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

Moscow, May 29—June 1, 2019

MORPHOLOGICAL PARSING OF LOW-RESOURCE LANGUAGES[1]

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

Moscow Institute of Physics and Technology, Neural Networks and Deep Learning Lab, Dolgoprudny, Russia;
Moscow State University, Faculty of Mathematics and Mechanics, Moscow, Russia

It this paper we study morphological parsing and lemmatization on the material of Evenk and Selkup language. We compare basic neural models with their extensions that attempt to utilize additional linguistic information from the training data. We show that the augmented model does not improve over the baseline even decreasing performance for the task of lemmatization. We hypothesize that to be helpful additional information should be extracted from external resources, if available, not the corpus itself.

Keywords: morphological parsing, lemmatization, low-resource languages, morphological guesser

МОРФОЛОГИЧЕСКИЙ АНАЛИЗ МАЛОРЕСУРСНЫХ ЯЗЫКОВ

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

Московский Физико-технический Институт, Лаборатория нейронных систем и глубокого обучения, Долгопрудный, Россия; Московский Государственный Университет, механико-математический факультет, Москва, Россия

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

Данная работа посвящена морфологическому анализу и лемматизации для эвенкийского и селькупского языка на материале соревнования LowResourceEval-2019 для малоресурсных языков. Мы сравниваем базовую нейронную модель с её расширениями, использующими лингвистическую информацию (морфологический словарь, извлечённый из корпуса), и показываем, что они не ведут к улучшению качества. Наша гипотеза состоит в том, что дополнительная информация должна извлекаться не из обучающего корпуса, а из внешних источников, в противном случае ту же самую информацию более эффективно извлекает сама нейронная сеть.

Keywords: морфологический анализ, малоресурсные языки, лемматизация, морфологические словари

1. Introduction

In recent years neural networks have dramatically improved the quality of natural language processing, especially in semantically-oriented tasks. One of the key advantages of neural models is their ability to extract knowledge from large amounts of unlabeled data, for example in the form of word embeddings or language models, or efficiently utilize patterns in raw data that are too complex or vague to be captured with handcrafted features. However, the applicability of neural approaches in low-resource setting is not that obvious. The main obstacle is the inclination of neural networks to overfit, especially on small datasets.

In the field of morphological tagging neural network models clearly outperform earlier approaches based on conditional random fields or local classifiers [2]. As discussed in [9], the key reason is the importance of word-level information which is readily captured by character-level embeddings, in contrast to tag-level interactions which were in the focus of hidden Markov models or conditional random fields. Several editions of CONLL shared tasks [13], [12] have demonstrated that neural networks are equally more efficient in high-resource or low-resource setting. Therefore we consider neural networks as a default choice for morphological analyzer without any need for further discussion.

What have to be discussed is the choice of information passed to the network. Usually neural models are trained on raw tokenized texts. On the contrary, earlier approaches to morphological tagging heavily relied on external morphological dictionaries and other resources. For example, the whole task of morphological tagging was treated as disambiguation, which is, the selection of the correct label from the ones presented in the dictionary. This approach is rarely applied in neural paradigm since it contradicts the main idea of neural NLP: make the model to learn arbitrarily complex patterns from data and do not impose restrictions on their form by external constraints. However, the studies for Russian language [1], [9] demonstrated, that passing the output of external morphological analyzer as additional input of the network improves tagging accuracy even in high-resource setting. Therefore we expected the benefits to be even higher in case when little data is given.

We tested our hypothesis on two datasets of LowResourceEval-2019 competition[2] [5] for Evenk and Selkup languages, solving the tasks of morphological tagging and lemmatization. Since the datasets were equipped with gold morpheme segmentation and most of the morphemes were found to correspond with morphological features, our strategy was restrict the set of potential tags given possible segmentations and pass these tags as additional inputs to the model (the approach successfully applied in the studies mentioned above). However, our hypothesis failed, since none of the complex models was able to outperform the basic ones[3].

