ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **21**, Number 4, 2018, 446–459

A Character-Based Part-of-Speech Tagger with Feedforward Neural Networks

Aliaksei Kolesau , Dmitrij Šešok , and Mindaugas Rybokas

Vilnius Gediminas Technical University
E-mails: aliaksei.kolesau@vgtu.lt, dmitrij.sesok@vgtu.lt, mindaugas.rybokas@vgtu.lt

Abstract This article presents a simple method to perform part-of-speech (POS) tagging with feedforward neural networks applied to learnable character embeddings. The motivation of the research is based on the fact that for some languages a human can find out the part of speech for a word just by its spelling even without knowing the meaning of the word (see C. Fries's example "woggles ugged diggles"). One of the goals was to achieve high accuracy tagging without using semantic information (e.g. without word embeddings). This allows performing tagging for out of vocabulary words. Also, the dependency of the performance from context size was studied. The plausibility of the method was proved by building a POS-tagger with the accuracy comparable to state-of-the-art results.

Key-words: POS tagging, character embeddings, feedforward neural networks.

1. Introduction

Part-of-speech (POS) tagging is one of the first steps for building systems for machine translation [1], speech synthesis [2], identifying biomedical concepts [3], machine translation [4, 5], sentiment analysis [6] and other natural language processing (NLP) tasks [7, 8].

For example, authors of [9] noted that even though many neural network models such as Convolutional Neural Network, Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) have been proposed to learn representation of phrase/sentence, more syntactic knowledge can be used to solve natural language processing problems. Specifically, they claim that using part-of-speech tags in RNN and LSTM allows improvement in sentiment classification.

In [10] the method of incorporating part-of-speech tags in recurrent language models was explored. Authors achieved more accuracy in speech recognition task with the use of such language models in rescoring stage. Many different statistical and machine learning (including deep learning) methods were applied to build high quality automatic POS-taggers, especially for rich corpus languages such as English. For example, various types of Markov models (see [11] or [12]) and conditional random fields (CRF) ([13]) were used.

These models often require some hand-engineering work in order to obtain high accuracy. Specifically [13] uses a rich feature set including word context, prefixes and suffixes of a current word up to length 4, an indicator value of a word having capital letter, hyphen, dash or digit and some linguistic features. Fine-tuning such a system requires some expert knowledge specific for each new language.

As it usually happens in such situations attempts to apply neural networks were made. Remarkably, these attempts are not that recent as in other tasks: see Schmid work of 1994 ([14]). In that work, neural networks were used to predict a tag of the word having probability distribution on tags for preceding words. A priori tag probability was incorporated in the decoding scheme.

A great number of recent works (*e.g.* [15] or [16]) on topic are using word embeddings such as word2vec. Dense representation of words learnt on (for example) a language modelling task helps to catch syntactic features which certainly are useful for part-of-speech tagging. The problem with this approach is that it requires having embeddings for all words in input meaning so there is no way to predict a tag for out-of-vocabulary words. Also, learning high quality word embeddings usually requires big size datasets.

The problem of out-of-vocabulary classification was also investigated in [17]. Authors focused on improving Farsi part-of-speech tagging. They proposed neural network-based solution which consistently outperforms baseline both on in-vocabulary and out-of-vocabulary tokens. The proposed tagger's performance is highly influenced by morphological features, which we aim to avoid in our research.

Another possible method for attacking the problem of POS-tagging is the family of evolution algorithms. The detailed description, analysis and comparison with neural network method can be found in [18].

In this article we argue that word embeddings built from learnable character embeddings (similar to [19]) are good enough to build a high-quality POS-tagger at least for languages with rich morphological features (such as Russian). Moreover, POS-tagging is a naturally sequential problem; nevertheless, such embeddings are sufficient to get high accuracy for words taken independently (adding context certainly helps, which is confirmed with experiments). A neural network with fully connected layers and parametric rectified linear unit (PReLU) activation was used.

Great overview of various methods of data-driven POS-tagging (ranging from n-gram models to CRF-s) can be found in [20].

This article goes as follows. Section 2 gives a brief description of related works; in Section 3 our method of POS-tagging is presented. In Section 4 we present our experiments and conclude with discussion in Section 5. Some deep learning terms we use can be found in Appendix C.

2. Related works

Most researchers reasonably state the problem of POS-tagging as a task to get the best sequence of part-of-speech tags t1, t2, . . . , t_n given the sequence of words (usually sentence) w1, w2, . . . , w_n. The main difference of our paper is to use word embedding to classify its part-of-speech. A Russian and English POS-tagger constructed following our method proves that word embeddings built from character embeddings are sufficient to get reasonably high quality.

The sequential formulation is the only possible one for some languages because of their spelling system. The obvious example would be any language with hieroglyphs-based writing

system. The only way to get part-of-speech information for these languages is to catch semantic of syntactic features.

For example, in [21] POS-tagging for Arabic language was explored. The authors formulated the problem in graph terms and investigated ant-colony based algorithms for the task. Authors of [22] use specific peculiarities of Chinese language to build a high-quality POS-tagger.

