Multilingual Named Entity Recognition Using Pretrained Embeddings, Attention Mechanism and NCRF

Anton A. Emelyanov
MIPT, Sberbank
Moscow, Russia
login-const@mail.ru

Ekaterina Artemova
National Research University
Higher School of Economics
echernyak@hse.ru
echernyak@hse.ru

Abstract

In this paper we tackle multilingual named entity recognition task. We use the BERT Language Model as embeddings with bidirectional recurrent network, attention, and NCRF on the top. We apply multilingual BERT only as embedder without any fine-tuning. We test out model on the dataset of the BSNLP shared task, which consists of texts in Bulgarian, Czech, Polish and Russian languages.

1 Introduction

Sequence labeling is one of the most fundamental NLP models, which is used for many tasks such as named entity recognition (NER), chunking, word segmentation and part-of-speech (POS) tagging. It has been traditionally investigated using statistical approaches (Lafferty et al., 2001), where conditional random fields (CRF) (Lafferty et al., 2001) has been proven to be an effective framework, by taking discrete features as the representation of input sequence (Sathiya and Sellamanickam, 2007). With the advances of deep learning, neural sequence labeling models have achieved state-of the-art results for many tasks (Peters et al., 2017).

For the purpose of this paper, we consider neural network solution for multilingual named entity recognition for Bulgarian, Czech, Polish and Russian languages for the BSNLP 2019 Shared Task (Piskorski et al., 2019). Our solution is based on BERT language model (Devlin et al., 2018), use bidirectional LSTM (Hochreiter and Schmidhuber, 1996), Multi-Head attention (Vaswani et al., 2017), NCRFpp (Yang and Zhang, 2018) (being neural network version of CRF++framework for sequence labelling) and Pooling Classifier (for language classification) on the top as additional information.

2 Task Description

2.1 Data Format

The data consists of raw documents and the annotations, separately provided by the organizers. Each annotation contains a set of extracted entities and their types without duplication. We convert each raw document and corresponding annotations to labeled sequence and predict named entity label for each token in the input sentence. The documents are categorized into topics. There are two topics in the dataset released first: named “brexit” and “asia_bibi”.

2.2 Tasks

The BSNLP Shared Task has three parts (Piskorski et al., 2019):

1. Named Entity Mention Detection and Classification;
2. Name Lemmatization;
3. Cross-lingual entity Matching.

For more details about the dataset and the task refer to the description on the web page[1]. We focused on Named Entity Mention Detection (Named Entity Recognition) in this work.

3 System Description

We propose modeling the task as both sequence labeling and language classification jointly with a neural architecture to learn additional information about text. The model consists of one encoder, which on its own is build from the pretrained multilingual BERT model, followed by several trainable layers and two decoders. While the first decoder generates output tags, the second decoder

[1] Full BSNLP Shared Task description available at http://bsnlp.cs.helsinki.fi/shared_task.html.

Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing, pages 94–99,
Florence, Italy, 2 August 2019. ©2019 Association for Computational Linguistics

identifies the language of the input sentence[2]. The system architecture is presented in Figure 1 and consists of seven parts:

1. BERT Embedder as pretrained multilingual language model;
2. Weighted aggregation of BERT output;
3. Recurrent BiLSTM layer to be trained for the NER task;
4. Multi-Head attention to take shorter dependencies between words into account;
5. linear layer as the head of the encoder part;
6. NCRF++ inference layer for decoding, i.e. final sequence labelling;
7. Concatenation operation of Max Pooling, Average Pooling and last output of Multi-Head attention layer, later passed to linear layer for classification as a second decoder for language identification.

3.1 Neural Network Architecture

3.1.1 BERT Embedder

The BERT embeddings layer contains Google's original implementation of multilingual BERT language model. Each sentence is preprocessed as described in BERT paper (Devlin et al., 2018):

1. Process input text sequence to WordPiece embeddings (Wu and Mike Schuster, 2016) with a 30,000 token vocabulary and pad to 512 tokens.
2. Add first special BERT token marked "[CLS]".
3. Mark all tokens as members of part "A" of the input sequence.

But instead of BERT's original paper (Devlin et al., 2018) we keep "B" ("Begin") prefix for labels and do a prediction for "X" labels on training stage. BERT neural network is used only to embed input text and don't fine-tune on the training stage. We freeze all layers except dropout here, that decreases overfitting.

We take hidden outputs from all BERT layers as the output of this part of the neural network and pass to the next level of the neural network. So the shape of output is 12×768 for each token of 512 length's padded input sequence.

