Computational Linguistics and Intellectual Technologies:
Proceedings of the International Conference “Dialogue 2018”

Moscow, May 30—June 2, 2018

# IMPROVING NEURAL MORPHOLOGICAL TAGGING USING LANGUAGE MODELS[1]

**Sorokin A. A.** (alexey.sorokin@list.ru)

Moscow Institute of Physics and Technology, Dolgoprudnyj, Russia
Lomonosov Moscow State University, Moscow, Russia

We offer a new neural architecture for character-level morphological tagging, combining character-level networks with the output of neural language model on morhological tags. Our proposal reduces tagging error up to 10% in comparison with baseline model and achieves state-of-the-art performance both on ru_syntagrus and MorphoRuEval datasets.

**Keywords:** morphological analysis, tagging, neural network, neural language model, character-based models

# АВТОМАТИЧЕСКИЙ МОРФОЛОГИЧЕСКИЙ АНАЛИЗ НА ОСНОВЕ НЕЙРОННЫХ МОДЕЛЕЙ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКОВЫХ МОДЕЛЕЙ

**Сорокин А. А.** (alexey.sorokin@list.ru)

Московский физико-технический институт,
Долгопрудный, Россия
Московский государственный университет
им. М. В. Ломоносова, Москва, Россия

[1] The research was conducted under support of National Technological Initiative Foundation and Sberbank of Russia. Project identifier 0000000007417F630002.

Данная работа посвящена автоматическому морфологическому анализу. Мы показываем, что комбинация символьных нейронных сетей с нейронными языковыми моделями улучшает качество морфологического анализа, снижая количество ошибок на 10%, при этом данный результат достигается без использования дополнительных ресурсов. Результат ещё улучшается в случае дополнительного использования морфологического словаря.

**Ключевые слова:** морфологический анализ, нейронные сети, символьные нейронные сети, нейронная языковая модель

## 1. Introduction

There is no exaggeration in saying that last decade in computational linguistics is the decade of «neural network turn». The works of Tomas Mikolov, e.g. [Mikolov, 2013], on vector representations of words revolutionized not only computational semantics, but entire computational linguistics (further, CL), stimulating the fast growth of embedding-based approach. Another breakthrough insight was the introduction of character-based networks by Santos and Zadrozny [Santos and Zadrozny, 2014], permitting the researchers to solve practically every task from scratch provided enough data is available. After only several years, the vast majority of CL tasks of different complexity, from machine translation to morphological tagging, is solved mostly using different neural network-based architectures.

In the present paper we focus on the task of automatic morphological tagging which takes as input the sequence of words and assigns each word a label (or tag) containing the morphological description of that word. In early years of CL only the part-of-speech information was labeled, therefore this task is sometimes referred as POS-tagging. For analytical languages like English the sets for coarse part-of-speech tagging and fine-grained morphological labeling does not differ much in size and complexity. However, most of the languages are far more complex in its morphology and have a wider inventory of morphological categories, which makes the task of detailed morphological analysis much harder than coarse POS-tagging. Downstream applications benefit more from detailed morphological information (words connected by syntactic dependency often agree in their morphology, which cannot be revealed using only part-of-speech tags), therefore we address this very task and use the term morphological tagging through the paper.

Despite the undoubtable evidence for superior performance of neural network models for the plenty of tasks, their usage for morphological tagging worths further discussion. Indeed, most neural models were tested for English which has unbounded amount of training data and very simple morphology. For more complex morphology the patterns might be the opposite: for example, already the pioneer work [Lafferty et al., 2001] on conditional random fields was compatible with other morphological taggers. On the contrary, only clever design of learning process and output space made them capable to achieve state-of-the-art tagging level [Muller, 2013]. There is no a-priori evidence, that the same neural architectures are suitable

for developed morphological structure of Russian or for small corpora in case of less widespread languages.

However, both the doubts are decisively disproved by recent research. What concerns the second problem, [Heigold et al., 2017] showed that a character-based neural tagger outperforms state-of-the-art CRF parser for a wide range of languages. The effect is rather clear even for training corpora with a thousand training sentences. That implies that LSTMs are more effective in capturing morphosyntactic patterns than CRFs possible due to their capability to learn long-distance dependencies.

