Using the LSTM-CRF Stack for Keyword Tagging in Data Science Job Posts

Ran Xia

Marymount University

Abstract

The purpose of this project is to extract technology keywords in the Data Science job requirements. The project first builds a sequential model that takes the tokenized job posts for word embedding. Then the project builds a functional model taking POS tags as additional features. All input features are fed to the LSTM-CRF stack for sequential labeling. The results are evaluated with the corresponding confusion matrix.

Keywords: technology keywords, WordEmbedding, LSTM-CRF stack, sequential model, functional model, POS tags, confusion matrix

# Overview and Problem Statement

With the quick expansion of Data Science jobs, many new required skills emerge and gain popularity all the time. Therefore, it is hard to use a simple hardcoding approach to retrieve key phrases. One possible solution is to use an RNN model that can label sequential data according to the linguistic structure in which the target phrases reside—according to Donghuo Zeng, Chengjie Sun, Lei Lin, & Bingquan Liu (2017), combining Bi-Directional LSTM and CRF yields promising results in Drug Named Entity Recognition. This project takes a similar approach to labeling Technology Named Entities.

# Project Methods

## Data Source and Preprocessing

All data is collected from JobsPikr.com, a company that provides job data solutions. The raw data is in CSV format and contains HTML and other formatting tags in the job descriptions. Therefore, the data needs to be cleaned by first removing HTML and other formatting tags. The following code uses the HTMLParser for tag stripping:

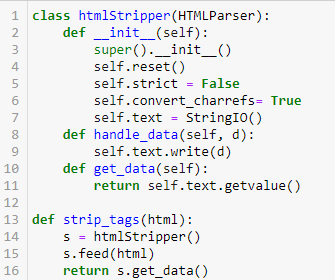


Figure 1. Creating the htmlStripper Class for tag removing

The texts are then processed with sentence tokenization. During the tokenization, each sentence is put into a tuple with the format of ‘(document id, sentence, token number).’ The mean and standard deviation of the token numbers are later used for deciding the max padding length. Each sentence is then processed with tokenization, converting to lowercase, removing stopwords and punctuation, and posing tagging.

The cleaned texts are then used for manual labeling to create the training dataset. In this project, around 5000 noun phrases are labeled, and around 600 noun phrases are identified as the target class. These targe noun phrases are then used as a lookup dictionary for the rest of the unlabeled phrases for keyword matching. All phrases labeled as the target are then tagged with the BOI-tagging. For example, the phrase ‘understanding machine learning algorithms’ are labels as ‘O, B-TECH, I-TECH, I-TECH.’ The final representation of each token is a tuple of ‘doc\_id, seq\_id, tokens, pos\_tags, boi\_tags.’ The following shows the resulted dataframe:

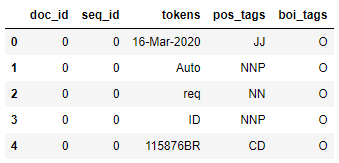


Figure 2. Resulting dataframe from data preprocessing

## Prepare the data and model construction

The first step in this process is reconstructing sentences. It is done by grouping the tokens by ‘seq\_id.’ The second step is converting textual data into a numerical representation; there are three pairs of lookup dictionaries needed: the token-to-index and index -to-token, the postag-to- index and index -to-postag, and the boitag-to- index and index -to-boitag. Then the two placeholders for padding and out-of-vocabularies are added to the dictionaries. After this, all tokens, pos tags, and boi tags are converted into index format. The following shows a reconstructed sentence and its numeric representation:

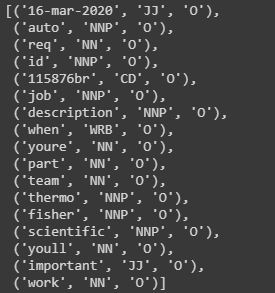


Figure 3. Reconstructed sentence



Figure 4. Numeric representation of the reconstructed sentence

The first two numeric arrays(tokens and pos tags) are then padded to 150 fixed lengths so the embedding layer can process them. The boi tag array is processed with one-hot-encoding for the CRF layer to lookup.

