Original Paper

Privacy-Preserving Deep Learning for the Detection of Protected Health Information in Real-World Data: Comparative Evaluation

Sven Festag[1,2]; Cord Spreckelsen[1,2]

[1]Department of Medical Informatics, Medical Faculty, RWTH Aachen University, Aachen, Germany

[2]Institute of Medical Statistics, Computer and Data Sciences, Jena University Hospital, Jena, Germany

Corresponding Author:
Cord Spreckelsen
Institute of Medical Statistics, Computer and Data Sciences
Jena University Hospital
Bachstraße 18
Jena,
Germany
Phone: 49 3641 9 398360
Fax: 49 3641 9 396952
Email: Cord.Spreckelsen@med.uni-jena.de

Abstract

Background: Collaborative privacy-preserving training methods allow for the integration of locally stored private data sets into machine learning approaches while ensuring confidentiality and nondisclosure.

Objective: In this work we assess the performance of a state-of-the-art neural network approach for the detection of protected health information in texts trained in a collaborative privacy-preserving way.

Methods: The training adopts distributed selective stochastic gradient descent (ie, it works by exchanging local learning results achieved on private data sets). Five networks were trained on separated real-world clinical data sets by using the privacy-protecting protocol. In total, the data sets contain 1304 real longitudinal patient records for 296 patients.

Results: These networks reached a mean F1 value of 0.955. The gold standard centralized training that is based on the union of all sets and does not take data security into consideration reaches a final value of 0.962.

Conclusions: Using real-world clinical data, our study shows that detection of protected health information can be secured by collaborative privacy-preserving training. In general, the approach shows the feasibility of deep learning on distributed and confidential clinical data while ensuring data protection.

(JMIR Form Res 2020;4(5):e14064) doi: 10.2196/14064

KEYWORDS

privacy-preserving protocols; neural networks; health informatics; distributed machine learning

Introduction

Background

Data protection is a major issue in health care, but clinical research and medical care also rely on high accessibility of patient data. The problem is aggravated in the context of data-driven medicine, where the amount of data needed exceeds the capacity of manual data curation and manual deidentification that would be necessary to protect patient privacy. In the United States, the Health Insurance Portability and Accountability Act of 1996 obliges data curators to remove protected health information (PHI) from medical records before they are shared with researchers. The same holds true for many other countries (see, for instance, §6 GDSG NW [Germany], SI 1438/2002 reg. 7 [England and Wales]). Computer-based data deidentification primarily needs to solve the task of detecting personal information like names, phone numbers, locations, etc. Deidentification systems must meet two opposed interests. To ensure privacy, they must work with high sensitivity (ie, avoid overlooking personal data). Additionally, these systems need to maintain high specificity (ie, avoid removing data unnecessarily). Otherwise, deidentified texts would contain little valuable information compared with the original inputs.

Many approaches to finding protected personal information in health records are based on supervised machine learning [1-3]. Such systems have proven to be very efficient for the

deidentification task. One presented by Dernoncourt et al [3] even outperformed other state-of-the-art approaches in 2016. The general shortcoming of such approaches in this context is that they depend heavily on labeled training data. These training data usually consist of original health records containing personal information. Thus, these data cannot be shared among researchers for distributed training due to the above-mentioned problem. Consequently, there are many small training data sets at medical research institutes which can only be used locally. An additional challenge arises from the fact that even trained neural networks (NNs) can be abused to recover PHI that has been used for training [4,5]. Hence, a trained network for deidentification cannot be shared with other researchers or the public without causing a threat to patient privacy.

Related Work

Several systems for the deidentification of PHI have been presented in the last 20 years. Meystre et al [6] state that most of the systems introduced before 2010 are rule-based or rely on pattern matching. The underlying patterns and rules are typically hand-crafted. Thus, domain experts are needed for the costly generation of such systems resulting in limited generalizability [6]. More recent approaches are mainly based on machine learning or, more precisely, conditional random fields and support vector machines [7]. For these systems, only features of the input data must be specified by hand. Several such tools, however, use additional rules for identifying certain classes of PHI or postprocessing [1,2,8]. The method proposed by Dernoncourt et al [3] is one of the first that solely makes use of NNs and thus is independent of manual feature selection. Liu et al [9] compare several rule-based and NN-based approaches to the deidentification task. Moreover, they introduce an ensemble method that combines these systems.

Collaborative privacy-preserving machine learning has already been studied by several authors. Many systems introduced in this field homomorphically encrypt local data and share the ciphered information for centralized training [10-12]. Thus, these systems need an authority trusted by all parties to start encrypted communication. Another issue of cryptographic centralized training is the cost for encryption and decryption which can become untenable for high-dimensional data in practical applications [10]. By using a decentralized training which obviates the need for sharing individually identifiable health information, these problems vanish. Chang et al [13] experimented with several nonparallel distributed training heuristics. In their proposed framework, local workers conceal their training data but share full sets of learned network parameters. According to Shokri and Shmatikov [4], this transfer of local parameters might still lead to indirect leakage of personal information. Moreover, Fredrikson et al [5] investigated model inversion attacks that infer sensitive information included in the training data from a given trained model. The authors distinguish between black- versus white-box attacks: black-box attacks exploit prediction/classification queries via functional access to a model without knowing its internal details, while white-box attacks additionally use details of the model (eg, network topology and weight matrices). Fredrikson et al [5] demonstrated the feasibility of model inversion attacks especially in the case of white-box approaches using real-world data and publicly available trained classifiers. Their experiments included NNs used for facial recognition. A model inversion attack was able to reconstruct faces of the training set after entering the corresponding names, which can well be considered a relevant violation of privacy.

Hence, we decided to use the collaborative and parallel training method introduced by Shokri and Shmatikov [4]. It reduces the risk of indirect leakage and allows the cooperation of institutes with different computational capacities. For this training scheme, no encryption or any other data protection is needed except the local data and computations must be secured. Like most other collaborative privacy-preserving learning approaches, it only protects from honest-but-curious adversaries. That means malicious participants can hamper the learning or even render it unusable, but they cannot gain information not meant for them. A similar training approach was used by Liu et al [14]. In contrast to our study, they trained feedforward NNs and investigated the effect of using mobile devices to conduct the local training.

