Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2019”

Moscow, May 29—June 1, 2019

BERT FINETUNING AND GRAPH MODELING FOR GAPPING RESOLUTION

Belkin I. (ilya.belkin-trade@yandex.ru)
Moscow Institute of Physics and Technology, Moscow, Russia

This paper reports our participation in the Automatic Gapping Resolution for Russian shared task (AGRR-2019) within Dialogue Evaluation 2019. Our team took the first place among other nine teams in all subtasks which includes gapping presence-absence classification, gap resolution and full annotation. The phenomenon of gapping is well theoretically studied. However, the problem of automatic gapping resolution is new and there is no baseline for it. We found it possible to bring this task into sentence classification and token tagging problems and solve them using recent advances in Natural Language Processing and deep learning. Training large language models with millions of parameters on small data became possible with the development of transfer learning methods. Using pretrained models for computer vision problems is straightforward and since BERT language model was realized it became possible to benefit from transfer learning in NLP. Our solution is heavily based on BERT, but we found that parsing gapping constructions, which are very structured, benefit from special postprocessing which includes modeling a gapping in the form of a directed graph. Our solution may be considered as the first public baseline for the task of automatic gapping resolution which is based on NLP modern practices.

Key words: gapping resolution, language model, transfer learning, ensembling, contextual embeddings

РАЗРЕШЕНИЕ ГЕППИНГА С ПОМОЩЬЮ ДООБУЧЕННОЙ ЯЗЫКОВОЙ МОДЕЛИ BERT И ГРАФОВОЙ МОДЕЛИ ЯЗЫКОВОГО ЯВЛЕНИЯ

Белкин И. (ilya.belkin-trade@yandex.ru)
Московский физико-технический институт, Москва, Россия

1. Introduction

Automatic analysis of natural language is complicated by many factors one of which is the presence of rare sentence-level constructions. Omission from a clause of some words may not affect the meaning of the sentence for humans but may puzzle automatic system. This construction is known in linguistics as ellipsis and may take different forms. One of them is gapping, that occurs in coordinate structures and elides a repeated predicate, typically from the second clause, with its participants remaining expressed. Gapping has been largely studied from theoretic point of view in [3], [6], [7], [8]. To recover the full meaning of the sentence one has to find the position of the elided predicate and the head of the corresponding predicate. Here an example of this construction in Russian:

(1) *Дайте мне две пятерки, а я вам [дам] десятку.*

The word in brackets is omitted in original sentence. Its absence does not affect the meaning of the sentence for human reader. That means that inner representation of the sentence in some system of automatic analysis of natural language with and without recovered word should be similar. To deal with, such constructions should be detected and parsed appropriately. This led to the problem of automatic gapping resolution.

Automatic gapping resolution is a new problem not only for Russian but for other languages too. There are no open source tools or published experiments on this task. One of the reasons is the lack of data. Gapping is a very rare phenomenon and collecting a large enough corpus is a complicated task.

Automatic Gapping Resolution for Russian shared task (AGRR-2019) within Dialogue Evaluation 2019 is a pilot event for gapping resolution task for Russian held for the first time. The data provided to the participants of the Shared task was not published anywhere before. We were free in problem interpretation and method selection. Our team decided not to dive into theoretical studying of gapping as it may take a long time, but to bring the problem to a known task in Natural Language Processing and apply modern methods to solve it. Our approach has shown the best results among other participants and here we give an overview of our system.

The paper is organized as follows. First, we describe the shared task setup. Second, we present our method. Third, we report on the quality of our system and results achieved.

2. Shared task overview

AGRR-2019 is the first public benchmark for algorithms of automatic gapping resolution for Russian. Usually, when reporting method performance on some dataset, there is no need to describe what data and markup we have, what evaluation procedure we follow. These are standardized for particular benchmark and usually well-known. But it is not our case. The dataset and even exact problem standing are new. In this section we give an overview of key points of the shared task which are essential for further reading and understanding of our approach. First, we describe three tasks, which compose the problem. Than we give a description of the data and its markup. And lastly, we cover the evaluation procedure for each of tasks. For more information refer to [13].

