E1-246 Natural Language Understanding Assignment-3

Sayambhu Sen
M.Tech (1st Year)
SE (Department of EE)
SR No. 14351
sayambhusen@iisc.ac.in

Abstract

This is a report for the third assignment of Natural Language Understanding. In this assignment we are trying to perform Named Entity Recognition using a conditional Random field and then using a neural language model called BiLSTM which is an LSTM with bidirectional state inputs and outputs. This is coupled with a CRF to try and predict the output of the token. We have to classify tokens as diseases('D'), or treatments('T') or other ('O') from some sentences.

1 Language Modelling tools

The language model implemented in this assignment is the CRF Model and the BiLSTM Model. We are using 'keras' to implement our BiLSTM and pysuite crf to implement the Conditional Random Field. The package 're' is the regular expression package which is used to preprocess our data. The 'nltk' package has been mainly used for using the snowball Stemmer.

2 Introduction

Named Entity Recognition is a task in which we take in as input a sequence of tokens in a sentence and assign a tag on that each token or phrase in that sequence. The importance of the conditional random field model is that it's output is an undirected graphical model and hence, it models sequences of outputs given corresponding sequences of inputs. In this assignment we will use CRF using just word features as well as using the neural model called BiLSTM. All data is preprocessed for keeping space for unknown words by removing some of the single frequency words. All words are lemmatized by the Snowball stemmer before processing.

3 CRF model

We use the inbuilt library of python called 'pycrfsuite'. The given input file is a text file with one word being followed by the tag with a space in between. End of sentences are indicated by an empty line. We take that data and store it in the form of lists of sentences which are essentially lists of words. Thus, the output tags are lists of tags.

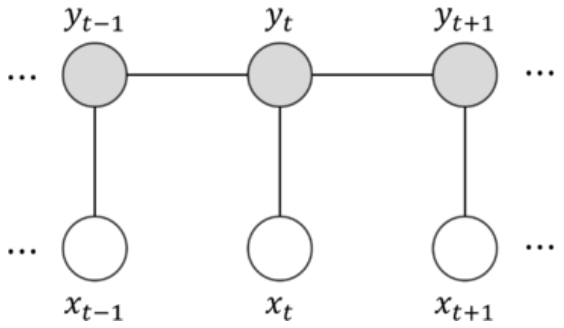
The inbuilt function takes in a sequence and tried to model the output as a distribution of $P(Y\|X)$ where the X is the input feature sequence and Y is the output label sequence which are connected by an undirected graphical model. X and Y are disjoint and also it is to be noted that X is a sequence of feature sets. Hence, X can be a tuple. In essence, this means X can be a very large feature vector. The main feature of the CRF model is that using the counts for each tag , it tries to fit in a tag for each output. Also, the features we give for each node of the input sequence matters a lot for correct classification

In our case, we have experimented with different feature sets for prediction and finally settled on a particular feature set. At first, we used the function for Mallet which seemed to give high accuracy a first. However, we must note that in our given text input, we have an overwhelmingly, large number of word tokens which are tagged as 'O'. Hence, accuracy will be a very misleading value as it might seem like it is giving a very good model. Indeed labelling all tokens as 'O' does indeed give above 90 percent accuracy. Thus, a better metric which we could use in our case could be the Precision and Recall for each individual label.

4 BiLSTM CRF Model

This is a neural network model. Unlike the previous model which uses counts to model the distribution of the graphical models, here we do not

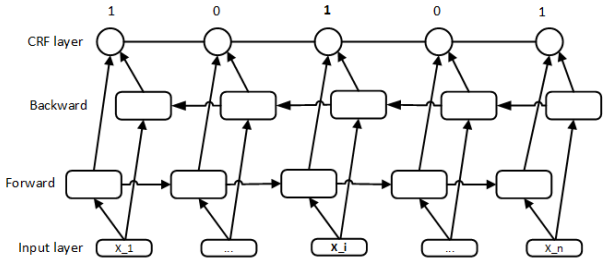
Figure 1: Graphical model for CRF



have to make any explicit counts in order to make the model. Rather, we use a neural network which given the input and outputs modifies it's weights to approximate the appropriate function.

In our case, we use a Bidirectional LSTM since our output is a sequence which is connected both ways and LSTMs are ideal for capturing sequences. Also, we use a model in which on top of the LSTM we stack a CRF. Hence, it is sort of like a Deep model but the later stage is not a neural network. We have found that the literature states this to be a good model for sequence tagging tasks. Here too, we give the sentence sequence as an input and train it on the output sequence tag. The only difference would be that the neural network has fixed sized inputs and outputs and hence, we must keep all inputs padded with <UNK >.

