Named Entity Recognition: initial analysis

Lodi Dodevksa, Viktor Petreski
University of Ljubljana
Faculty for computer and information science
Večna pot 113, SI-1000 Ljubljana
lodi.dodevska@gmail.com, vpetreski96@gmail.com

1 Introduction

After we did some research on classic and state-of-the-art Named entity recognition models, we chose to implement Conditional Random Fields (CRFs), which will represent the baseline for our work. Next, we decided to continue with a model of Bi-LSTM with a CRF layer. We chose to use rather simple embedding for the data, because first we wanted to get familiar with the Bi-LSTM models, explore the influence of the parameters on the performance and then use BERT and/or ELMO or similar embeddings for the final part of the assignment. We are using CoNLL-2003 and GMB datasets, which are popular for experimenting in NER-related tasks. We did our experiments on both of the obtained datasets at the same time.

2 Data preparation

Both of the datasets that we use have been preprocessed in the previous part of the assignment. Now, before we started with training the models we had to prepare appropriate train and test sets. CoNLL-2003 dataset already contains three subsets: the largest set is the training data and the two smaller sets (*testA* and *testB*) are for testing. On the other hand, the GMB dataset did not come split into subsets for training and testing, so we split the dataset into training and test subset with ratio 70:30. In the Bi-LSTM models, we used 10% of each training set as a validation set.

The input sequences for CRF and Bi-LSTM-CRF are different. CRF uses the words and the corresponding POS tags for creating the input features. Bi-LSTM-CRF uses only the tokens of the sentences. Bi-LSTM-CRF expects a numerical input, so we convert each token and label (tag) of the vocabulary to a corresponding integer index. The good side of this representation is that it does not require a lot of time to train. Another thing that we must take care of is the length of the sequences. In order to hand over the data as an input to Keras model, all the sequences need to have the same length. So, we found the longest sentence and set its length to be the maximum length of the input sequence. The sentences that are shorter than this are padded to get to the required length.

3 Models

As mentioned before, we are using two different models for Named entity recognition. The first model (CRF) is supposed to represent the baseline for our work and the second model (Bi-LSTM-CRF) is the one we will continue to work on and try to improve by the end of the third assignment.

3.1 CRF

Conditional Random Field is a statistical model which is used very often in the NER field. Up to some time ago the most successful NER models were based on CRF. We provide a list of features for each token in the sentence as an input to this model. Based on these features, the model tries to predict a label for each word in the sentence, or, better said, an optimal sequence of labels for the entire sentence. The hard part here is finding the types of features that are useful and can improve the predictions. Luckily, CRFs are widely popular and there is a lot of research done. We will use the feature dictionary suggested by *sklearn-crfsuite*. Some of the token's properties that are taken into account are whether it starts with uppercase or lowercase, what is its POS tag, is it a digit or not, etc.

3.2 Bi-LSTM with CRF layer

Bi-LSTM networks are used when we need to process sequences where the past and future features are equally important, just like in our case. The first layer in our network is the Input layer, which carries only the shape of the input data. Next comes the Embedding layer. It takes three parameters as an input: size of the vocabulary of the dataset, dimensions of the embedding (size of the vector space in which the words will be embedded) and the length of the input. Since all our sentences are padded, this length is actually the length of the longest sentence.

This layer is followed by Bi-LSTM layers. Bi-LSTM is comprised of two separate LSTM layers. One reads the sentence from the beginning to the end (left to right) and the other reads it in a reversed order. We are going to optimize this layer with respect to *units*, *dropout* and *recurrent_dropout*. The *return_sequence* parameter is set to True, which means that the layer will return the full sequence of the output, and not just a final value. That is why after the Bi-LSTM comes the TimeDistributed layer. This layer allows to apply the next layer to every element of the sequence independently. The outputs of the Bi-LSTM are actually scores of each label. These scores are handed over to the CRF layer and the label that has the highest score will be selected as a prediction for the token. We could easily create a model without the last CRF layer, but researchers have shown that using CRF can improve the performance of the network, because it allows to better capture the context of the sequence.

