Named Entity Recognition in Russian with Word Representation Learned by a Bidirectional Language Model

Georgy Konoplich[1], Evgeniy Putin[1], Andrey Filchenkov[1(✉)], and Roman Rybka[2]

[1] ITMO University, Saint Petersburg, Russia
konoplich@rain.ifmo.ru, putin.evgeny@gmail.com, afilchenkov@corp.ifmo.ru
[2] Kurchatov Institute, Moscow, Russia
RybkaRB@gmail.com

Abstract. Named Entity Recognition is one of the most popular tasks of the natural language processing. Pre-trained word embeddings learned from unlabeled text have become a standard component of neural network architectures for natural language processing tasks. However, in most cases, a recurrent network that operates on word-level representations to produce context sensitive representations is trained on relatively few labeled data. Also, there are many difficulties in processing Russian language. In this paper, we present a semi-supervised approach for adding deep contextualized word representation that models both complex characteristics of word usage (e.g., syntax and semantics), and how these usages vary across linguistic contexts (i.e., to model polysemy). Here word vectors are learned functions of the internal states of a deep bidirectional language model, which is pretrained on a large text corpus. We show that these representations can be easily added to existing models and be combined with other word representation features. We evaluate our model on FactRuEval-2016 dataset for named entity recognition in Russian and achieve state of the art results.

Keywords: NER · Word representation · Semi-supervised learning
Language modeling · Bi-LSTM

1 Introduction

Due to their simplicity and efficiency, pre-trained word embedding have become widespread in natural language processing (NLP) systems. Many prior studies have shown that such embedding capture useful semantic and syntactic information [1, 2] and including them in NLP systems has been shown to be highly helpful for a variety of domain tasks [3]. However, these approaches for learning word vectors only allow a single context independent representation for each word. Learning high quality representations can be challenging.

Previously proposed methods overcome some of the shortcomings of traditional word vectors by either enriching them with subword information [4, 5] or learning separate vectors for each word sense [6]. Other recent work has also focused on learning

D. Ustalov et al. (Eds.): AINL 2018, CCIS 930, pp. 48–58, 2018.
https://doi.org/10.1007/978-3-030-01204-5_5

context-dependent representations. Context2vec [7] uses a bidirectional Long Short Term Memory (LSTM) to encode the context around a pivot word. Other approaches for learning contextual embeddings include the pivot word itself in the representation and are computed with the encoder of either a supervised neural machine translation (MT) system [8] or an unsupervised language model [9]. Both approaches benefit from large datasets, although the MT approach is limited by the size of parallel corpora.

Previous work has also shown that different layers of deep bidirectional recurrent neural networks (biRNNs) encode different types of information. For example, introducing multi-task syntactic supervision (e.g., part-of-speech tags) at lower levels of a deep LSTM can improve overall performance of higher level tasks such as dependency parsing [10]. Authors of [11] showed that in an RNN-based encoder-decoder machine translation system, the representations learned at the first layer in a 2-layer LSTM encoder are better at predicting POS tags than the representations learned at the second layer. Finally, the top layer of an LSTM for encoding word has been shown to learn representations of word sense [7].

State of the art sequence tagging models typically include a biRNN that encodes word sequences into a context sensitive representation before making word specific predictions [10, 12, 13]. The problem with these models is that they are trained on a small amount of labeled data and do not fully take into account the context of each word. Authors of [12] have presented methods for jointly learning the biRNN with supplemental labeled data from other tasks.

There are many difficulties in processing the Russian language: free word order, morphological richness, polysemy, neologisms. The approach we suggest aims to handle with all these difficulties.

In this paper, we explore an alternate semi-supervised approach, which does not require additional labeled data. We use Embeddings from Language Models (ELMo) representations [14] (Sect. 2.3). Unlike previous approaches for learning contextualized word vectors [8, 9], ELMo representations are deep, in the sense that they are a function of all internal layers of the deep bidirectional language model (biLM, Sect. 2.1). The biLM architecture is described in Sect. 2.4. Our biLM train on large corpus of unlabeled Russian data, then we use fine-tuning of biLM model on task specific data, supervised labels are temporarily ignored. For fine-tuning we use approach similar to the one presented in paper [15]. We use discriminative fine-tuning and gradual unfreezing, techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning (Sect. 2.2). Then combined fastText word representation and ELMo embeddings are given to a bidirectional LSTM (Sect. 3).