Aim of the Study

Our study is aimed at showing the feasibility of a collaborative privacy-preserving learning approach for deep NNs trained to detect personal information in real-world clinical data. We adopted the network topology of the mentioned deidentification approach described by Dernoncourt et al [3]. To overcome the problem of small data sets and privacy issues, we used collaborative privacy-preserving learning suggested by Shokri and Shmatikov [4]. This technique allows the integration of many local data sets into the training while minimizing the risk of leaking protected information. It restricts the training to the local site of each data provider (ie, own data stay with each provider) and only relies on the sharing of small fractions of the local improvements.

We evaluated the performance of the deidentification network trained in a collaborative and privacy-preserving manner. For this purpose, we used an existing set of longitudinal clinical narratives published in the course of the Informatics for Integrating Biology and the Bedside (i2b2) 2014 deidentification challenge hosted by the National Center for Biomedical Computing [7,15] (deidentified clinical records used in this research were provided by the i2b2 National Center for Biomedical Computing funded by U54LM008748 and were originally prepared for the Shared Tasks for Challenges in Natural Language Processing for Clinical Data organized by Dr. Özlem Uzuner, i2b2, and the State University of New York).

Methods

Recurrent Neural Network for Deidentification

General Topology and Training Goal

The recurrent neural network (RNN) presented by Dernoncourt et al [3] is constructed to work on fragments of medical notes. The computed output is a sequence of labels with each label corresponding to one term (also called a token) of the input text. In our adapted version, a label can either be non-PHI or one of 28 PHI classes or subclasses adopted from the definitions of

the i2b2 deidentification challenge (Table 1). If there is at least one subclass, the general class is not used as a label.

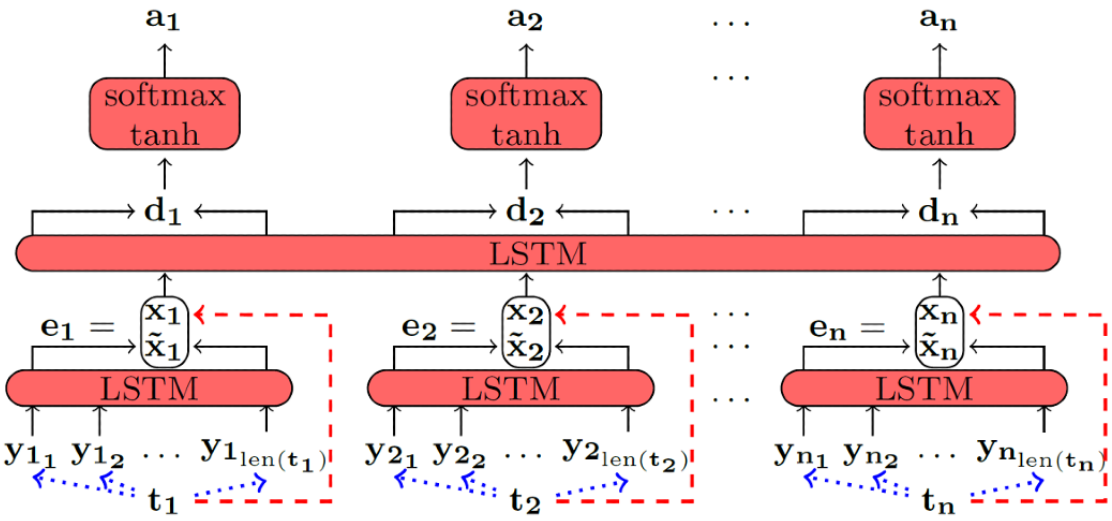
The network topology shows two different layers. The first layer consists of long short-term memory (LSTM) RNNs and is called the character-enhanced token embedding layer. The actual label assignment is conducted in the label prediction layer seen in Figure 1.

Table 1. Possible protected health information classes of terms.

Class	Subclasses
Name	Patient, clinician, username
Profession	—
Location	Hospital, organization, street, city, state, country, zip, other
Age	—
Date	—
Contact	Phone, fax, email, URL, Internet Protocol address[a]
Identification	Social security number[a], medical record number, health plan number, account number[a], license number[a], vehicle identification[a], device identification, biometric identification, identification number

[a]Classes that are not present in the published data set [7].

Figure 1. Network topology of the recurrent neural network used for deidentification [3].



Preprocessing

During preprocessing, the input text is subdivided into terms $\mathbf{t_i}$ which are then transformed into their vector representations $\mathbf{x_i}$ $\in \mathbb{R}^{300}$. For the transformation we use the Word2Vec approach which can map words into a low-dimensional vector space while keeping some semantic and syntactic information [16]. This transformation from word to vector is depicted by the red-dashed arrows (Figure 1). As a result of the preprocessing, the input text $T = (\mathbf{t_1}, \ldots, \mathbf{t_n})$ is available as a sequence of Word2Vec vectors $X = (\mathbf{x_1}, \ldots, \mathbf{x_n})$.

In the second preprocessing step, every term $\mathbf{t_i}$ is further divided into its individual characters ($t_{i\_1}, \ldots, t_{i\_len(ti)}$). Again, every character is translated into a vector $\mathbf{y_{i\_j}} \in \{0,1\}^{128}$ of low dimensionality. The translation is represented by blue-dotted arrows (Figure 1). In contrast to the word embedding, characters are encoded by a simple one-hot method (ie, by vectors with exactly one nonzero component). The dimension being nonzero is obtained by taking the Unicode representation of the corresponding character if it is smaller than 127. For all other characters, the last dimension is set to 1. In this way, all ASCII punctuations and symbols as well as the basic Latin alphabet can be encoded. All letters are mapped to corresponding one-hot vectors without taking any syntactic information into account.

To sum up, at the end of the preprocessing there are vectors $\mathbf{x_i}$ (word embeddings) representing every input term and vectors $\mathbf{y_{i\_j}}$ (character embeddings) each representing one occurrence of a character.

Character-Enhanced Token Embedding Layer

For every word, the first layer contains an independent instance of a bidirectional LSTM. Hence, the number of RNNs changes for every input text.

