Gapping parsing using pretrained embeddings, attention mechanism and NCRF

Emelyanov A. A. (login-const@mail.ru), Artemova E. L. (echernyak@hse.ru)

Moscow Institute of Physics and Technology,
Nationa Research University Higher School of Economics,
Moscow, Russia

The article is devoted to the problem of automatic gapping resolution for the Russian language. We use BERT Language Model as embeddings with bidirectional recurrent network, attention, and NCRF on the top. Unlike other models these are using BERT, we apply BERT only as embedder without any fine-tuning. As a result, our implementation took second place in the AGRR-2019 competition.

Key words: gapping resolution, BERT embeddings, attention, NCRF.

Разрешение гэппинга с использованием предобученных эмбеддингов, механизма внимания и NCRF

Емельянов А. А. (login-const@mail.ru), Артемова Е. Л. (echernyak@hse.ru)

МФТИ, Москва, Россия

Статья посвящена проблеме автоматического разрешения гэппинга для русского языка. Мы используем языковую модель BERT в качестве эмбеддингов с двунаправленной рекуррентной нейронной сетью, механизмом внимания и NCRF на верхнем слое сети. В отличие от других моделей, в которых используется BERT, мы применяем BERT только в качестве эмбеддингов без какого-либо дообучения. В результате наша реализация заняла второе место в конкурсе AGRR-2019.

Ключевые слова: разрешение гэппинга, BERT эмбеддинги, механизм внимания, NCRF.

1 Introduction

In natural language along with the surface level (the material that we read or hear), there is deep structure. The deep structure can differ in many aspects from the surface. One of the cases is omition of repeating elements. If the elements can be unambiguously restored from the previous linguistic context, such procedure is called ellipsis. Our work touches upon one type of ellipsis, namely gapping. Gapping is omition of repeating predicate in coordinate (and probably subordinate) structure while its arguments remain expressed. Consider the example 'John likes tea, and Mary coffee', where the second clause lacks the predicate 'likes', but has two of its arguments (“Mary“ and “coffee“) [4].

While having been studying theoretically for decades (Ross 1970, Hankamer 1979, Coppock 2001, [1, 2, 3]) the phenomenon still has been not illustrated with sufficient corpora which is a prerequisite for developing of automatic systems. In the framework of the Automatic Gapping Resolution for Russian competition (AGRR-2019) such corpus for the Russian language was presented.

The data consists labeled text sequences (see Section 2.1 for detailes). So we decided to address this problem as sequence labelling task and predict gap label for each token in the input sentence. For the purpose of this paper, we consider neural network solution for automatic gapping resolution for Russian language in proceedings of AGGR-2019 challenge [5]. Our solution based on BERT language model [6], use bidirectional LSTM [7], Multi-Head Attention [8], NCRFpp [9] (being neural network version of CRF++framework for sequence labelling) and Pooling Classifier (for classification) on the top.

2 Task description

2.1 Data format

Input data consists of sentences without any additional markup (raw texts). For each sentence output should contain 7 columns. First column should have 0 or 1 in it, depending on presence of gapping construction in the sentence. Other output cells separated with tab symbol correspond gapping element names (cV, cR1, cR2, V, R1, R2) and should contain char offsets (first symbol in each sentence has offset 0 1) for annotation borders (two numbers separated by colon (:) symbol) for each gapping element. If the provided sentence lacks certain gapping element, the corresponding cell should not contain any symbols. For example: “Аналогичным образом, среднегодовой прирост ВВП на душу населения, который в странах, расположенных к югу от Сахары, составлял в период с 1965 по 1973 год 3 процента, cV [упал cV] cR1 [с 1980 до 1986 года cR1] cR2 [на 2,8 процента cR2], R1 [в 1987 году R1] — V[] R2 [на 4,4 процента R2] и R1 [в 1989 году R1] — V[] R2 [на 0,5 процента R2]”. For the binary presence-absence classification for each sentence all the output cells except the first one are ignored. For gap resolution task cells in columns cR1, cR2, R1, R2 are ignored. For the full annotation task all output cells are evaluated [5].

2.2 Tasks

The task contains three parts [5]:

1. Binary presence-absence classification. For each sentence decide if there is a gapping construction in it.

2. Gap resolution. Predict the position of the elided predicate and the correspondent predicate in the antecedent clause.

3. Full annotation. In the clause with the gap predict the linear position of the elided predicate and annotate its remnants. In the antecedent, clause finds the constituents that correspond to the remnants and the predicate that corresponds the gap.

For more details about data and task see description on github[1].

3 System description

We propose modeling the task as both sequence labeling and classification jointly with a neural architecture.[2] The system's architecture is shown in Figure 1 and consists of seven parts:

1. BERT Embedder;

2. Weighted aggregation of BERT output;

[1]Full AGRR-2019 description available at https://github.com/dialogue-evaluation/AGRR-2019.
[2]Code is available at https://github.com/king-menin/AGRR-2019.