3.1.2 BERT Weighting

Here we sum all of BERT hidden outputs from previous part:

$$o_i = \gamma \times \sum_{i=0}^{m-1} b_i s_i \tag{1}$$

where

- o_i is output vector of size 768;
- $m = 12$ is the number hidden layers in BERT;
- b_i is output from i BERT hidden layer;
- γ and s_i is trainable task specific parameters.

As we do not fine-tune BERT, we should adapt its outputs for our specific sequence labeling task. The suggested weighting approach is similar to ELMo (Peters et al., 2018), with a lower number of weighting vectors parameters s_i. This approach can help to learn importance of each BERT output layer for this task and and network doesn't lose too much information about text, that was stored in all BERT outputs.

3.1.3 Recurrent Part

This part contains two LSTM networks for forward and backward passes with 512 hidden units so that the output representation dim is 1024 for each token. We use a recurrent layer for learning dependencies between tokens in an input sequence (Hochreiter and Schmidhuber, 1996).

3.1.4 Multi-Head Attention

After applying the recurrent layer, we use Self-attention mechanism to learn any other dependencies in a sequence for each token. This can be denoted as $D(d_h|S)$, where D is some hidden dependency; d_h is the h head of attention, and S is all sequence. each head can learn its dependencies such as morphological, syntactic or semantic relationships between words (tokens). Presumably, dependencies may look as shown at Figure 2. Also, mechanism attention can compensate limitations of the recurrent layer when working with long sequences (Bahdanau et al., 2015). In our

[2] Our code is available at https://github.com/anonymize/slavic-ner. This code is based on https://github.com/sberbank-ai/ner-bert.

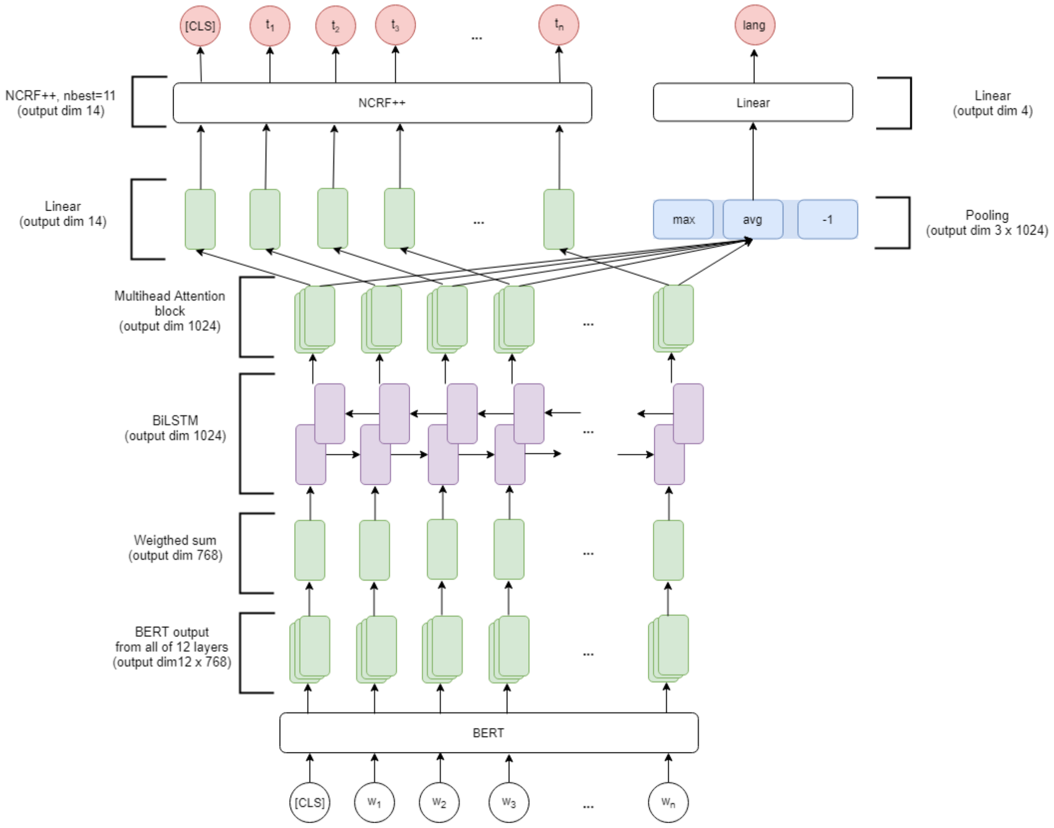

Figure 1: The system architecture

architecture, we use multihead-attention block as proposed in the paper “attention is all you need” (Vaswani et al., 2017). We took 6 heads and value and key dim 64.