LSTMs integrate new parts of an input sequence into their computation in every step. Furthermore, they keep some information of the already processed prefix of the sequence.

This makes them capable of learning long-term dependencies between vectors whose positions in the input are far apart [17]. Mathematically, a general unidirectional LSTM can be described as follows.

Let W_i, W_c, W_o be three weight matrices and $\mathbf{b_i}$, $\mathbf{b_c}$, $\mathbf{b_o}$ be three bias vectors. These parameters are learned during the training. If the input sequence equals (z_1, ... , z_m), the LSTM conducts m steps. In the $\mathbf{t}^{th}$ iteration, the hidden state $\mathbf{h_t}$ and the memory state $\mathbf{c_t}$ are computed using $\mathbf{z_t}$, $\mathbf{h_{t-1}}$, and $\mathbf{c_{t-1}}$ as inputs and the following formulas:

$\mathbf{i_t} = \sigma(W_i \cdot \text{concat}(\mathbf{z_t}, \mathbf{h_{t-1}}) + \mathbf{b_i} + \mathbf{1})$

$\mathbf{c_t} = \mathbf{i_t} \odot \mathbf{c_{t-1}} + (1 - \mathbf{i_t}) \odot \tanh(W_c \cdot \text{concat}(\mathbf{z_t}, \mathbf{h_{t-1}}) + \mathbf{b_c})$

$\mathbf{o_t} = \sigma(W_o \cdot \text{concat}(\mathbf{z_t}, \mathbf{h_{t-1}}) + \mathbf{b_o})$

$\mathbf{h_t} = \mathbf{o_t} \odot \tanh(\mathbf{c_t})$

where $\odot$ denotes the element-wise multiplication, σ the element-wise logistic sigmoid function, and tanh the element-wise hyperbolic tangent function. For the first iteration the hidden state $\mathbf{h_0}$ and the cell state $\mathbf{c_0}$ must be initialized arbitrarily. Sometimes the full sequence of hidden states ($\mathbf{h_1}$, ... , $\mathbf{h_m}$) is considered the output of the LSTM and sometimes just the last hidden state $\mathbf{h_m}$ is seen as the output.

A bidirectional LSTM consists of two independent unidirectional LSTMs. The first one works on the original input sequence ($\mathbf{z_1}$, ... , $\mathbf{z_m}$), whereas the second one computes the output for the reversed sequence ($\mathbf{z_m}$, … , $\mathbf{z_1}$).

By combining the two output sequences ($\mathbf{h_1}^{\mathbf{for}}$, ... , $\mathbf{hm}^{\mathbf{for}}$) and ($\mathbf{h_1}^{\mathbf{back}}$, ... , $\mathbf{hm}^{\mathbf{back}}$) one gets the overall output (concat($\mathbf{h_1}^{\mathbf{for}}$, ... , $\mathbf{h_1}^{\mathbf{back}}$),..., concat($\mathbf{hm}^{\mathbf{for}}$, ... , $\mathbf{hm}^{\mathbf{back}}$)) or concat($\mathbf{h_m}^{\mathbf{for}}$, ... , $\mathbf{h_m}^{\mathbf{back}}$), respectively.

The token embedding layer contains one bidirectional LSTM per term $\mathbf{t_i}$ of the input text. Every such LSTM works on the corresponding sequence of character embeddings ($\mathbf{y_{i\_1}}$,...,$\mathbf{y_{i\_len(ti)}}$). At this stage no information are exchanged between LSTMs working on different tokens. However, they all share the same weight matrices and bias vectors. The output of the $\mathbf{i}^{th}$ RNN can be written as $\tilde{\mathbf{x}}_\mathbf{i}$ =concat($\mathbf{h_{i\_len(t\_i)}}^{\mathbf{for}}$, $\mathbf{h_{i\_len(t\_i)}}^{\mathbf{back}}$).

This vector is used as an additional word embedding to overcome shortcomings of Word2Vec [3]. The presented computation always leads to a reproducible vector representation of a token, whereas Word2Vec cannot handle vocabulary that was not used during training. Moreover, this strategy keeps more semantic information than pure Word2Vec combined with a preceding lemmatization step. However, in comparison to the Word2Vec embedding, those LSTMs compute vectors that do not contain any information of interword dependencies.

To keep the advantages of both strategies, the two word embeddings of every token are combined in one vector $\mathbf{e_i}$ = concat($\mathbf{x_i}$, $\tilde{\mathbf{x}}_\mathbf{i}$). In summary, it can be stated that for the input sequence ($\mathbf{t_1}$, ... , $\mathbf{t_n}$) the character-enhanced token embedding layer computes the output ($\mathbf{e_1}$, ... , $\mathbf{e_n}$).

Label Prediction Layer

The subsequent layer evaluates the dependencies between different words of the input text. Again, a bidirectional LSTM is used for the task. In contrast to the previous layer, only one LSTM is used independent of the number of tokens. This RNN uses the output sequence ($\mathbf{e_1}$, ... , $\mathbf{e_n}$) of the previous layer as its input. The output consists of the full sequence ($\mathbf{d_1}$, ... , $\mathbf{d_n}$) where $\mathbf{d_i}$ = concat($\mathbf{h_1}^{\mathbf{for}}$, ... , $\mathbf{h_1}^{\mathbf{back}}$).

Every $\mathbf{d_i}$ is further processed by a two-layer, fully connected feedforward network. The parameters are the same for every input $\mathbf{d_i}$ and are trained jointly for every sequence ($\mathbf{d_1}$, ... , $\mathbf{d_n}$). The network works as follows:

$\mathbf{l_i} = \tanh(W_1 \cdot \mathbf{d_1} + \mathbf{b_1})$ hidden layer

$\mathbf{a_i} = \text{softmax}(W_2 \cdot \mathbf{l_i} + \mathbf{b_2})$ output layer

Thus, the results $\mathbf{a_i} \in [0,1]^{29}$ can be interpreted as conditional posterior probabilities of the possible labels given the input token $\mathbf{t_i}$. By choosing label($\mathbf{t_i}$) = $\text{argmax}_{j \in \{1,...,29\}} \mathbf{a_{i\_j}}$, every word is uniquely assigned one of the 28 PHI classes (Table 1) or the non-PHI label. The loss is defined pursuant to the cross-entropy and minimized during training.