The structure of our paper is the following: in **Section 2** we present the architecture of our basic model, **Section 3** describes the feature extraction process, **Section 4** is devoted to data and experiments description, in **Section 5** we present the results and discuss them. We conclude in **Section 6** with the directions for future work.

2. Model architecture

2.1. Morphological tagging

Our basic model is the implementation of Heigold's character-based network [2]. For the sake of completeness we briefly describe the architecture below. Note that similar approach was also pursued in the work [4] on neural language modelling. The model consists of two subnetworks: the first transforms the words to their vector representations, the second uses the obtained embeddings to predict morphological labels.

1. Each character is encoded as a 1-hot row vector with n_c dimensions, n_c being the number of characters. Thus the word is represented by a sequence of L such vectors $x_{i_1}, \ldots, x_{i_L}$, which is a matrix X with L rows and n_c columns with 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 $x'_{i_1}, \ldots, x'_{i_L}$. X_{i_j} is the i_j-th column of the embedding matrix U, which is a dense representation of i_j-th character in the alphabet.
3. X' is passed through parallel convolutional layers with different window size $w_1, \ldots, w_K$ and filters number $f_1, \ldots, f_K$. After this step K vectors of dimensions $f_1, \ldots, f_K$ are accociated 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 with L rows and F columns.
5. A maximum over each row is taken via max-pooling layer, finally encoding a word as a vector h' of fixed dimension F.

[2] https://lowresource-lang-eval.github.io

[3] Nonetheless, our models took the first place on all tasks where we participated

6. Several highway layers [10] are applied to this vector. Highway layer performs the transformation $h = s \odot g(Vh') + (1 - s) \odot h'$, where V is a square matrix with V rows, g a non-linear function and $\odot$ denotes coordinate-wise product. The idea is both to produce useful combinations of features using one-layer perceprtron output $g(Vh')$ and keep relevant dimensions of h' at the same time. The contributions of both components are balanced using s vector, which is obtained by another one-layer perceptron with sigmoid activation: $s = \sigma(Sh')$.

The second component of the network transforms the obtained sequence of word vectors $h_1, \ldots, h_n$ into n propability distributions $\pi_1, \ldots, \pi_n$, π_j being probabilities of tags for j-th word in the sentence. First, two LSTMs are applied[4], the first processing the sentence from left to right and the second from right to left. The first produces vectors $\vec{y}_1, \ldots, \vec{y}_n$ and the second produces $\overleftarrow{y}_n, \ldots, \overleftarrow{y}_1$, thus each word is encoded by two vectors $\vec{y}_i, \overleftarrow{y}_i \in \mathbb{R}^{n_y}$. These vectors are multiplied by a projection matrix W with n_t rows and n_y columns, n_t being the number of tags. Applying softmax layer produces the required probability distribution:

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

In [2] this architecture is proved to be successful for languages of different morphological structure even with only several thousands of tagged sentences available for training. It is also flexible enough to encode additional linguistic information i. e. from a morphological dictionary. This information is encoded in a vector form, for example, using a 0/1 vector of size n_t where nonzero elements correspond to the positions of possible dictionary tags. Such a vector $z_i^{\mathbf{feat}}$ is concatenated to the output z_i of bidirectional LSTM[5].

2.2. Lemmatization

In general, the task of lemmatization (the recovery of word normal form given the inflected one) is a string-to-string transduction problem. As demonstrated in several works [7] on string inflection, such a problem should be solved by sequence-to-sequence (seq2seq) neural networks. However, in most cases the transduction changes the material only on word edges (certainly, it is wrong for Semitic languages or for stem vowel alternations in German or Spanish), therefore it can be described using finite amount of information. This reduces the transduction problem "reconstruct the basic form letter by letter" to a classification problem "guess the transformation pattern" which can be solved even without neural networks. Such an approach is used, for example, in [11]. However, in the languages of the current study the problem becomes

[4] We omit the equations, interested reader may consult [4].