2 Word Representation

Unlike the most widely used word embeddings [1, 2], ELMo word representations are functions of the entire input sentence. They are computed on the top of two-layer biLMs with character convolutions, as a linear function of the internal network states. This setup allows to perform semi-supervised learning, where the biLM is pretrained at a large scale and incorporated into a wide range of existing neural NLP architectures. We also fine-tune the biLM on domain specific data to increase performance of the model for NER.

2.1 Bidirectional Language Models

Given a sequence of N tokens, $(t_1, t_2, \ldots, t_N)$, a forward language model computes the probability of the sequence by modeling the probability of token t_k given the history $(t_1, \ldots, t_{k-1})$:

$$p(t_1, t_2, \ldots, t_N) = \prod_{k=1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1}).$$

Recent state-of-the-art neural language models [16] compute a context-independent token representation x_k^{LM} (via token embeddings or a CNN over characters) then pass it through L layers of forward LSTMs. At each position k, each LSTM layer outputs a context-dependent representation $\overrightarrow{h_{k,j}^{LM}}$, where $j = 1, ..., L$. The top layer LSTM output $\overrightarrow{h_{k,L}^{LM}}$ is used to predict the next token t_{k+1} with a Softmax layer. A backward LM is similar to a forward LM, except it runs over the sequence in reverse, predicting the previous token given the future context. It can be implemented in an analogous way to a forward LM, with each backward LSTM layer j in a L layer deep model producing representations $\overleftarrow{h_{k,j}^{LM}}$ of t_k given $(t_{k+1}, ..., t_N)$.

A biLM combines both the forward and backward LM. This formulation jointly maximizes the log likelihood of the forward and backward directions [14]:

$$\sum_{k=1}^{N} \left(\log p(t_k | t_1, \ldots, t_{k-1}; \Theta_x, \overrightarrow{\Theta_{LSTM}}, \Theta_S) + \log p(t_k | t_{k+1}, \ldots, t_N; \Theta_x, \overleftarrow{\Theta_{LSTM}}, \Theta_s) \right).$$

The parameters tied for both the token representation (Θ_x) and Softmax layer (Θ_s) in the forward and backward direction while maintaining separate parameters for the LSTMs in each direction. Overall, this formulation is similar to the approach presented in [9], with the exception that some weights shared between directions instead of using completely independent parameters.

2.2 Fine-Tuning BiLM

For fine-tuning, we use approach similar to the one presented in [15]. We use discriminative fine-tuning and gradual unfreezing techniques to retain previous knowledge and avoid catastrophic forgetting during fine-tuning.

Discriminative Fine-Tuning. As different layers capture different types of information [17], they should be fine-tuned to different extents. To this end, we use discriminative fine-tuning. Instead of using the same learning rate for all layers of the model, discriminative fine-tuning allows us to tune each layer with different learning rates. For context, the regular stochastic gradient descent (SGD) update of a model parameters θ at time step t looks like the following [18]:

$$\theta_t = \theta_{t-1} - \eta \cdot \nabla_\theta J(\theta),$$

where η is the learning rate and $\Delta_\theta J(\theta)$ is the gradient with regard to the model objective function. For discriminative fine-tuning, we split the parameters θ into $\{\theta^1, ..., \theta^L\}$, where θ^l contains the parameters of the model at the l-th layer and L is the number of layers of the model. Similarly, we obtain $\{\eta_1, ..., \eta_L\}$ where η_l is the learning rate of the l-th layer.