3. Recurrent BiLSTM layer;
4. Multi-Head Attention;
5. linear layer;
6. NCRF++ inference layer for 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.

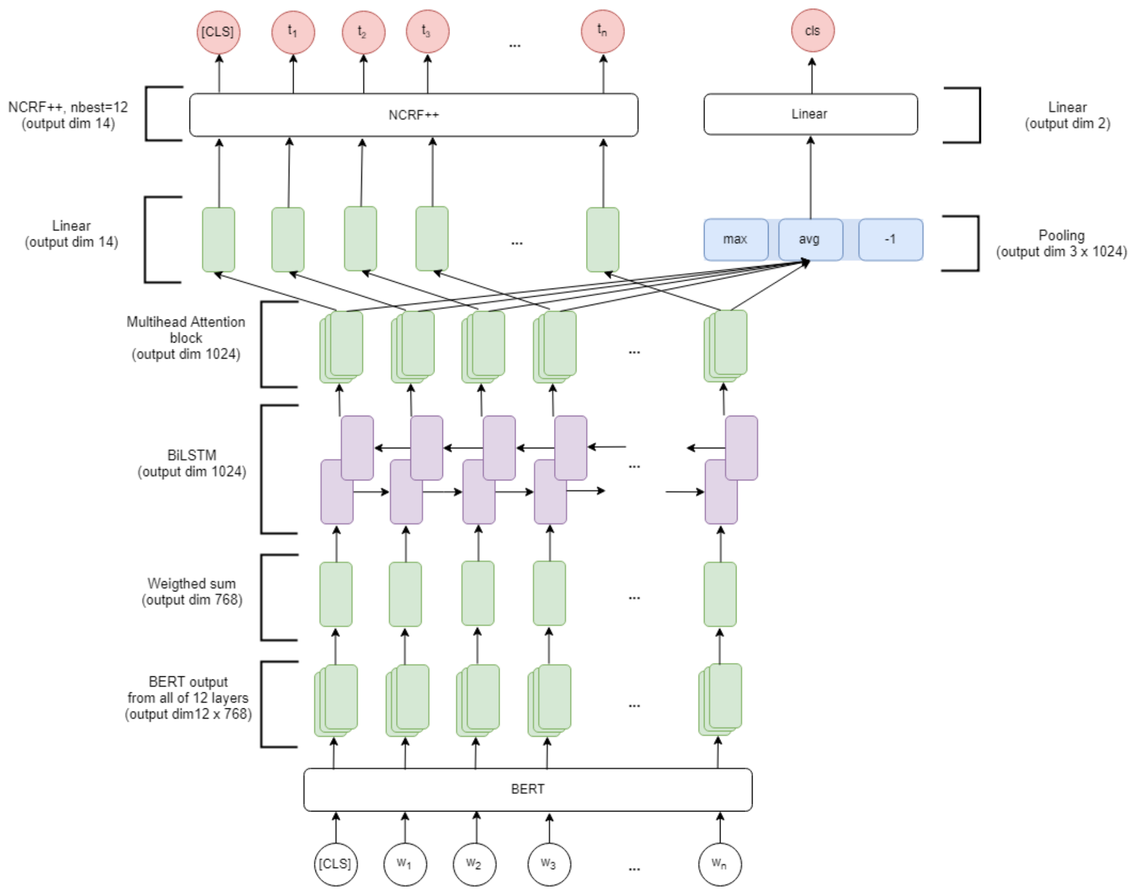
[CLS]
t_1
t_2
t_3
...
t_n
cls
NCRF++, nbest=12 (output dim 14)
NCRF++
Linear
Linear (output dim 2)
Linear (output dim 14)
max
avg
-1
Pooling (output dim 3 x 1024)
Multihead Attention block (output dim 1024)
BiLSTM (output dim 1024)
Weigthed sum (output dim 768)
BERT output from all of 12 layers (output dim12 x 768)
BERT
[CLS]
w_1
w_2
w_3
w_n

Figure 1: The system architecture.

3.1 Neural network architecture

3.1.1 BERT Embedder

The BERT embeddings layer contains Google's original implementation of BERT language model. Each sentence is preprocessed as described in BERT paper [6]:

1. Process input text sequence to WordPiece embeddings [10] 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.

Entities labels converted to format IOBX format. For example:

Jim Hen ##son was a puppet ##eer

B-PER I-PER X O O O X

But instead of BERT's original paper [6] 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.

Because we do not fine-tune BERT, we should adapt it outputs for our specific sequence labeling task. The suggested weighting approach is similar to ELMo [11], but with a lower number of weighting vectors parameters s_i.