Neural networks are often used to build high quality part-of-speech taggers. To our best knowledge the first attempt is described in [14]. In this work, authors train a neural network that takes the probability distribution on POS-tags of previous words in sentence as an input and tries to predict the tag of a new word. Smartly built lexicon is used to incorporate a priori probability of word's tag.

More recent works often use word embeddings to help with classification. For example, in [16] authors propose a novel method to train word embeddings from unlabelled data. Authors of [23] use both word embeddings and character embeddings. In [15] a smart method for online updating of word embeddings was proposed. One can use this method to classify out-of-vocabulary words, but the quality for words just seen naturally is not as good as for known words even if they are quite similar morphologically (we can be sure that "great-great-great-grandfather" has the same tag as "great-great-grandfather", which is easy to see from spelling but is hardly (or not) ever used in abovementioned approaches).

In SyntaxNet [24] a neural network takes word, prefix and suffix embeddings as an input for POS-tagging.

To our knowledge the state of the art result in POS-tagging belong to "Parsey Mc-Parseface" https://github.com/tensorflow/models/tree/master/syntaxnet. It is interesting to see that the authors achieved these wonderful results after updating their models to incorporate character embedding information (it is not the only change they have made).

As was stated above, all these works solve sequential problems. The contribution of this paper is that character embeddings are powerful enough to catch syntactic features needed for POS-tagging.

3. Materials and methods

We state the task of POS-tagging as a classification task. We have a word w with possible context (L ≥ 0 words preceding this word in the sentence and R ≥ 0 words going after w in the sentence) as an input.

The output of the model is a probability vector p with $p_i \geq 0,\ \sum_i p_i = 1$. p_i — the probability of the word w having the part-of-speech tag i according to our model.

We use greedy strategy to evaluate our model (meaning that we are comparing $\text{argmax}_i p_i$ with the correct answer instead of sampling it from p).

We use feedforward neural networks for this task. This implies that the input must be a fixed size vector. Therefore, we are limiting the length of the word by l (meaning that we cut the word if it has more than l letters in its spelling). In order to test our hypothesis we represent each character as a 1-hot vector with dimension n = $|A|$ + 2 (where A is an alphabet, including punctuation, digits, letters and special characters). Two other places are reserved for special symbols: "blank" (we ensure that each word has the size of exactly l by placing as many blanks at the end of the word as necessary) and "unknown" (for handling characters that have not been seen during training). To get reasonable behaviour for "unknown" characters we filter out all

the characters that have a small number of occurrences in a trainset (less than 0.003% trainset occurrence ratio in our experiments).

In a model with context we concatenate resulting matrices into one. This means that we have $(L + R + 1)\ l \times n$ matrix I as an input with exactly one 1 in each column.

We multiply this matrix by learnable matrix E that we call an embedding matrix with size $n \times k$ (k is an embedding size). Note that it is equivalent to replacing each column of I with k-vector specific to the character in that column. After we train the model (in inference time) we can just pull a needed vector for each character. This behaviour is widely used and implemented; *e.g.*, in tensorflow as a "tf.embedded_lookup()" method.

After that we flatten our matrix to $(L+R+1) \cdot n \cdot k$ vector and apply several fully connected layers to it. Each layer has PReLU [25] activation. Experiments with *tanh* and σ were also conducted, but the results were not satisfactory so they are not presented in this paper.

In the last layer an affine transformation with softmax activation is performed in order to get a probability distribution.

We use batch versions of Adam [26] and RmsProp algorithms to optimize the cross entropy criterion between the 1-hot vector of correct answer and output of the model.

In Section 4 we explore the performance depending on values of L, R, k, n, batch size, optimization algorithm and layer sizes.

Note that for $L = R = 0$ we have only word spelling for POS-tagging. As we can see from the experiments section this setup can also show reasonably high accuracy but adding context words helps a lot. It means that syntax features are very useful for POS-tagging and it is very interesting that despite this fact it is possible to get high results with only spelling (*i.e.*, not incorporating syntax features like word dependencies in a network input).

It is also worth noting that no prior or expert information is used in our setup. This allows training our model for new language very easily; we do not need linguistic knowledge for it.

Certainly it does not mean that it is impossible to get better results with handcrafted features. For example, we observed that we achieve higher quality if we reverse all the words. In Russian language many adjectives and verbs have specific endings, which make it easier to distinguish them from other parts-of-speech. But in our setup these ending occur in different positions of the input matrix I. Affine transforms must learn these endings in different positions.

Having the words reversed allows learning them only in the "beginning" of the word.

Visualization of the method is presented in Figure 1.

In the diagram, we present an example with classification of one word (L=R=0) into three classes (*e.g.*, "noun", "adjective", "verb). In the given example, our alphabet is composed of the letters "e", "y", and the blank (which we marked "_" in the diagram). The symbol "Unknown" has been skipped to simplify the diagram. Let the length of the chosen work be l=5 and the word being classified be "eye".