The results of MorphoRuEval challenge [Sorokin et al., 2017] demonstrated that a deep neural model of [Anastasiev et al., 2017] defeats by a huge margin the second ranked tagger of [Sorokin and Yankovskaya, 2017], combining a hidden Markov model with linguistically motivated rules for reranking. Both these systems extensively used external knowledge in the form of morphological dictionary, the first one also utilized the output of a closed rule-based semantic parser as feature, while the second heavily relied on feature engineering. Interestingly, other neural models except the winner were clearly behind second place.

Before describing our approach I would like to emphasize the difference between the behavior of neural and Markov tagging models on the example sentence

| | | | | | |
|---|---|---|---|---|---|
| | *его* | *решение* | *задачи* | *было* | *неправильным* |
| (1) | *ego* | *reshenie* | *zadachi* | *bylo* | *nepravil'nym* |
| | his | solution+Sg+Masc | problem+Sg+Gen | be+Past+Sg+Neut | incorrect+Sg+Masc+Ins |

Due to extensive regular homonymy in Russian it has more than 100 variants of tagging as summarized in the table below.

**Table 1.** Regular homonymy in Russian for the sentence *Его решение было неправильным*

| Word | Number of tags | Tags |
|---|---|---|
| его | 5 | PRON, Gender=Masc, Case=Gen<br>PRON, Gender=Masc, Case=Acc<br>PRON, Gender=Neut, Case=Gen<br>PRON, Gender=Neut, Case=Acc<br>**DET** |
| решение | 2 | **NOUN, Gender=Neut, Case=Nom**<br>NOUN, Gender=Neut, Case=Acc |
| задачи | 3 | **NOUN, Number=Sing, Case=Gen**<br>NOUN, Number=Plur, Case=Nom<br>NOUN, Number=Plur, Case=Acc |
| было | 2 | **AUX, Gender=Neut**<br>PART |
| неправильным | 3 | ADJ, Number=Sing, Gender=Masc, Case=Ins<br>**ADJ, Number=Sing, Gender=Neut, Case=Ins**<br>ADJ, Number=Plur, Case=Dat |

When parsing this sentence, an HMM relies on dictionary word-tag statistics and tag trigram frequencies. It decides that было should be an AUX since it is a dominant label of this word. An AUX,Gender = Neut tag is often followed by a neutral adjective in instrumental case and preceded by a NOUN, Gender=Neut, Case=Nom NOUN, Number=Sing, Case=Gen bigram. Finally, a determiner is more probable to occur before a noun, than a personal pronoun. Thus, the correct labeling is uncovered using only the tag cooccurence statistics. However, consider another sentence:

| | | | | | |
|---|---|---|---|---|---|
| | *его* | *решение* | *задачи* | *будет* | *неправильным* |
| (2) | *ego* | *reshenie* | *zadachi* | *budet* | *nepravil'nym* |
| | his | solution+Sg+Masc | problem+Sg+Gen | be+Fut+Sg | incorrect+Sg+Masc+Ins |

Russian verbs do not change by gender in non-past tense, consequently, the adjective *неправильным* has nothing to support the Gender=Neut hypothesis among its two preceding tags. Therefore HMM and CRF model are likely to fail since they do not take remote context into account.

On the contrary, neural models rely on the lexemes themselves, not the tag statistics. The word *было* (not its tag) forces the network to assign Number=Sing, Gender=Neut label to the next adjective *неправильным* and Case=Nom label to the word *решение* in its left vicinity. This latter word supports DET reading for *его* and *case=Gen* reading for *задачи* since genitives nouns often follow *решение*. Actually, this evidence is confirmed by other words ending by *–ние* as well since character-based networks capture graphical similatity. Moreover, a neural model probably captures the dependency between *решение* and *неправильным* making the second example less problematic.

Summarizing, lexical information is more important than grammar constraints that HMMs are trying to capture. The main advantage of neural network with respect to HMMs and CRFs is its ability to compress this information. It is usually "stored" in the states of a bidirectional LSTM, often being the principal layer of the model. To produce the probability distribution of word tags these states are usually passed through a one-layer perceptron with softmax activation. However, the tags predicted for different words do not directly affect each other. Consequently, the network has no direct mechanism to impose grammatical constraints on tag cooccurrences which may potentially limit its performance.

