Bull. Nov. Comp. Center, Comp. Science, 47 (2023), 33-41
@ 2023 NCC Publisher

Comparison of ELMO-based models on the named entity recognition task

Artem Skiba, Tatiana Batura

Abstract. This paper presents a comparison of ELMO-based models. The comparison was performed on data in the Russian language for the task of named entity recognition (NER). The paper also discusses a comparison of the architectures based on the Simple Recurrent Unit (SRU) and Gated Recurrent Unit (GRU). All the models compared were trained on a corpus of news texts in the Russian language taken from the Wikinews resource and were assessed on the Russian subsets of the WikiNEuRal and MultiCoNER datasets. The datasets and original code are available at https://github.com/Abiks/distributed-semantic-models. The results obtained suggest that the SRU architecture is promising for solving the NER task and it provides a high training speed. Also, the quality of NER models based on uncontextualized char-based embeddings was found to be comparable with other models discussed herein. This highlights the advantage of using character-based embeddings as part of the RNN or a transformer-based model because of their robustness to typos. However, the architectural details of the char-based block need further research.

Keywords: neural network architectures, distributional semantic models, ELMO, SRU, GRU, named entity recognition task.

Introduction

In recent years, machine learning models, such as neural network models, have been developing actively. The same refers to models working with natural language texts, which have found applications in commercial chatbots, processing of email and search queries, spam generation and filtering, summarization of a large number of comments and reviews on products, etc.

In most cases, such models are trained using the transfer learning approach, with a model trained on a large corpus of texts used as the base.

One of the first language models capable of considering context was the ELMO (Embeddings From Language Models) architecture [1] comprising two models based on the LSTM (Long Short-Term Memory) recurrent architecture [2], one of which processed the text in the forward direction, and the other, in reverse.

Further progress in the development of vectorizers was driven by the appearance of the attention mechanism and Transformer architecture. Among recurrent architectures, one of the newest is the SRU (Simple Recurrent Unit) [3]. In their publication, the authors show that the SRU-based models are of higher quality than those based on other recurrent architectures. The comparison was performed on a sentiment analysis task. In our opinion, however, the named entity recognition (NER) task is more important as it is central to text processing. Therefore, the main contribution of this work is the comparison of models with different architectures on the task of named entity recognition for the Russian language.

1. Related work

1.1. ELMO architecture

ELMO (Embeddings From Language Models) [1], a classic neural network that implements the approach of obtaining vector representations from autoregressive language models, consists of three blocks, as shown in the figure.

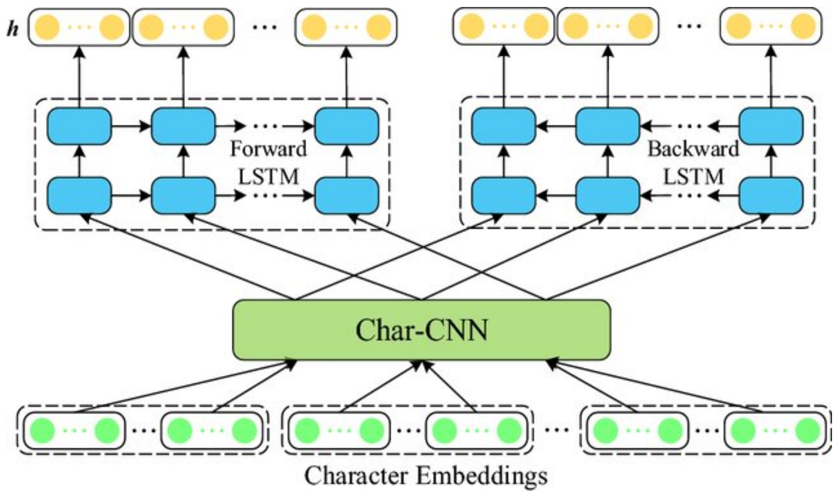

Figure. ELMO Neural Network Architecture [4]

1. Char-CNN (character-level CNN) forms preliminary vector representations of words without consideration of the surrounding context.
2. Forward LSTM processes the resulting sequence in the forward direction, adding the contextual information of all the previous words to the vector representation of each word.
3. Backward LSTM processes the resulting sequence in the reverse direction, from end to the beginning, adding the contextual information of all subsequent words to the vector representation of each word.

Both Forward LSTM and Backward LSTM consist of two LSTM layers and represent the fragments of two independent autoregressive language models, each of which is trained on the task of predicting the next input element. However, the other fragments of these language models are not included in ELMO, as they are not involved in the formation of useful vector representations of words after training is completed.

