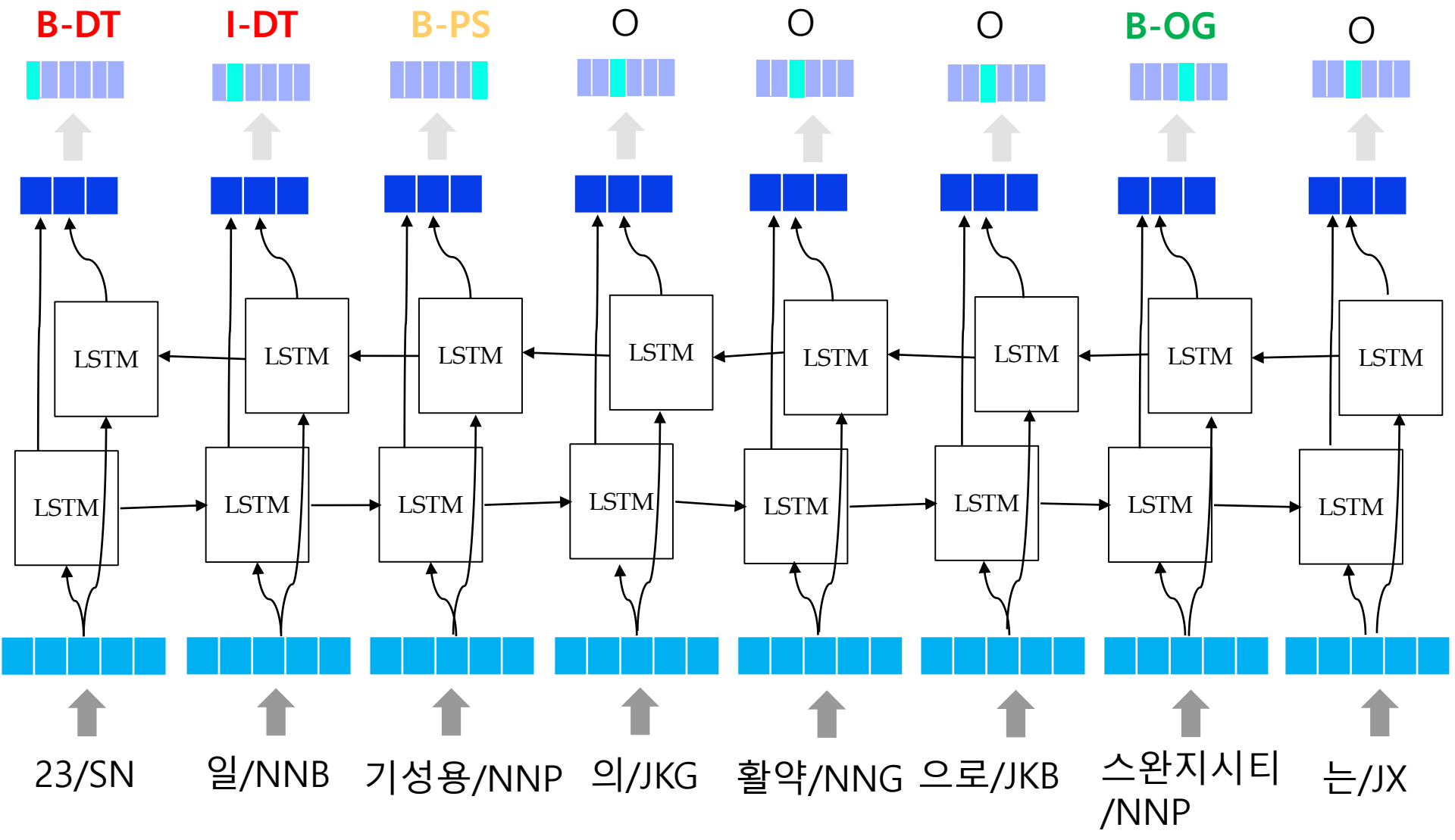
개체명 / 개체명 태그 / 형태소 분석결과에서의 개체명 위치


```
{
  "id": 1501,
  "resever_str": "",
  "text": "한편, AFC챔피언스리그 E조에 속한 포항 역시 대회 8강 진출이 불투명하다 .",
  "morp": [
    { "id": 0, "lemma": "한편", "type": "NNG", "position": 0},
    { "id": 1, "lemma": ",", "type": "SP", "position": 6},
    { "id": 2, "lemma": "AFC", "type": "SL", "position": 8},
    { "id": 3, "lemma": "챔피언스", "type": "NNG", "position": 11},
    { "id": 4, "lemma": "리그", "type": "NNG", "position": 23},
    { "id": 5, "lemma": "E", "type": "SL", "position": 30},
    { "id": 6, "lemma": "조", "type": "NNG", "position": 31},
    { "id": 7, "lemma": "에", "type": "JKB", "position": 34},
    { "id": 8, "lemma": "속하", "type": "VV", "position": 38},
    { "id": 9, "lemma": "ㄴ", "type": "ETM", "position": 38},
    { "id": 10, "lemma": "포항", "type": "NNP", "position": 45},
    { "id": 11, "lemma": "역시", "type": "MAJ", "position": 52},
    { "id": 12, "lemma": "대회", "type": "NNG", "position": 59},
    { "id": 13, "lemma": "8강", "type": "NNG", "position": 66},
    { "id": 14, "lemma": "진출", "type": "NNG", "position": 71},
    { "id": 15, "lemma": "이", "type": "JKS", "position": 77},
    { "id": 16, "lemma": "불투명", "type": "NNG", "position": 81},