Collaborative Training

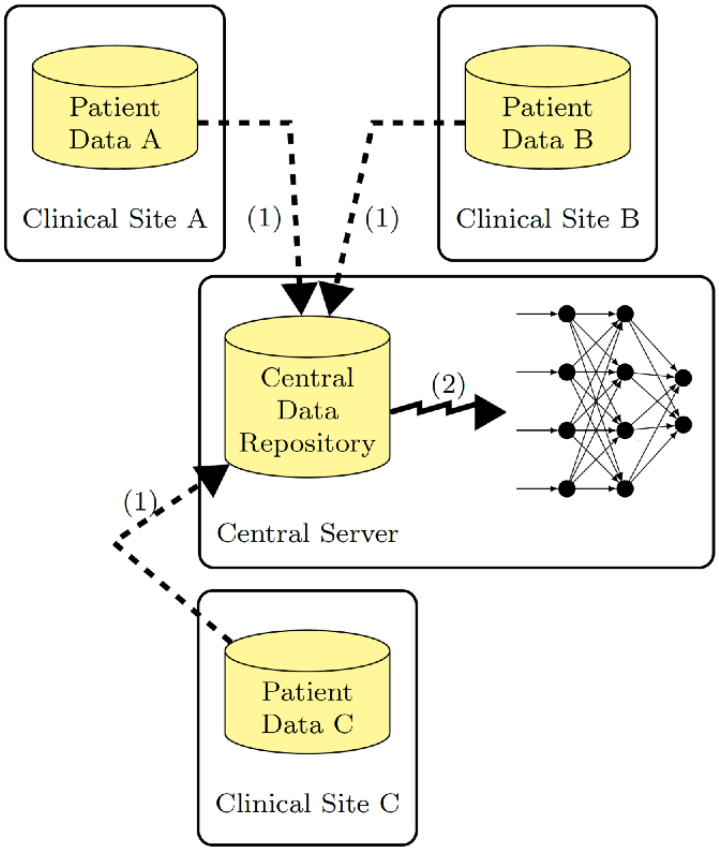
Aim

The aim of collaborative learning is the integration of many (private) training sets into the learning process. Note that this goal is different from the one of distributed training that solely aims at faster training through the use of several computation nodes. The following sections summarize three alternative collaborative approaches: the nonprotective standard method, a round robin technique, and the privacy-preserving distributed selective stochastic gradient descent (DSSGD).

Nonprotective Training

The nonconfidential training (Figure 2) relies on central data processing. The dashed arrows denote the disclosure of the local training data, which leads to a large central data set. Usually only one entity controls the training by distributing tasks and data to workers of a large computing cluster [18]. The jagged arrow marks the distributed training supervised by the central server. The server has full control over all data and over the training procedure. Hence, this method is not suitable for protecting local data but enables fast training with full information integration.

Figure 2. Nonprotected training with shared data.

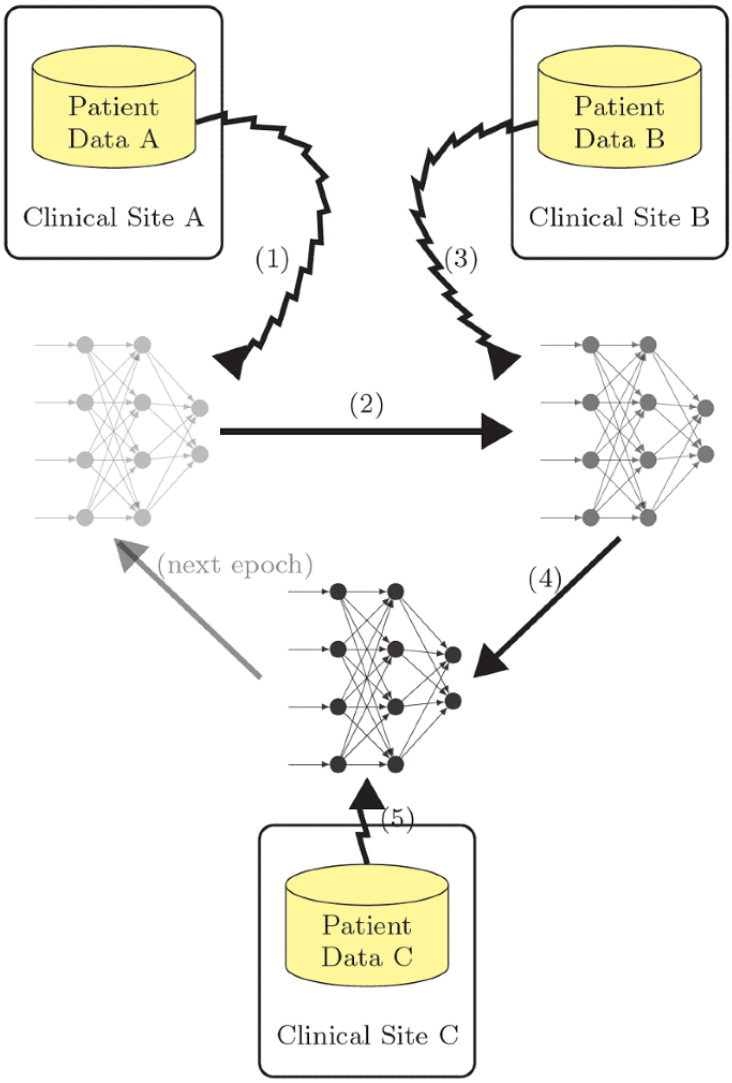


Semiprotective Round Robin Training

An initial step toward privacy protecting training is to keep local training data secret and just share the NN that is trained by applying a round robin algorithm (Figure 3). In a first step, worker A trains the network using only its local data (jagged arrow [1]). After it is finished, the trained network is handed to the second worker (solid arrow [2]) who improves the network further by training with its local data (jagged arrow [3]). The other workers contribute analogously until the first training epoch is finished. Afterward the next round can be conducted in the same way. This training is similar to the cyclical weight transfer presented by Chang et al [13].