1.2. LSTM architecture

The Long Short-Term Memory (LSTM) mechanism [2] is an extension of the idea of using the information about the previous elements of the sequence processed. The approach proposed by the authors of the LSTM architecture is based on copying the human memory mechanisms: it is important to remember both some general information ("long-term memory") and specific information about previous elements ("short-term memory") during sequence processing. Thus, the LSTM implements the following approach:

$$\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) && \text{forget gate}\\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) && \text{input gate}\\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) && \text{output gate}\\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma(W_c x_t + U_c h_{t-1} + b_c) && \text{long-term memory}\\
h_t &= o_t \circ \sigma(c_t) && \text{hidden state}
\end{aligned}$$

Here, the symbol ◦ denotes the element-wise multiplication of vectors (Hadamard product).

The LSTM cell consists of a forget gate, input gate, output gate, and two vectors (c_t, h_t) for storing information about the previous elements of the sequence processed. The output of the cell used in subsequent layers is the vector h_t.

Further development of the LSTM architecture takes into account the current state of the long-term memory when it calculates the gates:

$$\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + V_f ct - 1 + b_f) && \text{forget gate}\\
i_t &= \sigma(W_i x_t + U_i h_{t-1} + V_i ct - 1 + b_i) && \text{input gate}\\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + V_o ct - 1 + b_o) && \text{output gate}\\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma(W_c x_t + U_c h_{t-1} + b_c) && \text{long-term memory}\\
h_t &= o_t \circ \sigma(c_t) && \text{hidden state}
\end{aligned}$$

Sometimes, a forget gate is used instead of an input gate. Thus, if the cell "forgets" some information, it immediately replaces it with new information:

$$\begin{aligned}
f_t &= \sigma(W_f x_t + U_f h_{t-1} + V_f ct - 1 + b_f) && \text{forget gate}\\
r_t &= \sigma(W_r x_t + U_r h_{t-1} + V_r ct - 1 + b_r) && \text{reset gate}\\
c_t &= f_t \circ c_{t-1} + (1 - f_t) \circ \sigma(W_c x_t + U_c h_{t-1} + b_c) && \text{long-term memory}\\
h_t &= r_t \circ \sigma(c_t) && \text{hidden state}
\end{aligned}$$

1.3. SRU architecture

The Simple Recurrent Unit (SRU) architecture [3] proposes a number of improvements to the LSTM. One of the LSTM's drawbacks is its weak scalability: in order to process the next element of a sequence, the model needs to wait until the processing of the previous element is completed. In the SRU, this problem is solved as follows:

$$f_t = \sigma(W_f x_t + v_f \circ c_{t-1} + b_f) \qquad \text{forget gate} \qquad (1)$$

$$c_t = f_t \circ c_{t-1} + (1 - f_t) \circ (W_c x_t) \qquad \text{memory} \qquad (2)$$

$$r_t = \sigma(W_r x_t + v_r \circ c_{t-1} + b_r) \qquad \text{reset gate} \qquad (3)$$

$$h_t = r_t \circ \sigma(c_t) + (1 - r_t) \circ x_t \qquad \text{hidden state} \qquad (4)$$

1. The vectors of parameters and element-wise multiplication are used instead of weight matrices when calculating the contribution of the vector to the internal memory, as shown in equations (1) and (2).
2. Equations (1) and (2) describe the mechanism of lightweight recurrence allowing the calculation of the internal memory vectors for the entire input sequence without waiting until the processing of the previous elements is completed.
3. Equations (3) and (4) describe the mechanism of additive modification of input data (highway), in which the network output is mixed with its input, allowing the training of the deeper architectures of neural networks.

These differences allow for a much higher speed of the SRU model compared to the LSTM: $O(L \cdot B \cdot d)$ for the SRU versus $O(L \cdot B \cdot d^2)$ for the LSTM (where B is the batch size, L is the sequence length, d is the internal dimension of the block).

Before describing the experiments with different architectures, let us briefly discuss the datasets that were used for training and evaluating the models.

1.4. GRU architecture

In comparison with the LSTM, the GRU (Gated Recurrent Unit) [5] neural network architecture is simpler and has fewer parameters: it does not have an output gate, and the input gate and forget gate are combined into the update gate.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \qquad \text{update gate}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \qquad \text{reset gate}$$

$$\hat{h}_t = \phi(W_h x_t + U_h (r_t \circ h_{t-1}) + b_h) \qquad \text{candidate activation vector}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \hat{h}_t \qquad \text{hidden state}$$

The advantage of the GRU compared to the LSTM is its faster training owing to a simpler architecture and, hence, fewer different operations. At the same time, the GRU networks can perform better than the LSTM on the same data [6].