We add two blanks to get "eye_", and then build an l-hot encoding representation. Afterwards, we multiple the resulting matrix (l x n) by the character embedding matrix of size n x k, where k is the embedding size equal to 2 in our example.

Note that such multiplication is equivalent to changing each symbol with the corresponding vector representation or the row in the embedding matrix.

Then, we "stretch" the matrix to a numerical vector of size l x k and apply several fully-connected layers with the activation function PReLU. The last layer applies the activation softmax, thus receiving the predicted probability for each class.

You can see an implementation of our model at https://bitbucket.org/kolesov93/pos-tagger.

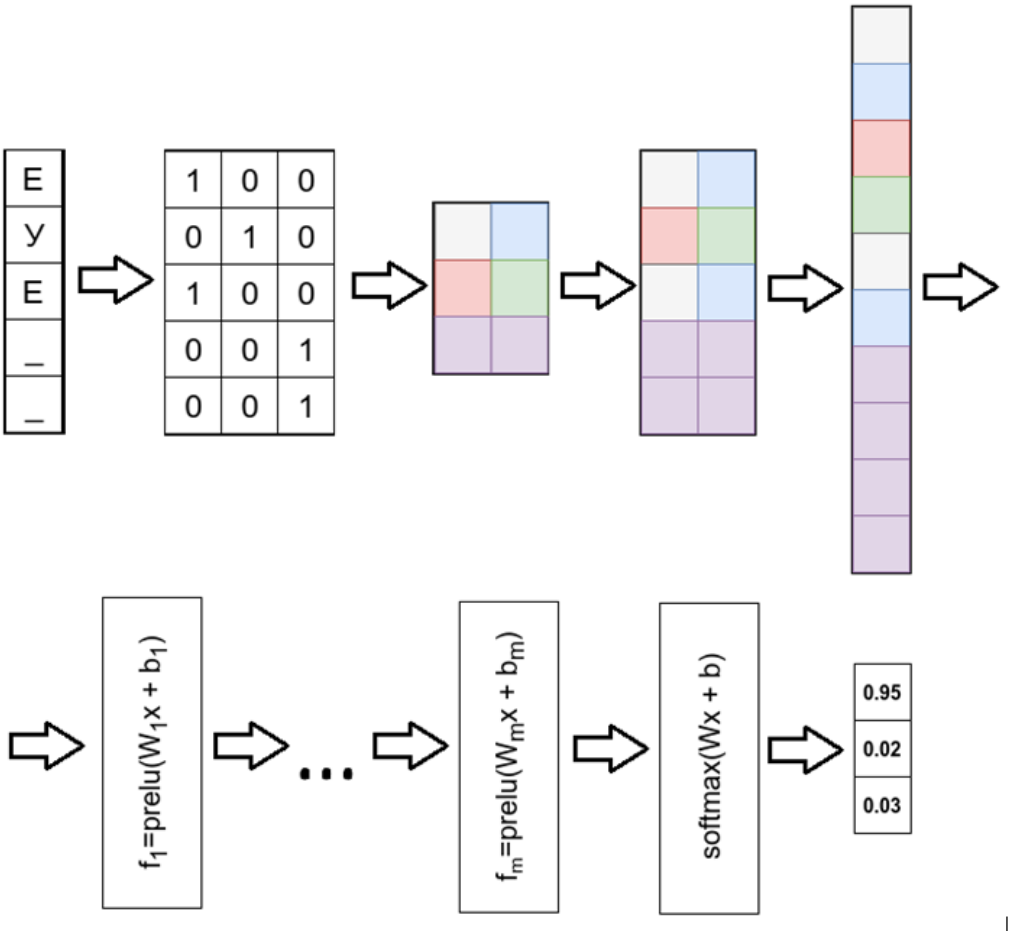

Figure 1. Visualization of the suggested method.

4. Results

Experiments for two languages were conducted (Russian and English) on the Universal Dependencies data (http://universaldependencies.org/). We used SynTagRus corpus for Russian language which has around 1M words in 66000 sentences. For English we used "Original" corpus (254K words in 16K sentences). Train/validation/test separation was used as it is proposed on the UD site. There are 17 part-of-speech tags for English and Russian.

For a baseline we computed the most probable part of speech for all words in trainset (by the number of occurrences) and predicted this tag for test set (we use "noun" for out of vocabulary words as it's the most probable tag in dataset), We've got 81% accuracy for Russian and 84% for English.

The experiments were conducted in two stages. First, we used a wide range of hyperparameters and trained each network for 5 epochs. The model with the best epoch validation loss is reported for each experiment at this stage. A whole range of parameter options is reported in Ta-

ble 1. Also, the experiments with the choice of activation function were performed, but PReLU was strictly superior to tanh and sigmoid as expected, so these result are not reported. The maximal length of a word is set to 12 in each experiment because it leaves at least 99% of words in a trainset with the length equal or less than 12.