Although this protocol allows the workers to initially keep the local data secret, it does not prevent leakage of personal information through the passing on of the trained network. The parameters might still reveal parts of confidential information that have been used for training.

Figure 3. Semiprotective round robin training with local data.



Distributed Selective Stochastic Gradient Descent

DSSGD enables an exclusively local training and additionally allows every participant to decide how much information about local parameters is shared (Figure 4). The following description of DSSGD solely corresponds to the version used for our experiments. We would like to point out that there are several other options suggested by Shokri and Shmatikov [4].

The basic idea of gradient descent in general is to find a (local) minimum of the loss function by adapting the weights along their negative gradients. In the fully stochastic version, every gradient is computed over only one sample at a time. Hereafter, the phrase stochastic gradient descent refers to this kind of gradient computation.

Let $\mathbf{p}$ be the flattened vector representing all parameters of the network (ie, W_i, W_c, W_o, $\mathbf{b_i}$, $\mathbf{b_c}$, $\mathbf{b_o}$ of both LSTM types and W_1, W_2, $\mathbf{b_1}$, $\mathbf{b_2}$ of the feedforward layers). The cross-entropy error of the weights with respect to one training text T is defined as

$$E(\mathbf{p}) = -\sum_{i=1,\ldots,n} (\text{class}(\mathbf{t_i}) \cdot \ln(\mathbf{a_i}))$$

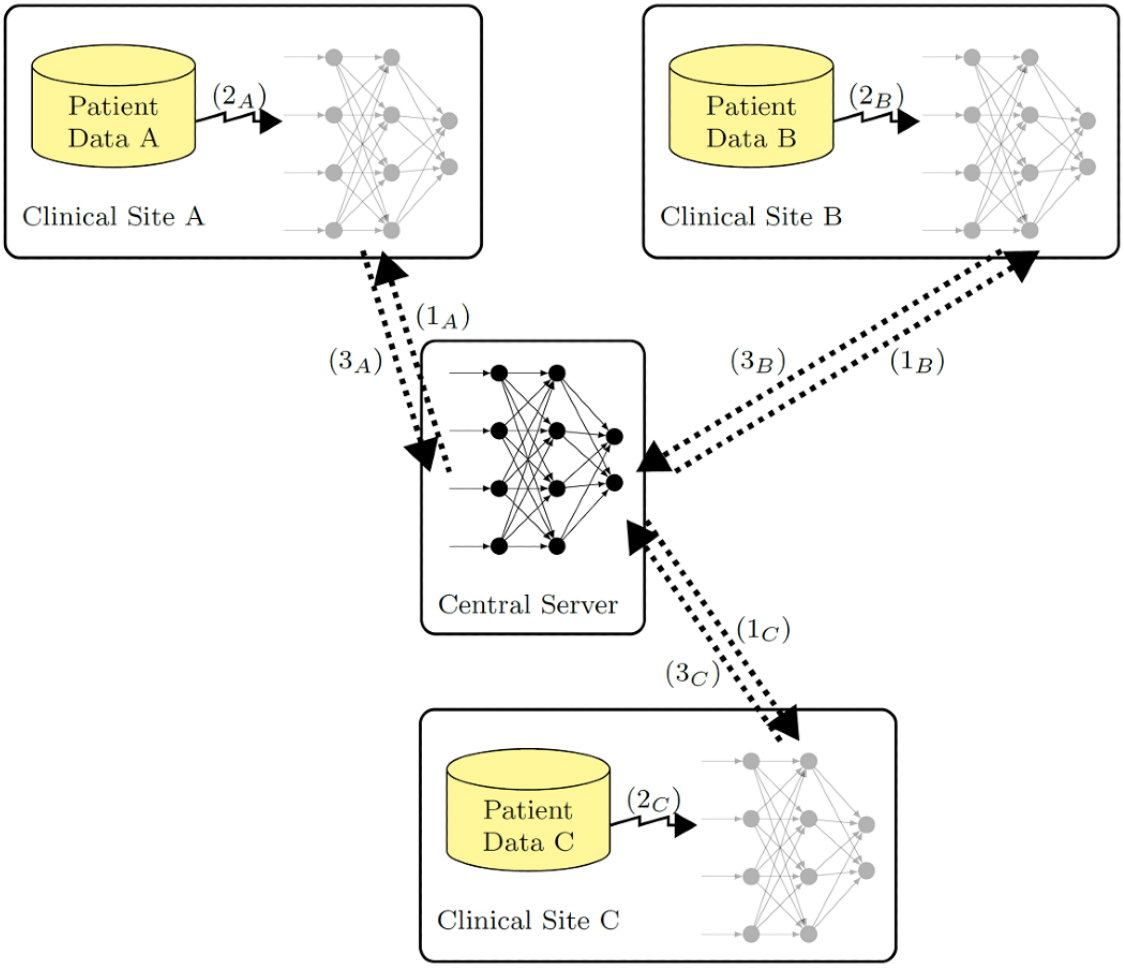
where $\text{class}(\mathbf{t_i}) \in \{0,1\}^{29}$ is the one-hot representation of the correct class of token $\mathbf{t_i}$.

After the error has been computed in the forward pass, the gradient of $E(\mathbf{p})$ can be determined in the backward pass using backpropagation. The individual parameters are then updated in the following way:

$$\mathbf{p}_j = \mathbf{p}_j - \eta \cdot \partial E / \partial \mathbf{p}_j (\mathbf{p})$$

where η denotes a fixed learning rate.

Figure 4. Distributed selective stochastic gradient descent with local data.



The peculiarity of DSSGD is that the learning is conducted in a collaborative fashion but without a central manager (Figure 4). Every participating worker computes error gradients by using only its local data (jagged arrows). A certain number of local partial derivatives is shared with the other workers (dotted arrows) via the central server after every epoch. This fraction of shared derivatives θ can be set by local data curators. In order to prevent unwanted divulgation of private information, the absolute values of shared scalars can be clipped to an individually defined minimum/maximum γ. Moreover, some lower bound τ can be chosen to omit sharing updates with information that is too small. All those additional settings are unknown to other workers.

