# 1. Introduction

### \* 소프트웨어 명칭

Bi-LSTM-CRF\_NER

### \* 소프트웨어 설명

Named Entity Recognition with Bi-LSTM-CRF architecture on word level.

Named-Entity Recognition을 위해 Bi-LSTM-CRF 모델을 이용한 개체명 인식기 입니다.

### \* 사용 환경

* python==3.5
* tensorflow==1.12.0
* numpy==1.15.4
* pandas==0.23.4
* cloudpickle==0.5.5
* pickleshare==0.7.4
* gensim==3.6.0
* Keras==2.2.4
* Keras-Applications==1.0.6
* Keras-Preprocessing==1.0.5

# 2. File Structure

### \* major file description

1. 실행 파일
   * main.py ------------------------------------------ 소프트웨어 실행 파일
2. 데이터 파일
   * exobrain*dev.bio*ner ------------------------------------------ validation data file
   * exobrain*test.bio*ner ------------------------------------------ test data file
   * exobrain*train.bio*ner ------------------------------------------ train data file
   * tag\_enc.pickle ---------------------------------------------- tag lookup dictionary
3. 사전 파일
   * /Dictionary/ne\_dict.pickle ------------------------------------------ named entity dictionary file
4. 결과 파일
   * result.csv ---------------------------------------------- result file

### \* 전체 구조

* (프로그램에 의해 생성되는 파일 및 폴더는 소괄호로 표기해 놓았습니다.)
* Bi-LSTM-CRF\_NER/
  + Corpus4model/ ------------------------------------------ data set 폴더
    - exobrain*dev.bio*ner
    - exobrain*test.bio*ner
    - exobrain*train.bio*ner
    - test.ner
  + Dictionary/ ------------------------------------------ 개체명 사전을 저장할 폴더
    - ne\_dict.pickle
  + (embeddings/) ------------------------------------------ 프로그램 실행 시 'embedding\_utils.py'에 의해 생성
    - char\_emb.model
    - morph*pos*emb.model
    - pos\_emb.model
  + (final/) ------------------------------------------------------ saving final model. generated by 'model.py'
  + (sentences/) ---------------------------------------------- 프로그램 실행 시 'ner*data*utils.py'에 의해 생성
    - test*data*real\_sentences.pickle
    - test*data*sentences.pickle
    - train*data*sentences.pickle
    - valid*data*sentences.pickle
  + (tmp/) ---------------------------------------------- saving intermediate stages of model. generated by 'model.py'
  + config.py ---------------------------------------------- configuration file
  + embedding\_utils.py ---------------------------------------------- making pre-trained embedding models
  + main.py ---------------------------------------------- main program
  + metrics.py ---------------------------------------------- metrics for evaluating the model
  + model.py ---------------------------------------------- NER Tagger model(with Bi-LSTM-CRF architecture)
  + model*utils.py ---------------------------------------------- utilities for the NER Tagger model(padding, mini*batch)
  + ner*data*utils.py ---------------------------------------------- utilities for data pre-processing
  + README.md ---------------------------------------------- read me file(markdown ver.)
  + requirements.txt ---------------------------------------------- requirements specification file
  + (result.csv) ---------------------------------------------- result file(csv format)
  + (tag*enc.pickle) ---------------------------------------------- tag lookup dictionary. generated from 'ner*data\_utils.py' file.

# 3. How to Use

### \* 준비할 파일

* data set(/Corpus4model/)
  + data set for model training and evaluating
    - 정답 열('TAG')이 있는 corpus
    - Format : CoNLL form(['RAW', 'MORPH', 'POS', 'TAG']) 형태로 이루어진 파일
  + corpus for labeling ('test.ner')
    - 정답 열이 없는 corpus
    - Format : 'TAG'를 제외한 형태(['MORPH', 'POS'] 또는 ['RAW', 'MORPH', 'POS'])로 이루어진 파일
* NE dictionary(/Dictionary/)
  + 개체명 사전
    - [NE] 와 [NE label] 두 열로 이루어진 사전 파일 입니다.
    - 사전을 변경할 경우 'config.py'의 '\_DICTIONARY\_'에 변경한 사전 경로를 넣어주세요.

### \* main.py

* model = NERTagger(...)

1. Training the model
   * model.train(train, valid)
2. Evaluating the model
   * model.test(test)
   * model.metrics("test")
3. NE labeling
   * model.test\_real(test\_real)
   * model.result() : making 'result.csv' file