Table 1. Options for the first stage of experiments

Parameter	Options
Layer sizes	[128, 128], [256, 256],
	[128, 128, 64],
	[64, 64, 64, 64]
Embedding size	4, 8, 16, 32
Left context	0, 1, 2
Right context	0, 1, 2
Optimizer	adam, rmsprop
Initial learning rate	10^{-3}, 10^{-4}
Was word reversed	True, False

The results for the first stage of experiments are presented in Appendix A.

It is easy to see that embedding, layer and context sizes matter. For example, shifting from embedding of size 8 to size 16 gives at least one per cent accuracy almost for every setting.

Subsequent doubling of the embedding size also gives more accuracy, but not so dramatically. Bigger embedding sizes were not used as they do not give much accuracy, but require more resources for network inference. The usual learning rate of 10^{-3} seems to be close to optimal judging by the plot of validations loss. The trick with word reversing also helps to achieve higher accuracy.

The best test accuracy for each language and context size is presented in Table 2.

Table 2. Options for the first stage of experiments

Language	Context size	Accuracy
English	(1, 1)	89.11%
English	(0, 1)	88.68%
English	(1, 0)	87.66%
English	(0, 0)	86.11%
Russian	(1, 1)	95.32%
Russian	(1, 0)	95.15%
Russian	(0, 1)	94.84%
Russian	(0, 0)	94.70%

In order to investigate the influence of context size, the second stage of the experiment was conducted. The networks with embedding size of 32 and two fully connected layers with 256 neurons each were trained with an Adam optimizer until convergence.

For single best model and Russian language the final result is 95.45% accuracy on the test set, and for English it is 89.11%. You can find more details in Appendix B.

Also, we build several model ensembles using random subsets of trained models and achieved 96.2% for Russian and 89.4% for English. For ensemble inference we ran all networks in the ensemble and averaged their probability vectors in logarithmic domain. In addition, experiments for a Lithuanian UD set were conducted with the goal to check our method viability for small

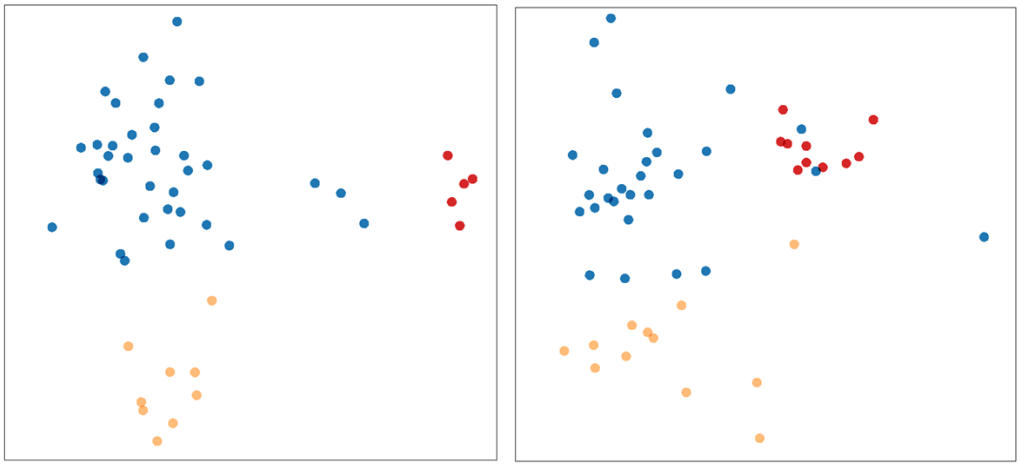

Figure 2. The embeddings for Russian language (on the left) and English language (on the right) embedded in 2D with principal component analysis (PCA).

resources learning. The trainset for Lithuanian only consists of fewer than 3500 words, which is not enough to produce a high quality tagging. The best result with a single model is 70.3% and 71% for a model ensemble.

5. Discussion

In this paper, a simple way to build a POS-tagger exclusively from character embeddings was presented. This method is trivial to train and apply but, as can be seen from experiments, it is not equally suitable for all languages. For example, for Russian it achieves a very strong 95.45% accuracy, but for English the results are not so good: 89.11%. It can be explained by the fact that in English it is very often that one word can be a noun and the verb in the different contexts (the word "store" illustrates this point well). Only complex systems that capture syntax features (like SyntaxNet) could provide high quality in such a setting. Still 89.11% is a good result; SyntaxNet achieves 90.48% on the same set.

We can empirically see that character embeddings have explainable properties (see Figure 2). For example, we can see a cluster of digits (red colour) and the cluster of punctuation (yellow colour).

On the other hand, our model is easy to train (note that for the L = R = 0 case we only need a vocabulary, but for other methods we need full sentences and often with syntax labels), and it captures morphological features that are not derived from syntax. Our model confidently tags quite famous Russian sentence "глокая куздрая штеко будланула бокра и кудрячит бокрёнка" that was created by linguists to illustrate the fact that a human is able to vaguely understand the sentence even when all the words are made up but placed and composed correctly. This gives an opportunity to use our method in TTS engines, which should be able to pronounce words that are not in the vocabularies (proper names or neologisms).