The central server is solely used to upload and download subsets of local derivatives asynchronously. It does not manage the local workers like a central manager and does not have access to the local training data.

In the following text, the local training as well as the parameter exchange is described in more detail. In the beginning, different parties agree upon a topology of the network and a learning rate. On this basis, the global weights $\mathbf{p}_{glo}$ are initialized at the server. Beside the single set of weights, the server maintains a statistic for every weight indicating how often it has been updated by workers.

Figure 5 summarizes the work conducted by one single worker where *D* denotes the local training set. In the beginning, a worker builds a local copy of the network collectively agreed upon. Additionally, the local learning rate is set to the globally fixed value. A single epoch of the local training begins with the download of a subset of the global weights. The size of this set is specified by the local hyperparameter θ_d. The $\theta_d \cdot \text{len}(\mathbf{p}_{glo})$ parameters with the highest global update statistics are chosen for the download. By adjusting θ_d, the worker can decide how much other workers can influence its final result. In the subsequent step, simple stochastic gradient descent is used to improve local weights. After one epoch, the weight updates that are sent to the server are chosen. For this purpose, the first step is to check for every update whether its absolute value exceeds the threshold τ determined in the beginning. If this is not the case, this single update value is not communicated. Second, the remaining updates are clamped to the interval $[-\gamma,+\gamma]$. The third and last part of the selection process is based on randomness. To meet the upload rate θ_u, the set of possible updates is sampled uniformly at random. The training is ended when a minimum or any other stopping criterion is reached. A comprehensive assessment of the passive protection assured by the presented method can be found in the original paper [4].

Figure 5. Local training procedure [4].

Require: $\theta_d, \theta_u, \gamma, \tau$
  Initialize local network
  repeat
    Download $\theta_d \cdot \text{len}(\mathbf{p_{glo}})$ parameters
    update = 0
    for all $T \in D$ **do**
      $\mathbf{d} = -\eta \cdot \nabla E(\mathbf{p_{loc}})$
      $\mathbf{p_{loc}} = \mathbf{p_{loc}} + \mathbf{d}$
      update = ***update*** + **d**
    for all ***update***$_j$ **do**
      if $|\mathbf{update}_j| < \tau$ **then**
        update$_j$ = 0
      else
        update$_j$ = clamp (***update***$_j$, $(-\gamma, +\gamma)$)
    Sample $\theta_d \cdot \text{len}(\mathbf{p_{glo}})$ nonzero parameters from ***update***
    Update global parameters by adding chosen updates
  until local minimum is reached

Experiments

To evaluate the performance of the network for deidentification and the different training methods, we used a data set also used by Dernoncourt et al [3] for training the noncollaborative version. The basis of this data set is 1304 real longitudinal patient records for 296 diabetic patients provided by Partners Healthcare and adapted by Stubbs and Uzuner [15] for the 2014 i2b2 deidentification challenge. These notes were collected at different hospitals and are written in natural language (English) by caregivers to document or communicate patients' medical history, physical and mental state, prescriptions, and other events. During the generation of this gold standard set, several human annotators marked critical PHI in the notes, which was later replaced by realistic surrogate data. Names, for example, have been replaced by imaginary names. See Stubbs and Uzuner [15] for a detailed description of the data set and the replacement methods.

The full training subset contains around 598,000 tokens with approximately 17,000 instances of PHI, while the subset used for testing consists of approximately 387,000 tokens including more than 11,000 instances of PHI. Records of patients represented in the training set are not contained in the test set. Due to the class imbalance, the F1 measure was used to quantify performances during experiments.

For the tokenization of the data sets, we made use of the tokenize package published as part of the Natural Language Toolkit. The Word2Vec embedding was generated with the help of the Gensim library and was trained on all English Wikipedia articles available in the beginning of November 2018. All networks were implemented with the help of the open source software library TensorFlow. We used graphics processing unit support based on the CUDA platform to accelerate training.

The first experiment (A) was conducted to get baseline results. We trained one RNN as described in section Recurrent Neural Network for deidentification using the full training data set. This training corresponds to the nonprotective case outlined in section Nonprotective Training. Plain stochastic gradient descent with a learning rate of 0.9 was used for the optimization. After every epoch, the performance was determined with respect to the test set. The hidden states, the cell states, and the biases of LSTMs used in the token embedding layer were of size 128. Thus, the weight matrices were in space $\mathbb{R}^{128\times256}$ and $\mathbf{e_i} \in \mathbb{R}^{556}$. During training, dropout with probability 0.5 was applied to the sequence ($\mathbf{e_1}$,..., $\mathbf{e_n}$) to counteract overfitting that might have been introduced by the large number of training epochs. The single LSTM of the label prediction layer held parameter vectors that were all of size 100 and hence output the sequence ($\mathbf{d_1}$,..., $\mathbf{d_n}$) with $d_i \in \mathbb{R}^{200}$. For the final two feedforward layers, the weights were $W_1 \in \mathbb{R}^{100\times200}$ and $W_1 \in \mathbb{R}^{29\times100}$.

Experiment B was performed according to the description in section Semiprotective Round Robin Training. The full training set was subdivided into 5 disjoint private sets of similar size. All records of one patient were kept in the same subset. The training was performed in a round robin fashion using stochastic gradient descent with a learning rate of 0.9 at each worker. The test set was, as in all experiments, left untouched and tested against after every full epoch (ie, after 5 local epochs).

In a third experiment (C_0.1), the collaborative privacy-preserving training (Distributed Selective Stochastic Gradient Descent section) was tested. For this purpose, the global network topology was chosen, as in the previous experiments. We trained 5 RNNs collaboratively that asynchronously shared some of their weight updates. The training data were distributed as in experiment B. In contrast to the previous experiment, the nets were tested against the global test set after every local epoch, since there is no global epoch in this setup.

Thus, there are 5 results for every epoch. The training hyperparameters for all nets were θ_d=0.1, θ_u=0.5, γ=10, τ=0.0001, while the global learning rate of DSSGD was set to

0.9. Afterward we conducted a similar experiment with the only difference being that θ_d was set to 0.5 (experiment C_0.5).