2.1. Problem description

The meaningful statement of the problem was discussed in the introduction, and in this section, we give a more formal statement. The general problem of gapping resolution was represented to participants as three tasks, each of which was evaluated separately.

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

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

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

Participants were free in method selection for each of tasks.

2.2. Data description

Participants were provided with three datasets: Train, Dev, Add. The first two were labeled manually and include 16,406 and 4,142 sentences respectively. Add dataset consists of 115,536 samples. It was obtained automatically by ABBYY Compreno [11] and, as were mentioned by organizers, may contain some mistakes. Test data was realized by the end of the AGRR-2019 and includes 20,951 samples. Each sentence with the gapping was marked and annotated with two remnants R1 and R2, their correlates in the antecedent clause cR1 and cR2, the position of the elided predicate V and the head of the correspondent predicate cV.

(2) *Тогда я **cV[** принял **cV]** **cR1[** ее **cR1]** **cR2[** за итальянку **cR2]**, а **R1[** его **R1]** **V[]** **cR2[** за шведа **cR2]**.*

Gap resolution task includes prediction of positions of tokens labeled as cV and V; participation in full annotation task requires to recover the whole markup for the sentence. Not all sentences contain both remnants, some contain only the first.

(3) *Но Христианин сказал Упрямому: — Нет, сосед, лучше ты последуй примеру Сговорчивого. Мы в самом деле там **cV[** получим **cV]** **cR1[** то, о чем я сейчас говорил **cR1]** , а сверх того — **V[]** **R1[** великую славу **R1]** .*

Others may contain more than one clause with the gap.

(4) *Индекс **cR1[** промышленного производства **cR2]** за январь-февраль 2008 г. **cV[** составил **cV]** **cR2[** 106,0% **cR2]** , **R1[** инвестиций в основной капитал **R1]** — **V[]** **R2[** 120,2% **R2]** и **R1[** оборота розничной торговли **R1]** — **V[]** **R2[** 116,3% **R2]**.*

Also, it is worth noting that the position of elided predicate is always the beginning of the second remnant R2, or, in sentences with only one remnant—the beginning of the first one R1. Sentences without gapping are taken into account only in binary presence-absence classification task.

2.3. Evaluation and metrics

Evaluation of participants' systems was performed on the Test dataset which was released two days before the deadline and did not contain markup. The main metric for binary classification task was standard f-measure. Gapping element annotations was measured by symbol-wise f-measure. E.g. if the gold standard offset for certain gapping element is 10:15 and the prediction is 8:14, we have 4 true positive chars, 1 false negative char and 2 false positive chars and the resulting f-measure equals 0.727.

3. Our approach

We consider the problem of gapping resolution as sequence labeling and sentence classification problem. Our solution is based on BERT and we follow model design and finetuning procedure as described in [5]. We use special post-processing procedure to obtain predictions which satisfy strict conditions observed from real data. And we use ensembling to clean automatically collected sentences.

3.1. Input Representation

Each sentence was converted to appropriate form for BERT model. This process includes WordPiece tokenization, adding special token [CLS] for sentence level classification as the first token and [SEP] as the last one, padding with [PAD] token to fixed length (128 in our experiments) and converting into indices of model vocabulary.

Also, in early experiments, we found that some additional preprocessing steps improve validation quality and accelerate convergence. First, we replace comma with point in decimals. And second, we replace dash with hyphen since the first one is not presented in the vocabulary.

3.2. Model

We follow model design proposals from the original paper [5]. By incorporating BERT with one additional linear layer model may be adapted for various tasks which include single sentence classification and single sentence tagging tasks. However, instead of training separate models for gapping absence-presence classification and full annotation tasks we combine these two models as shown in the **Fig. 1**. Distributed representation for the first token is used for binary sentence classification. Other tokens are classified into five classes. We do not predict the position of the elided predicate because it is always the beginning of R2 or R1. For more information on this see Data description section.