The SGD update with discriminative finetuning is then the following:

$$\theta_t^l = \theta_{t-1}^l - \eta^l \cdot \nabla_{\theta l} J(\theta).$$

We choose learning rate η_L of the last layer. The learning rate for lower layers is chosen as $\eta^{l-1} = \frac{\eta^l}{2.6}$ guided by an empirical research.

Gradual Unfreezing. Rather than fine-tuning all layers at once, which risks catastrophic forgetting, we use to gradually unfreeze the model starting from the last layer as this contains the least general knowledge [17]. We first unfreeze the last layer and fine-tune all unfrozen layers for one epoch. We then unfreeze the next lower frozen layer and repeat, until we finetune all layers until convergence at the last iteration. This is similar to "chain-thaw" [19], except that we add a layer at a time to the set of "thawed" layers, rather than only training a single layer at a time.

2.3 ELMo Embeddings

ELMo is a task specific combination of the intermediate layer representations in the biLM. For each token t_k, a L-layer biLM computes a set of $2L + 1$ representations

$$R_k = \{x_k^{LM}, \overrightarrow{h_{k,j}^{LM}}, \overleftarrow{h_{k,j}^{LM}} | j = 1, ..., L\} = \{h_{k,j}^{LM} | j = 0, ..., L\},$$

where $h_{k,0}^{LM}$ is the token layer and $h_{k,j}^{LM} = \left[\overrightarrow{h_{k,j}^{LM}}; \overleftarrow{h_{k,j}^{LM}}\right]$ for each biLSTM layer. For inclusion in a downstream model, ELMo collapses all layers in R into a single vector, $ELMo_k = E(R_k; \Theta_e)$. In the simplest case, ELMo just selects the top layer, $E(R_k) = h_{k,L}^{LM}$, as in TagLM [9] and CoVe [8]. More generally, word representation using all biLM layers computed as:

$$ELMo_k = E(R_k; \Theta) = \gamma \cdot \sum_{j=0}^{L} s_j h_{k,j}^{LM},$$

where, s are softmax-normalized weights and the scalar parameter γ allows the task model to scale the entire ELMo vector. Parameter γ is of practical importance to aid the optimization process. Considering that the activations of each biLM layer have a different distribution, in some cases it also helped to apply layer normalization [20] to each biLM layer before weighting.

2.4 Pre-trained Bidirectional Language Model Architecture

In this paper, we use following BiLM AllenNLP TensorFlow implementation[1]. This BiLM architecture is similar to the architectures described in [16], but modified to support joint training of both directions and add a residual connection between LSTM layers (Fig. 1).