These constraints can in principle be learnt by neural language models on morphological tags. Such models are known to capture long-distance dependencies using memory mechanisms [Tran et al., 2016]. We propose two combinations which combine an underlying BiLSTM model of [Heigold et al., 2017] with a neural language model via the topmost layer of the tagger.

We apply our approach to UD2.0 [Nivre et al.] and MorphoRuEval [Sorokin et al., 2017] datasets for Russian language. Our paper is organized as follows: Section 2 introduces the baseline BiLSTM model, section 3 explains our extension of it, section 4 describes the experimental setup and presents tagging results, section 5 discusses the results obtained and we conclude with directions for future work.

## 2. Baseline model

Morphological tagging is a task of predicting a correct sequence of morphological tags $\boldsymbol{t} = t_1, ..., t_n$, given the words $\boldsymbol{v} = v_1, ..., v_n$. A character-based approach of [Heigold et al., 2017] addresses this problem from scratch and does not require any other resources except for a morphologically annotated corpus. Their model consists of two parts: the first encodes each word $v_i$ (a sequence of characters) as a fixed-width embedding vector $h_i$, while the second transforms the obtained sequence of vectors $h_1, ..., h_n$ to the morphological tags.

First, we describe the encoding component of the model. The paper of Heigold uses two architectures for word representation: the first is a 2-layer LSTM while the second combines several convolutional and highway layers. The first one slightly outperforms the second for most languages, however, we selected the second due to memory requirements. We refer the reader to the original paper for the full description, see also [Kim et al., 2016], applying the same ideas to neural language modelling. Briefly, the architecture is the following:

1. Each character is encoded as a 1-hot row vector with $n_c$ dimensions, where $n_c$ is the number of characters. A word of length $L$ is represented by a sequence of $L$ such vectors $x'_{i_1}, \ldots, x'_{i_L}$, that form a matrix $X$ with $L$ rows and $n_c$ columns and exactly one unit in each row.
2. This matrix is multiplied by a matrix $U$ of size $n_c \times n_e$, producing a sequence $X'=XU$ of $L$ embeddings $x'_{i_1}, \ldots, x'_{i_L}$. $j$-th element of this sequence is a dense representation of $i_j$-th character in the alphabet.
3. $X'$ is passed through $K$ parallel convolutional layers with different widths. After this step $K$ vectors of dimensions $f_1,...,_k$ are associated with each position of the word. Roughly speaking, $k$-th of these vectors contains information of useful ngrams of length $w_k$ around current position.
4. All the vectors from the previous step are concatenated, producing a vector of length $F = \sum_j f_j$ for each symbol of the word. A word is now a matrix wit $L$ rows and $F$ columns.
5. A maximum-over-time (max-pooling) layer is applied to each row, finally encoding the word as a vector $h'$ of fixed dimension $F$.
6. Several highway layers [Srivastava et al., 2015] are applied to this vector. Highway layer performs the transformation $h=s\odot(Vh')+(1-s)\odot h'$, where $V$ is a square matrix with $F$ rows, $g$ is a non-linear function and $\odot$ denotes coordinate-wise product. The highway layer simultaneously produces useful combinations of features using one-layer perceptron ($Vh'$) and keeps relevant dimensions of $h'$. The contribution of both components is balanced by means of vector $s$, which is obtained by another one-layer perceptron with sigmoid activation: $s=(Sh')$.

The second component of the network transforms the obtained sequence of word vectors $h_1, ..., h_n$ into $n$ probability distributions $\pi_1, ..., \pi_n$. Here $\pi_j$ contains tag probabilities for $j$-th word in the sentence. First, two LSTMs are applied, the first processing the sentence from left to right and the second from right to left. The first produces vectors $\vec{y}_1,...,\vec{}_n$ and the second outputs $\overleftarrow{y}_n, ..., \overleftarrow{y}_1$, thus each word is encoded by two vectors $\vec{y}_i, \overleftarrow{y}_i \in \mathbb{R}^{n_y}$. The concatenation of these vectors is multiplied by a projection

matrix $W$ with $n_t$ rows and $2n_y$ columns, $n_t$ being the number of tags. A softmax layer yields the required probability distribution:

$$\begin{aligned} y_i &= [\vec{y}_i, \overleftarrow{y}_i] \quad \text{(concatenation)} \\ z_i &= Wy_i \\ \pi_{ij} &= \frac{e^{z_{ij}}}{\sum e^{z_{ik}}} \end{aligned}$$

In [Heigold et al., 2017] this architecture is proved to be successful for languages of different morphological structure even with only several thousands of tagged sentences available for training.

## 3. Our proposal

### 3.1. Language models

As has already been said, the described architecture does not care about for the probability of the tag sequence "as a whole", it only tries to predict the most probable tag in each position. Hidden Markov models follow the opposite approach: they rewrite the probability of tag sequence given the word sequence as

$$p(t_1 \dots t_n \mid v_1 \dots v_n) \sim p(v_1 \dots v_n \mid t_1 \dots t_n) p(t_1 \dots t_n)$$

and further decompose this probability as

$$\begin{aligned} p(v_1 \dots v_n \mid t_1 \dots t_n) &= p(v_1 \mid t_1) \dots p(v_n \mid t_n) \\ p(t_1 \dots t_n) &= p(t_1) p(t_2 \mid t_1) p(t_3 \mid t_1 t_2) p(t_4 \mid t_2 t_3) \dots p(t_n \mid t_{n-2} t_{n-1}) \end{aligned}$$

assuming mutual independence of lexical probabilities $p(v_i|t_i)$. Morphological tags are supposed to be generated by a trigram model. Restricting lexical probabilities to single word-tag pairs is a drawback of HMMs since the morphological tag depends not only on the word it is assigned to, but also on the whole context (see discussion in the introduction). But proposed Char-LSTM architecture lacks this component at all, which limits its possibilities in an opposite way. Though, n-gram language models used in HMMs require much data for training and cannot access inner structure of morphological tags. However, language models can be based on neural networks as well, not only on n-grams: the probability of current tag $t_i$ given the preceding tags $t_1,\dots,t_{i-1}$ might be obtained as an output of recursive neural network.

Neural language models were successful in modelling sequences of words (see [Tran et al., 2016], [Kim et al., 2016] and multiple references there) in large-scale tasks. We apply them to model sequences of morphological tags. We adapt the model of [Tran et al., 2016] which uses a variant of memory networks [Sukhbaatar et al., 2015] to attend the recent past.

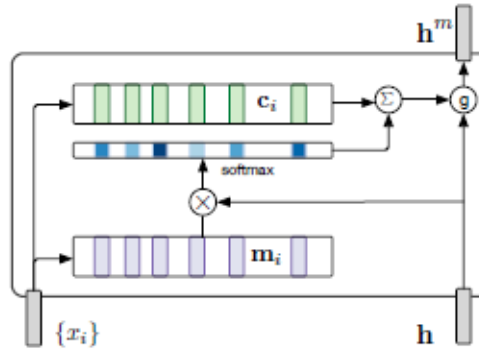


**Figure 1.** Memory block from [Tran et al., 2016]

The goal is to extract information which is the most relevant to predict further tags from the immediate left context of the current one. LSTM itself does this processing the sentence from left to right, but only partially, an additional memory block encoding context can capture more. The context in position $i$ is a matrix $X_i$ with $d$ rows and $m_v$ columns containing $d$ preceeding elements $x_{i-d+1},\ldots,_i$. We multiply this context by two matrices $M$ and $C$ of size $m_v \times m_e$ obtaining two dense representations of the context $M_i=X_iM$ and $C_i=X_iC$. Actually, $j$-th row of $M$ is the "input" embedding of $j$-th element in the vocabulary while $j$-th row of $C$ is its output embedding. $M_i$ is used to define the attention distribution over $d$ preceding elements, which is calculated as:

$$\mathbf{p}_i = [p_{i,1}, \ldots, p_{i,d}] = \text{softmax}((M_i + T)h_i)$$

Informal explanation is the following: softmax favors those rows of $M_i=T$ which are the most similar to $h_i$. $T$ is the bias which forces the model to attend particular positions independent from their content. Since $h_i$ indirectly encodes the information about the past, the selected rows are the most relevant for this past. These rows should contribute the most to the context representation, so we use $p_i$ as weights to produce an output representation of the context:

$$s_i = C_i^T p_i$$

$s_i$ and $h_i$ are concatenated to produce a joint embedding of the context in $i$-th position including both the global information from $h_i$ and the relevant local information from $s_i$. As suggested in [Tran et al., 2016], this encoding is propagated through another LSTM layer:

$$\begin{aligned} h' &= [s_i, h_i], \\ z_i^{LM} &= LSTM(h'_1, \ldots, h'_i), \\ \pi_i^{LM} &= \text{softmax}(W_{LM} z_i^{LM}). \end{aligned}$$

In experiments of [Tran et al., 2016] $x_i$ are just one-hot word encodings. However, morphological tags possess inner structure, therefore we apply encoding scheme summarized in Table 2. Additionally one can add an additional embedding layer before passing feature vectors to the LSTM. NOUN

**Table 2.** Input encoding of morphological tags

| Feature dimension | Value |
|---|---|
| NOUN | 1 |
| VERB | 0 |
| … | 0 |
| NOUN, case=Nom | 1 |
| NOUN, case=Gen | 0 |
| ADJ, case=Nom | 0 |
| | 0 |
| NOUN, gender=Fem | 1 |
| NOUN, gender=Neut | 0 |
| NOUN, gender=Fem | 0 |

Language model allows us to discriminate between probable and improbable combinations of tags, the next step is to apply it to the output of Char-LSTM to filter out inconsistent sequences.

### 3.2. Model combination

Now for each word in the sentence we have two probability distributions that predict its morphological tag. The first is the one of the Char-LSTM model, while the second generates current morphological label given already predicted tags $\pi^L(t_i)=p^{LM}(t_i|t_1 \ldots t_{i-1})$. They should be combined to produce the output distribution over tags. A naive way is to sum their logarithmic probabilities $\log(t_i)=\log \pi^{base}(t_i)+\log \pi^{LM}(t_i)$, assuming the independence of distributions under consideration. Obviously, these two distributions are not independent, therefore we take their weighted combination:

$$\log \pi(t_i) \sim s \log \pi^{base}(t_i) + (1-s) \log \pi^{LM}(t_i)$$

$s$ itself is not a constant: obviously, the reliability of both distributions depends from internal states of corresponding models as well as from the position in the sentence. Informally, for some words the Char-LSTM model is already a good predictor, so the language model weight should not be large. In the beginning of the sentence neural LM is also irrelevant since there is no history it can rely on. On the contrary, when observing rare or homonymous words we should trust the LM more. Summarizing, we choose $s$ to be a vector of weights, not a single weight. It is predicted using a single-layer perceptron with sigmoid activation:

$$\begin{aligned} z_i^w &= [z_i^{base}, z_i^{LM}, pos_i], \\ \mathrm{s}_i &= \sigma(S^w z_i^w + b^w) \end{aligned}$$

Here $pos_i=log(1+i)$ is a scalar encoding current position; $z_i^{base}$ and $z_i^{LM}$ are the states of the topmost LSTMs for Char-LSTM tagger and neural LM, respectively; $s_i$ and $b^w$ are vectors of dimension $n_{tags}$ and $S^w$ is a matrix with $n_{tags}$ rows and $d^{base}+d^{LM}+1$ columns. Here $d^{base}$ and $d^{LM}$ are hidden state dimensions for char-LSTM and neural LM,

respectively. In principle, a multilayer network instead of a single layer could be applied. We refer to this architecture as Char-Weight in the further.

However, the weighting scheme helps only if at least one probability distribution is reliable, it is not capable to correct synchronous errors. Analogously to [Gulcehre, 2015], we use another approach. The output distributions of both the neural LM and the CharLSTM tagger are obtained by projecting their states by means of one-layer perceptron. It implies that all the probability information is already encoded in these states. The idea is to fuse the states of CharLSTM and neural LM into a single state and then process it using a separate network. We choose a two-layer perceptron with ReLU activation as such a network, formally:

$$\begin{aligned} z_i^w &= [z_i^{base}, z_i^{LM}, pos_i], \\ z'^w_i &= \max(S_1 z_i^w + b_1, 0), \\ \pi_i &= \mathrm{softmax}(S_2 z'^w_i + b_2). \end{aligned}$$

We refer to the second model as CharFusion.