A great advantage of our model is that it is easy to train for each new language because all you need to know is its alphabet. You can significantly improve accuracy of proposed model

by adding context information if you have access to labelled sentences. But even without using context our method gives a boost over the statistical baseline.

A great advantage of our model is that it is easy to train for each new language because all you need to know is its alphabet. You can significantly improve accuracy of proposed model by adding context information if you have access to labelled sentences. But even without using context our method gives a boost over the statistical baseline.

There are several aspects in which we want to extend our work. First of all, it is desirable to use more complex neural network architectures such as convolution neural networks (we believe that it is possible to catch specific letter combinations with them) or bidirectional recurrent neural networks (which make it easier to combine all the information of word spelling). We hope that it is possible to at least halve the gap between our model accuracy and the state-of-the-art, while preserving simplicity of training and only a necessity of basic labelling (without syntax tags).

Also, it is interesting to extend our experiments for small corpora and achieve good results; *e.g.*, for a challenging Lithuanian dataset.

6. Conclusions

First of all, we empirically proved that the character embeddings catch enough information to be used for part-of-speech tagging. We see improvement over the statistical baseline for both Russian and English languages.

Not surprisingly, the quality of the resulting tagger is not better than state-of-the-art solutions and we see two main reasons for this:

- Our method does not use sentence-level information making it impossible to classify the word correctly in some cases (performance gains in context-using scenarios prove this point)
- We use a rather simple neural network architecture (related works in language modelling suggest using convolutional or recurrent models for the task)

Nevertheless, our method achieves reasonably high accuracy and makes it easy to train a POS-tagger: all you need is the vocabulary with part-of-speech tags for words, which is easy to get (in comparison to labelled sentences). Finally, we investigated a simple extension of our method in order to incorporate context information. As it was expected, such method gave a performance boost.

Appendix A: The First Stage of Experiments

In the following tables, the results for the first stage of experiments are presented (Table 3 for English, Table 4 for Russian). Only ten best and worst scenarios are presented here, the reader can read the full report at https://bitbucket.org/kolesov93/pos-tagger.

Wide networks with a rich embedding space enjoy a much better resulting accuracy.

Table 3. First stage of experiments for English

Layers	Embedding	Context	Reversed	Optimizer	Learning rate	Accuracy
256, 256	32	(1, 1)	+	adam	0.001	89.11%
256, 256	16	(1, 1)	+	adam	0.001	88.93%
256, 256	32	(1, 1)	+	rmsprop	0.001	88.88%
256, 256	32	(1, 1)	-	adam	0.001	88.77%
256, 256	32	(0, 1)	+	adam	0.001	88.68%
256, 256	16	(1, 1)	-	adam	0.001	88.47%
256, 256	16	(0, 1)	+	adam	0.001	88.46%
256, 256	32	(1, 1)	-	rmsprop	0.001	88.26%
256, 256	16	(1, 1)	+	rmsprop	0.001	88.19%
256, 256	16	(1, 1)	-	rmsprop	0.001	88.18%
...	...	...	...	...	...	...
64, 64, 64, 64, 64	16	(1,0)	+	adam	0.0001	53.56%
64, 64, 64, 64, 64	8	(1, 1)	+	rmsprop	0.0001	53.55%
64, 64, 64, 64, 64	4	(1, 1)	-	rmsprop	0.0001	53.21%
64, 64, 64, 64, 64	4	(1, 1)	+	adam	0.0001	53.13%
64, 64, 64, 64, 64	4	(0, 0)	+	adam	0.0001	52.84%
64, 64, 64, 64, 64	4	(1, 1)	+	rmsprop	0.0001	52.83%
64, 64, 64, 64, 64	4	(0, 1)	+	rmsprop	0.0001	52.51%
64, 64, 64, 64, 64	4	(0, 1)	-	rmsprop	0.0001	52.05%
64, 64, 64, 64, 64	8	(0, 0)	+	adam	0.0001	52.01%
64, 64, 64, 64, 64	8	(0, 1)	+	adam	0.0001	51.51%