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 long time dependencies in an input sequence [7].

3.1.4 Multi-Head Attention

After applying the recurrent layer, we should learn any other dependencies in a sequence for each token. We can achieve this result with Self-Attention (Figure 2). That can be

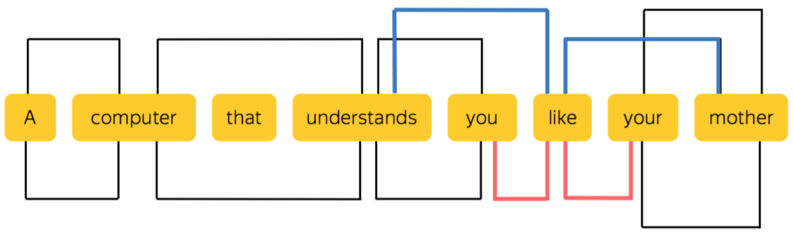

Figure 2: David Talbot's example of Self-Attention.

formulated as $D(d_h|S)$, where D is some hidden dependency; d_h is the h head of attention, and S is all sequence. In our architecture, we use Multihead-Attention block as proposed in the paper "Attention is all you need" [8]. We took 3 heads and value and key dim 64.

3.1.5 Inference for sequence task

After the input sequence was encoded, we gave 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 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 [9].

3.1.6 Inference for classification task

For the classification inference, we use Pooling Linear Classifier block as proposed in ULMFiT paper [12]. We pass output sequence representation H from Multihead-Attention part to different Poolings and concat (as shown in Figure 1):

$$h_c = [h_T, maxpool(H), meanpool(H)] \tag{2}$$

where $[]$ is concatenation;

h_T is last output significant vector of Multihead-Attention part (which does not have "pad" label).

The result of concat Pooling (3×1024) is passed to Linear layer, and that predicts binary classification.

3.2 Postprocessing prediction

After getting labels for the sequence of WordPiece tokens, we should convert prediction to word level labels in sequence labeling task. 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 the final prediction we have two strategies of making full gapping annotation:

1. Before the submission deadline: in this submission we don't use classification result for sequence label prediction, but train joint model and select classification result by the following rule: if any word's predicted labels of all sequence contains any label except "O", "[CLS]" and "pad" mark sample as 1 (has gapping), 0 - otherwise. Sequence labels are not changed.

2. After the submission deadline: first, classification result is taken into account and all words in sequence are marked as "O" , next if binary classification prediction is not 0 (gapping present) and predicted sequence labeling are returned..

For the Full annotation task, we should predict “V“ label. We use the following rule because in data all labels “V“ have zero length representation in text: mark word with “V“ label if the word was marked by “R2“. If the word wasn’t marked, mark word with “V“ label if the word was marked by “R1“.

4 Training the system

4.1 Data conversion

Because train, dev and test datasets contain symbolwise markup, but BERT take words sequence as input we convert datasets to word level IOB markup [13]. 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 as described in section 2.2. After this, we transform word predictions to symbolwise markup.

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 [9] for the sequence labeling task and $\mathcal{L}_{clf}$ is Binary Cross Entropy Loss for the classification task.

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 warmup, and linear decay of the learning rate. Also, gradient clipping was applied for weights with $clip = 5.0$.

Training of proposed neural network architecture was performed on one GPU with the batch size equal to 16, the number of epochs equal to 100, which required only around 5 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 dev set on each epoch.

The final model was trained on train and dev datasets.

5 Results and discussion

5.1 Evaluation results

The evaluation of the training stage was produced on dev dataset. Tabel 1 shows word-level metrics precision, recall, and f1-measure. The evaluation metric of AGGR-2019 [5] competition is symbolwise and measured by organizations evaluation script. For dev set, we obtained the following scores: Binary classification quality (f1-score): 0.958 and Gapping resolution quality (symbol-wise f-measure): 0.958. This difference in word and symbolwise is because words prediction isn’t used classification results.

The evaluation of test dataset presented in Tabel 2. In this submission, we do not use classification result. After deadline submission takes into account classification result and marks all words in sequence as “O“ than Binary Classification prediction is 0 (no gapping) and select predicted word labels otherwise. This difference in evaluation metrics means that single neural network architecture (for Gapping Resolution only) is overfitted on “O“ label.

Table 1: Dev dataset evaluation metrics.

label	precision	recall	f1-score	support
B_O	0.98	0.98	0.98	74268
B_{R1}	0.95	0.92	0.94	1500
I_{R1}	0.93	0.92	0.93	1769
B_{R2}	0.94	0.95	0.94	1473
I_{R2}	0.94	0.91	0.93	3255
B_{cR1}	0.86	0.85	0.86	1382
I_{cR1}	0.86	0.86	0.86	2022
B_{cR2}	0.89	0.90	0.89	1355
I_{cR2}	0.87	0.92	0.89	2395
B_{cV}	0.96	0.94	0.95	1382
I_{cV}	0.00	0.00	0.00	1
avg / total	0.832	0.831	0.833	90802

Table 2: Test dataset evaluation metrics.

