Federated Learning in Named Entity Recognition

Efim Luboshnikov\textsuperscript{1(✉)} and Ilya Makarov\textsuperscript{1,2(✉)}

\textsuperscript{1} HSE University, Moscow, Russia
eaaluboshnikov@edu.hse.ru, iamakarov@hse.ru
\textsuperscript{2} Big Data Research Center, National University of Science and Technology MISIS, Moscow, Russia

Abstract. This article is devoted to the implementation of the federated approach to named entity recognition. The novel federated approach is designed to solve data privacy issues. The classic BiLSTM-CNNs-CRF and its modifications trained on a single machine are taken as baseline. Federated training is conducted for them. Influence of use of pretrained embedding, use of various blocks of architecture on training and quality of final model is considered. Besides, other important questions arising in practice are considered and solved, for example, creation of distributed private dictionaries, selection of base model for federated learning.

Keywords: Federated learning · Federated averaging · Named entity recognition · BiLSTM-CNNs-CRF

1 Introduction

In response to the recent increase in incidents involving leaks, illegal use and sale of client data, governments are creating and enacting laws to protect the rights of clients, both people and companies. Such laws impose significant restrictions on the transfer, storage and processing of customer data. Most of the solutions based on machine learning in business are built using client data. Companies directly use the data for training and model validation. With the adoption of laws, it will become more difficult, if not impossible, to continue implementing models.

One possible solution is to use federated learning. The main idea is to share the resources of many independent parts and then benefit all participants. Federated learning is called distributed training of the general model, where data is always stored on clients during training. Despite its recent appearance, federated learning is already being applied in various areas. Projects such as Google Keyboard, joint medical and financial research are good examples. In all these projects, private data was successfully and safely used for model training and evaluation.

This work is devoted to solving the practical problem of extracting named entities in a federated approach. This task more often arises as a subtask of more
complex, for example, machine translation or question answering. Most studies in the field of federated learning use multi-layer perceptron, the simplest CNN, which is of little practical interest. NLP tasks are more difficult to implement because of the complexity of general, secure preprocessing and preparation for learning, for example, creating common dictionaries. Besides, the models themselves have much more complicated architecture. In this study, we will try to find answers to several questions in this complex task and take a step towards safe and effective learning.

2 Related Work

This work has two logical parts: federated learning and named entity recognition. Analysis of related work, current approaches, benchmarks and results also has two semantic parts. Therefore, at the beginning of this part, we will give an overview of the NER area, and then we will move on to the issue of federated learning, which is the main interest of the work.

**Named Entity Recognition.** Named Entity Recognition (NER) is a machine learning task that consists of identifying and segmenting named objects, then classifying or classifying them according to various predefined classes. This task appears both independently and as a subtask in other tasks, for example: text understanding, information retrieval, automatic text summarization, question answering, natural language understanding, knowledge base construction.

New state-of-the-art results are obtained with the help of deep learning approaches. According to the article [1] there are three key aspects of deep learning success in this area: complex non-linear mapping between network input and output, automatic features creation and end-to-end approach.

Most neural network architectures for this task have a 3-step structure. In the first stage, we get a good representation of input. Not only the word itself is important in the sentence, but also the context surrounding it. So, CNN, RNN, LM or Transformer are used for this purpose in the second step. In the final step, we use decoder to get the predictions themselves - tags for words. By convention, we can call these three steps as input, encoder, decoder.

The work BiLSTM-CNNs-CRF [2] was introduced in 2016 and marked a new trend in the solution. So, BiLSTM-CRF is the most common architecture now. The latest SOTA results were achieved with really huge models. The Bi-directional Transformer in a cloze-style manner [3] reaches (93.5%) on the CoNLL03 dataset. Instead of treating the NER problem as a sequence marking problem, the article [4] suggests formulating it as a machine reader’s understanding problem (MRC). As result, models achieve significant amount of performance boost over current SOTA models on nested NER datasets for chinese, english languages. The success of NER systems strongly depends on input representation. The use of pre-trained language model embeddings is prevalent in new developments and research.
Federated Learning. The growing popularity of federated learning is primarily due to restrictions on the use of data. GDPR in European Union [5], PDPA in Singapore [6], CCPA [7] in the USA and many other laws and regulations limit the use and transfer of client data. Those who work with this data have to look for solutions. Using the full amount of distributed, private data for training would allow for better models, new tasks, and ultimately, better lives.