Table 4. First stage of experiments for Russian

Layers	Embedding	Context	Reversed	Optimizer	Learning rate	Accuracy
256, 256	32	(1, 1)	+	adam	0.001	95.32%
256, 256	32	(1, 0)	+	adam	0.001	95.15%
256, 256	8	(1, 1)	+	adam	0.001	94.95%
128, 128	32	(1, 1)	+	adam	0.001	94.94%
256, 256	16	(1, 1)	+	adam	0.001	94.91%
256, 256	16	(0, 1)	+	adam	0.001	94.84%
256, 256	32	(0, 1)	+	adam	0.001	94.83%
128, 128	32	(0, 1)	+	adam	0.001	94.80%
128, 128, 64	32	(1, 1)	+	adam	0.001	94.79%
256, 256	16	(1, 0)	+	adam	0.001	94.72%
...	...	...	...	...	...	...
64, 64, 64, 64, 64	4	(0,0)	-	rmsprop	0.0001	73.43%
64, 64, 64, 64, 64	4	(1, 0)	-	adam	0.0001	72.18%
64, 64, 64, 64, 64	8	(1, 1)	-	adam	0.0001	72.13%
64, 64, 64, 64, 64	4	(0, 0)	-	adam	0.0001	71.72%
64, 64, 64, 64, 64	4	(1, 1)	-	rmsprop	0.0001	69.97%
64, 64, 64, 64, 64	8	(0, 1)	-	adam	0.0001	69.12%
64, 64, 64, 64, 64	4	(1, 0)	-	rmsprop	0.0001	68.65%
64, 64, 64, 64, 64	4	(1, 1)	-	adam	0.0001	66.66%
64, 64, 64, 64, 64	4	(0, 1)	-	adam	0.0001	66.26%
64, 64, 64, 64, 64	4	(0, 1)	-	rmsprop	0.0001	65.02%

Appendix B: The Second Stage of Experiments

In the following tables, the results of the second stage of experiments are provided.

Table 5. Results for English

Layers	Embedding	Context	Reversed	Optimizer	Learning rate	Accuracy
256, 256	32	(1, 1)	+	adam	0.001	89.11%
256, 256	32	(0, 1)	+	adam	0.001	88.70%
256, 256	32	(1, 0)	+	adam	0.001	87.70%
256, 256	32	(0, 0)	+	adam	0.001	86.31%

Table 6. Results for Russian

Layers	Embedding	Context	Reversed	Optimizer	Learning rate	Accuracy
256, 256	32	(1, 1)	+	adam	0.001	95.45%
256, 256	32	(0, 1)	+	adam	0.001	95.24%
256, 256	32	(1, 0)	+	adam	0.001	95.11%
256, 256	32	(0, 0)	+	adam	0.001	94.71%

For a more accurate distribution of correct answers over examples, see Figures 3-6. They display confusion matrices: j-th value in i-th row is proportional to the number of cases when the model finds that the word with the i-th POS-tag has the j-th POS-tag. Thus, correct answers are displayed on the diagonal of the confusion matrix.

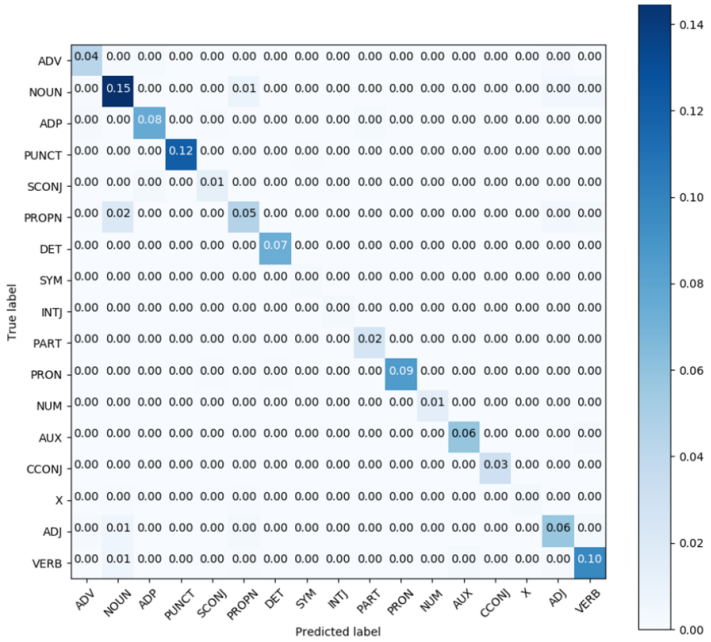

Figure 3. The confusion matrix for the best model for English normalized on the total amount of text samples (color online).

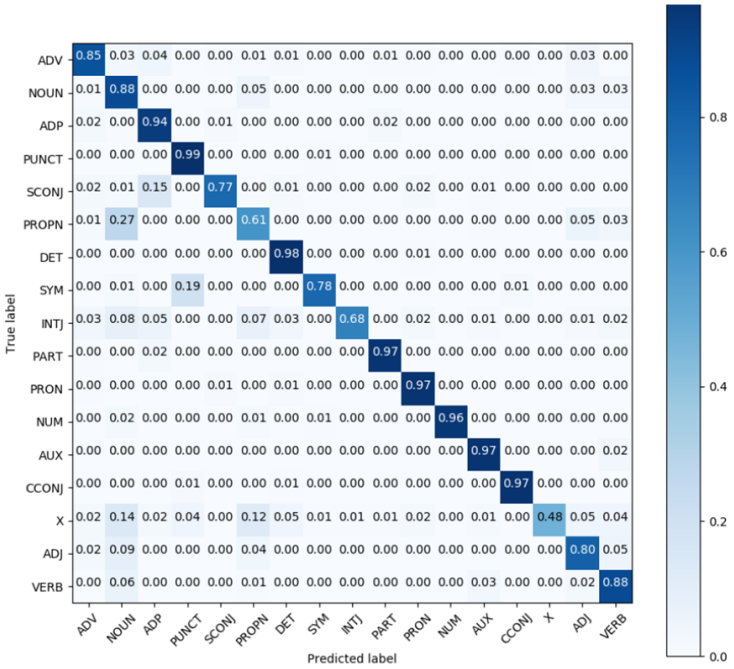

Figure 4. The confusion matrix for the best model for English, where each row is normalized on the amount of text samples with the i-th POS-tag (color online).