Team	binary			gap resolution	full
	precision	recall	f-measure	f-measure	f-measure
$fit_predict$	0.969	0.95	0.959	0.900	0.892
EXO (**ours**) after deadline	0.946	0.946	0.946	0.859	0.836
EXO (**ours**) before deadline	0.899	0.964	0.931	0.815	0.786
Koziev Ilya	0.774	0.903	0.834	0.677	0.647
Derise	0.801	0.906	0.850	0.665	0.622
Meanotek	0.891	0.781	0.832	0.635	0.514
МГУ-DeepPavlov	0.934	0.644	0.762	0.600	0.588
Vlad	0.778	0.915	0.841	0.574	
MorphoBabushka	0.763	0.619	0.683	0.466	0.440
nsu-ai	0.485	0.123	0.196	0.037	0.036

5.2 Error analysis

First of all, we have some errors with converting from origin data format (symbolwise markup) to word markup and back to origin after prediction. For example with extra spaces, bad Unicode symbols and there are some symbols, which are absent in WordPiece vocabulary.

Another error connected with neural network prediction mistakes. Even though we use classification results in after deadline submission network was overfitted on label "O" and there are many false positives in prediction as shown in Section 4.

The last kind of errors connected with the bad learned structure of gapping.

6 Related work

The related work has several parts: first, the phenomena of gapping has received some attention of the NLP community recently, see [4, 14]. Secondly, our work follows the recent trend of using trained neural languages models, such as [6, 11, 12]. Thirdly we model the task of gapping resolution as a joint sequence labeling and classification task following other joint architectures[15, 16].

7 Conclusion and future work

We have proposed neural network architecture that solves three tasks (described in Section 1) without any additional data. However, it heavily exploits GPU memory cost on train and prediction steps. Our method took second place on AGGR-2019 competition [5]. This neural network architecture can be used for other tasks, that can be reformulated as a sequence labeling task for Russian or any other language (listed in BERT documentation [6]).

As improvements of the system, we can fine-tune BERT embeddings and try to do different layers after BERT or pass other modern language models as an input.

Acknowledgements

We are grateful to Sesame Street for their fruitful inspiration.

References

[1] Elizabeth Coppock. 2001. *Gapping: In deffence of deletion.*

[2] Jorge Hankamer. 1979. *Deletion in coordinate structuresn.*

[3] John Robert Ross. 1970. *Gapping and the order of constituents.* In Manfred Bierwisch and Karl Erich Heidolph, editors, Progress in Linguistics. De Gruyter, The Hague.

[4] Sebastian Schuster, Matthew Lamm, Christopher D. Manning. 2017. *Gapping Constructions in Universal Dependencies v2.* Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017), pages 123–132,Gothenburg, Sweden.

[5] Ponomareva M. Smurov I., Shavrina T. O., Droganova K., Bogdanov A. *AGRR: Automatic Gapping Resolution for Russian.*
`https://github.com/dialogue-evaluation/AGRR-2019`

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. 2018. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* Google AI Language.

[7] Hochreiter, S. and Schmidhuber, J. 1997. *LSTM can solve hard long time lag problems.* In Advances in Neural Information Processing Systems 9. MIT Press, Cambridge MA. Presented at NIPS 96.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin. 2017. *Attention Is All You Need.* 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

[9] Jie Yang, Shuailong Liang, Yue Zhang. 2018. *Design Challenges and Misconceptions in Neural Sequence Labeling.* Proceedings of COLING 2018, Santa Fe, New Mexico, USA.

[10] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc VLe, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. *Google's neural machine translation system: Bridging the gap between human and machine translation.* arXiv preprint arXiv:1609.08144.

[11] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, Luke Zettlemoyer. 2017. *Deep contextualized word representations.* arXiv:1802.05365.

[12] Jeremy Howard, Sebastian Ruder. 2018. *Universal Language Model Fine-tuning for Text Classification.* arXiv:1801.06146v5.

[13] Lance A. Ramshaw, Mitchell P. Marcus. 1995. *Text Chunking using Transformation-Based Learning.* arXiv:cmp-lg/9505040v1.

[14] Sang-Hee Park. 2016. *Towards a QUD-Based Analysis of Gapping Constructions.* Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers.

[15] Bing Liu, Lane Ian. 2016. *Attention-based recurrent neural network models for joint intent detection and slot filling.* arXiv:1609.01454.

[16] Thien Huu Nguyen, Kyunghyun Cho, Ralph Grishman. 2016. *Joint event extraction via recurrent neural networks.* Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.