2. Description of model training and comparison on datasets

2.1. Data for model training

To train language models, a Russian-language corpus of news texts was collected from the Wikinews resource[1]. The following preprocessing steps were performed:

1. All texts were divided into individual sentences using the razdel library[2].
2. Sentences longer than 200 words were discarded.
3. Short sentences were combined to obtain the longest possible text fragment not exceeding 200 words.
4. All whitespace characters (tabulation, line breaks, etc.) were replaced with regular spaces.
5. All texts were converted to lowercase.
6. Lemmatization of each word was performed using the UdPipe library [7].

The total size of the resulting corpus is 8.8 million texts and 294 million tokens.

2.2. Data for model comparison

The comparison of models was performed on the Russian subsets of the WikiNEuRal and MultiCoNER datasets.

The WikiNEuRal dataset [8] was created by collecting and processing the information from Wikipedia using the additional BabelNet knowledge base [9]. The dataset includes collections in German, English, Spanish, French, Italian, Dutch, Polish, Portuguese and Russian languages. The Russian part consists of 123,000 texts divided into the training, validation, and test collections in the ratio 8:1:1.

An interesting feature of the MultiCoNER dataset [10] is a very large size of the test part and a very small percentage of matching entities in the training and test parts. Additionally, the test part includes examples from mixed domains, such as search queries, which are not present in the training part. The dataset includes texts in 11 languages: Bengali, Dutch, German, Chinese, Korean, Turkish, Russian, English, Spanish, Farsi, and Hindi. Paper [11] describes some experiments with the Russian part of MultiCoNER, though there are many unexplored questions in the solution of the named entity recognition task on such datasets.

Both datasets are annotated in the BIO (begin-inside-out) format, where each word in the text is assigned with a label indicating whether it is part of an entity or not. The first word of an entity is labeled as "B-" + "entity type name", and all subsequent words are labeled "I-" + "entity type name". All other words not participating in any entity are labeled "O". This type of annotation is one of the most commonly used. Table 1 shows the sizes of the training, validation, and test parts for WikiNEuRal and MultiCoNER.

[1] https://www.wikinews.org/

[2] https://github.com/natasha/razdel

Table 1. Split of WikiNEuRal and MultiCoNER datasets

Dataset	Training Size	Validation Size	Test Size
WikiNEuRal	92 352	11 544	11 544
MultiCoNER	15 513	813	218 958

3. Comparison of ELMO-based models

To conduct an experiment on data in the Russian language, the ELMO-Taiga model used the Char-CNN vectorization block. Initially, the ELMO-Taiga model was trained on the Taiga corpus as part of the RusVectores project [12]. The weights of the vectorization block did not change during the training. The internal dimensions of both recurrent blocks were fixed at 1024. This model was trained for one epoch on the Wikinews text corpus described in the previous section.
Separate models were trained for testing the named entity recognition without changing weights during training. The two-layer architecture (LSTM+CRF) was chosen as the architecture for the named entity recognition models. For the GRU, the built-in implementation of the Fully Gated Unit of the Pytorch library[3] was used. Each of these models was trained for 20 epochs. Table 2 shows the results of the comparison of the architectures with the LSTM, SRU, and GRU. Their quality was compared using the token-level F1 score, meaning that the named entity recognition task is treated as a multiclass classification task of BIO (Begin, Input, Out) labels on individual tokens.

Table 2. The results on Russian subsets of WikiNEuRal and MultiCoNER

Architecture	Number of layers	Number of parameters (million)	F1 WikiNEuRal	F1 MultiCoNER
CharCNN +LSTM(2lrs)	2	30	0.685	0.348
CharCNN +GRU(2lrs)	2	22.6	0.693	0.402
CharCNN +SRU(2lrs)	2	11.1	**0.763**	0.456
CharCNN +SRU(4lrs)	4	23.7	0.753	**0.458**

[3] https://pytorch.org/

CharCNN	0	0	0.755	0.456

These results are in line with those presented in [3]. An interesting result is the low quality of the LSTM and GRU architectures, which is even lower than that of the models with non-contextual vector representations using the Char-CNN block. A possible reason is that these architectures do not directly imply the residual connection, unlike the SRU; therefore, they cannot learn quickly to use good vector representations coming to them as input.

Another conclusion from this experiment is that the quality of the SRU model with 4 layers is lower than the quality of the model with 2 layers, which rather indicates a more limited training time than the disadvantage of having a larger number of layers. In the original paper [3], the authors showed that the best quality was achieved with 12 layers of the SRU.