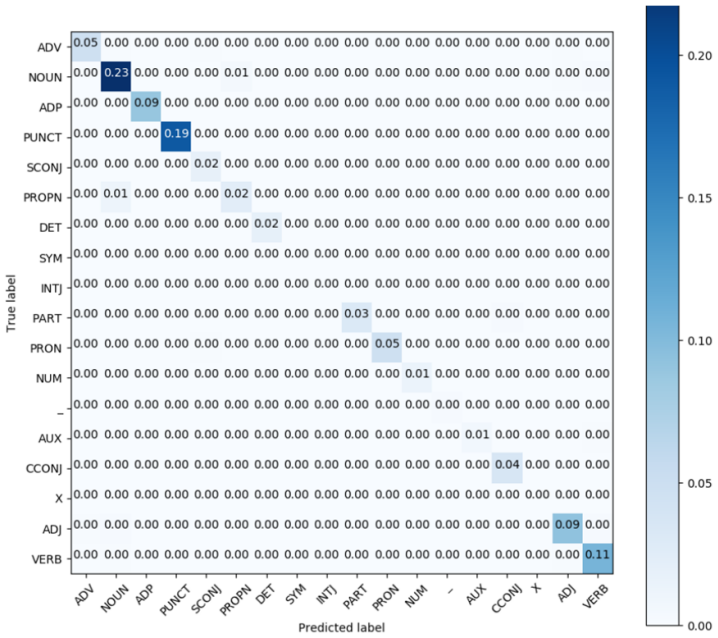

Figure 5. The confusion matrix for the best model for Russian normalized on the total amount of text samples (color online).

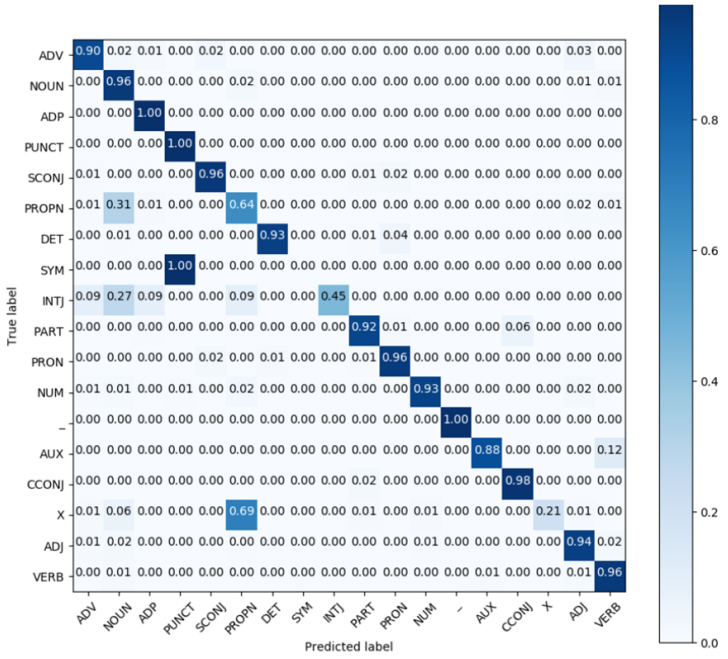

Figure 6. The confusion matrix for the best model for Russian, where each row is normalized on the amount of text samples with the i-th POS-tag (color online).

Appendix C: Deep Learning Terms

In this section some used deep learning terms are explained.

An activation function is a non-linear vector function applied after an affine transformation in neural networks. For a long time $\sigma(x) = 1/(1 + e^{-x})$ and $than(x) = (e^{2x} - 1)/(e^{2x} - 1)$ were the most popular activation functions. Nowadays, piecewise linear functions are widely used, because they do not have a saturation problem (neural network optimization is almost exclusively performed with gradient methods, but a gradient for aforementioned functions is close to zero almost everywhere except a relatively small area around the origin). A popular choice is a rectified linear unit (ReLU), which is zero in the negative domain and is an identity in the positive one. The most beneficial activation function in this article is a parametric rectifier linear unit (PReLU), which is linear in the negative domain and is an identity in the positive one.

Neural network optimization is often performed with modification of stochastic gradient descent method. Adam and RmsProp are popular examples of such modifications. They both make use of calculated statistics of performed steps. The reader can find more detailed overview in cited articles.

References

[1] SENNRICH R., HADDOW B., Linguistic Input Features Improve Neural Machine Translation, arXiv preprint arXiv:1606.02892.