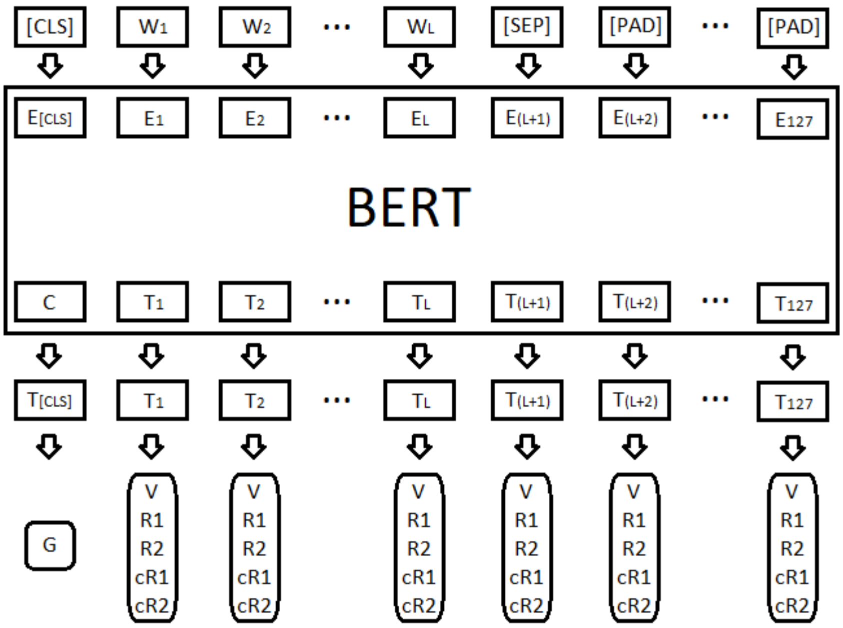

Fig. 1. Model architecture for gapping resolution problem

3.3. Training

Instead of training from scratch, we use pretrained weights [12] for BERT layers and finetune the model in three stages. First, we freeze BERT part and train only top linear layers for two cycles [9, 10]. In one cycle we linearly adjust learning rate up to 0.001 for the first 20% of cycle and then linearly decrease to the initial value of 1e-5. In the second stage, we train full model for two cycles. One cycle lasts two epochs and includes linear learning rate warmup for the first 10% of the time to 2e-5 and linear decreasing to the initial value. In the last stage, we train our model for two cycles with cycle size of three epochs and a linear warmup to 1e-6 for the first 30% of the cycle.

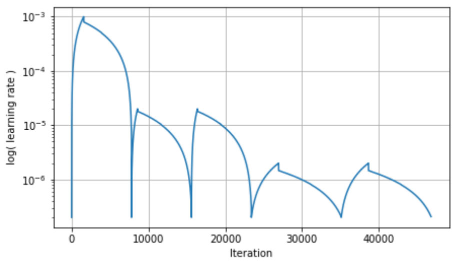

Fig. 2. Learning rate schedule of final model

Training procedure of models in ensemble is quite similar. First, we train only top layers for one cycle. Cycle size equals to two epochs, learning rate linearly increases to 0.001 during the first 20% of cycle. Finetuning stage includes three similar cycles with cycle size of two epochs and linear warmup of learning rate up to 2e-5 for the first 10% of cycle.

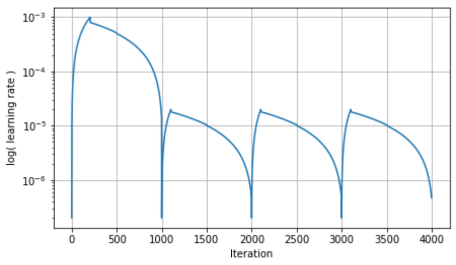

Fig. 3. Learning rate schedule of models in ensemble

3.4. Post-processing

To obtain resulting predictions we perform some additional steps on the raw outputs of the model. One possible way is to take argmax over output distributions for each token. However, this approach does not address some peculiar properties of the problem and lead to prediction errors which may be easily avoided.