    { "id": 17, "lemma": "하", "type": "VV", "position": 90},
    { "id": 18, "lemma": "다", "type": "EC", "position": 93},
    { "id": 19, "lemma": ".", "type": "SF", "position": 97}
  ],
  "word": [
    { "id": 0, "text": "한편,", "type": "", "begin": 0, "end": 1},
    { "id": 1, "text": "AFC챔피언스리그", "type": "", "begin": 2, "end": 4},
    { "id": 2, "text": "E조에", "type": "", "begin": 5, "end": 7},
    { "id": 3, "text": "속한", "type": "", "begin": 8, "end": 9},
    { "id": 4, "text": "포항", "type": "", "begin": 10, "end": 10},
    { "id": 5, "text": "역시", "type": "", "begin": 11, "end": 11},
    { "id": 6, "text": "대회", "type": "", "begin": 12, "end": 12},
    { "id": 7, "text": "8강", "type": "", "begin": 13, "end": 13},
    { "id": 8, "text": "진출이", "type": "", "begin": 14, "end": 15},
    { "id": 9, "text": "불투명하다", "type": "", "begin": 16, "end": 18},
    { "id": 10, "text": ".", "type": "", "begin": 19, "end": 19}
  ],
  "NE": [
    { "id": 0, "text": "E조", "type": "OG", "begin": 5, "end": 6}
  ],
  "chunk": [],
  "dependency": [],
  "SRL": [],
  "relation": [],
  "SA": []
},
```


```
한편/NNG O
,/SP O
AFC/SL O
챔피언스/NNG O
리그/NNG O
E/SL B-OG
조/NNG I-OG
에/JKB O
속하/VV O
ㄴ/ETM O
포항/NNP O
역시/MAJ O
대회/NNG O
8강/NNG O
진출/NNG O
이/JKS O
불투명/NNG O
하/VV O
다/EC O
./SF O
```