Experiment D was run in the same way as experiment C_0.1 except for the fact that this time the 5 networks did not communicate at all and just trained on their local sets. Again, we made use of simple stochastic gradient descent with a learning rate of 0.9. A summary of all experiments can be found in Table 2.

Table 2. Summary of the experiments.

Name	Learning strategy
A	Centralized training using stochastic gradient descent (learning rate: 0.9)
B	Collaborative training of 5 workers using the round robin method (learning rate: 0.9)
C_0.1	Collaborative privacy preserving training of 5 workers with DSSGD[a] (θ_d = 0.1, θ_u = 0.5, γ = 10, τ = 0.0001; learning rate: 0.9)
C_0.5	Collaborative privacy preserving training of 5 workers with DSSGD (θ_d = 0.5, θ_u = 0.5, γ = 10, τ = 0.0001; learning rate: 0.9)
D	Local training without collaboration of 5 workers using stochastic gradient descent

[a]DSSGD: distributed selective stochastic gradient descent.

Results

To obtain results that can be compared to the outcomes achieved by Dernoncourt et al [3], we adopted their scoring. Thus, every token labeled as PHI by the network was considered a true positive result if and only if this token was actually some PHI. The scoring, therefore, does not consider the PHI classes but only the decision PHI versus non-PHI.

In all experiments the nets underwent 200 epochs of training. The illustration on the left in Figure 6 depicts the development of F1 scores achieved on the test set. The black line corresponds to experiment A and the magenta one to experiment B. Note that both lines are very similar and partially overlap. For the other experiments, there are 5 results per time point. The solid green line depicts the mean F1 values achieved in experiment C_0.1, while the area colored in green indicates the full range in which the values fell. Similarly, the results of experiment D are represented. For these measurements, the color red was used. For reasons of clarity, the results of experiment C_0.5 are depicted in the image on the right. Again, these results are shown in green. The remaining lines are the same as in the illustration on the left. The final scores are given in Table 3.

Figure 6. Performances achieved on the test set during the experiments including C_0.1 (left) or C_0.5 (right).

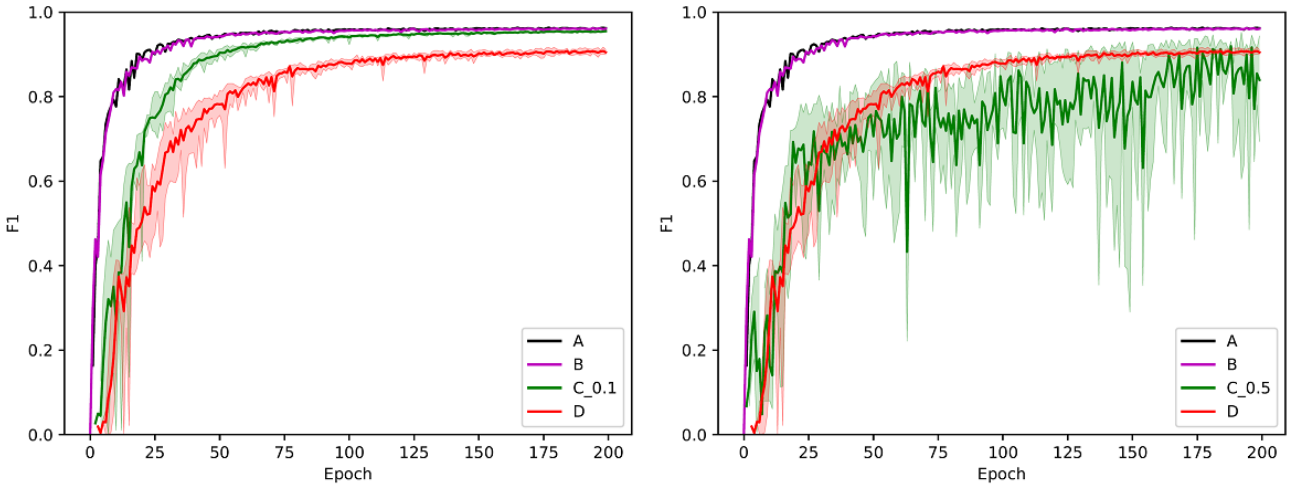


Table 3. Scores after 200 training epochs.

Experiment	F1		
	Minimum	Mean	Maximum
A	—	0.962	—
B	—	0.961	—
C_0.1	0.955	0.955	0.956
C_0.5	0.694	0.840	0.944
D	0.900	0.905	0.911

Discussion

Using real-world clinical data, our study shows that collaborative privacy-preserving training enables a NN-based detection of PHI in clinical free-text records. The approach, thus, solves the dilemma of requiring machine learning techniques for easily adapting a PHI detection system to heterogeneous clinical documentation while avoiding disclosure of locally stored patient data.

Comparisons of the results achieved by the networks trained with plain stochastic gradient descent (A and D) underline the intuitive assumption that larger training data sets lead to better performances.

The similarity between the results of experiments A and B originates from the fact that both training methods are equivalent except for the kind of shuffling of samples during training. If the only aim is to keep the exact local data private and not to protect as much private information as possible, the round robin technique (or cyclic weight transfer) is the best choice. If, however, a real privacy-preserving collaborative learning mechanism is need as in the presented medical domain, DSSGD is the correct training algorithm. It offers both privacy preservation and good performance. The results of experiments C_0.1 and D show that a lack of sufficiently large local training data sets can be compensated by applying collaborative privacy-preserving training based on DSSGD. This finding is in line with the observations made by Shokri and Shmatikov [4] and the results obtained by Liu et al [14] who tested the protocol with some smaller nonrecurrent neural networks. By applying the collaborative training, all workers reach similar outcomes rapidly. After 25 epochs, all nets perform better than the ones trained with the corresponding noncollaborative version (D). The local F1 values all tend to the results achieved in the centralized version (A). During all experiments, a rather high learning rate was used that might have led to nonoptimal solutions. However, since it was kept constant throughout all 5 experiments, the insights gained by comparison are still valid. Another shortcoming of our experiments is that the γ value was set without using a calibrating data set as is suggested by Shokri and Shmatikov [4].