[5] we tested other ways to append this information, but this showed the higher performance.

even simpler: in Evenk the inflected form is always formed by attaching several (possibly zero) letters to the end of the word. Consequently, the lemma is always an initial segment of the word under consideration. Therefore to perform lemmatization one suffices to predict the end position of the stem. We model it by predicting a probability distribution $\mathbf{p}_w = [p_1, \ldots, p_{|w|}]$ over word positions where p_i is the probability that word stem ends after its i-th letter. Then the end of the initial word form is the maximum of this distribution.

For Selkup the pattern is slightly more complex: inflection also includes infixation, which is the insertion of morphemes inside the stem, though the percentage of such cases is rather low. For example, the inflected form of *amrsat* "*bowl*" is *amĭrsat*'. Hence, the stem is no longer a prefix of the word, but its (possibly discontinuous) subsequence. We model this by predicting two vectors $\mathbf{p}_w$ and $\mathbf{q}_w$: the first has the same meaning as for Evenk, while $q_i \in [0; 1]$ is the probability of i-th letter to be the result of epenthese, which implies that it is not present in the initial form.

Summarizing, the network architecture is the following:

1. Each symbol is encoded as a 0/1-vector of size n_t, which is transformed to a dense vector by multiplying an embedding matrix.
2. As in the tagging model, we pass the embeddings throw several convolutional layers. Each layer contains filters of different width, whose outputs are collected together in a single vector. To facilitate learning we insert batch normalization [3] between consecutive layers[6].
3. Each positional output h_i of the convolutional layer is multiplied by a trainable vector w to obtain a number $s_i = \langle w, h_i \rangle$.
4. The vector $\mathbf{s}$ of obtained scores is passed through a softmax layer to get the final probability distribution $\mathbf{p} = softmax(\mathbf{s})$:

$$p_i = \frac{e^{s_i}}{\sum_{j=1}^{|w|} e^{s_j}}$$

When we additionally predict the vector q of deletion probabilities, then the probability that i-th symbol is omitted is $q_i = \sigma(\langle w_{del}, h_i \rangle)$, where σ is the sigmoid activation function. To predict the lemma we find the maximum value of p_i: $I = \text{argmax}_i p_i$ and return w[:I] as lemma. When modelling the infixation, we additionally delete all symbols in positions j such that $q_j \geq \frac{1}{2}$.

3. Additional features

Theoretically, context-dependent morphological taggers should benefit from information, available from morphological dictionaries, lexicons and/or context-free analyzers. When a dictionary is not available, the most common tool to apply is a suffix guesser, which determines possible morphological features using word suffixes.

[6] This solution is crucial for deep convolutional network, without batch normalization the network often fails to learn at all due to gradient decay.

However, this method is not easily adapted to agglutinative languages since to extract a particular morpheme one may need to observe up to 8 final symbols. We selected another strategy: in addition to the morphological tags, the training data contained the morpheme segmentation of the form:

ne:jamtli ne: ja_DIM m_ACC tłi_3SG

Some of the morphemes can be converted to morphological features, for example, **3SG** is **Number=Sing|Person=3**. This mapping can be reconstructed automatically, by calculating probabilities of morphological features which cooccur with a given morph in a training corpus. This leads to the following method of feature extraction: given a word, we extract all possible morpheme combination on its right edge, checking not only the correctness of morph segmentation, but also the validity of corresponding sequence of morpheme types. Then for each morpheme type we extract the corresponding values of morphological features. We select as possible all morphological tags whose feature values do not contradict the selected features and cooccur with them in at least 3 training examples. To prevent overfitting we randomly replace this vector by all zeros with a fixed probability to allow the model to generalize to out-of-vocabulary inputs.

In the case of lemmatization we also experimented with either adding the part of speech label as additional input during lemmatization or multitask learning approach: we trained the network to predict word part-of-speech and detect stem boundary simultaneously, sharing all the embedding and convolutional layers between them.

4. Data and experiments

In case of neural tagging we run two models: the basic one and the one augmented with possible tag information. We also test three models for lemmatization: the basic one, the one augmented with morphological tags as input and the one with guessed possible morpheme boundaries.

4.1. Model parameters

Following [2] and [9], we choose the following parameters of morphological tagger: character embeddings are of size 32, convolutional window size changes from 1 to 7, the number of filters for width w is $\min(200, 50w)$. The number of convolutional layers is 2 with 0.2 dropout between layers and highway layer following the final convolution. This yields word embeddings of final size 1100, which are passed through a bidirectional LSTM with 128 units in each direction. We also tested several other parameter combinations but found these to give higher accuracy.

The lemmatizer had two convolutional layers of with 5 and 192 filters with no dropout (we found it useless in contrast to several previous studies). Other parameters are completely determined by the algorithm.

In preliminary studies we divided the dataset to train and development subsets and found 20 epochs of training to be optimal for lemmatization and 25 epochs for morphological tagging. Therefore our final models were trained for this number of epochs

without early stopping. We used Adam optimizer and batches of size 16. All networks are implemented using Keras framework, our implementation is open-source[7].

When we used the guesser, we additionally memorized all the word-lemma pairs that appear 5 or more times in the training data. Since the precision of the guesser is much lower than its recall (see **Section 5** below), we restricted its output to 5 most frequent tags.

4.2. Data

We test our models on Evenk and Selkup dataset of LowResourceLangEval contest[8]. The parameters of the dataset are given in **Table 1**. All the datasets were converted to CONLL-U format [9] by the Shared Task organizers, they provided the tokenization as well. We consider as tags the concatenation of part-of-speech label and morphological features, for example, ADP and NOUN, are both examples of possible tags. We also tried to predict the value of each feature, e. g. case and gender, separately, but this significantly deteriorated performance.

Table 1: Dataset parameters

Language	Dataset	words	sentences	unique tags	OOV words	hapaxes
2*Evenk	train == xbby xtby	25,869	5,527	873	0	8,063
	test == xbby xtby	2,697	548	319	814	272
2*Selkup	train == xbby xtby	13,436	2,394	316	0	5,088
	test == xbby xtby	2,426	425	151	912	246

Lemmatization algorithm is trained and tested on the same datasets. To reduce overfitting we downsample frequent words: if a word occurs $n > 5$ times in the dataset we include it to the training sample only $5 + \lceil \log_2(n-5) \rceil$ times.

We would like to note that in comparison to low-resource languages in CONLL 2018 Shared Task [12] Evenk and Selkup corpora are substantially larger, since most of low-resource corpora there do not exceed 1000 words. That implies that several questions relevant for actual low-resource parsing do not arise in our current study.

5. Results and discussion

In case of morphological tagging we compare two approaches, the basic one (BASIC) and the one using dictionary (DICTIONARY) information. Since we have no separate morphological dictionaries, all the word-tag pairs are extracted from the training data. We report accuracy both for morphological tags (which is, the percentage

[7] https://github.com/AlexeySorokin/NeuralMorphoTagger1/tree/low-resource

[8] https://lowresource-lang-eval.github.io

[9] https://universaldependencies.org/format.html

of words whose full morphological descriptions are predicted correctly) and sentences (the fraction of sentences where all words obtain correct morphological tags). For each metric we report two numbers: the average across 3 randomly initialized models (left) and the ensemble of these models (right).

Table 2: Results of morphological tagging

Model	Evenk		Selkup	
	Tag acc.	Sent acc.	Tag acc.	Sent acc.
BASIC	81.30 83.98	45.25 50.18	80.75 82.81	40.32 43.06
DICTIONARY	81.48 83.13	45.80 48.54	80.65 82.32	39.14 42.12

For lemmatization we compare 3 models: the basic one (BASIC), the tag-augmented one (TAGS) which takes gold morphological labels as additional inputs and the joint one (MULTITASK) which tries to predict these tags as an auxiliary task. As in case of tagging, we show the average accuracy across 3 runs and the accuracy of ensemble of 3 models.

Table 3: Results of lemmatization

Model	Evenk		Selkup	
	Single	Ensemble	Single	Ensemble
BASIC	91.32	93.33	88.35	89.94
TAGS	91.73	92.66	87.68	89.40
MULTITASK	90.88	91.88	86.26	87.79

5.1. Discussion

As shown in Table 2 and Table 3, the baseline network method either is on the par with linguistically informed extensions or even outperforms them. As demonstrated earlier [1], [9], in case of several other languages morphological dictionaries and guessers do improve performance (the boost is especially valuable for Russian and Pymorphy [6] analyzer). Actually, our results do not show that external morphological knowledge is useless, it only shows that dictionary cannot be extracted from the same training corpus. It can be viewed as the kind of overfitting: the dictionary information is available in training time, but the model may lack it in test phase due to OOV words. Usually dropping a fraction of dictionary inputs in training time fixes this issue at least partially, but the present study it was not the case.

To understand the phenomenon better we tested the guesser itself. As mentioned above, we restricted the output of the guesser to 5 most probable tags. This yields to the coverage of 72% for Selkup and 66% for Evenk even on the training set itself, since other hypotheses are too rare to occur between top 5 that do not contradict with morpheme segmentation. However, the coverage on the development is not much lower: 70% and 60%, which means that our morpheme guesser is able to generalize

to unseen words. Hence, it is not poor coverage that causes decrease in performance. Further, omitting the top 5 variants restriction produces $20 - 30$ variants for a word in average, which makes the guesser helpless (the choice between 30 tags is not easier than between the original 200). And the learning curve shows that dictionary-augmented model trains significantly faster on the first few iterations (which confirms that dictionary information is used), however, achieves lower performance in the end.

Probably, the problem lies in the datasets themselves. We compare the characteristics of the datasets with other language presented in UD 2.3 corpora [8]. Evenk belongs to Tungus family, which has no other UD corpora available, while Selkup is Uralic, though it belongs to Samoyed outgroup. **Table 4** contains the characteristics of Selkup in comparison with hu_szeged corpora of Hungarian, which also belongs to Uralic family of languages, and SST corpus of Slovenian, which is Indo-European (Slavic), but whose corpora is of the same order of size[10].

Table 4: Dataset comparison

Language	Dataset	words	sentences	unique tags	OOV words	hapaxes
2*Evenk	train == xbby xtby	25,869	5,527	873	0	8,063
	test == xbby xtby	2,697	548	319	814	272
2*Selkup	train == xbby xtby	13,436	2,394	316	0	5,088
	test == xbby xtby	2,426	425	151	912	246
2*Hungarian	train == xbby xtby	20,166	910	581	0	5,883
	test == xbby xtby	10,448	449	446	3233	513
2*Slovenian	train == xbby xtby	19,473	2,078	645	0	3,021
	test == xbby xtby	10,015	1,110	506	1,631	316

We observe that the main feature of Selkup corpus is smaller length of sentences. The ratio of unique tags and corpus length is almost the same as for Hungarian and lower than for Slovenian, but larger than for typical corpora of Universal Dependencies. The percentage of out-of-vocabulary words and hapax legomena is also much larger than in Hungarian which obviously makes tagging harder and makes it more useless to memorize training set in form of dictionaries. The quantitative properties of Evenk corpora are even more extreme than of Selkup. Last, but not the least is the origin of texts appearing in corpus, while most UD corpora are collected from media and fiction, Evenk and Selkup corpora are more informal by nature and mostly consist of native speakers oral speech records, which also makes the corpus less standartized. Summarizing, our hypothesis is that the informal nature of the corpora and dialectal variation increase the proportion of out-of-vocabulary and rare words, thus making it harder to memorize corpus via dictionaries.

All these concerns apply to lemmatization as well. The only thing we would like to mention is that multitask learning both on lemmatization and part-of-speech tagging task had failed, showing inferior performance. Probably this is due to low capacity of networks we used, however, our experiments show that even gold morphological

[10] The corpora for comparison are chosen randomly just to show the general pattern

tags does not improve lemmatization accuracy which implies that part-of-speech information is practically useless for this task. That contradicts our intuition and requires further study.

6. Conclusions and further work

We compared different methods of augmenting neural models with additional information for lemmatization and part-of-speech tagging of Selkup and Evenk languages. Our results show that basic neural models outperforms its extensions. We expect this to be not a general phenomena, but the feature of particular datasets. However, a wider cross-lingual study is required to reveal the factors that affect the applicability of morphological dictionaries in tasks of computational morphology. The first experiment to perform is to use not the dictionary, extracted from the training set, but independently constructed one. However, the author does not know such dictionaries for Evenk and Selkup. Another direction of study is the usage of unlabeled corpora. Such corpora were available in the shared task, but their orthography was different from the morphology corpus. Given recent success of minimally supervised neural language models, probably we can extract more from unlabeled data than from dictionaries and grammars.

However, the main problem is to find the cheapest and quickest way for field linguists to create resources which will allow high-quality morphological analysis. For example, it is questionable whether it is easier to collect an unlabeled corpus of required size or an example grammar. The author is not a practical linguist to resolve this question, however the adoption of neural networks from industrial NLP for main world languages to the low-resource studies still has to be done.

Acknowledgements

The author thanks the Shared Task for giving the opportunity to participate and for their helpful cooperation during the Shared Task. I am also very grateful to the stuff of MIPT Neural Networks laboratory for warm and inspiring atmosphere during the work on the problem.

References

1. *Anastasiev D., Gusev I., Indenbom E.* Improving part-of-speech tagging via multitask learning and character-level word representations (2018), International conference on computational linguistics "Dialogue", Moscow, Russia, Vol 1., pp. 14–28.
2. *Heigold G., Neumann G., van Genabith J.* An extensive empirical evaluation of character-based morphological tagging for 14 languages (2017), Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, Valencia, Spain, Vol. 1., pp. 505–513.
3. *Ioffe S., Szegedy C.* Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015), arXiv preprint arXiv:1502.03167, available at https://arxiv.org/pdf/1502.03167.pdf.

4. *Kim Y. et al.* Character-Aware Neural Language Models (2016), AAAI., pp. 2741–2749.
5. *Klyachko E. et al.* LowResourceEval-2019: a shared task on morphological analysis for low-resource languages. (2019), Proceedings of International conference on computational linguistics "Dialogue", online articles, Moscow, Russia, to appear.
6. *Korobov M.* Morphological analyzer and generator for Russian and Ukrainian languages (2015), International Conference on Analysis of Images, Social Networks and Texts, Ekaterinburg, Russia, pp. 320–332.
7. *Makarov P., Ruzsics T., Clematide S.* Align and copy: UZH at SIGMORPHON 2017 shared task for morphological reinflection (2017), arXiv preprint arXiv:1707.01355, available at https://arxiv.org/pdf/1707.01355.pdf.
8. *Nivre, J. et al.* Universal Dependencies 2.3. (2018), LINDAT/CLARIN digital library at the Institute of Formal and Applied Linguistics (ÚFAL), available at http://hdl.handle.net/11234/1-2895.
9. *Sorokin A.* Improving neural morphological Tagging using Language Models (2018), International conference on computational linguistics "Dialogue", Moscow, Russia, Vol 1., pp. 707–720.
10. *Srivastava R. K., Greff K., Schmidhuber J.* Highway networks (2015), arXiv preprint arXiv:1505.00387 available at https://arxiv.org/pdf/1505.00387.pdf.
11. *Straka M., Straková J.* Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe (2017), Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies., Vancouver, Canada, pp. 88–99.
12. *Zeman D. et al.* CoNLL 2018 shared task: Multilingual parsing from raw text to universal dependencies (2018), Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies., Brussels, Belgium, pp. 1–21.
13. *Zeman D. et al.* CoNLL 2017 shared task: Multilingual parsing from raw text to universal dependencies (2017), Proceedings of the CoNLL 2018 Shared Task: MultilingualParsing from Raw Text to Universal Dependencies, Vancouver, Canada, pp. 1–20.