The main idea of federativity is cooperation of many independent participants. Federated systems include concepts of federated learning, federated database systems [8] and federated cloud [9]. The idea of federativity is inherent in all of them, but there are some peculiarities. In federated databases there is a focus on distributed data management, in federated cloud there is a focus on resource planning, in federated learning systems (FLS) there is a secure computation between all components of the system.

Research in this area has just begun, so in most works do not experiment with the choice of the optimizer. Default stochastic gradient descent has theoretical grounds and is well researched. Strategies of averaging local model improvements into global ones are studied much more. The simplest and most reliable way is to average local weights of the model changed after local training, the so-called FedAvg. In the article [10] the authors of the algorithm investigate the question of improving the quality of the global model with a federated approach on very simple models. The authors put forward two hypotheses to improve the quality of the global model:

1. Increase the number of epochs on local clients.
2. Increase the number of transfers between local clients and the central server where the global model is stored.

Logical prerequisites are justified by experiments. The authors study optimal parameters, as the increase in the number of epochs leads to the increase in the cost of local calculations, and the growth of transfers leads to an increase in the cost of communication activities.

In works [11,12] the authors, relying on FedAvg, achieve better results with changing the target function for optimization. In FedProx, changes in the local loss function are limited, this approach has theoretical grounds. In the Agnostic FL method, a centralized model optimized for any target class distribution is trained. The main problem for practical application of federated training is efficiency. In [13] authors reduce the size of the reference at each iteration. The work [14] is devoted to finding the necessary number of communication rounds. Trade-off between local computations and weight exchange is still unexplored.

Only one article [15] was found on the subject of this study. Team of authors claim to have created a comprehensive model containing two modules. One of the modules is taught in a federated approach, respectively, after graduation trained weights and coefficients are common to all clients. The second module is trained on local data only. Thus, they offer to use the advantage of federated training to obtain aggregated information on private data distributed among clients, and additional training on local data will allow to take into account the context of a particular client. Unfortunately, this article has no source code.
and was published quite recently, so there is no possibility to judge about the reliability of the results.

3 Dataset and Metrics

The CoNLL03 dataset [16] is based on common themes and has a split into train (14041), valid (3250), test (3453). There are four entities: LOC (location), ORG (organization), PER (person) and MISC (miscellaneous). Original tagging scheme is BIO, but classic experiments results with IOBES is better. So, in all our experiments we use IOBES.

In most cases, accurate estimates are used for NER tasks. In extracting entities, there are two subtasks that need to be solved: defining exactly the boundaries and type of entity. For accurate estimates, only an extraction where boundaries and type are correctly marked is correct. Similar to the classification tasks, the concepts True Positive, False Positive, False Negative, True Negative are introduced. The obtained values are used to calculate metrics: precision, recall, F-measure. These metrics are defined for each class separately. In practical tasks it is always more convenient to focus on one number as a measure of quality. For this purpose, different types of F-measure averaging are used. In macro average we count F-score for each tag, and then take the arithmetic mean. We use micro-average, so, we sum up TP, FP, FN, TN all over the body without tags, and then take the average.

4 Baseline Model

Bi-directional LSTM-CNNs-CRF [2] is an strong baseline for NER tasks. There is no special preprocessing, feature-engineering in the model. This model can be trained for any sequence labeling task. Common NER solution are based on three-step architecture as shown on Fig. 1. At the beginning of the model, convolutional layers are used to obtain information from words at the symbol level. Then we combine word and character based views into one common one and send it to BiLSTM. With RNN, we aggregate context information from specific words and characters. BiLSTM outputs are fed into a random field to account the correlation of neighbor labels in order to obtain the most likely sequence of labels for the input sentence.

![Fig. 1. Typical three-step architecture for NER. Used blocks are shown in blue. (Color figure online)](image-url)
For the baseline word vectors were trained from scratch, not used character level features. Parameters were initialized in the same way and were optimized with stochastic gradient descent (SGD). We set initial learning rate $\eta_0 = 0.015$, momentum $m = 0.9$. Also, we used multiplicative learning rate scheduler with factor $f = 0.95$. The model achieves 85.33% F1 quality for NER on CoNLL03 dataset. In this article [17] more than 50,000 experiments have been conducted to select the optimal parameters. The value we have obtained corresponds to the distribution of the test F1 for the corresponding set of model and optimizer hyperparameters.