By analyzing sentences with gapping, we found two such properties. First, possible tags at each position depend on all previous tags. Second, given a sentence the number of segments labeled as R1 and R2 are equal. To visualize possible sequences of tags we build a directed graph where each node corresponds to one or more sequential tokens of input. Nodes “_start” and “_end” correspond to [CLS] and [SEP] tokens respectively. Edges correspond to tokens with no label.

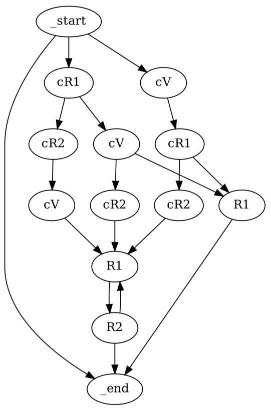

Fig. 4. Graph representation of gapping construction

Given a probability distribution over tags for each position of input we may compute a likelihood of any path in this graph. But we interested in the max-likelihood path. Fortunately, this may be easily found by Dijkstra algorithm [2].

3.5. Data cleaning

Since deep neural networks perform better with more data it is straightforward to concatenate Train and Add datasets and use all available data for training. But wrong labeled samples may harm convergence and lead to worse results and generalization. To deal with we use ensembling to filter the large dataset. In detail, we concatenate train and dev sets, shuffle them and split in five folds. Then we five times select one for validation and the rest for training. Since we use cycling learning rate scheduling it is straightforward to use checkpoint ensembling [4] to diversify ensemble by using weights of the network at the end of each cycle. For each train/validation split, we save three checkpoints and thus obtain 15 trained models. Then we filter the large dataset as follows. Given a sentence we calculate predictions of 15 networks and include this sentence to resulting training set if at least one condition holds: prediction of at least one model matches with its markup; predictions of at least 8 networks are the same, in this case, we omit markup provided and use ensemble answer as a ground truth. Following this procedure does not increase dataset diversity as networks are already able to give the right answer. However, even 'similar' samples may be good for training as they provide regularization and may increase generalization. Total amount of data used for training our final model was 126,202 sentences. We refer to this enlarged dataset as "Train + Add".

4. Results

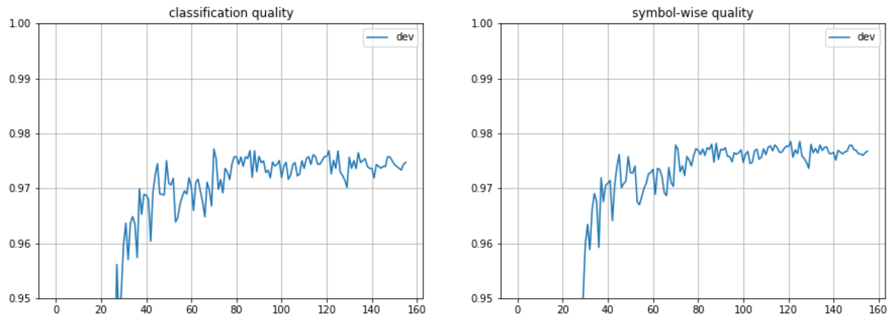

Fig. 5. Final validation metrics

In this section we report quality achieved on Train/Dev/Test datasets as well as give some analysis of model errors. Our result was the best among competitors and brought our team the first prize. The best quality on validation was achieved on 71 iteration and weights saved at the end of this iteration were used to obtain test predictions. Validation quality plot does not show signs of overfitting such as increasing of model error, but there is a large gap between train and validation/test quality.