Bi-LSTM for NER



23일 기성용의 활약으로 스완지시티는 리버풀전에서 승리를 얻었다.

전체 파일 설명

- **utils.py**
  - Data build
  - Configuration
- **model.py**
  - Bi-LSTM model
  - Build
  - Train
  - Restore
  - Evaluate
- **train.py**
  - Training the model with training data, development data
- **evaluate.py**
  - Test/evaluate the model with test data

Utils.py

- **def load_vocab(filename)**
- **def write_vocab (vocab, filename)**
- **def data_build()**
  - train, test, dev 데이터에 대한 generator 생성
  - word, character, label에 대한 vocabulary 생성
- **def get_processing_word(vocab_words=None, vocab_chars=None, chars=False, allow_unk=True)**
  - string 형태로 된 sentence를 word idx로 바꿔줌
  - character 자질 사용 시, string 형태로 된 word를 character idx로 바꿔줌
- **Class Config()**
  - configuration 정보가 담긴 클래스
  - vocabulary 생성, vocabulary에 있는 string을 idx로 바꿈, pre-trained embedding 가져오기 등의 기능을 함
- **Class data_read(object)**

Utils.py - Class Config()

```
class Config():
    # general config
    dir_output = "results/test/"
    dir_model  = dir_output + "model.weights/"
    path_log   = dir_output + "log.txt"

    # embeddings
    dim_word = 50
    dim_char = 50

    filename_embedding = "data/korean_news_100MB_word2vec.txt".format(dim_word)
    filename_trimmed = "data/korean_embedding.trimmed.npz".format(dim_word)

    use_pretrained = True
    use_chars = True

    # dataset
    filename_dev = "./data/NER_dev.txt"
    filename_test = "./data/NER_test.txt"
    filename_train = "./data/NER_train.txt"

    max_iter = None # if not None, max number of examples in Dataset

    # vocab (created from dataset with build_data.py)
    filename_words = "data/words.txt"
    filename_tags = "data/tags.txt"
    filename_chars = "data/chars.txt"
```

```
    # training
    nepochs          = 10
    dropout          = 0.5
    batch_size       = 20
    lr               = 0.005 #learning rate
    lr_decay         = 0.9
    nepoch_no_imprv  = 3

    # model hyperparameters
    hidden_size_char = 25 # lstm on chars
    hidden_size_lstm = 100 # lstm on word embeddings
```

model.py

- **def minibatches(data, minibatch_size)**
  - data를 minibatch_size만큼만 가져올 수 있게 함
- **def _pad_sequences(sequences, pad_tok, max_length)**
- **def pad_sequences(sequences, pad_tok, nlevels=1)**
  - 네트워크의 입력이 일정하도록 같은 길이를 갖게끔 padding
- **def get_chunks(seq, tags)**
  - NE tag processing

- **Class NERmodel(object)**
  - Bi-LSTM 기반의 NER 모델을 생성/학습/평가 하는 기능을 가진 Class
  - **def build(…) : 모델 그래프 생성**
  - def train(…) : 모델 학습
  - def restore_session(…) : 학습된 모델을 load
  - def evaluate() : 모델 test 및 성능측정
  - def get_feed_dict(…) : 모델 학습 시 입력 데이터를 feed
  - def predict_batch(…) : batch 단위로 test
  - def predict(…) : word list(sentence)를 입력하면, 해당 list의 NER test 결과를 반환

train.py

```
import utils
import model

# data build for word embeddings
utils.data_build()

# create instance of config
config = utils.Config()

# build model
model = model.NERmodel(config)
model.build()

# read datasets
dev   = utils.data_read(config.filename_dev, config.processing_word,
                  config.processing_tag, config.max_iter)
train = utils.data_read(config.filename_train, config.processing_word,
                  config.processing_tag, config.max_iter)

# train model
model.train(train, dev)
```

model.py – def train()