During experiments C_0.1 and C_0.5, the importance of well-chosen communication parameters θ_u, θ_d became apparent. If the download rate is set to high, the global results interfere too much with local training sessions. This leads to high fluctuations and prevents the local weights from converging.

Acknowledgments

SF and CS were affiliated with the Department of Medical Informatics, Medical Faculty at RWTH Aachen University at the time of writing and are currently affiliated with the Institute of Medical Statistics, Computer and Data Sciences at Jena University Hospital.

Conflicts of Interest

None declared.

References

1. Aberdeen J, Bayer S, Yeniterzi R, Wellner B, Clark C, Hanauer D, et al. The MITRE Identification Scrubber Toolkit: design, training, and assessment. Int J Med Inform 2010 Dec;79(12):849-859. [doi: 10.1016/j.ijmedinf.2010.09.007] [Medline: 20951082]
2. Deleger L, Molnar K, Savova G, Xia F, Lingren T, Li Q, et al. Large-scale evaluation of automated clinical note de-identification and its impact on information extraction. J Am Med Inform Assoc 2013 Jan 01;20(1):84-94 [FREE Full text] [doi: 10.1136/amiajnl-2012-001012] [Medline: 22859645]
3. Dernoncourt F, Lee JY, Uzuner O, Szolovits P. De-identification of patient notes with recurrent neural networks. J Am Med Inform Assoc 2017 May 01;24(3):596-606. [doi: 10.1093/jamia/ocw156] [Medline: 28040687]
4. Shokri R, Shmatikov V. Privacy-preserving deep learning. In: Ray I, Li N, editors. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press; 2015:1310-1321.
5. Fredrikson M, Jha S, Ristenpart T. Model inversion attacks that exploit confidence information and basic countermeasures. In: Ray I, Li N, editors. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press; 2015:1322-1333.
6. Meystre S, Friedlin F, South B, Shen S, Samore M. Automatic de-identification of textual documents in the electronic health record: a review of recent research. BMC Med Res Methodol 2010:70-86. [doi: 10.1186/1471-2288-10-70]
7. Stubbs A, Kotfila C, Uzuner Ö. Automated systems for the de-identification of longitudinal clinical narratives: overview of 2014 i2b2/UTHealth shared task Track 1. J Biomed Inform 2015 Dec;58 Suppl:S11-SS9 [FREE Full text] [doi: 10.1016/j.jbi.2015.06.007] [Medline: 26225918]
8. Liu Z, Chen Y, Tang B, Wang X, Chen Q, Li H, et al. Automatic de-identification of electronic medical records using token-level and character-level conditional random fields. J Biomed Inform 2015 Dec;58 Suppl:S47-S52 [FREE Full text] [doi: 10.1016/j.jbi.2015.06.009] [Medline: 26122526]
9. Liu Z, Tang B, Wang X, Chen Q. De-identification of clinical notes via recurrent neural network and conditional random field. J Biomed Inform 2017 Dec;75S:S34-S42 [FREE Full text] [doi: 10.1016/j.jbi.2017.05.023] [Medline: 28579533]

10. Chen T, Zhong S. Privacy-preserving backpropagation neural network learning. IEEE Trans Neural Netw 2009 Oct;20(10):1554-1564. [doi: 10.1109/TNN.2009.2026902] [Medline: 19709975]
11. Bansal A, Chen T, Zhong S. Privacy preserving Back-propagation neural network learning over arbitrarily partitioned data. Neural Comput & Applic 2010 Feb 12;20(1):143-150. [doi: 10.1007/s00521-010-0346-z]
12. Yuan JW, Yu SC. Privacy preserving back-propagation neural network learning made practical with cloud computing. IEEE Trans. Parallel Distrib. Syst 2014 Jan;25(1):212-221. [doi: 10.1109/TPDS.2013.18]
13. Chang K, Balachandar N, Lam C, Yi D, Brown J, Beers A, et al. Distributed deep learning networks among institutions for medical imaging. J Am Med Informat Assoc 2018;25:945-954. [doi: 10.1093/jamia/ocy017]
14. Liu M, Jiang H, Chen J, Badokhon A, Wei X, Huang M. A collaborative privacy-preserving deep learning system in distributed mobile environment. In: Arabnia H, Deligiannidis L, editors. 2016 International Conference on Computational Science and Computational Intelligence. Washington: IEEE Computer Society, Conference Publishing Services; 2016:192-197.
15. Stubbs A, Uzuner Ö. Annotating longitudinal clinical narratives for de-identification: the 2014 i2b2/UTHealth corpus. J Biomed Inform 2015 Dec;58 Suppl:S20-S29 [FREE Full text] [doi: 10.1016/j.jbi.2015.07.020] [Medline: 26319540]
16. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Burges CJ, Bottou L, editors. Advances in Neural Information Processing Systems. Cambridge: MIT Press; 2013:3111-3119.
17. Greff K, Srivastava R, Koutnik J, Steunebrink B, Schmidhuber J. LSTM: A Search Space Odyssey. IEEE 2017;28:2222-2232. [doi: 10.1109/tnnls.2016.2582924]
18. Dean J, Corrado G, Monga R, Chen K, Devin M, Le Q, et al. Large scale distributed deep networks. In: Pereira F, Burges CJ, editors. Proceedings of the 25th International Conference on Neural Information Processing Systems. Philadelphia: Proceedings of the 25th International Conference on Neural Information Processing Systems; 2012:1223-1231.

Abbreviations

DSSGD: distributed selective stochastic gradient descent
i2b2: Informatics for Integrating Biology and the Bedside
LSTM: long short-term memory
NN: neural network
PHI: protected health information
RNN: recurrent neural network

Edited by CL Parra-Calderón; submitted 27.03.19; peer-reviewed by M Mizani, H Wu, M Kohlweiss; comments to author 20.06.19; revised version received 25.08.19; accepted 16.12.19; published 05.05.20

Please cite as:
Festag S, Spreckelsen C
Privacy-Preserving Deep Learning for the Detection of Protected Health Information in Real-World Data: Comparative Evaluation
JMIR Form Res 2020;4(5):e14064
URL: https://formative.jmir.org/2020/5/e14064
doi: 10.2196/14064
PMID: