A Multiclass Words Classification by the Recurrent Neural Network with Memory (LSTM) as Applicable to the Named Entity Recognition Problem

Vladimir Vakurin
Tula State University
Tula, Russia
vakourinvl@yandex.ru

Andrey Kopylov
Tula State University
Tula, Russia
and.kopylov@gmail.com

Oleg Seredin
Tula State University
Tula, Russia
oseredin@yandex.ru

Konstantin Mertsalov
Rensselaer Polytechnic Institute
Troy, NY, USA
kmertsalov@gmail.com

***Abstract*—This study considers back propagation neural networks (NN) training for named entity recognition using multilayer NN architectures and various feature spaces on character strings. Experimental results showing the relation between the generalizing properties and the intersection of the training and test named entity sets while solving the conventional named entity recognition problem are presented. We also propose a method for improving the model predictive ability to recognize named entities not used in the training.**

Keywords—recurrent neural network, character feature spaces, long short-term memory architecture

I. INTRODUCTION

The paper proposes a new method and investigates the key disadvantages of the existing named entity (NE) recognition solutions. Named entity recognition is a well-known problem, a part of the text mining domain [1].

Within the text mining domain, named entity recognition is used to locate and identify identical information objects contained in the text either directly, or indirectly. The general named entity recognition (NER) problem is the identification of words/word sequences in a text that belongs to a specified group, such as company names, geographic names, proper names, etc. The problem has many specific formulations and is significant for automated text processing systems. The common problems mentioned in the available references are proper name recognition, drug name recognition (bio-NER, drug-NER) [2], and chemical entity recognition (chem-NER) [3]. Since developing syntax rules and dictionaries for such problems is difficult, and proper names and formulas often contain errors, the problems are usually solved with machine learning [3,4]. For the last three to four years, more advanced named entity recognition methods emerged. The new methods use the most advanced long short-term memory neural network architectures [5] and are extensively investigated. An application of such a neural network architecture to the Russian language is presented in [6].

A commonly used optimization method for neural network training is the stochastic gradient descend (SGD) [7]. It is iteratively controlled by a numeric loss function value [8]. On one hand, the method is based on a random distribution of changes to the neural network coefficients. It means that the model parameter vector randomly oscillates around the common path since it is updated as a new entity enters the network (with some noise relative to the generalized pattern; it is a so-called "online update", refer to [9]). With this, the expected global error minimum can be found faster [9]. On the other hand, the ground truth and the loss function should match the NN learning objective.

The problem statement for this research is improving the quality of the models used for the recognition of named entities not presented at the NN training phase by using a multiclass loss function along with a probabilistic representation of the specific named entity strings. We also present the experimental results showing the relation between the generalizing properties and the intersection of the training and test named entity sets while solving the conventional named NE recognition problem, and the extremely poor generalizing ability of such conventionally trained models when applied to texts that contain new, unknown NEs which is common in actual (commercial) NE recognition applications.

II. RELATED WORKS

There are several approaches to the named entity identification problem: grammar templates [10]; a classifier based on support vectors [11], statistical models, namely, hidden Markov models [12], conditional random fields [13, 14], and a range of deep learning NN models [15-18]. To overcome the limitations of using recurrent neural networks used for NE string prediction [15], neural network cells with long short-term memory (LSTM) were introduced [5].

The latest trend is combining various neural network architectures as layers of a top-level multilayer neural network [19]. Lately, it has been considered as deep learning. This is presented in [16]; the first results obtained with a convoluted network are shown in [17] as applied to advanced neural network architectures [18]. Despite the relatively NER solution high quality compared to the above-listed conventional methods, the researchers note a disadvantage attributed to random errors introduced to the features of an entity to be recognized. The paper [20] notes that expanding the feature space by introducing capital letters and part of speech attributes do not improve the quality. A solution that brings LSTM neural networks to a state-of-the-art level is the architectures that do not require manual feature engineering or pre-processing. Instead, they are end-to-end architectures that process character strings directly and generate a feature space with a sufficient dimensionality [20, 21, 22] for the top LSTM layers that recognize the string (containing a NE.)