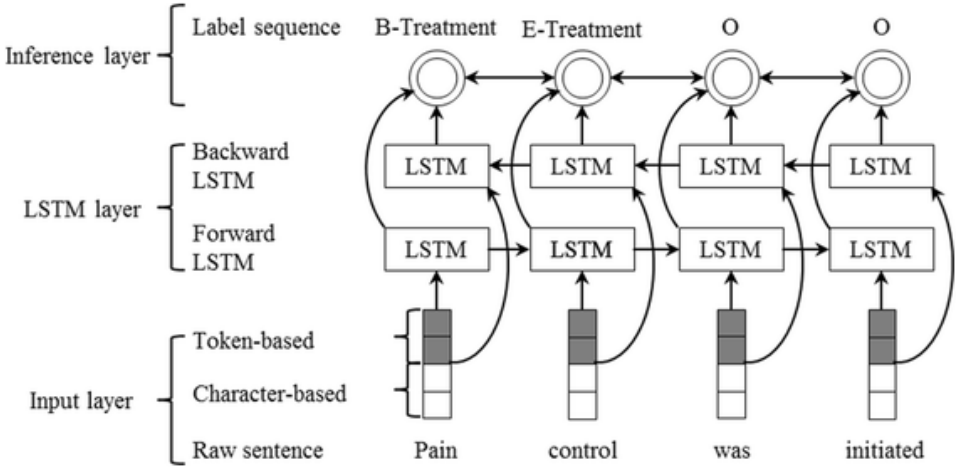
```
def train(self, train, dev):

    best_score = 0
    nepoch_no_imprv = 0 # for early stopping

    for epoch in range(self.config.nepochs):
        print("Epoch {:} out of {:}".format(epoch + 1, self.config.nepochs))

        batch_size = self.config.batch_size

        for i, (words, labels) in enumerate(minibatches(train, batch_size)):
            fd, _ = self.get_feed_dict(words, labels, self.config.lr,
                                       self.config.dropout)

            _, train_loss = self.sess.run(
                [self.train_op, self.loss], feed_dict=fd)

            # print(i + 1, [("train loss", train_loss)])

        metrics = self.run_evaluate(dev)
        msg = " - ".join(["{} {:04.2f}".format(k, v)
                          for k, v in metrics.items()])
        print(msg)
        score = metrics["f1"]
        self.config.lr *= self.config.lr_decay

        if score >= best_score:
            nepoch_no_imprv = 0
            if not os.path.exists(self.config.dir_model):
                os.makedirs(self.config.dir_model)
            self.saver.save(self.sess, self.config.dir_model)
            best_score = score
            print ("new best score")
        else:
            nepoch_no_imprv += 1
            if nepoch_no_imprv >= self.config.nepoch_no_imprv:
                print ("- early stopping {} epochs without ""improvement".format(nepoch_no_imprv))
                break
```

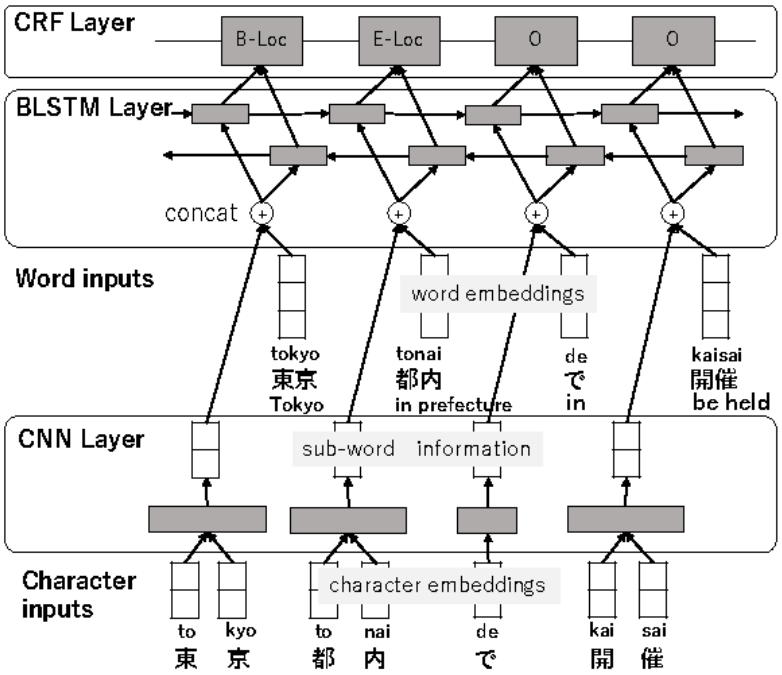
evaluate.py

```
# -*- coding: utf8 -*-

import model
import utils

# create instance of config
config = utils.Config()

# build model
model = model.NERmodel(config)
model.build()
model.restore_session(config.dir_model)

# create dataset
test  = utils.data_read(config.filename_test, config.processing_word,
                        config.processing_tag, config.max_iter)

# evaluate and interact
model.evaluate(test)
```