Additionally, an experiment was conducted to compare the ELMO and SRU models with 12 layers. The training time for the SRU model was limited (approximately 24 hours) and the available resource was 1 graphics card Nvidia 4090. The results are shown in the table below.

Table 3. The comparison of ELMO-Taiga and ELMO-SRU models

Architecture	# RNN layers	Internal dimension of RNN block	# parameters of RNN block	# of training epochs	Training corpus size	F1 WikiNEuRal	F1 MultiCoNER
ELMO-Taiga	2	2048	75.6	3	~5000	0.781	**0.505**
ELMO-SRU	12	1024	74.1	3	~300	**0.792**	0.465

We can see that the ELMO-SRU model performed better than the ELMO-Taiga model on the WikiNEuRal dataset in terms of F1 score. However, the performance of the ELMO-SRU model is worse on the MultiCoNER dataset. This may be due not only to the size of the training corpus but also to the data structure of the dataset. A more detailed interpretation of the results requires additional research.

Conclusion

The experiments conducted demonstrate the potential of the SRU architecture as a recurrent architecture of distributional semantic models for solving the named entity recognition task. Another important quality of the ELMO-SRU architecture is its high training speed, which allows experiments with different architecture variants in a short time. However, current results also reveal the limitations of the experimental design: deep architectures in one epoch of training fail to achieve the same quality as smaller ones.

The experiments demonstrate, on the one hand, the importance of the vectorization blockand, on the other hand, the need for its improvement and modernization. In distributional language models based on the SRU architecture, the computation time of this block takes 35% of the total model runtime, while the block itself accounts for approximately 19% of the entire model. Such resource consumption appears suboptimal. The CharCNN block essentially solves the problem of encoding a sequence of characters of arbitrary length into a single vector. However, the architecture of this block was developed quite a long time ago, even before the Transformer architecture. Therefore, it can be improved using modern architectures and approaches, which we intend to do in the future.

References

[1] Peters M. et al. Deep contextualized word representations // Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Proc. — 2018. — Vol. 1 (Long Papers). — P. 2227–2237.

[2] Hochreiter S., Schmidhuber J. Long short-term memory // Neural Computation. — 1997. — Vol. 9 (8). — P. 1735–1780.

[3] Lei T. et al. Simple recurrent units for highly parallelizable recurrence // Empirical Methods in Natural Language Processing. Proc. EMNLP. — 2018. — P. 4470–4481.

[4] Wang Y., Hou Y., Che W. et al. From static to dynamic word representations: a survey // International Journal of Machine Learning and Cybernetics. — 2020. — Vol. 11. — P. 1611–1630. DOI: 10.1007/s13042-020-01069-8.

[5] Cho K., Van Merrienboer B., Gulcehre C., Bahdanau D., Bougares F., Schwenk H., Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. — 2014. — (Preprint / arXiv; 1406.1078).

[6] Yamak P.T., Yujian L., Gadosey P.K. A comparison between ARIMA, LSTM, and GRU for time series forecasting // Proc. 2nd International Conference on Algorithms, Computing and Artificial Intelligence. — 2019. — P. 49-55.

[7] Straka M. UDPipe 2.0 prototype at CoNLL 2018 UD shared task // Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies. Proc. CoNLL-2018. — Brussels, Belgium. — 2018. — P. 197–207. DOI: 10.18653/v1/K18-2020.

[8] Tedeschi S. et al. WikiNEuRal: Combined neural and knowledge-based silver data creation for multilingual NER // Findings of the Association for Computational Linguistics. Proc. EMNLP-2021. — 2021. — P. 2521–2533.

[9] Navigli R., Ponzetto S.P. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network // Artificial Intelligence. — 2012. — Vol. 193. — P. 217–250.

[10] Malmasi S. et al. MultiCoNER: A large-scale multilingual dataset for complex named entity recognition // 29th International Conference on Computational Linguistics. Proc. — 2022. — P. 3798–3809.

[11] Miftahova A., Pugachev A., Skiba A., Artemova E., Batura T., Braslavski P., Ivanov V. NamedEntityRangers at SemEval-2022 Task 11: Transformer-based approaches for multilingual complex named entity recognition // 16th International Workshop on Semantic Evaluation. Proc. SemEval-2022. — 2022. — P. 1570-1575.

[12] Kutuzov A., Kuzmenko E. WebVectors: A toolkit for building Web interfaces for vector semantic models // Analysis of Images, Social Networks and Texts: 5th International Conference. Proc. AIST-2016. — Yekaterinburg, Russia, April 7-9. — 2017. — P. 155–161. — (Revised Selected Papers. Cham: Springer International Publishing).