## 4. Experiments and Results

### 4.1. Experimental setup

For CharLSTM model we use the setup of [Heigold et al., 2017] with minor modifications. Namely, character encoding dimension is 32, there are 7 convolutional layers applied in parallel with their width ranging from 1 to 7. The number of filters on layer with width $w$ is min(200, 500$w$), so each position of the word is encoded by vector with 1100 elements after passing the convolution. On the word level we use LSTMs with 128 units in each direction. To prevent overfitting we apply dropout to word embeddings and to the outputs of topmost LSTM layer, the dropout probability is 0.2. We use the shallow variant of the architecture, which means only 1 convolutional and highway layers are applied on character level and only one LSTM layer on the word level.

In the neural language model dense tag embeddings have dimension 96 as well as the memory embeddings in the attention layer, the history window to be attended is 5. Output LSTM has 128 hidden units. 0.2 dropout is applied to the outputs of all embedding layers. In the CharFusion model we use 256 units on the hidden layer of the output perceptron.

All models are implemented in Keras library [Chollet et al.] with Tensorflow backend. The models are optimized using Adam optimizer with Nesterov momentum [Dozat, 2016], the learning rate and other optimizer parameters are set to default. The taggers are trained for 75 epochs, language models are trained for 50 epochs, the conventional cross-entropy loss (negative logarithmic probability of correct sequence) is used. When training CharWeight and CharFusion models, we train the basic CharLSTM component of them as well, the weight of the basic model loss is 0.25. We stop

training when the loss on development set have not improve for 10 epochs, saving the model with the best performance on the validation set.

We did not perform exhaustive hyperparameter search. However, preliminary experiments has shown that character embeddings of size 16 as in the original paper lead to worse performance and using recurrent networks with more hidden states or layers slightly deteriorates tagging accuracy. We have also found that regularizing the output probability distribution with L2 loss makes this distribution smoother and prevents overfitting, the regularization coefficient was set to 0.005.

Searching for the optimal tag in position $i$ requires the knowledge of preceding morphological label. In the training phase we feed the model with golden tags, which are not available in test time. Therefore during testing we predict the tags one-by-one from left-to-right and return the sequence with maximal sum of logarithmic tag probabilities. To make the model capable to recover from its errors we apply a beam search with beam width 5. That raises the problem of exposure bias: when training, the model sees only the correct tags as the left context. However, if in the test phase the models fails to predict a correct tag in position $i$, all the predictions in positions $i$+1, $i$+2, ... will be done with incorrect tag history. Neither the tagger, nor the language model, are able to deal with such histories since they were trained only on gold contexts. This problem is called the exposure bias, to alleviate it we replace a 20% fraction of tags in the left context by a vector of all zeros forcing the model to operate correctly even if it lacks complete information about tag history.

### 4.2. Dataset

We evaluate our model on ru_syntagrus subcorpus of Universal Dependencies 2.0 corpus [Nivre et al.], the train subsection was used for training, the development one for validation and the test part for evaluation. We lowercase all the words, in case a word starts with a capital letter or consists of all capitals special pseudoletters <FIRST_UPPER> or <ALL_UPPER> were added in the beginning. All the letters appearing less than 3 times were replaced by special <UNK> symbol.

The size of the corpus in sentences and words is given in 3. Experimental results are presented in Table 4, we evaluate both per-tag and per-sentence accuracy. We observe that CharWeight model reduces error rate by about 6% depending on the corpus while the CharFusion error reduction exceeds 10%. It demonstrates that our model indeed improves the quality of morphological tagging.

**Table 3.** ru_syntagrus corpus statistics

| Corpus | Words | Sentences |
|---|---|---|
| Train | 870,033 | 48,814 |
| Development | 118,427 | 6,584 |
| Test | 117,276 | 6,491 |

**Table 4.** Evaluation on UD2.0 dataset. ERR—error rate reduction

| Model | Tag accuracy | ERR | Sentence accuracy | ERR |
|---|---|---|---|---|
| CharLSTM (baseline) | 95.19 | 0.0 | 52.22 | 0.0 |
| | 95.22 | 0.0 | 50.98 | 0.0 |
| CharWeight | 95.54 | 7.3 | 54.95 | 5.7 |
| | 95.52 | 6.3 | 53.66 | 5.5 |
| CharFusion | 95.70 | 10.6 | 57.15 | 10.3 |
| | 95.70 | 10.0 | 56.29 | 10.8 |

### 4.3. Using morphological dictionary

Roughly speaking, morphological tagging for dictionary words simply selects the most appropriate tag from a predefined set of dictionary tags of the current word. Therefore we enrich the data which the model accesses with the output of morphological analyzer PyMorphy [Korobov, 2015]. For each word we compute the set of possible tags using PyMorphy, transform these tags to UD2.0 format by means of freely available russian-tagsets package and extract all UD2.0 categories that are compatible with the labels obtained. The list of categories is encoded using one-hot scheme and then embedded into a dense vector of length 256. This vector is concatenated to word embedding that is obtained from the character-level network.