Bi-directional LSTM 모델 생성

- 목적 : **Bi-directional LSTM**을 사용한 개체명 인식기 생성
  - Bi-LSTM은 Sequence Labeling문제에서 좋은 성능을 보임
  - 높은 성능의 개체명 인식기들이 대부분 Bi-LSTM 기반





Bi-directional LSTM 모델 생성

model.py

```
import numpy as np
import os
import tensorflow as tf

UNK = "$UNK$"
NUM = "$NUM$"
NONE = "O"

def minibatches(data, minibatch_size):...

def _pad_sequences(sequences, pad_tok, max_length):...

def pad_sequences(sequences, pad_tok, nlevels=1):...

def get_chunks(seq, tags):...

class NERmodel(object):

    def __init__(self, config):...

    def build(self):...

    def train(self, train, dev):...

    def get_feed_dict(self, words, labels=None, lr=None, dropout=None):...

    def predict_batch(self, words):...

    def run_evaluate(self, test):...

    def restore_session(self, dir_model):...

    def evaluate(self, test):...
```

Bi-directional LSTM 모델 생성

model.py – class NERmodel – def build()

Step 1.

- placeholder 생성하기
  - sequence_lengths
    - batch size
  - labels
    - batch size * max length of sentence in batch
  - dropout, lr (learning rate)
    - 단일 값
  - word_ids
    - batch size * max length of sentence in batch

```
self.sequence_lengths = tf.placeholder(tf.int32, shape=[None], name="sequence_lengths")
self.labels = tf.placeholder(tf.int32, shape=[None, None], name="labels")
self.dropout = tf.placeholder(dtype=tf.float32, shape=[], name="dropout")
self.lr = tf.placeholder(dtype=tf.float32, shape=[], name="lr")
self.word_ids = tf.placeholder(tf.int32, shape=[None, None], name="word_ids")
```

Bi-directional LSTM 모델 생성

model.py – class NERmodel – def build()

Step 2.

- 단어 입력 표현 정의

1. Random initialize 후 fine-tuning
   - Embedding정보를 저장하는 Tensor를 선언
     - 변수 이름 : _word_embeddings
     - config.nwords * config.dim_word 차원
   - ***tf.nn.embedding_lookup (Tensor, index of word)***를 사용하여 embedding 생성
     - self.word_embeddings에 저장

2. pre-trained embedding 불러오기
   - Utils.py – class Config의 use_pretrained가 True 일 때, 미리 학습 된 embedding을 load

Bi-directional LSTM 모델 생성

model.py – class NERmodel – def build()

- Step 2.

```
with tf.variable_scope("words"):

    if self.config.use_pretrained is False:
        print ("Randomly initializing word vectors")
            _word_embeddings = tf.get_variable(
                name="_word_embeddings",
                dtype=tf.float32,
                shape=[self.config.nwords, self.config.dim_word])
    else:
        print("Using pre-trained word vectors :"+self.config.filename_embeddi
        _word_embeddings = tf.Variable(
                self.config.embeddings,
                name="_word_embeddings",
                dtype=tf.float32)

    word_embeddings = tf.nn.embedding_lookup(_word_embeddings,
            self.word_ids, name="word_embeddings")

self.word_embeddings =  word_embeddings
```

방법 1

방법 2

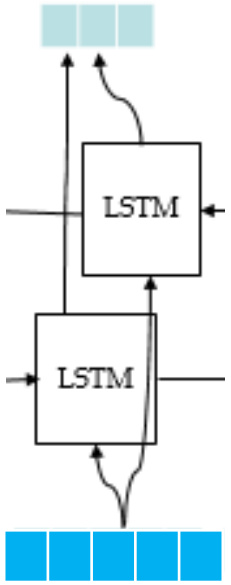
Bi-directional LSTM 모델 생성

model.py – class NERmodel – def build()

Step 3.

