Deep Convolutional Networks for Supervised Morpheme Segmentation of Russian Language

Alexey Sorokin[1,2], Anastasia Kravtsova[1]

[1] Moscow State University, Faculty of Mechanics and Mathematics
[2] Moscow Institute of Physics and Technology

Abstract. The present paper addresses the task of morphological segmentation for Russian language. We show that deep convolutional neural networks solve this problem with F1-score of 97% over morpheme boundaries and beat existing non-neural approaches.

Many successful approaches of modern NLP treat words as mere sequences of symbols, not as atomic units, for example, the FastText model constructs word embedding from the embeddings of its ngrams. However, not all symbol ngrams are of the same utility, the most important ones often correspond to morphemes or pseudomorphs: the root is essential in capturing word semantics, while affixes reflect morphological and syntactic relations. It brings the problem of automatic morphological segmentation to the fore of computational linguistics. Recently, morpheme segmentation was used as a part of machine translation system in [9] and in [1] for constructing word embeddings.

In earlier years of NLP this task was usually solved using no or minimal supervision. Researchers tried to utilize letter variety statistics [3], or modeled the sequence of morphological segments using either HMM [2] or adaptor grammars [6]. Being linguistically motivated, this approach obviously suffers from its unsupervised nature and predetermined constraints imposed by the probabilistic model. As any segmentation task, morpheme segmentation can be transformed to a sequence tagging problem using BMES-scheme. However, for most languages the amount of supervised data is too low for such treatment. Fortunately, Russian does not have this problem since the morphological dictionary of A. N. Tikhonov [8] which includes more than 90000 lexemes is freely available .

As most sequence tagging tasks, morphological segmentation was successfully addressed using conditional random fields [4]. For other sequence labelling tasks, such as NER or morphological tagging, neural network approaches tend to outperform CRFs. Therefore our choice is to apply neural networks to morpheme segmentation. Since morphological segmentation is essentially local — the boundary position depends mostly on the immediate context — we apply convolutional neural networks (CNNs) due to their excellent ability to capture local phenomena. Neural networks were applied to morpheme segmentation [5], [7], however, we do not know any works testing CNNs for this problem.

Our contribution is threefold: we release a cleared version of A. N. Tikhonov morphological dictionary[1], we test a multilayer convolutional network as a baseline approach for this task and show its effectiveness; also we demonstrate that additional memorizing of morphemes slightly improves performance.

1 Model architecture

We treat morpheme segmentation as a sequence labeling task. Segments are encoded using BMES-scheme, where B stands for Begin, M — for Middle and E and S for End and Single respectively. As in named entity recognition, we also encode types of the morphemes to be tagged, for example, the encoding of the word *учитель* (*teacher*) and its segmentation *уч/ROOT/и/SUFF/тель/SUFF/* is

у	ч	и	т	е	л	ь
B-ROOT	E-ROOT	S-SUFF	B-SUFF	M-SUFF	M-SUFF	E-SUFF

Our network starts from 0/1 encodings of the input letters. These vectors are passed through several convolutional layers. For each window width we have its own filters to deal with symbol ngrams of different length. The outputs of all convolutions are concatenated and passed through a (possibly) multilayer perceptron. The final layer of the perceptron is followed by softmax which outputs probability distribution over possible labels in each position of the word.

Describing the model formally, for a word $w = w_1 \dots w_n$ and one-hot encoding of its letters $e_1 \dots e_n$, we have

$$\begin{aligned}
(z_1^1)_i &= \mathrm{CONV}(e_{i-d_1/2}, \dots, e_i, \dots, e_{i+d_1/2}), i = 1, \dots, n \\
&\dots \\
(z_r^1)_i &= \mathrm{CONV}(e_{i-d_r/2}, \dots, e_i, \dots, e_{i+d_r/2}), \\
&\dots \\
(z_j^k)_i &= \mathrm{CONV}((z_j^{k-1})_{i-d_1/2}, \dots, (z_j^{k-1})_i, \dots, (z_j^{k-1})_{i+d_1/2}),
\end{aligned}$$

Here $k = 2, \dots, K$ is the number of layer, $d_1, \dots, d_r$ are different window widths and i is position in the sequence. For each i we concatenate all convolutions as $z_i = [(z_1^K)_i, \dots, (z_r^K)_i]$. z_i encodes the context around i-th position and is further passed through a two layer perceptron that outputs a probability distribution p_i over all possible classes:

$$\begin{aligned}
h'_i &= \max(W' z_i + b', 0), \\
h_i &= W h_i + b \\
p_{ij} &= \frac{e^{h_{ij}}}{\sum\limits_r e^{h_{ir}}}
\end{aligned}$$

Weights of all the layers are optimized during model training.