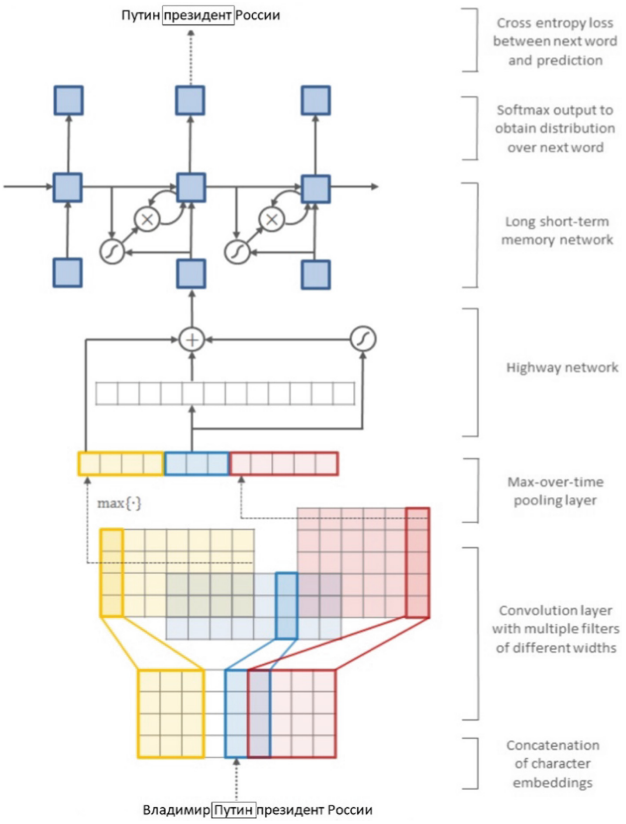

Fig. 1. BiLM architecture

To balance overall language model perplexity with the model size and computational requirements for downstream tasks while maintaining a purely character-based input representation, we change all embedding and hidden dimensions from the single best model CNN-BIG-LSTM in [16]. The final model uses $L = 2$ biLSTM layers with 2048 units and 512 dimension projections and a residual connection from the first to the second layer. The context insensitive type representation uses 2048 character n-gram convolutional filters followed by two highway layers [21] and a linear projection down to a 512 representation.

[1] https://github.com/allenai/bilm-tf.

3 Bidirectional LSTM + CRF Model for NER

Following recent state-of-the-art systems [9, 12], the baseline model uses pre-trained word embeddings, two biLSTM layers and a conditional random field (CRF) loss [22], similar to [3]. The different word representation is passed through two biLSTM layers, the first with 40 hidden units and the second with 20 hidden units before a final dense layer. During training, we use a CRF loss and at test time perform decoding using the Viterbi algorithm. The architecture of the model is presented on Fig. 2.

The implementation of the model can be found in online repository[2].

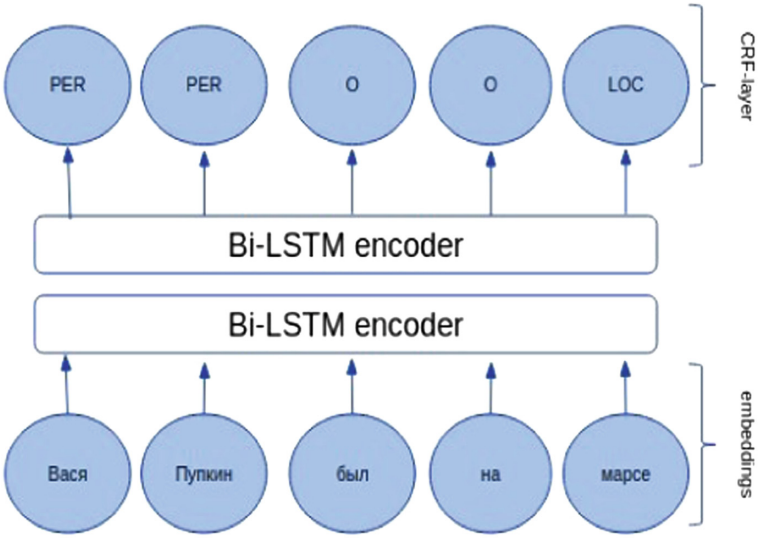

Fig. 2. The main network architecture for NER. Word embeddings are given to a bidirectional LSTM. Example of input is in Russian: "Vasya Pupkin was on Mars"

4 Experiments and Results

4.1 Dataset and Evaluation

We use the FactRuEval corpus of data[3]. The corpus consists of newswire and analytical texts in Russian dealing with social and political issues. The texts were gathered from Private Correspondent[4] and Wikinews[5].

[2] https://github.com/ctlab/ML/BiLSTM_for_NER.

[3] https://github.com/dialogue-evaluation/factRuEval-2016.

[4] http://www.chaskor.ru/.

[5] https://ru.wikinews.org.

The corpus is split into two parts—a demo corpus of 122 texts and a test corpus of 133 texts. The demo corpus contains about 30 thousand tokens and 1700 sentences. The test corpus contains about 60 thousand tokens and 3100 sentences. We train our models on the demo corpus and test the on the test corpus. Entities of the following three types have to be recognized: person, location, organization. We use official FactRuEval-2016 evaluation scripts to calculate metrics of performance (micro-averaged F_1).

4.2 Baseline

As a baseline solution, we tried to train a neural network only on morphological and syntactic features.

For word representation, we use the information contained in the syntax tree, the surrounding words in the sentence with the window size 5, various morphological features: prefix, postfix, post-tag, lemma and others. The input dimension was 825.