3.1.5 Inference for NER Task

After the input sequence was encoded, we achieve the final representation of each token in a sequence. This representation is passed to Linear layer with $tanh$ activation function and gets a vector with 14 dim, that equals to the number of entities labels (include supporting labels “pad” and “[CLS]”). The inference layer takes the extracted token sequence representations as features and assigns labels to the token sequence. As the inference layer, we use Neural CRF++ layer instead of vanilla CRF. That captures label dependencies by adding transition scores between neighboring labels. NCRF++ supports CRF trained with the sentence-level maximum log-likelihood loss. During the decoding process, the Viterbi algorithm is used to search the label sequence with the highest probability. But also, NCRF++ extends the decoding algorithm with the support of $nbest$ output (Yang and Zhang, 2018). We chose the $nbest$ parameter equal to 11, because we have 11 meaningful labels. In this decision we followed the original article (Yang and Zhang, 2018).

3.1.6 Inference for Language Classification

We train our system for language classification. For the classification inference, we use Pooling Linear Classifier block as proposed in ULMFiT paper (Howard and Ruder, 2018). We pass output sequence representation H from Multihead-attention part to different Poolings and concat (as shown in Figure 1):

$$h_c = [h_0, maxpool(H), meanpool(H)] \qquad (2)$$

where [] is concatenation;

h_0 is first output significant vector of Multihead-attention part (which does have “[CLS]” label).

The result of concat Pooling (3×1024) is passed to Linear layer, and that predicts probability for four language classes (Bulgarian, Czech, Polish and Russian).

3.2 Postprocessing Prediction

After getting labels for the sequence of WordPiece tokens, we should convert prediction to word level labels extraction named entities. Each WordPiece token in the word is matched with neural network label prediction. We use ensemble classifier on labels by count all predicted labels for one word except “X” and select label for a word with the higher number of votes.

For final prediction we unite token’s sequences which have not “O” (“Other”) label to spans and write to result of entities set.

4 Training the System

4.1 Data Conversion

On the training stage we divide the input data into two parts: the training set (named “brexit”) and development set (named “asia_bibi”). Hence we train the system on one topic and evaluate the system on another topic. Because the input contains raw text and annotation, but BERT take words sequence as input, we convert data to word level IOB markup (Ramshaw and Marcus, 1995). After that, each word was tokenized by WordPiece tokenizer and word label matched with IOBX labels.

On the prediction stage result, labels were received by voice classifier. After this, we transform word predictions to spans markup. The results of develop evaluation stage described in Table 1.

After evaluation stage we train our network on all input data (“brexit” and “asia_bibi”) to make final predictions on the blind test set.

4.2 Training Procedure

The proposed neural network was trained with joint loss:

$$\mathcal{L} = \mathcal{L}_{SL} + \mathcal{L}_{clf} \tag{3}$$

where $\mathcal{L}_{SL}$ is maximum log-likelihood loss (Yang and Zhang, 2018) for the sequence labeling task and $\mathcal{L}_{clf}$ is Cross Entropy Loss for the language classification.

We use Adam with a learning rate of $1e - 4$, $\beta_1 = 0.8$, $\beta_2 = 0.9$, $L2$ weight decay of 0.01, learning rate warm up, and linear decay of the learning rate. Also, gradient clipping was applied for weights with $clip = 1.0$.

Training of proposed neural network architecture was performed on one GPU with the batch size equal to 16, the number of epochs equal to 150, but stopped at epoch number 80 because the loss function has ceased to decrease. The model required only around 3 GB of memory instead of fine-tuning all BERT model, which would have required more than 8 GB GPU memory. All training procedure lasted around five hours on one GPU with the evaluation of development set on each epoch.

The final model was trained on unit of training and development datasets.

5 Results and Discussion

5.1 Evaluation Results

As baseline for BSNLP Shared Task we use a simple CRF tagger and obtain exact word level f1-score 0.372 on the development dataset.

Finally we use joint model for named entity recognition task and language classification task because the model without part of the classification gave a result by several percent less than proposed final model. This means that the joint model pays attention to a specific language morphology and some connections between words within one language.

label	precision	recall	f1-score
PER	0.733	0.725	0.729
PRO	0.384	0.547	0.451
EVT	0.385	0.370	0.377
LOC	0.648	0.872	0.744
ORG	0.550	0.630	0.587
avg/total	0.540	0.629	0.578

Table 1: Evaluation metrics on development dataset

For proposed neural network architecture the evaluation of the training stage was produced on development dataset. Table 1 shows span-level metrics precision, recall, and f1-measure. For development set, we obtained the following scores: language classification quality (f1-score): 0.998 and Multilingual Named Entity Recognition quality (f1-score): 0.70 for exact word level matching and 0.578 for exact full entities matching. Also we train model without language classification, which resulted in f1-score equal to 0.66 . This confirms the impact of language classification. Our model significantly outperforms the CRF baseline.