Figure 2: BiLSTM CRF layers



4.1 Training of the models

For CRF we just try to make feature files and get the corresponding outputs. At each line we make the features of each word and the corresponding label output. We add or subtract features based on the training data and validate on the validation error. Finally for the lowest validation error , we test on the test error. The test and validation data are each 10 percent of the data while the training error consists of 80 percent of the data. Hence, the data is split based on sentences.

For neural models again, we make sentences but we take as input only the words. There is no feature engineering involved here. Rather we create a dictionary of words and give it as input to the model's embedding layer. Thus, we let the model take care of the embedding instead of worrying about making the embeddings ourselves.

Each sequence or sentence is independent of other sentences and hence, we can train our model on each sequence of sentences padded accordingly with the <UNK >token. Thus, the neural model can be trained for any size of data and we choose to train it on batch sizes of 25.

4.2 Results

4.2.1 CRF

Now, we will show addition of different features adds to the accuracy of the CRF model. The reason we only show these features are because these are the features which contribute to the maximum accuracy, all other features that we tried to use reduced the accuracy of the model.

Table 1: Features and corresponding accuracy values

$Featureused$	$TrainAccuracy$
Word only	67 %
Word, Tag	72 %
Word,Tag,Capital	74 %
Word,Tag and,Capital,Embedding	78 %

Finally, we perform an ablation study as told and find which features are important. We always keep Word, Tag, Capitalization Indicator and Embedding at the beginning and check for error after removing one of these features.

Table 2: Features removal effect (Ablation study)

$Featureused$	$TrainAccuracy$
Word removal	75 %
Tag removal	73 %
Capital removal	75 %
Embedding removal	69 %

Thus, of all these models, the embedding contributes to the major factor since, it is captures a lot of semantic as well as syntactic meaning and thus, is a very good identification factor for Named Entity recognition tasks.

Table 3: Confusion matrix for CRF(True label vs Predicted label)

	O	D	T
O	3798	51	77
D	514	569	19
T	832	19	291

Also, the confusion matrix is as follows:

Thus, as we can see there is a lot of confusion on predicting T as most of the time the model predicts a T as O. The precision and recall for D and T is given below.

Table 4: Precision and Recall for D and T (Macro)

	$Precision$	$Recall$
D	51.6%	89%
T	25.4%	75.1%

4.2.2 BiLSTM CRF

Here we take in as input a sequence (a sequence of words) and give as output a sequence of tags. The sequence is in the form of a fixed length since neural networks can take in fixed length sequences. Thus, it uses padding to fix the input length. The model is trained for 5 epochs in batch sizes of 25 sequences. The results are as given below.

Table 5: Confusion matrix for CRF(True label vs Predicted label)

	O	D	T
O	25676	37	23
D	347	511	7
T	578	15	181

Table 6: Precision and Recall for D and T (Macro)

	$Precision$	$Recall$
D	61.1%	90.77%
T	31.9%	85.78%

Thus, we see it improves on all counts but it still has a marginal improvement in precision of T.

5 Discussion

It is to be noted that in both methods we have used some kind of neural network methods. In the first method (CRF), we trained word embedding vectors using a neural model for feature representation, which was used in the CRF itself. In the second model, we created a word dictionary and let the model itself create the embeddings according to the task required (in this case NER tagging). Since word embeddings capture local information it turns out to be a very good feature to be used for NER tagging.

In this assignment we could not use the semantic labelling of tokens using wordnet or MESH since, we did not how to assign semantic labels to tokens. However, it could be a useful feature to increase accuracy. Shallow chunking or unlabelled data for creating embeddings were not used.

In the case of neural models, we could have used a deep modelling method which uses a stacked LSTMs with a lower layer giving as output the POS tags and an upper layer giving as output NER tags. But we did not know how to get an output from a middle layer so it was not used. Another method could have been that we input a concatenation of a one hot vector for a word and tag to output the NER tag. We use nltk's inbuilt POS tagger so that could be simple enough.

References

- picture of CRF
  https://ko.wikipedia.org/wiki/%EC%A1%B0%EA%B1%B4
- picture of BiLSTM CRF
  https://www.researchgate.net/figure/BI-LSTM-CRF-model
- https://github.com/scrapinghub/python-crfsuite/blob/master/examples/CoNLL%202002.ipynb
- https://www.depends-on-the-definition.com/sequence-tagging-lstm-crf/