The approach is supported by the paper [23]. It notes that the feature space generated by such a model can distinguish word suffixes, capitalized words, prefixes, and perform tokenization automatically. With such an approach, the NN training seems to be similar to the way people learn words: an explicit character string is matched to a test list of words hidden from the observer. It is abstract and not obvious at the initial phases of learning, but as the learning is completed, the word list contains a set of words and the rules of their usage. In this paper, we will experimentally verify if this approach is valid. We will also experimentally verify the controlled vertical addition of layers to a neural network. As the number of layers is determined by the architecture, there is a problem of representing the linear operator for multiple NN layers (applied to the NN layers considered as elements: as it would have been applied to the elements of a specific NN layer in the conventional problem formulation.) The problem is solved with such architectures as shown in [24, 25] that resulted in the emergence of highway neural networks.

III. GENERAL ARCHITECTURE OF THE PROPOSED NEURAL NETWORK

A. *Encoder Architecture*

The features are represented with a convolutional encoder [9]. The encoder input is the letter features encoded by natural numbers [21]. Each word is encoded by a vector. Its length is equal to the length of the longest word (21 letters in our experiment). The vector elements are the letter sequential numbers in the alphabet. An empty position is coded as 1.

As it is noted in [21], sequence convolutions (usually called 'time convolutions') are used to process natural language texts in contrast to spatial convolutions used to process images. For this reason, a feature representation $\mathbf{f}^k \in R^{l-w+1}$ of the neural network middle layer for the word k is generated as follows: where $\mathbf{C}^k[*, i:i+w-1]$ are columns of the $\mathbf{C}^k$ matrix from i to $i+w-1$, $\langle \mathbf{A}, \mathbf{B} \rangle = \mathbf{Tr}\left(\mathbf{A}\mathbf{B}^T\right)$ is the Frobenius scalar product.

The most significant features for each word k are to be selected from the feature vector $\mathbf{f}^k$: $\mathbf{y}^k = \max_i \mathbf{f}^k[i]$ (max-over-time) for k, located at the center of a letter window wide [21].

The most efficient method to represent the generated n-gram character sequences for a convoluted neural network is to use several such filters concurrently. The filters have various bandwidths proportional to the expected n-gram length (a word length expressed in characters.) We used the same parameters as in the paper [21]: seven filters with [50, 100, 150, 200, 200, 200, 200] dimensions. As the authors note, the key concept is to identify the most significant features for a specific n-gram input and each filter with various dimensions.

For the filters $\mathbf{H}_1, \mathrm{K}, \mathbf{H}_h$ ($h = 7$ in this case), the convoluted neural network output for a character representation is $\mathbf{y}^k = \left[y_1^k, \mathrm{K}, y_h^k\right]$ for the input representation of the word k, max. length of 21 characters. As the paper [21] specifies, for many natural language processing applications is the dimension of the output middle layer (usually between 100 and 1,000.) In our experiment, the value is 650.

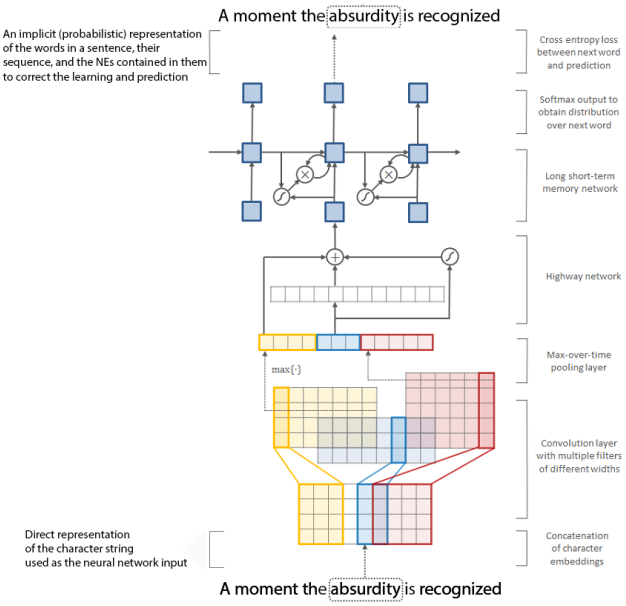

Fig. 1. The general arrangement of a char-cnn-lstm encoder based on the arrangement presented in [19].