[1] all code and data is put on Github, the link is omitted due to anonymity reasons

2 Experiments

2.1 Data

We used the electronic version of Tikhonov dictionary available in the Web. Since downstream tasks often require morpheme types (affixes for morphological tasks and roots for semantic ones), we labeled the morphemes using the data from `slovolit.ru`. This data was manually postprocessed and cleared to avoid errors and inconsistencies.

We used 7 types of morphemes: prefix, root, suffix, ending, postfix (*-ся* in *целоваться*), link (*средн-е-русский*) and hyphen (англо-русский). Data was partitioned in 3/1 proportion and the same partition was held for all the experiments. Training part contained 72036 words and the test part includes 24012 ones. All the data is available on our Github page.

2.2 Model implementation

We implement our model using Keras library with Tensorflow backend. 20% of the training data was left for validation. We stopped learning when accuracy on this development set did not improve for 10 epochs and trained the model for maximum of 75 epochs. The size of batch was 32. We used standard Adam optimizer with default parameters except for gradient clipping whose threshold was set to 5.0. In addition to the architecture described in the previous section we used ReLU activations on all the layers and inserted a dropout layer with dropout rate 0.2 between consecutive convolutional layers.

2.3 Experiments

We tested different parameters of our neural network: the number of convolutional layers $(1, 2$ or $3)$ and the distribution of filters on each layer. In preliminary experiments we selected optimal total number of filters of 192 and further tried different combinations of window sizes. We evaluated 4 combinations: 64 filters of width $3, 5, 7$, 96 filters of width 5 and 7 and 192 filters either of width 5 or 7. For 3 layers we tested only two best combinations from previous tests. We report 5 evaluation measures: the usual precision, recall and F1-measure over morpheme boundaries, accuracy of BMES labels and percentage of correctly segmented words. To be ranked as true, not only the position of the morpheme, but also its type should coincide with the correct one. In contrast to [4], we take the final word boundary into account since we may fail to predict it properly by choosing an incorrect morpheme type.

Results of evaluation are shown in 1, where the leftmost column stands for the number of layers and the next one for the combination of filters. For a single layer width 7 is optimal since 5 symbols are too little to capture long roots. With 2 and 3 layers width 5 is better since two layers of width 5 collect information from 9 consecutive symbols. Using 2 layers instead of 1 leads to error reduction over 30% for all the metrics, the improvement between 2 and 3 layers is much

Table 1: Quality of morpheme segmentation.

# layers	Filter combination	Precision	Recall	F1-score	Accuracy	Word accuracy
1	64-64-64	96.14	95.63	95.88	92.74	76.21
	96-96	96.31	*95.90*	96.10	93.12	77.47
	192(5)	96.17	95.38	95.77	92.59	75.82
	192(7)	*96.49*	95.73	*96.11*	*93.17*	*77.62*
2	64-64-64	97.12	97.2	97.16	94.93	83.13
	96-96	97.18	97.41	97.29	95.16	83.80
	192(5)	*97.18*	*97.59*	*97.38*	*95.34*	*84.48*
	192(7)	96.79	97.40	97.09	94.85	82.69
3	96-96	97.47	97.70	97.58	95.76	85.67
	192(5)	**97.67**	**97.82**	**97.74**	**95.99**	**86.42**

less but also clear. We tested a bidirectional LSTM instead of final convolutional layer but it deteriorated performance. That shows that morphological segmentation is essentially a local task and does not require memorizing long-distance dependencies.

We suggest that convolutional filters have enough power to learn morphs, but also experimented with memorizing morphemes directly. The context of each position is encoded with a 15-dimensional vector. This vector contains three boolean values for each principal morpheme type (prefix, root, suffix, ending and postfix): whether there is a morpheme ngram that begins in this position, ends in it or whether current letter can be a single-letter morpheme. Additionally, we checked that a sequence of morpheme types is well-formed, verifying that:

1. A prefix starts in the beginning of the word.
2. A postfix occurs in the end of the word.
3. An ending occurs in the end of the word or preceeds a postfix.
4. A suffix occurs in the end of the word or immediately before a postfix, an ending or another suffix. At most three suffixes can occur consecutively.
5. A root occurs either after a prefix (possibly empty), or before a sequence SUFF*END*POSTFIX*, which ends on the right edge of the word.