- 모델 생성
  - LSTM cell 두 개로 Bi-LSTM cell을 만든다.
    - 변수 이름 : cell_fw, cell_bw
    - ***tf.contrib.rnn.LSTMCell(size of cell)***
  - ***tf.nn.bidirectional_dynamic_rnn(…)*** 함수를 통해 LSTM cell의 결과 저장
  - 각 LSTM cell에서 출력된 결과를 concatenate하여 출력을 저장
    - 변수 이름 : output

```
cell_fw = tf.contrib.rnn.LSTMCell(self.config.hidden_size_lstm)
cell_bw = tf.contrib.rnn.LSTMCell(self.config.hidden_size_lstm)
(output_fw, output_bw), _ = tf.nn.bidirectional_dynamic_rnn(
                                                    dtype=tf.float32)
output = tf.concat([output_fw, output_bw], axis=-1)
output = tf.nn.dropout(output, self.dropout)

nsteps = tf.shape(output)[1]
```



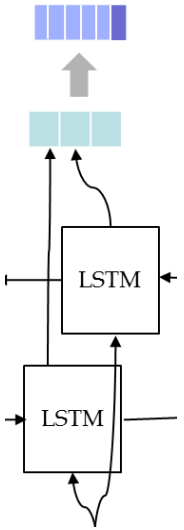
Bi-directional LSTM 모델 생성

model.py – class NERmodel – def build()

Step 4.

- 출력 레이어 정의
  - Bi-LSTM 출력 값 계산
    - 최종 예측 값(변수이름 : pred) : output *W + b
    - ※ output의 shape에 주의하세요!
  - Bi-LSTM의 출력을 softmax 하기 위한 레이어를 정의
    - W, b 변수 생성
    - Bi-LSTM output size : 2*self.config.hidden_size_lstm
  - pred 값을 reshape 하여 self.logits에 저장
    - -1 * nsteps * self.config.ntags 차원
    - softmax의 입력으로 사용 됨

```
output = tf.reshape(output, [-1, 2*self.config.hidden_size_lstm])
W = tf.get_variable("W", dtype=tf.float32,
                    shape=[                                , self.config.ntags])
b = tf.get_variable("b", dtype=tf.float32,
                    shape=[                    ,initializer=tf.zeros_initializer())
pred = tf.matmul(output, W) + b

self.logits = tf.reshape(pred, [-1, nsteps, self.config.ntags])
```



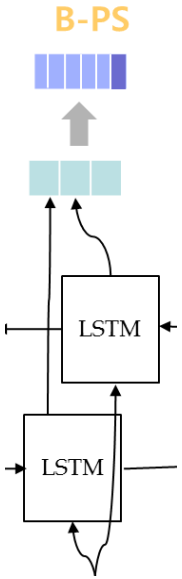
Bi-directional LSTM 모델 생성

model.py – class NERmodel – def build()

Step 5.