As new sentences are supplied to the training window 100 sentences long an internal covariance shift may occur [9]. To minimize it, and to accelerate the training, we used mini-batch normalization [26].

After normalization, the convoluted encoder output can be complemented by layers with linear transfer functions and a carry gate that excludes several linear layers based on the value of the function G [24, 25]:

$$\mathbf{y} = H\left(\mathbf{x}, \mathbf{W}_H\right) \cdot G\left(\mathbf{x}, \mathbf{W}_G\right) + x \cdot \left(1 - G\left(\mathbf{x}, \mathbf{W}_G\right)\right),$$

where x is the input, $H\left(\mathbf{x}, \mathbf{W}_H\right)$ is the transform gate, $G\left(\mathbf{x}, \mathbf{W}_G\right)$ is the carry gate: $H(\mathbf{x}) = \left(\mathbf{W}_H \mathbf{x} + b_H\right)$, $G(\mathbf{x}) = \sigma\left(\mathbf{W}_G \mathbf{x} + b_G\right)$, where σ is the sigmoidal function.

We used two such layers in the experiments.

LSTM cells were applied for the sequence recognition. A layer with LSTM cells [6] replaces the NN hidden layer coefficients ($\mathbf{W}$) with a system of equations that connects the LSTM elements horizontally and enables short-term long memory (refer to Fig. 2).

B. *Decoder. Using the Estimated vs. Reference Mismatch Vector for Backpropagation*

A language model that estimates the next word probability w_{t+1} (a named entity or another word) from a character sequence $\mathbf{w} = \left[w_1, \ldots w_t\right]$ was developed as follows.

Upon every neural network weights update as new features (character strings) are presented, an error function is estimated. The error function checks the match or mismatch of the class index (the word number in the dictionary) in the training set and the estimated class index (the word number

in the dictionary) for each character string that represents the word:

$$y^* = \arg\max_{y \in Y(z)} p(y|z;\mathbf{W},b) .$$

A result of successful training is matching the character string segments being words as individual elements [23].

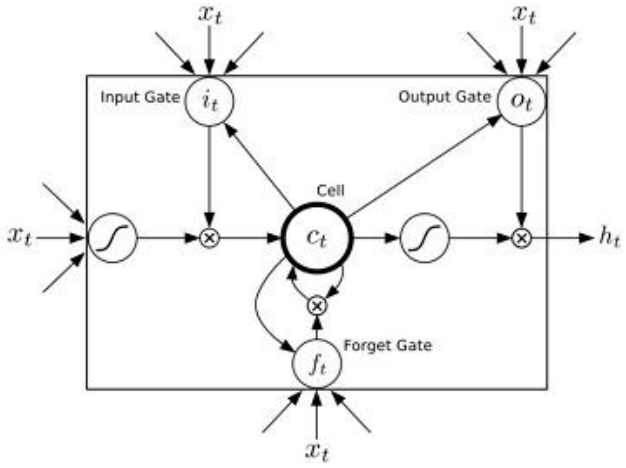

Fig. 2. A short-term long memory cell structure (*from [4]*).

Estimating a word class (or a NE class) in a sentence (text representation hidden from the NN input) as a character string containing the word is presented, or, if the prediction is wrong, a set of characters not related to the expected word is as follows. Two extra layers are added to the recurrent neural network output: a dropout layer with a 0.5 dropout probability, and a so-called linear layer with its dimension equal to the dictionary size:

$$P(\mathbf{x}) = \mathbf{W}_P\mathbf{x} + b_P .$$

In other words, the neural network output as a $S \times N$ matrix is multiplied by a $N \times T \times P$ matrix, where S is the number of sentences (100), N is the neural network output dimension, T is the number of words in the sentence (35), P is the dictionary size.