We tried different architectures of neural networks and used GridSearch for hyperparameter optimization. The best architecture of neural network consists of three recurrent layers (LSTM), two dense layers and multi-layer perceptron at the end. The architecture of the model is presented on the Fig. 3.

Full set of parameters for this model consists of parameters of neural network layers (weight matrices, biases, word embedding matrix). All these parameters are tuned during training stage by back propagation algorithm with stochastic gradient descent ($lr = 0.1, decay = 10^{-7}, momentum = 0, clipvalue = 3$). Variational Dropout is applied to avoid overfitting and to improve the system performance.

After training 30 epochs, RNN + MLP model achieves 72.90 *precision*, 76.60 *recall* and 74.71 F_1. Input features encode insufficient knowledge of natural language and the model is trained on a small amount of labeled data. It requires a lot of word knowledge, and a deep understanding of grammar, semantics, and other elements of natural language.

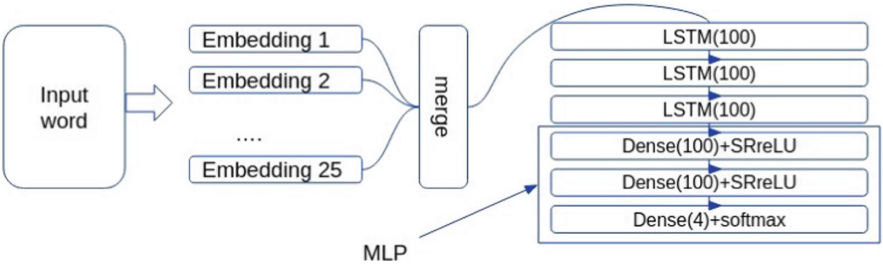

Fig. 3. Baseline solution. Input word representation combine syntactic and morphological features and feed it to 3-layers LSTM model with MLP at the end.

Then for improving results, we add distributive word representation for each word. However, word representations only give a single context independent representation for each word. And at the end we will show how Embeddings from Language Models

(ELMo) improve results. ELMo give a context dependent representation for each word and solve the problem of polysemy.

4.3 Training BiLM Model and Fine-Tuning

We train biLM model on large corpus of data. Data are taken from conll2017[6]. After preprocessing, we have about 200 million tokens and 15 million sentences, the vocabulary is 60 thousand most frequent words. After training for 5 epochs, the average forward and backward perplexities is about 47. Once pretrained, the biLM can compute representations for any task.

In some cases, fine-tuning the biLM on domain specific data leads to significant drops in perplexity and an increase in downstream task performance. This can be seen as a type of domain transfer for the biLM. To fine-tune the model, we take raw sentences of demo corpus. The biLM is fine-tuned as described in Subsect. 2.2 for 3 epochs and evaluated on the test corpus sentences. Test corpus perplexity after fine-tuning improved from 90 to 40 (lower is better). Once fine-tuned, the biLM weights were fixed during task training.

4.4 Training Bi-LSTM + CRF

FastText word representations and some morphological features are given to model. Full set of parameters for this model consists of parameters of Bi-LSTM layers (weight matrices, biases, word embedding matrix) and transition matrix of CRF layer. All these parameters are tuned during training stage by back propagation algorithm with stochastic gradient descent ($lr = 0.1, decay = 10^{-7}, momentum = 0, clipvalue = 3$). Variational Dropout is applied to avoid over-fitting and improve the system performance. After training for 30 epochs, Bi-LSTM + CRF model achieves 77.23 *precision*, 85.19 *recall* and 81.02 F_1.

Then we combine FastText word representations and ELMo embeddings before fine tuning and repeated training model. After training for 30 epochs, Bi-LSTM + CRF model achieves 82.32 *precision*, 84.04 *recall* and 83.17 F_1.

Finally, we fine-tune biLM model and take ELMo embeddings. After training for 30 epochs, Bi-LSTM + CRF model achieves 83.19 *precision*, 85.41 *recall* and 84.29 F_1.