- Loss 계산
  - 예측 값과 정답 값을 가지고 cross entropy로 loss를 구해서 저장한다.
    - tf.nn.sparse_softmax_cross_entropy_with_logits(logits= ?,labels= ???)
    - 변수 이름 : losses $H_{y'}(y) = -\sum_i y'_i \log(\text{softmax}(y_i))$
  - tf.reduce_mean으로 loss를 계산해서 self.loss에 저장
- Test 결과 확인을 위한 예측 레이블을 self.labels_pred에 저장

```
losses = tf.nn.sparse_softmax_cross_entropy_with_logits(
                    logits=self.logits, labels=self.labels)
self.loss = tf.reduce_mean(losses)
self.labels_pred = tf.cast(tf.argmax(self.logits, axis=-1), tf.int32)
```



Bi-directional LSTM 모델 생성

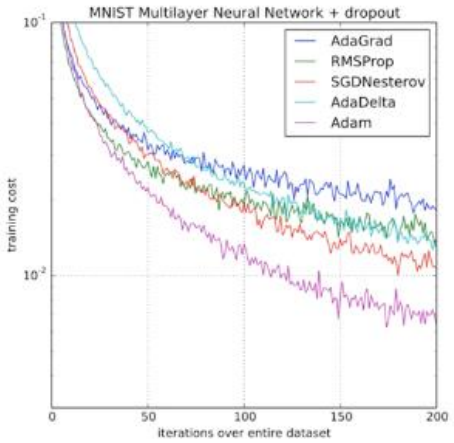
model.py – class NERmodel – def build()

Step 6.

- Optimizer 정의와 Session 초기화
  - Adam optimizer 사용
  - tf.train.AdamOptimizer (learning rate)

ADAM: a method for stochastic optimization

[Kingma et al. 2015]




```
optimizer = tf.train.AdamOptimizer(self.lr)
self.train_op = optimizer.minimize(self.loss)
self.sess = tf.Session()
self.sess.run(tf.global_variables_initializer())
self.saver = tf.train.Saver()
```

Bi-directional LSTM 모델 생성

train.py 실행

```
Epoch 1 out of 10
acc : 85.31 - f1 : 0.00
new best score
Epoch 2 out of 10
acc : 86.17 - f1 : 8.94
new best score
Epoch 3 out of 10
acc : 87.96 - f1 : 20.14
new best score
Epoch 4 out of 10
acc : 88.78 - f1 : 29.74
new best score
Epoch 5 out of 10
acc : 89.23 - f1 : 38.97
new best score
Epoch 6 out of 10
acc : 89.69 - f1 : 42.06
new best score
Epoch 7 out of 10
acc : 90.00 - f1 : 44.34
new best score
Epoch 8 out of 10
acc : 90.07 - f1 : 45.32
new best score
Epoch 9 out of 10
acc : 90.06 - f1 : 46.38
new best score
Epoch 10 out of 10
acc : 90.64 - f1 : 48.55
new best score
```

evaluate.py 실행

```
Reloading the latest trained model...
Testing model over test set
acc : 92.26 - f1 : 52.11
23/SN B-DT
일 /NNB I-DT
기 성 용 /NNP B-PS
의 /JKG O
활 약 /NNG O
으 로 /JKB O
스 완 지 시 티 /NNP B-PS
는 /JX O
리 버 풀 /NNP O
전 /NNG O
에 서 /JKB O
승 리 /NNG O
를 /JKO O
얻 /VV O
었 /EP O
다 /EC O
./SF O
```

Bi-directional LSTM 모델 생성

숙제 2

- 주어진 코드에 앞에서 설명한 모델 생성 부분을 구현하고, 성능을 측정한다.
- 주석을 반드시 포함할 것

최종 제출물

- 2016lipexpo_NERcorpus_*.json을 CoNLL 데이터 형식으로 바꾸는 코드
- 2016lipexpo_NERcorpus_*.json을 CoNLL 데이터 형식으로 바꾼 결과 파일
- 최종 NER 모델
  - 초기 주어진 data 폴더는 낼 필요 없음, python 파일만 제출
- 보고서
  - 코드 각 부분에 대한 상세 설명
  - 초기에 주어진 데이터로 NER을 실행하고 난 결과 이미지
  - Korean_*_CoNLL.txt로 NER을 실행하고 난 결과 이미지

참고 자료

- **Tesorflow API : https://www.tensorflow.org/api_docs/**
- **코드참고 : https://guillaumegenthial.github.io/sequence-tagging-with-tensorflow.html**