The resulting matrix contains non-normalized values of the dictionary word degree of membership to the classes recognized in the array of sentences that the neural network (not receiving the "right" term numbers directly) gets as a sequence of characters. In the course of optimization the network is trained to recognize the sequences of characters as indivisible fragments (words) and to predict each such word, and also to predict (whether correctly or erroneously) the class of an index 0 named entity.

To decrease the P dimension, we can estimate the softmax index by assigning it to the respective element of the $S \times T$ array: the index is the expected word (class) index in the dictionary used to compare the current neural network output with the referenced one.

The stochastic gradient descend (SGD) method is used to optimize the neural network layer coefficients. The SGD argument is the error value, i.e., the cross-entropy function value estimated for the probability of membership in each word of the language:

$$H(p,q) = -\sum_y p(y)\log q(y) ,$$

that is to be transformed back (with some error) into the coefficients of an LSTM recurrent neural network.

IV. EXPERIMENTAL PROCEDURE

Two language corpora were used: Penn Treebank [27] and English NER task CoNLL2003 [28]. Refer to Table 1 for their summary data. For the CoNLL2003 corpus, NE-PER (Personal, person, human) were used. To estimate the named entity recognition quality we used conventional metrics: general accuracy for all the classes, accuracy, completeness, F1-score for the first class represented by the NEs [29]. Also, refer to [30].

TABLE I. EXPERIMENTAL DATA SET STATISTICS

Dataset	Text element type	Penn Treebank	CoNLL2003
Training	Sentences	42068	14987
	Words	887.521	204.567
Validation	Sentences	3370	3466
	Words	70.390	51.578
Test	Sentences	3761	3684
	Words	78.669	46.666

V. EXPERIMENTAL PROCEDURE

A. *Experiment No.1: Standard NE recognition problem*

Refer to Fig. 3 for the test set recognition results achieved with the multiclass loss function.

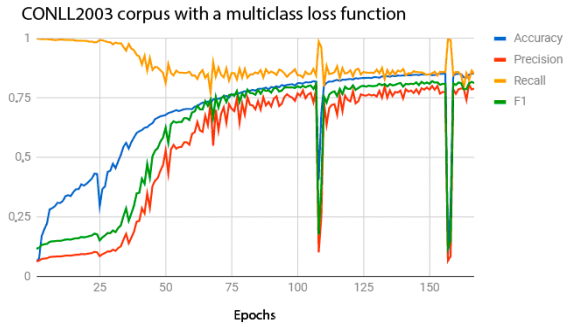

Fig. 3. CoNLL2003 test set recognition result.

B. *Experiment No.2: Random NE recognition*

Feature space for the CoNLL2003 corpus is constructed in such a way as to make the named entity character strings composed of 3 - 20 random characters for training and testing. Refer to Fig. 4 for the results.

We will further check if the experimental result is a mistake.

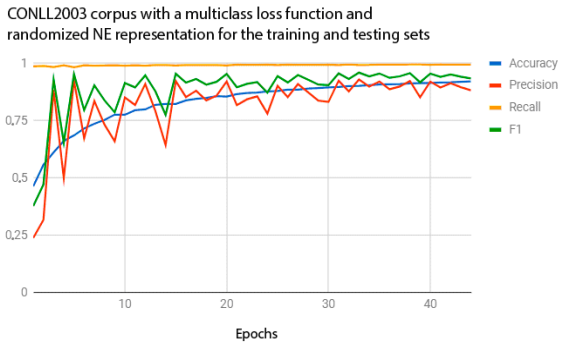

Fig. 4. The CoNLL2003 corpus test set recognition result with randomly misspelled NE character features during the training and the testing.