4 Results

In almost all of the cases we obtained about 97% - 99% accuracy. But, we could not use this measure to evaluate our models because it is not appropriate for this type of predictions. We are expecting that most of the tokens are assigned the label **O** since it is the most common label and it outnumbers the other labels which are much more important. That is why we are using F1-score for evaluation, which is a better metric in the case of imbalanced labels.

For the CRF model we used *lbfgs (Gradient descent using the L-BFGS method)* algorithm for optimization, because it is supposed to achieve better solution than regular Gradient descent with less iterations. We optimized the regularization parameters *c1* and *c2*. We also combined the training dataset of CoNLL-2003 with one of its test sets, in order to see whether that will contribute to better performance. In Table 1 are the obtained results on the CoNLL-2003 dataset. We achieved better F1-score (89%) when we trained the CRF on the original training set and on the combined original set + test_B. Table 2 shows the best F1-scores for the GMB dataset. 84%-85% F1-score is achieved with all the given parameter combinations.

c1	c2	F1-score	Train set	Test set
0.0910	0.0029	89%	Train	A
0.1000	0.0010	81%	Train	B
0.5000	0.1000	81%	Train, A	B
0.0677	0.1949	82%	Train, A	B
0.3724	0.3135	88%	Train, B	A
0.1561	0.3552	89%	Train, B	A

Table 1: F1-scores for CRF models on CoNLL-2003 dataset

c1	c2	F1-score	Train set	Test set
0.0910	0.0029	84%	Train	A
0.1000	0.0010	84%	Train	B
0.5000	0.1000	85%	Train, A	B
0.0677	0.1949	85%	Train, A	B
0.3724	0.3135	84%	Train, B	A
0.1561	0.3552	85%	Train, B	A

Table 2: F1-scores for CRF models on GMB dataset

For the Bi-LSTM model, we used relatively small dropout and recurrent dropout rates, somewhere between 0.1 and 0.2. The performance did not improve with increasing these values. We also tried different embedding sizes, but the best F1-score was achieved for embedding size 45-50. Even though we started with fairly small number of epochs, eventually we realized that with more epochs we can obtain better results, but, as always, there is a point of diminishing return. After increasing the number of epochs over 15, the improvement was negligent.

The best results for CoNLL-2003 are presented in Table 3 and for Groningen Meaning Bank dataset in Table 4 . We achieved F1-scores of 85% and 82% for CoNLL and GMB respectively. We expect to improve these values with using more complex embedding in the future.

Emb. size	Drop-out	Rec. dropout	Epochs	LSTM units	F1 score	Train set	Test set
40	0.1	0.1	10	50	83%	Train, B	A
30	0.1	0.1	8	50	62%	Train, B	A
40	0.1	0.1	8	50	75%	Train, A	B
45	0.15	0.1	12	80	85%	Train, B	A
50	0.2	0.15	15	90	85%	Train, B	A
40	0.1	0.1	8	50	73%	Train, A	B
45	0.1	0.1	30	85	85%	Train, B	A

Table 3: F1-scores for Bi-LSTM-CRF on CoNLL-2003 dataset

Emb. size	**Drop-out**	**Rec. dropout**	**Epochs**	**LSTM units**	**F1 score**	**Train set**	**Test set**
20	0.5	0.5	5	40	77%	GMB train	GMB test
30	0.3	0.3	5	40	77%	GMB train	GMB test
30	0.3	0.3	5	50	79%	GMB train	GMB test
30	0.3	0.3	10	50	81%	GMB train	GMB test
40	0.1	0.1	10	80	82%	GMB train	GMB test
45	0.15	0.1	12	80	82%	GMB train	GMB test

Table 4: F1-scores for Bi-LSTM-CRF on GMB dataset

5 Conclusion and future work

Now that we are familiar with Bi-LSTMs, we are willing to explore this field furthermore. The first step will be to use BERT and/or ELMO embeddings (Straková et al., 2019) and to optimize the parameters for the new models. Next, we could try to develop a different kind of model, i.e. Bi-LSTM-CNN (Chiu and Nichols, 2016), if we have enough time.

References

Jason P.C. Chiu and Eric Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370.

Jana Straková, Milan Straka, and Jan Hajic. 2019. Neural architectures for nested NER through linearization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5326–5331. Association for Computational Linguistics.