4.5 Results and Other Works

Traditional approaches to named entity recognition in Russian are based on hand-crafted rules and external resources. So, in work [23], regular expressions and dictionaries were used to solve the problem. The next step was the application of statistical training methods, such as conditional random fields (CRF) and supporting vector machines (SVM) for classifying entities. CRF over linguistic features, considered as the baseline in the studies [24]. In [25], a two-stage algorithm of conditional random fields was proposed: at the first stage, the input for the CRF was on hand-crafted linguistic

[6] http://universaldependencies.org/conll17/.

features. Then on the second, the same linguistic features were combined with global statistics, calculated at the first stage and submitted to the CRF. In work [26], SVM was applied to the distributive vector representations of words and phrases. These representations were obtained by extensive unsupervised pre-training on different news corpora. Simultaneous use of dictionary-based features and distributed word representations was presented in [27]. Dictionary features were retrieved from Wikidata and word representations were pre-trained on Wikipedia. Then these features were used for classification with SVM.

In [28], LSTM networks were applied to NER task in Russian. In the study [29], a modern model of the neural network for the English NER is applied to open data mappings for NER in Russian. The model consists of three main components: bidirectional recurrent networks (bi-LSTM), CRF and distributed vector representation of words. Such topology is widely used in for sequential data processing in different domains [30, 31]. A deeper study on how these models can be combined was presented in [32].

Table 1 presents the results of studies in which the evaluating phase corresponds to the requirements of the competition factRuEval-2016. We compare our results with methods, proposed in [27, 29, 32] as showing state-of-the-art results. We did not include [26], because the authors achieved the results on strings that were known to contain NER term, while we solve the problem without such knowledge, for entire text, some strings of which may not contain any NER term.

Table 1. Comparison of different models.

Model	Precision	Recall	F1-score
Rubaylo and Kosenko [29]	77.70	78.50	78.13
Basic + dictionary + w2v features + SVM [27]	82.57	74.08	78.10
Bi-LSTM + CRF + Lenta [32]	**83.8**	80.84	82.10
NeuroNER + Highway char [32]	80.59	80.72	80.66
RNN + MLP	72.90	76.60	74.71
Bi-LSTM-CRF	77.23	85.19	81.02
Bi-LSTM-CRF + ELMo	82.32	84.04	83.17
Bi-LSTM-CRF + ELMo + fine-tuning	83.19	**85.41**	**84.29**

As it can be seen, usage of ELMo helped to increase model performance dramatically in terms of precision at the cost of decreasing its recall. Fine-tuning increased both precision and recall even greater, resulting in the best model we used in comparison.

5 Conclusions

In this work, we showed that the context sensitive representation captured in the ELMo embeddings is useful in named entity recognition in Russian. When we fine-tuned our bidirectional language model and included ELMo embeddings in our Bi-LSTM, we achieve state-of-the-art results.

There is no clear understanding between the performance of the language model and the accuracy in solving specific domain tasks. It is not clear when the language model has learned better and its application will yield better results on the inherited model. Further it is supposed to continue studying language models. Training them in the Russian language, experiments with different architectures, obtaining deep context-dependent representations. After all, such representations are useful in many tasks of natural language processing.

Acknowledgements. The authors would like to thank Ivan Smetannikov for helpful conversation and unknown AINL reviewers for useful comments.

The research was supported by the Government of the Russian Federation (Grant 08-08).