C. *Experiment No.3: Unique NE recognition refined problem statement*

Using the information on Chem-NER [3], we can refine the NE recognition problem with the CoNLL2003 corpus as follows: first, the NN is trained; then, it recognizes NEs not present in the training set, only in the test one. The resulting problem is more complicated: the network will be trained with the NE character features not found in the test set NEs. For this, every corpus CoNLL2003 named entity is a string composed of 3 - 20 random characters. It is transfer learning [30] for named entity recognition.

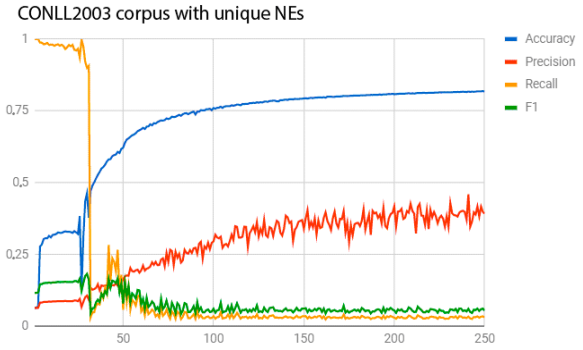

Fig. 5. The result of the CoNLL2003 corpus test recognition with the NN trained on NEs with randomly misspelled character features.

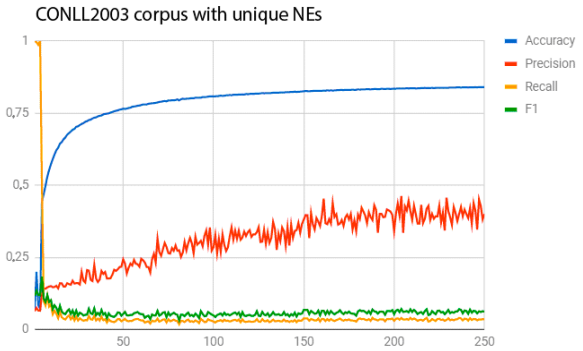

Fig. 6. The result of the CoNLL2003 corpus validation set recognition with the NN trained on NEs with randomly misspelled character features.

The results of this experiment and the previous one are controversial.

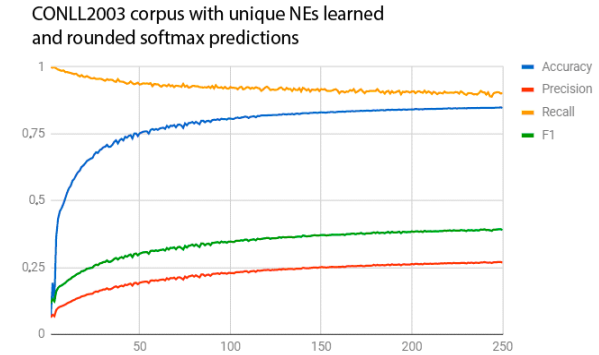

Fig. 7. The result of the CoNLL2003 corpus test recognition with the NN trained on NEs with randomly misspelled character features. The NN modified the prediction and loss functions.

D. *Experiment No.4: The algorithm adaptation for unique NE recognitions*

Using the feature space building conditions from Experiment No. 3, we will change the predictive function from the softmax class as follows: if the confidence factor in favor of at least one class is less than 50%, then class 0 (named entity) would be predicted. It means that the NN cannot recognize the unique string with a high probability:

$$y^* = \mathrm{ROUND}\left(\arg\max_{y \in Y(z)} p(y|z;\mathbf{W},b) \right)$$

In this case, while in the training the error function skips the recognition errors associated with the randomly changed NE characters.

E. *Experiment No.5: Solution verification with the Penn TreeBank corpus*

Experiment No. 4 is repeated with the Penn TreeBank corpus. The hypothesis is: with each named entity misspelled we will avoid the well-known <UNK> (unknown) character recognition problem. Every named entity is encoded by these characters. The text corpus (stock reports and financial news) is huge and homogeneous; that is why it is suitable to learn the unique named entity recognition accuracy with the method proposed in Experiment No. 4.