Table 5 contains results of model evaluation in case a morphological dictionary is added. We find that there is no clear gain from using a language model in this case. This effect is surprising to us and we plan to investigate it further.

**Table 5.** Tagging accuracy when using a language model

| Model | Tag accuracy | ERR | Sentence accuracy | ERR |
|---|---|---|---|---|
| CharLSTM(baseline) | 95.19 | 0.0 | 52.22 | 0.0 |
| | 95.22 | 0.0 | 50.98 | 0.0 |
| CharLSTM+PyMorphy | 96.30 | 23.0 | 60.48 | 17.3 |
| | 96.43 | 25.3 | 60.01 | 18.4 |
| CharWeight+PyMorphy | 96.26 | 22.2 | 60.65 | 17.6 |
| | 96.43 | 25.3 | 60.21 | 18.8 |
| CharFusion+PyMorphy | 96.34 | 23.9 | 61.80 | 20.0 |
| | 96.46 | 25.8 | 60.70 | 19.8 |

## 5. MorphoRuEval 2017 Dataset

We also evaluate our models on MorphoRuEval-2017 dataset [Sorokin et al., 2017]. We compare against two best models, the deep learning one of [Anastasiev et al., 2017] and the HMM-based rule reranker [Sorokin and Yankovskaya, 2017]. The results of comparison are in Table 6. We used the results mentioned in the papers and the official evaluation script of the contest.

**Table 6.** Results on MorphoRuEval-2017 dataset

| Model | MorphoRuEval dev | | MorphoRuEval test | |
|---|---|---|---|---|
| | Tags | Sentences | Tags | Sentences |
| Anastasiev et al., 2017 | 97.8 | NA | 97.1 | 83.3 |
| Sorokin, Yankovskaya, 2017 | 96.3 | 78.5 | 94.8 | 69.3 |
| CharLSTM [Heigold et al., 2017] | 95.8 | 74.9 | 94.6 | 67.0 |
| CharFusion | 96.1 | 77.0 | 94.9 | 68.0 |
| CharLSTM+PyMorphy | 96.3 | 77.4 | 95.1 | 68.8 |
| CharFusion+PyMorphy | 96.6 | 79.8 | 95.4 | 71.1 |

We observe that our best model outperforms the second system of MorphoRuEval-2017 being sufficiently behind the first one. Note, that the model of [Anastasiev et al., 2017] used an additional training corpus and complex representations from in-house parser. Therefore our tagger demonstrates state-of-the-art performance on MorphoRuEval dataset as well.

## 6. Conclusions and future work

The present work mainly is a "proof-of-concept": we have demonstrated that character-level morphological tagging can be significantly improved using neural language models on morphological tags. Our work establishes a new state-of-the-art for tagging from scratch without access to external morphological resources. The natural direction is to test our approach on other languages with less data available analogously to the previous work. However, tagging almost a half of the sentences erroneously is a significant problem. Actually, some of these errors are not relevant since UD morphological tags contain categories which cannot be determined from the context (e. g., verb aspect) or does not have a clear bound from other categories (e. g., participles, which are treated as verbs), therefore it would be more natural to exclude them from being evaluated.

We have also demonstrated that adding the information from morphological dictionary can further improve performance. Actually, we have tried the simplest way to do it and further analysis is required. Another way to boost the model is to utilize task-independent embeddings obtained from large unlabeled corpora. The next challenge is to achieve the state-of-the art quality of closed systems using only open resources. We plan to address this question in the future work.