5 Federated Approach

The goal of the series of experiments was to train the classical BiLSTM-CNNs-CRF and its modifications in a federated approach, to identify and find a solution to the problems encountered. Different metrics can serve as optimization functions in the implementation of the federated approach. In examples of model training on mobile devices, model weight and computational complexity are crucial. There are cases when the server cannot regularly receive model updates or send out a new version. Then it is critical to understand the minimum number of clients or their share required for training convergence. The experience of using federated training in the NER task speaks more about other problems.

Examples of applications are: training on customer medical records, medical records, payment documents, customer service calls. In all such cases, the distributed training will be conducted by the software provider or an independent participant for the common good of all participants in the training. This is why we do not consider improper motives or playing by the participants. Also, there is no question of communication, as there is constant communication between participants in such cases. The main issue in our research was cost reduction, which is primarily related to the learning time of the distributed model. For these reasons and for narrowing the space of the experiment, the number of users and tuned number of local epochs were fixed.

Model Capacity. Federated learning is often used when data is private. But a company may have some data for different reasons: open data, a special agreement with some customers to collect data, a way to ensure that no private data is collected. Suppose we have this situation and we have dataset. Before implementing the federated approach, it is necessary to check its appropriateness. We should proceed from the assumption that the quality of the model under the federated approach tends to the quality of the model trained on the whole dataset, which is now distributed. It is suggested that a simple sketch should be made to find out whether adding data will improve the quality of the current model. For this purpose, we will break down the data into a small number of identical parts. Then, we will train the model in 1 part and check the quality in the deferred sample. Then, we will train the model in 2 parts and evaluate it again. And so on, before we train and validate the model on 100% of available data.
Let us look at possible results obtained for two different models in such an experiment in the Fig. 2. In the left side we see that adding data will most likely not lead to an increase in quality. Therefore, for such a model, it makes no sense to build federated training on clients. On the right picture, we can see that the quality of the model grows with the data quantity. Therefore, adding even more data may be reasonable. Such a model should be trained in a federated approach, because adding data would rather improve its quality.

There are two things we should note. First, the experiment should be conducted for each new model or architecture configuration, since each model has its own capacity. If the model has a large capacity, it needs more data to achieve a plateau of quality on validation. Models with low capacity learn fewer patterns from data, so they need less data. Second, by extrapolating model quality growth on more data, we involuntarily assume that data will come from a similar distribution, which of course is not always done. This point should be taken into account and have at least an approximate estimate of the nature of data on distributed clients.

**Distributed Data Preparation.** In the process of teaching one model on the whole dataset, most preprocessing steps are simple and even routine. With a federated approach, training becomes somewhat more difficult. Obviously, the initial data on all clients should be processed in the same way, i.e. we need the same preprocessing, creation of fiche, pretrained embedding if necessary. When creating NLP models, we often use dictionaries. Such dictionaries help to map tensor representation in the model (embedding) and real token or symbol. In the baseline model we also used such dictionaries for tokens and symbols. To create such a dictionary, we have collected tokens in the dataset and assigned them unique numbers corresponding to their embedding in the model.

Now, as part of a federated approach, we can not collect tokens from all clients, so it would violate the conditions of privacy. To create a single dictionary with a privacy condition, we can implement the client-central server interaction scheme shown in this Fig. 3. Every user after this procedure will be able to transform raw data to the model correct input.
Remote secure mapping. 

1. Each client preprocesses data and makes a frequency dictionary of tokens.
2. Client hashes the keys in such dictionary and saves the pairs \{token : hash\} with itself. Then all clients send hashed dictionaries to the server.
3. On the central server we sum up all the dictionaries and create a single mapping by selecting the desired minimum frequency of token. Also, we add to the mapping special tokens, for example <unk>.
4. Send the mapping to all clients. Each client now has two dictionaries: its own \{token : hash\} and common \{hash : token_id\}. Applying one by one dictionary, each user will be able to get the correct input for the distributed model even for those tokens that were not in his original sample.

5.1 Experiments

Federated Model Training. In all our experiments with federated approach we will compare learning process and final model performance with baseline. We will estimate overall time to reach baseline model performance, number of global communication rounds, local training epochs in process. Also, we will check influence of different factors to model training, for example, usage of pretrained word vectors, neural model blocks.

During the experiments on federated model training for the NER, the quality of the baseline model was achieved. Obviously, this was longer than usual model training. The training time of the federated model has exceeded 2.58 times the training time of a single instance. The cost of model training is linearly related to the duration of training, so we measured and compared the duration of training in all experiments with the conventional and federated approach. It should be understood that with distributed parallel learning, the real learning speed is certainly higher. It is proportional to the number of users without taking into account the time spent. But to calculate the cost of training, you must use the total time spent on all the clients, which we have received.

Learning Process and Results. As already mentioned, the number of local epochs and users has been fixed in order to reduce the space of experiments. We take 10 for both parameters. The data divided equally between users. The division is fixed in all experiments. We conducted several experiments on initializing local
optimizer. It was decided to initialize the local optimizer according to the number of passed global training rounds. The scheme of updating the model weights was taken from the article [10]. This scheme consists in simple taking the arithmetic mean of weights of all models that have locally learned the same number of epochs on the given clients. After averaging, we distribute the averaged model to the participants again. So the learning process lasts to the required quality. On the Fig. 4 we can see the average value of loss for clients during the training. We can see that every tenth local epoch of loss increases because the local model is replaced by a global model. Every round of local training epochs we overfit on local data, but in general we reduce losses as we learn, reaching the right quality.

![Fig. 4. Local losses. Mean value and 3 sigma interval for user distribution.](image)

![Fig. 5. Baseline and federated learning process.](image)
The received training process is shown in Fig. 5. The training time of the federated model has exceeded 2.58 times the one-instance training time. It should be understood that with distributed parallel learning, the real learning speed is certainly higher. It is proportional to the number of users without taking into account the time spent. But to calculate the cost of training, we must use the total time spent on all the clients, which we have received.

**Pretrained Word Embeddings.** It is well known that the addition of pretrained embeddings reduces the training duration of models, and often increases the final quality. Nowadays, most modern models for NER are obtained using BERT, ELMO, their modifications as well as large language models. The addition of pretrained embeddings to BiLSTM-CNNs-CRF also leads to an increase in quality. As you can see from the previous section, the implementation of training in a federated approach leads to a multiple increase in training duration. Use of pretrained vectors could speed up the learning process. Two experiments were conducted to test this hypothesis. BiLSTM-CNNs-CRF was taken as a baseline, word embedding was initialized by pretrained GloVe-100 vectors. Word embeddings were left as a variable parameter.

The results of the experiment are shown in Table 1 and the training graph in Fig. 6a. The learning time has diminished 2.19 times for one-instance and increased 1.64 times for federated. This is due to the fact that the model was better trained to quality, and recent epochs took a long time. For an honest assessment, it is necessary to compare the learning time at a certain quality level. This is done in Table 1. In third line there we can see how various model modifications achieve the quality of the initial level (85.33%) by learning the federated approach. We see that the use of pretrained vectors and convolutional blocks allows us to significantly increase the learning speed. Modification BiLSTM-CNNs-CRF is the fastest. The final quality increased by 4.67%. This allows us to speak about the high efficiency of the use of pretrained embeddings for federated learning.

**Char Embeddings.** The next experiment was not followed by a preliminary hypothesis or an expected result. The aim of the experiment was to compare the behaviour of convolutional and recurrent architectures for processing symbolic embedding in a federated approach. It was the block of symbolic embedding processing in the initial architecture that we changed during the experiment. RNN at its time was a breakthrough in sequence processing. But subsequent results of CNN application in the text allowed us to consider convolutional layers as an alternative. First of all, it is related to the speed of learning. In our experiments, learning speed is almost the most important criterion.

BiLSTM-CRF with pretrained embedding were used, also, CNN or RNN was used as a character processing block. Use of recurrent block to process symbolic text representation allowed to increase quality by 0.32% in valid Fig. 6b. In the course of this experiment, we did not get significant differences in the quality of the model, which cannot be said about the learning rate. The use of CNN allowed
us to increase the training speed by 4% and 19% respectively for single-instance and federated approaches.

5.2 Results

In Table 1, the main results are the last 3 rows. For ease of comparison, we have scaled all experiments against the baseline experiment. The first line shows the training time for different modifications of the model. In the second line, we see how much longer training time increases when you move from one-instance to federated learning. This data is not so representative, because we need to remember that the final quality of models with pretrained embedding is higher, so they learn longer.

In order to estimate how quickly the models are trained to a specific level, the third line were created. In this line, we see how long various modifications of the models achieve the quality of baseline (85.33%) by learning the federated approach. We see that the use of pretrained vectors and convolutional blocks allows us to significantly increase the learning speed. At the same time, we should understand that the final quality of modification with convolutional is little worse to that with recurrent blocks.

Line 4 in Table 1 shows how time costs increase when moving from baseline one-instance to federated learning. The last line shows the $F_1$ value on the validation dataset. If we speak about the final quality, the models with the use of pretrained vectors have bypassed those trained from scratch by 5%.
Table 1. Comparison of learning time. Symbol $x$ means 5.97 h on GCP Instance (8CPU, 64 GB)

|                  | CNN, scratch | CNN, GloVe | LSTM, GloVe |
|------------------|--------------|------------|-------------|
| Baseline         | x            | 0.44x      | 0.46x       |
| Federated        | 2.58x        | 4.23x      | 5.23x       |
| Federated, 85%   | x            | 1.4x       | 1.75x       |
| Time increases   | 2.58         | 3.18       | 3.8         |
| F1, valid        | 85.33        | 90.00      | 90.32       |

6 Conclusion

In this work, a full process of preparation and training in a federated approach was carried out. A solution was found to the problems encountered. Such conclusions can be drawn for practical use.

– Before implementing a federated approach, it is necessary to conduct tests for the capacity module and, if possible, find out the nature of data distribution among the participants of training.
– For the model’s auxiliary dictionaries, you can use the scheme described in this article before you start training.
– It is better to use pretrained embeddings and convolutional blocks to increase the learning speed. In case of federated learning is even more important than in case of single machine learning.

Acknowledgments. The research was supported by the Russian Science Foundation grant 19-11-00281.

References

1. Li, J., Sun, A., Han, J., Li, C.: A survey on deep learning for named entity recognition. IEEE Trans. Knowl. Data Eng. 1 (2020). https://doi.org/10.1109/TKDE.2020.2981314
2. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354 (2016)
3. Baevski, A., Edunov, S., Liu, Y., Zettlemoyer, L., Auli, M.: Cloze-driven pretraining of self-attention networks. arXiv preprint arXiv:1903.07785 (2019)
4. Li, X., Feng, J., Meng, Y., Han, Q., Wu, F., Li, J.: A unified MRC framework for named entity recognition. arXiv preprint arXiv:1910.11476 (2019)
5. Voigt, P., Von dem Bussche, A.: The EU general data protection regulation (GDPR). A Practical Guide, 1st edn. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57959-7
6. Chik, W.B.: The Singapore Personal Data Protection Act and an assessment of future trends in data privacy reform. Comput. Law Secur. Rev. 29(5), 554–575 (2013)
7. Pardau, S.L.: The California consumer privacy act: towards a European-style privacy regime in the United States. J. Tech. L. & Pol’y 23, 68 (2018)
8. Sheth, A.P., Larson, J.A.: Federated database systems for managing distributed, heterogeneous, and autonomous databases. ACM Comput. Surv. (CSUR) 22(3), 183–236 (1990)
9. Kurze, T., Klems, M., Bermbach, D., Lenk, A., Tai, S., Kunze, M.: Cloud federation. In: Cloud Computing 2011, pp. 32–38 (2011)
10. McMahan, H.B., Moore, E., Ramage, D., Hampson, S., et al.: Communication-efficient learning of deep networks from decentralized data. arXiv preprint arXiv:1602.05629 (2016)
11. Mohri, M., Sivek, G., Suresh, A.T.: Agnostic federated learning. arXiv preprint arXiv:1902.00146 (2019)
12. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018)
13. Konečnỳ, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency. arXiv preprint arXiv:1610.05492 (2016)
14. Zhu, H., Jin, Y.: Multi-objective evolutionary federated learning. IEEE Trans. Neural Netw. Learn. Syst. 31(4), 1310–1322 (2020)
15. Ge, S., Wu, F., Wu, C., Qi, T., Huang, Y., Xie, X.: FedNER: medical named entity recognition with federated learning. arXiv preprint arXiv:2003.09288 (2020)
16. Sang, E.F., De Meulder, F.: Introduction to the CoNLL-2003 shared task: language-independent named entity recognition. arXiv preprint arXiv:cs/0306050 (2003)
17. Reimers, N., Gurevych, I.: Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. arXiv preprint arXiv:1707.06799 (2017)