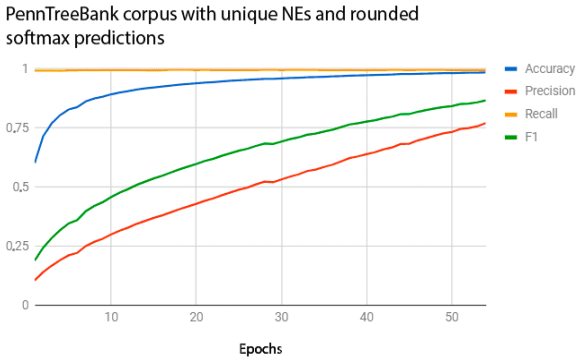

Fig. 8. The result of the PennTreeBank corpus test recognition with the NN trained on NEs with randomly misspelled character features. The NN modified the prediction and loss functions.

F. *Experiment No.6: The method improvement and the comparative metrics estimation*

During the experiments, we identified and confirmed the existence of the problem that was reviewed in [32]. Unfortunately, our team found it out too late, when experiments 1-5 had been completed. It is an independent confirmation that the problem does exist in the industry. Initially, we introduced a more radical problem statement and offered an EN representation-agnostic solution, even if the recognition quality is not perfect. Thus, to estimate the comparative characteristics, the loss function will be left as in experiments 5-6, and the convolutional encoder will get NE character strings as input. The NEs that were used in training are deleted from the test set for the quality assessment as proposed in [32]. Since gazetteers are used in [32], we also used them for this experiment. Refer to Table 2 for the comparative characteristic of this method with and without gazetteers. There are 1,500 training epochs for this model. The NE recognition target classes are Person, Organization, Location, as in [32].

TABLE II. THE QUALITY CHARACTERISTICS OF THE METHOD

Corpus	no gazetters			with gazetters		
	Prec.	Recall	F1	Prec.	Recall	F1
CONLL test A	0.56	0.78	0.649	0.59	0.75	0.657
CONLL test B	0.43	0.87	0.571	0.57	0.85	0.579

The recognition quality is higher if a NE generalized pattern is generated through training. Refer to Table 3 for the comparison of the results with [32]. Refer to Table 14 for a comparison of the results. (Table 14: Out of domain performance: F1 of NERC with different models).

TABLE III. RESULTS COMPARED TO [29]

The results	Precision	Recall	F1-score
Proposed method			
CONLL test B	0.59055	0.75364	0.6537
CONLL test A	0.44853	0.85251	0.57881
Memorization			
CONLL test B	0.5314	0.2236	0.3148
CONLL test A	0.5585	0.2249	0.3207
CRF Suite			
CONLL test A	0.6712	0.3857	0.4899
CONLL test B	0.6794	0.3641	0.4741
SENNA			
CONLL test A	0.6862	0.5868	0.6326
CONLL test B	0.6461	0.5194	0.5758

The experimental numerical results are presented in Table 4. The specified natural language models quality refers to the epoch indicated in the Table.

TABLE IV. EXPERIMENTAL RESULTS

Exp No.	Fig No.	Training epoch	General accuracy	NER precision	NER recall	F1 score
1	3	150	0.849	0.7859	0.8495	0.81
2	4	44	0.9214	0.8825	0.9950	0.934
3	5	250	0.8174	0.3921	0.0302	0.054
3	6	250	0.8401	0.4003	0.0346	0.061
4	7	250	0.8466	0.2681	0.9023	0.39
5	8	54	0.9852	0.7708	0.9943	0.866
6	--	1500	--	0.5637	0.7809	0.649

VI. RESULTS AND DISCUSSION

Interpreting Experiment No. 2 results as a success is a mistake because it contradicts Experiment No. 3 results. A possible reason for the contradiction is a feature of the tensorflow softmax software package function that processes the NN output:

- the class occurrence probability P is estimated from the NN output values with the class 0 features. The standard class index for NER-Person class is 0. The estimated probability is low, but still, it is higher than for the other n classes representing the words.
- or it assigns class index 0 (Person) if the probabilities of the term being a member of each class in the set are equal.

Nevertheless, as the classifier finds a non-random NE representation in a character string (refer to Experiment No. 3), it will assign to it an index of the class (word) that differs from the NE class but is more similar to that of another non-random word. A trivial example is: we need to recognize the proper noun: the Snowball dessert name. The NN model was trained with the names of other desserts. It was also trained with fairy tales used as counterexamples where the word Snowball represents a ball of snow for the winter game, but not the dessert.

This problem shows that the existing named entity recognition training methods have a significant disadvantage: the recognition quality depends on whether the NE lists for the training and recognition sets intersect or not. In this case, the contradiction is between the possible uniqueness of the NE representation and the statistical method of recognition applied.

These results mean that it is possible to formulate the problem of NE recognition by searching the character string that was not used while in training.

The most obvious solution for this contradiction is increasing the classifier sensitivity threshold to, e.g., 50% probability of accurate identification of previously known, standard words in a sentence. As experiments 4 and 5 show, this aim is achievable. For a big training set (Experiment 5) the recognition quality is equal to that of the non-unique NE recognition.

VII. CONCLUSIONS AND FURTHERRESEARCH

The experiments show that multilayer neural networks can be applied to named entity recognition even if the NEs greatly differ from the training set. The unique NE recognition for the CoNLL2003 corpus complex text is possible with accuracy 0.5637, completeness 0.7809, and F1-score 0.6492.

Nevertheless, the researchers should consider two different problems: the recognition of known or similar NEs, and the recognition of unknown NEs not similar to those used for the training. The paper [32] also confirms that the problem exists. Our results are comparable to those presented in [32]. Our experiments showed that the conventional substitution or a substitution refined with extra statistical data (gazetteers and additional features) can just significantly improve the recognition of known NEs (e.g., included in the dictionaries.) It is the case for the more complex, advanced accuracy improvement algorithms. The extra statistical data used in Experiment No. 6 increased F1-score by 0.7%...0.8% through reducing the recognition completeness. The achievable metrics of any new method for the conventional problem depends on the amount of intersection between the NE training set and the testing one. The recognition of general text patterns located between NEs is a more natural problem statement. We also identified an issue with the softmax function (particularly tensorflow tf.nn.softmax) as applied to NN output layer factors that represent NEs since it leads to lower accuracy.

REFERENCES

[1] A. Kao and S. Poteet, "Natural Language Processing and Text Mining," London: Springer-Verlag, 2007.

[2] J. Patrick and M. Li, "High accuracy information extraction of medication information from clinical notes: 2009 i2b2 medication extraction challenge," Journal of the American Medical Informatics Association, vol. 17, pp. 524 -527, 2010.

[3] M. Krallinger, "The CHEMDNER corpus of chemicals and drugs and its annotation principles," Journal of cheminformatics, vol. 7, no. 1, pp. 1-17 , 2015. DOI:10.1186/1758-2946-7-S1-S.

[4] A. Glazkova, "Russian Person Names Recognition Using the Hybrid Approach," Supplementary Proceedings of the Seventh International Conferencem on Analysis of Images, Social Networks and Texts (AIST), pp. 34-41, 2018.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Comput., vol. 9, no. 8, pp. 1735-1780, 1997.

[6] L. Anh, M. Arkhipov and M. Burtsev, "Application of a Hybrid Bi-LSTM-CRF model to the task of Russian Named Entity Recognition," Proceedings of the AINL, 2017.

[7] H. Robbins and S. Monro, "A Stochastic Approximation Method," The Annals of Mathematical Statistics, vol. 22, no. 3, pp. 400-407, 1951.

[8] A. Wald, "Statistical Decision Functions," Wiley, 1950.

[9] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, “Gradient based learning applied to document recognition,” Proceedings of the IEEE, pp. 2278-2324, 1998.