For better understanding of memorization features we give a detailed example of their calculation. Consider a position after prefix *учит* of the word *учитель* '*teacher*'. The first five coordinates stand for morphemes that start in this position. Such morpheme cannot be a prefix or a root (*учит-* is not a valid prefix, as it should be in case a root started there). -*ель* occurs as a suffix, which produces a 1 in the third element of the encoding. Since -*ель* is not a concatenation of (possibly empty) ending and postfix, next two features are 0.

The next five positions encode, whether a particular type of morphemes ends in current position. Since -*чит* occurs as a root and -*ит* as a suffix, they are 01100. The five remaining features enumerate possible morpheme types for one-letter morpheme -*е*-, following the current position. It is present in the training set as root, suffix and ending; however, the latter variant does not satisfy the well-formedness conditions, so the remaining five features equal 01100. Summarizing, the memorization vector is 001000110001100.

We extract all morphemes that occur at least 3 times in the training set. We tested two variants of memorization: the basic described above and the one equipped with ngram counts: if an ngram *чит* was labeled as root 5 times and appears 10 times in corpus, it will get score 0.5 for root features. For prefixes we use only ngrams in the beginning of the word, for endings and postfixes – only in the end. We observed that there is no difference between performance of these two models. Table 2 shows that memorizing is beneficial for a single layer model, while for 3 layers the improvement is only marginal. It proves that deep convolutional architecture has enough power to learn useful symbol ngrams.

Table 2: Effect of memorization on morpheme segmentation for different number of layers.

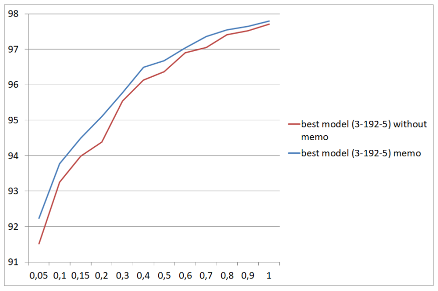
# layers	Filter combination	Precision	Recall	F1-score	Accuracy	Word accuracy
1	192(7)	96.49	95.73	96.11	93.17	77.62
	192(7)+memo	95.96	97.09	96.52	93.91	80.14
2	192(5)	97.18	97.59	97.38	95.34	84.48
	192(5)+memo	97.22	97.80	97.51	95.61	85.29
3	192(5)	97.67	97.82	97.74	95.99	86.42
	192(5)+memo	97.67	97.97	97.82	96.15	87.03

For the top score we ensembled three models with different random initializations and averaged the probabilities they predict. As shown in Table 3, that improved performance additionally; note that averaging has higher effect than memorization.

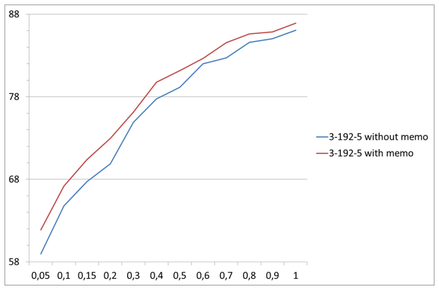
Table 3: Effect of ensembling and memorization on morpheme segmentation for best model configuration.

# layers	Filter combination	Precision	Recall	F1-score	Accuracy	Word accuracy
3	192(5)	97.67	97.82	97.74	95.99	86.42
	192(5)+ensemble	97.99	98.00	98.00	96.45	87.99
	192(5)+memo	97.67	97.97	97.82	96.15	87.03
	192(5)+memo+ensemble	97.86	98.35	98.10	96.64	88.62

Since usually morpheme segmentation is done in low-resource setting, we evaluated our model on different amounts of training data. The learning curves are presented on Figure 1. Already 20% of the training data (about 14000 words) are enough to outperform CRF-based model of [4] (see comparison below).



(a) F1-score over morpheme boundaries



(b) Word accuracy

Fig. 1: Dependence of quality from training data fraction

We compared our model against CRF-based state-of-the-art semi-supervised system of [4]. We used our own evaluation script and report F1-score over morpheme boundaries in two variants:

1. Without evaluating morpheme types and taking final boundary into account, as it is usually reported.
2. With morpheme types for our system and without morpheme types for the model of [4] since it does not use and output them.

We also report word-level accuracy with and without morpheme types.