The evaluation of test dataset presented in Table 2 (relaxed partial matching) and Table 2 (relaxed exact matching) is measured by the BSNLP Shared Task organizers.

Relaxed partial matching

label	precision	recall	f1-score
PER	0.84955	0.87119	0.86023
LOC	0.77526	0.93197	0.84642
ORG	0.62642	0.87170	0.72898
PRO	0.42079	0.81416	0.55483
EVT	0.24074	0.15476	0.18841
All	0.90142	0.69917	0.78752

Relaxed exact matching

label	precision	recall	f1-score
PER	0.76835	0.74023	0.73317
LOC	0.87747	0.73014	0.79705
ORG	0.71390	0.52295	0.60369
PRO	0.34439	0.18506	0.24075
EVT	0.10714	0.16667	0.13043
All	0.56225	0.46901	0.50102

Table 2: Evaluation metrics on test dataset

5.2 Error Analysis

First of all, we face some errors with converting from origin data format (raw and annotations) to word markup and back to origin format after predictions were made. This problems stand for extra spaces, bad Unicode symbols and symbols, absent in WordPiece vocabulary. Other errors are caused by neural network prediction failures. The model turns to be overfitted on the negative label “O” so that there are many false positives in the prediction. Lastly, the infrequent labels “PRO” and “EVT” are often confused.

6 Related Work

The related work has several parts: firstly, our work follows the recent trend of using pretrained neural languages models, such as (Devlin et al., 2018; Peters et al., 2018; Howard and Ruder, 2018). The main difference between original BERT’s approach for named entity recognition task (Devlin et al., 2018) we use its only as input embeddings of sequence without fine-tuning. From ELMo paper (Peters et al., 2018) we use weighting approach for different outputs from network and getting final representation of sequence. From ULMFiT work we took part which is related to the final decoding for classification (Pooling Classifier) without proposed language model (Howard and Ruder, 2018). Secondly we model the task of NER as a joint sequence labeling and classification task following other joint architectures (Liu and Lane, 2016; Nguyen et al., 2016).

7 Conclusion and Future Work

We have proposed neural network architecture that solves Multilingual Named Entity Recognition without any additional labeled data for Bulgarian, Czech, Polish and Russian languages. This implementation allows to train the model even on a modern personal computer with GPU. This neural network architecture can be used for other tasks, that can be reformulated as a sequence labeling task for any other language.

As the next steps in the study of the underlying architecture, we can increase or decrease the number of units on each layer or remove the recurrent layer or multihead-attention layer. As improvements of the system, we can fine-tune BERT embeddings and put additional layers on top of BERT or pass other modern language models as an input.

Acknowledgments

The article was prepared within the framework of the HSE University Basic Research Program and funded by the Russian Academic Excellence Project “5-100”. We are thankful to the Muppets and to the BSNLP shared task organizers.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.

Sepp Hochreiter and Jürgen Schmidhuber. 1996. Lstm can Solve Hard Long Time Lag Problems. In *NIPS*.

Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *ACL*.

John D. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *ICML*.

Bing Liu and Ian Lane. 2016. Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling. In *INTERSPEECH*.

Thien Huu Nguyen, Kyunghyun Cho, and Ralph Grishman. 2016. Joint event extraction via recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page 00309, San Diego, California. Association for Computational Linguistics.

Matthew E. Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. 2017. Semi-supervised sequence tagging with bidirectional language models. In *ACL*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Jakub Piskorski, Laska Laskova, Micha Marciczuk, Lidia Pivovarova, Pavel Pib, Josef Steinberger, and Roman Yangarber. 2019. The second cross-lingual challenge on recognition, classification, lemmatization, and linking of named entities across Slavic languages. In *Proceedings of the 7th Workshop on Balto-Slavic Natural Language Processing*, Florence, Italy. Association for Computational Linguistics.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text Chunking using Transformation-Based Learning. *CoRR*, cmp-lg/9505040.

Keerthi Sathiya and Sundararajan Sellamanickam. 2007. *Crf versus svm-struct for sequence labeling*, volume 1. Yahoo Research Technical Report.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*.

Yonghui Wu and Quoc VLe Mohammad Norouzi Wolfgang Macherey Maxim Krikun Yuan Cao Qin Gao Klaus Macherey Mike Schuster, Zhifeng Chen. 2016. *Googles neural machine translation system: Bridging the gap between human and machine translation*, volume arXiv:1609.08144.

Jie Yang and Yue Zhang. 2018. Ncrf++: An Open-source Neural Sequence Labeling Toolkit. In *ACL*.