Table 1. Results of final model on different subsets

	Binary classification quality	Gapping resolution quality
Train + Add	0.9988	0.9895
Train	0.9973	0.9784
Dev	0.9778	0.9313
Test	0.9590	0.8901

We have looked at some mistakes of our final model and observe that some of them are really ambiguous and difficult to avoid. However, it also makes some silly mistakes. We argue this is because of some overfitting. We give some examples of false negatives and false positives in Russian language and provide glosses(word-by-word translation) and translation into English to make sentences clear for non-Russian-speaking reader.

False negatives:

(5) *Russian: В жестах — нет песен, в музыке — ритма.*
Gloss: In gestures — no songs, in music rhythm.
English: There are no songs in gestures, in music — rhythm.

The word "нет" in Russian may be a particle or a verb depending on the context. In this sentence this is a verb and a part of gapping construction.

(6) *За 15 лет добросовестной службы в МЧС всё стало родным и все родными.*
Gloss: For 15 years of conscientious service in the Ministry of Emergency Situations, everything became relatives and all relatives.
English: We all became a family and everything became native for 15 years of conscientious service in the Ministry of Emergency Situations.

This example can illustrate model overfitting. In most sentences gapping construction is highlighted by punctuation, but not in this case

False positives:

(7) *Но к тому времени Джек был влюблен в Синди Пейдж, сейчас — миссис Джек Свайтек.*
Gloss: But by that time Jack was in love with Cindy page, now — Mrs. Jack Switek.
English: Jack was in love with Cindy Page, now Mrs. Jack Switek, by that time.

In this sentence "*Синди Пейдж*" and "*миссис Джек Свайтек*" are two names of the same person before and after wedding.

(8) *Раздался вой, а потом удаляющийся топот.*
Gloss: Rung howl, and then receding tramp.
English: There was a howl, and then the receding tramp.

This sentence is an example of markup mistake. It contains gapping but it is not labeled and thus considered as an algorithm error.

5. Conclusion

In this paper we present our winning solution of the shared task on automatic gapping resolution for Russian within Dialogue Evaluation 2019. This is the first time when the data for this problem was published and there are no public results for this task. To set up the first one we follow recent practices in natural language processing and training neural models in general. Our solution does not lean on gapping research, instead we follow data-driven approach. We think that combining these two approaches will lead to significant progress in gapping resolution.

References

1. *Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin* (2017) Attention Is All You Need, available at https://arxiv.org/abs/1706.03762.
2. *Dijkstra E. W.* (1959) A note on two problems in connexion with graphs, Numer. Math, Vol. 1, Iss. 1, P. 269–271.
3. *Elizabeth Coppock* (2001) In deffence of deletion, available at: https://www.linguistics.northwestern.edu/documents/award-winners/linguistics-undergraduate-award-past-winner-coppock.pdf.
4. *Hugh Chen, Scott Lundberg, Su-In Lee* (2017) Checkpoint Ensembles: Ensemble Methods from a Single Training Process, available at https://arxiv.org/abs/1710.03282.
5. *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova* (2018) BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, available at https://arxiv.org/abs/1810.04805.
6. *John Robert Ross* (1970) Gapping and the order of constituents, Progress in linguistics: A collection of papers, 43:249–259.
7. *Jorge Hankamer* (1979) Deletion in coordinate structures, Garland Publishing, Inc., New York & London.
8. *Kyle Johnson* (2014) Gapping, available at: http://people.umass.edu/kbj/homepage/Content/gapping.pdf.
9. *Leslie N. Smith* (2017) Cyclical Learning Rates for Training Neural Networks, available at https://arxiv.org/abs/1506.01186.
10. *Leslie N. Smith* (2018) A disciplined approach to neural network hyper-parameters: Part 1 — learning rate, batch size, momentum, and weight decay, available at https://arxiv.org/abs/1803.09820.
11. https://www.abbyy.com/ru-ru/science/technologies/compreno/.
12. https://github.com/huggingface/pytorch-pretrained-BERT.
13. https://github.com/dialogue-evaluation/AGRR-2019.