[2] SUN M., BELLEGARDA J. R., Improved POS Tagging for Text-to-Speech Synthesis, Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference, 5384–5387.

[3] GONG L., YANG R., LIU Q., *et al.*, *A Dictionary-Based Approach for Identifying Biomedical Concepts*, International Journal of Pattern Recognition and Artificial Intelligence, **31**(9), 2017.

[4] KHOLOOSI M. M., REZAPOUR A., SADREDDINI M. M., *A Multi-Level Reordering Model for Statistical Machine Translation Using Source Side Parse Trees*, International Journal of Computer Science and Network Security, **17**(7), pp. 281–288, 2017.

[5] BOROŞ T., TUFIŞ D., *Romanian English Speech Translation*, Proceedings of the Romanian Academy, Series A., **15**(1), pp. 68-–75, 2014.

[6] WANG X., LIU Y., LIU M., SUN C., WANG X., *Understanding Gating Operations in Recurrent Neural Networks through Opinion Expression Extraction*, Entropy, **18**(8), 294, 2016.

[7] HEEMAN P. A., *POS Tags and Decision Trees for Language Modeling*, Proceedings of the Joint SIGDAT conference on empirical methods in natural language processing and very large corpora, pp. 129–137, 1999.

[8] WU F., *Dependency Parsing with Transformed Feature*, Information, **8**(1), 2017.

[9] HUANG M., QIAN Q., ZHU X., *Encoding Syntactic Knowledge in Neural Networks for Sentiment Classification*, ACM Transactions on Information Systems, **35**(3), 2017.

[10] SHI Y., LARSON M., PELEMANS J., et al., *Integrating Meta-Information into Recurrent Neural Network Language Models*, Speech Communication, **73**, pp. 64–80, 2015.

[11] CHURCH K. W., *A Stochastic Parts Program and Noun Phrase Parser for Unrestricted Text*, Proceedings of the Second conference on Applied Natural Language Processing, Association for Computational Linguistics, pp. 136–143, 1988.

[12] BRANTS T., *TnT: a Statistical Part-of-Speech Tagger*, Proceedings of the Sixth Conference on Applied Natural Language Processing, 2000; Association for Computational Linguistics; 224–231.

[13] SILFVERBERG M., RUOKOLAINEN T., LINDEN K., KURIMO M., *Part-of-Speech Tagging Using Conditional Random Fields: Exploiting Sub-Label Dependencies for Improved Accuracy*, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, **2**, pp. 25–264, 2014.

[14] SCHMID H., *Part-of-Speech Tagging with Neural Networks*, Proceedings of the 15th conference on Computational linguistics, Association for Computational Linguistics, **1**, pp. 172–176, 1994.

[15] YIN, W., SCHNABEL T., SCHÜTZE H., *Online Updating of Word Representations for Part-of-Speech Tagging*, arXiv preprint arXiv:1604.00502.

[16] WANG P., QIAN Y., SOONG F. K., HE L., ZHAO H., *Part-of-Speech Tagging with Bidirectional Long Short-Term Memory Recurrent Neural Network*, arXiv preprint arXiv:1510.06168.

[17] PASSBAN P., LIU Q., WAY A., *Boosting Neural POS Tagger for Farsi Using Morphological Information*, ACM Transactions on Asian and Low-Resource Language Information Processing, **16**(1), 2016.

[18] LV C., LIU H., DONG Y., *et al.*, *Corpus Based Part-of-Speech Tagging*, International Journal of Speech Technology, **19**(3), pp. 647–654, 2016.

[19] JÓZEFOWICZ R., VINYALS O., SCHUSTER M., SHAZEER N., WU Y., *Exploring the Limits of Language Modeling*, arXiv preprint arXiv:1602.02410.

[20] TUFIŞ D., *An Overview of Data-Driven Part-of-Speech Tagging*, Romanian Journal of Iinformation Science and Technology, **19**(1-2), pp. 78–97, 2016.

[21] OTHMANE B., ZRIBI C., FERIEL B., LIMAM I., *POS-Tagging Arabic Texts: A Novel Approach Based on Ant Colony*, Natural Language Engineering, **23**(3), pp. 419–439, 2017.

[22] SUN W., WAN X., *Towards Accurate and Efficient Chinese Part-of-Speech Tagging*, Computational Linguistics, **42**(3), pp. 391–419, 2016.

[23] MA X., HOVY E. H., *End-to-End Sequence Labeling via Bi-directional lstm-cnns-crf*, arXiv preprint arXiv:1603.01354.

[24] ANDOR D., ALBERTI C., WEISS D., SEVERYN A., PRESTA A., GANCHEV K., PETROV S., COLLINS M., *Globally Normalized Transition-Based Neural Networks*, arXiv preprint arXiv:1603.06042.

[25] HE K., ZHANG X., REN S., SUN J., *Delving deep into rectifiers: Surpassing Human-Level Performance on Imagenet Classification*, Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034, 2015.

[26] KINGMA D, BA J. A., *A Method for Stochastic Optimization*, arXiv preprint arXiv:1412.6980.