References

1. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
2. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
3. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. **12**, 2493–2537 (2011)
4. Wieting, J., Bansal, M., Gimpel, K., Livescu, K.: Charagram: embedding words and sentences via character n-Grams. arXiv:1607.02789 (2016)
5. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv:1607.04606 (2016)
6. Neelakantan, A., Shankar, J., Passos, A., McCallum, A.: Efficient non-parametric estimation of multiple embeddings per word in vector space. arXiv:1504.06654 (2015)
7. Melamud, O., Goldberger, J., Dagan, I.: context2vec: learning generic context embedding with bidirectional LSTM. In: Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning, pp. 51–61 (2016)
8. McCann, B., Bradbury, J., Xiong, C., Socher, R.: Learned in translation: contextualized word vectors. In: Advances in Neural Information Processing Systems, pp. 6297–6308 (2017)
9. Peters, M.E., Ammar, W., Bhagavatula, C., Power, R.: Semi-supervised sequence tagging with bidirectional language models. arXiv:1705.00108 (2017)
10. Hashimoto, K., Xiong, C., Tsuruoka, Y., Socher, R.: A joint many-task model: growing a neural network for multiple NLP tasks. arXiv:1611.01587 (2016)
11. Belinkov, Y., Durrani, N., Dalvi, F., Sajjad, H., Glass, J.: What do neural machine translation models learn about morphology? arXiv:1704.03471 (2017)
12. Yang, Z., Salakhutdinov, R., Cohen, W.W.: Transfer learning for sequence tagging with hierarchical recurrent networks. arXiv:1703.06345 (2017)
13. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv:1603.01360 (2016)
14. Peters, M.E., et al.: Deep contextualized word representations. arXiv:1802.05365 (2018)
15. Howard, J., Sebastian, R.: Fine-tuned language models for text classification. arXiv:1801.06146 (2018)

16. Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the limits of language modeling. arXiv:1602.02410 (2016)
17. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)
18. Ruder, S.: An Overview of gradient descent optimization algorithms. arXiv:1609.04747 (2016)
19. Felbo, B., Mislove, A., Søgaard, A., Rahwan, I., Lehmann, S.: Using millions of Emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. arXiv:1708.00524 (2017)
20. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv:1607.06450 (2016)
21. Srivastava, R.K., Greff, K., Schmidhuber, J.: Training very deep networks. In: Advances in Neural Information Processing Systems, pp. 2377–2385 (2015)
22. Lafferty, J., McCallum, A., Pereira, F.C.: Conditional random fields: probabilistic models for segmenting and labeling sequence data (2001)
23. Trofimov, I.V.: Person name recognition in news articles based on the per-sons1000/1111-F collections. In: 16th All-Russian Scientific Conference Digital Libraries: Advanced Methods and Technologies, Digital Collections, RCDL 2014, pp. 217–221 (2014)
24. Gareev, R., Tkachenko, M., Solovyev, V., Simanovsky, A., Ivanov, V.: Introducing baselines for russian named entity recognition. In: Gelbukh, A. (ed.) CICLing 2013 Part I. LNCS, vol. 7816, pp. 329–342. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-37247-6_27
25. Mozharova, V., Loukachevitch, N.: Two-stage approach in Russian named entity recognition. In: Proceeding of IEEE International FRUCT Conference on Intelligence, Social Media and Web (ISMW FRUCT 2016), pp. 1–6 (2016)
26. Ivanitskiy, R., Shipilo, A., Kovriguina, L.: Russian named entities recognition and classification using distributed word and phrase representations. In: SIMBig, pp. 150–156 (2016)
27. Sysoev, A.A., Andrianov, I.A.: Named entity recognition in Russian: the power of wiki-based approach. In: Dialog Conference (2016, in Russian)
28. Malykh, V., Ozerin, A.: Reproducing Russian NER baseline quality without additional data. In: CDUD@ CLA, pp. 54–59 (2016)
29. Rubaylo, A.V., Kosenko, M.Y.: Software utilities for natural language information retrievial. Alm. Mod. Sci. Educ. **12**(114), 87–92 (2016)
30. Huang, Z., Xu, W., Yu, K.: Bidirectional LSTM-CRF models for sequence tagging. arXiv:1508.01991 (2015)
31. Tutubalina, E., Nikolenko, S.: Combination of deep recurrent neural networks and conditional random fields for extracting adverse drug reactions from user reviews. J Healthc. Eng. 2017 (2017)
32. Anh, L.T., Arkhipov, M.Y., Burtsev. M.S.: Application of a hybrid Bi-LSTM-CRF model to the task of russian named entity recognition. arXiv:1709.09686 (2017)