Table 4: Comparison with [Ruokolainen, 2014]

Metrics	[Ruokolainen]	[Ruokolainen, Harris features]	Our model (+memo)
morph boundary F1 (without last boundary and types)	92.17	92.95	97.16
morph boundary F1 (with last boundary and types)	94.24	94.77	97.9
word accuracy (without types)	65.29	68.19	87.53
word accuracy (with types)	65.29	68.19	87.03

In addition to its lower accuracy, even the basic model from [4] trained for more than 1 hour on a single core Intel Xeon CPU with 512GB RAM[2] and ran out of memory on 6GB RAM laptop, while our model takes about 30 minutes to be trained on 2GB GPU on the same laptop. Actually, a neural network has about 10 times fewer parameters than CRF, which explains its lower requirements.

2.4 Error analysis

Though our system sets a new benchmark for Russian morpheme segmentation, it still has much space to improve. Actually, for more than 10% of words the predicted morpheme structure is incorrect. We present a short analysis of our topmost model (ensemble of 3 models with 3 layers and 192 neurons on each, window width 5) errors and start with the distribution of error types, given in Table 5. Note a pleasant fact, that only in $1,67\%$ of cases (402 of 24012) our model fails for more than 1 morpheme boundary.

Table 5: Distribution of error types

# bound errors	error type	count	percentage
0	correct	21280	88.62
	morpheme type	109	0.45
1	overgeneration	1107	4.61
	undergeneration	893	3.72
	bound position	221	0.92
≥ 2	overgeneration	192	0.81
	undergeneration	99	0.41
	other	111	0.46

We also inspect the dependency of model performance from the number of morphemes. The results (see Table 6) are quite surprising: the lowest quality is observed for 1 or 2 morphemes in the word. The model typically overgenerates, some examples of its errors are given in Table 7.

Table 6: Performance quality depending on the number of morphemes

# Morphemes	1	2	3	4	5	≥ 6
Word accuracy	83.27	78.75	85.24	92.40	93.30	86.33
# in training set	3710	7599	15983	22472	16380	5890

[2] the model equipped with Harris features takes more than 2 hours.

Table 7: Examples of annotation errors.

(a) Segmentation errors caused by homonymy.

Correct	Predicted
недосуг	не-до-суг
облучок	об-луч-ок
помост	по-мост
аполог-ет	апол-о-гет
буханк-а	бух-ан-к-а

(b) Segmentation errors caused by desemantization or annotation errors.

Correct	Predicted
окоп-ник	о-коп-ник
поступательн-ый	поступ-а-тельн-ый
очевидн-ый	очевид-н-ый
оклад	о-клад
ловк-ий	лов-к-ий

3 Conclusions and future work

We have developed a convolutional tagging model for Russian morpheme segmentation task. It clearly outperforms all earlier approaches consuming less computational resources. Further directions are multifold: the first one is to extend the model to operate in semi-supervised fashion with very little data available. Another task to address is reconstruction of deep morpheme structure which requires allomorphy reduction (e.g., Russian *с-* and *со-* or English *-s* and *-es* should be mapped to the same morpheme) as in [5]. The third task is to test whether obtained morpheme segmentations are useful in downstream morphological or semantic tasks such as morphological tagging or checking semantic relatedness.

References

1. Botha J., Blunsom P. Compositional morphology for word representations and language modelling //International Conference on Machine Learning. – 2014. – pp. 1899-1907.
2. Creutz M., Lagus K. Unsupervised morpheme segmentation and morphology induction from text corpora using Morfessor 1.0. – Helsinki : Helsinki University of Technology, 2005.
3. Harris Z. S. Morpheme boundaries within words: Report on a computer test //Papers in Structural and Transformational Linguistics. – Springer, Dordrecht, 1970. – pp. 68-77.
4. Ruokolainen T. et al. Painless semi-supervised morphological segmentation using conditional random fields //Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers. – 2014. – pp. 84-89.
5. Ruzsics T., Samardzic T. Neural Sequence-to-sequence Learning of Internal Word Structure //Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017). – 2017. – pp. 184-194.
6. Sirts K., Goldwater S. Minimally-supervised morphological segmentation using adaptor grammars //Transactions of the Association of Computational Linguistics. – 2013. – T. 1. – pp. 255-266.

7. Shao Y. Cross-lingual Word Segmentation and Morpheme Segmentation as Sequence Labelling //arXiv preprint arXiv:1709.03756. – 2017.
8. Tikhonov A. N. Morphemno-orfograficheskij slovar' // M.: ACT Publishing. - 2002. - 704 c. (in Russian)
9. Vylomova E. et al. Word representation models for morphologically rich languages in neural machine translation //arXiv preprint arXiv:1606.04217. – 2016.