[10] J. Jang, “Information extraction from text,” Mining Text Data, Springer, 2012, 524 p.

[11] H. Isozaki and H. Kazawa, “Efficient support vector classifiers for named entity recognition,” Proceedings of the 19th international conference on Computational linguistics, vol. 1, pp. 1-7, 2002.

[12] G.D. Zhou and J. Su, “Named entity recognition using an hmm-based chunk tagger,” Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, pp. 473-480, 2002.

[13] R. Klinger, “Automatically selected skipedges in conditional random fields for named entity recognition,” Proceedings of the 8th International Conference on Recent Advances in Natural Language Processing, pp. 580-585, 2011.

[14] W. Chen, Y. Zhang and H. Isahara, “Chinese named entity recognition with conditional random fields,” Proceedings of the 5th Special Interest Group of Chinese Language Processing Workshop, pp. 118-121, 2006.

[15] Y. Bengio, P. Simard and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” IEEE Transactions on Neural Networks, vol. 5, pp. 157-166, 1994.

[16] A. Ivakhnenko, “Grouped Arguments Handling for Solving Prognostic Problems,” Automatics, no. 6, pp. 24-33, 1976.

[17] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard and L. Jackel, “Handwritten Digit Recognition with a Backpropagation Network,” Proceedings of NIPS, 1989.

[18] Y. Bengio, “Learning Deep Architectures for AI,” Foundations and Trends in Machine Learning, vol. 2, no. 1, pp. 1-127, 2009. DOI: 10.1561/2200000006.

[19] J. Li, A. Sun, J. Han and C. Li, “A Survey on Deep Learning for Named Entity Recognition,” IEEE Trans. Knowl. Data Eng., 2020. DOI: 10.1109/TKDE.2020.2981314.

[20] X. Ma and E. Hovy, “End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF,” Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, vol. 1, pp. 1064-1074, 2016.

[21] Y. Kim, Y. Jernite, D. Sontag and A. Rush, “Character-Aware Neural Language Models,” Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 2741-2749, 2016.

[22] M. Cho, J. Ha, C. Park and S. Park, “Combinatorial feature embedding based on CNN and LSTM for biomedical named entity recognition,” J. Biomed. Inform., vol. 103, no. 2019, 103381, 2020.

[23] J.Chiu and E. Nichols, “Named entity recognition with bidirectional lstm-cnns,” Transactions of the Association for Computational Linguistics, vol. 4, pp. 357-370, 2016.

[24] R. Srivastava, K. Greff and J. Schmidhuber, “Highway networks,” arXiv preprint: 1505.00387, 2015.

[25] G. Pundak and N. Tara, “Sainath: Highway-LSTM and Recurrent Highway Networks for Speech Recognition,” Proc. Interspeech, ISCA, 2017.

[26] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Proceedings 32nd ICML, pp. 448-456, 2015.

[27] M. Marcus, B. Santorini and M. Marcinkiewicz, “Building a large annotated corpus of English: the Penn Treebank,” Computational Linguistics, vol. 19, no. 2, pp. 313-330, 1993.

[28] E. Tjong, K. Sang and F. De Meulder, “Introduction to the conll-2003 shared task: Language independent named entity recognition,” Proceedings of CoNLL, vol. 4, pp. 142-147, 2003.

[29] C. Van, “Rijsbergen, Information Retrieval,” Butterworth-Heinemann, 1979.

[30] H. He, “Learning from imbalanced data,” IEEE Transactions on Knowledge and Data Engineering, pp. 1263-1284, 2009.

[31] L. Pratt, “Discriminability-based transfer between neural networks,” NIPS Conference: Advances in Neural Information Processing Systems 5. Morgan Kaufmann Publishers, pp. 204-211, 1993.

[32] L. Augenstein, L. Derczynski and K. Bontcheva, “Generalisation in Named Entity Recognition: A Quantitative Analysis,” Computer Speech & Language, 2017. DOI:10.1016/j.csl.2017.01.012.2017.