Another direction of research is improving neural models for morphological tags. Actually, not all government constraints can be addressed by the language model. For example, prepositions in Russian require different cases to their right ("*без друга*" *without the friend*+Gen vs "*про друга*» *about the friend*+Acc). The information about preposition cases is not encoded in their UD tags therefore in this case the CharLSTM component has to do the job more appropriate to the tag language model. Even a harder problem arises with verb government, consider *солгал другу* vs *обманул друга* both meaning "told+3 a lie to a friend" but with different case forms of the word *друг* ("*friend*"). Such examples demonstrate that actually we cannot separate "lexical"

and "morphological" part of tagging models. Probably, morphological tagging should be tackled using more complex architectures for sequence-to-sequence learning.

Summarizing, we have introduced a straightforward and linguistically motivated way to improve the quality of morphological tagging without having access to any external resources except the annotated corpus. Using external morphological dictionaries further improves performance. Our architecture is language-independent and does not have task-specific parameters, which makes it useful for applying "out of the box".

## 7. Acknowledgements

The author thanks Ekaterina Yankovskaya for invaluable help in editing and improving the first version of the paper. He is also grateful to his collegues in the laboratory of Neural Systems and Deep Learning of Moscow Institute of Physics and Technology and especially to Mikhail Arkhipov for helpful discussions.

## References

1. *Anastasiev D. G., Andrianov A. I., Indenbom E. M.* (2017), Part-of-speech tagging with rich language description, Computational linguistics and intellectual technologies: Proceedings of the International Conference "Dialog 2017". [Komp'yuternaya lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2017"], Moskva, pp. 2–13. http://www.dialog-21.ru/media/3895/anastasyevdgetal.pdf
2. *Gulcehre C. et al.* (2015), On using monolingual corpora in neural machine translation. arXiv preprint arXiv:1503.03535.
3. *Dozat T.* (2016), Incorporating nesterov momentum into adam. Available at https://openreview.net/pdf?id=OM0jvwB8jIp57ZJjtNEZ
4. *Heigold G., Neumann G., van Genabith J.* (2017), An extensive empirical evaluation of character-based morphological tagging for 14 languages, Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics. Valencia, Long Papers, Vol. 1., pp. 505–513.
5. *Chollet F. et al.,* Keras, available at https://github.com/keras-team/keras
6. *Kim Y., Jernite Y., Sontag D., Rush A. M.* (2016), Character-Aware Neural Language Models, AAAI, pp. 2741–2749.
7. *Lafferty J., McCallum A., Pereira F. C. N.* (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data, available at https://repository.upenn.edu/cgi/viewcontent.cgi?article=1162&context=cis_papers
8. *Mikolov T., Chen K., Korrado G., Dean J.* (2013), Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.
9. *Müller T., Schmid H., Schütze H.* (2013), Efficient higher-order CRFs for morphological tagging, Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, pp. 322–332.
10. *Nivre J. et al.,* (2017), Universal dependencies 2.0., available at https://lindat.mff.cuni.cz/repository/xmlui/handle/11234/1-1983

11. *Santos C. D., Zadrozny B.* Learning character-level representations for part-of-speech tagging (2014), Proceedings of the 31st International Conference on Machine Learning (ICML-14), Bejiing, pp. 1818–1826.
12. *Sorokin A. A. et al.* (2017), MorphoRuEval-2017: an evaluation track for the automatic morphological analysis methods for Russian, Computational linguistics and intellectual technologies: Proceedings of the International Conference "Dialog 2017". [Komp'yuternaya lingvistika i Intellektual'nye Tekhnologii: Trudy Mezhdunarodnoy Konferentsii "Dialog 2017"], Moscow, pp. 297–313, available at http://www.dialog-21.ru/media/3951/sorokinaetal.pdf
13. *Sorokin A. A., Yankovskaya E. V.* (2017), Using Context Features for Morphological Analysis of Russian, available at https://www.researchgate.net/publication/319623361_Using_Context_Features_for_Morphological_Analysis_of_Russian
14. *Srivastava R. K., Greff K., Schmidhuber J.* (2015), Highway networks, arXiv preprint arXiv:1505.00387.
15. *Sukhbaatar S., Szlam A., Weston J., Fergus R.* (2015), End-to-end memory networks, Advances in neural information processing systems, Montreal, pp. 2440–2448.
16. *Tran K., Bisazza A., Monz C.* (2016), Recurrent memory networks for language modeling, arXiv preprint arXiv:1601.01272.