There are two models constructed in this project to compare the influence of adding pos tags as additional input features to the Bi-LSTM layer. The following shows the model structures:

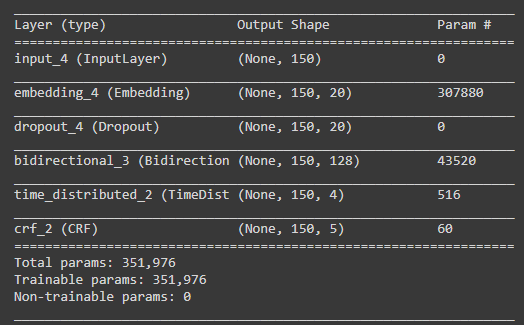


Figure 5. The sequential LSTM-CRF model takes one input from the embedding layer

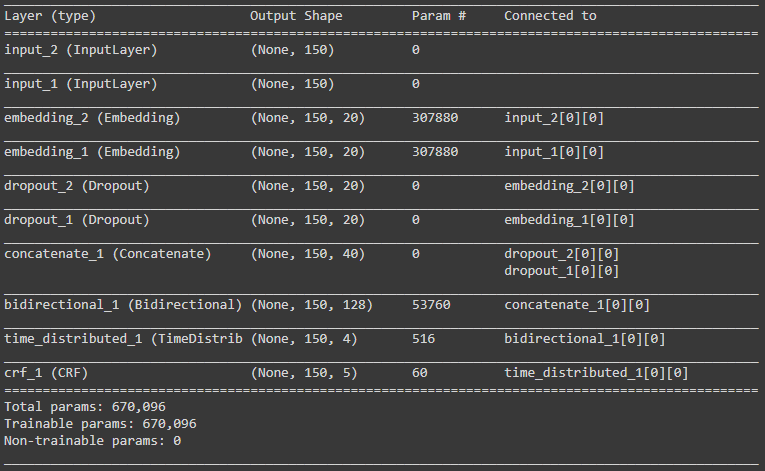


Figure 6. The functional LSTM-CRF model takes two inputs from the embedding layers

## Model Evaluation

## The following shows the confusion matrixes of both models:

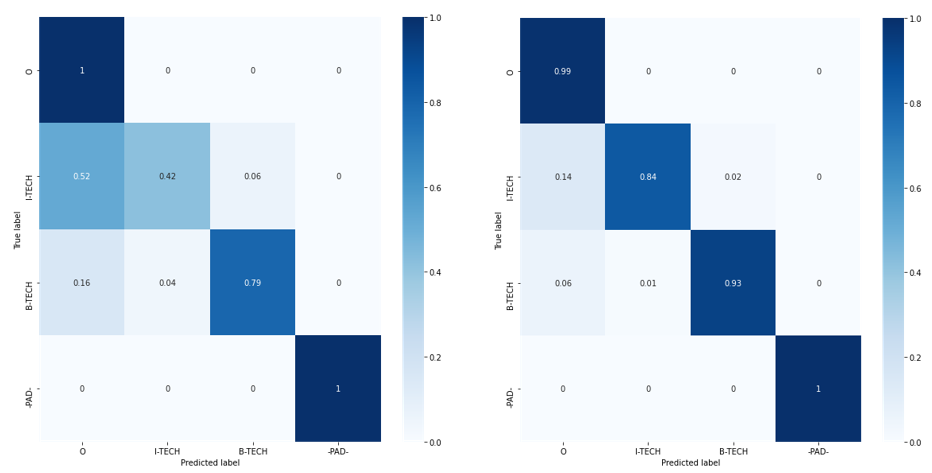


Figure 7. The confusion matrixes for the sequential model(left) and the functional model(right)

As shown above, adding pos tags as additional input features significantly increases the true positive rate for both ‘B-TECH’ and ‘I-TECH’ classes. Without the pos tags, the model incorrectly predicts 52% of the ‘I-TECH’ classes as ‘O’ and 6% as ‘B-TECH.’ Also, there are 16% of the ‘B-TECH’ samples are misclassified as ‘O’ and 4% as ‘I-TECH.’ In comparison, adding pos tags to the model reduces the misclassification rate of ‘I-TECH’ to 14% and 2%, and the misclassification rate of ‘B-TECH’ to 6% and 1%.

In conclusion, the LSTM-CRF model can effectively hand sequential labeling for customized NER tasks, and adding pos tags as the additional feature can significantly improve the performance.

References

Donghuo Zeng, Chengjie Sun, Lei Lin, & Bingquan Liu. (2017). LSTM-CRF for Drug-Named Entity Recognition. Entropy, 19(6), 283. https://doi-org.proxymu.wrlc.org/10.3390/e19060283