Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

Master's Thesis

Data Protection Compliant Exchange of Training Data for Automatic De-Identification of Medical Text

Language Technology (LT) Group
Fachbereich Informatik
MIN-Fakultät
Universität Hamburg

submitted by
Max Friedrich
Informatik
645555

November 20, 2018

Examiners: Prof. Dr. Chris Biemann
Dr. Gregor Wiedemann
Advisor: Arne Köhn

Abstract

Due to data protection and privacy laws, medical records can only be shared for research if they are de-identified. Automated de-identification tools aim to enable cost-effective de-identification of medical records at scale. These tools are trained on medical records that are manually pseudonymized and often come from a single source. However, existing tools do not generalize well to data from new sources. Bigger and more diverse datasets are required to train more general de-identification tools. We examine approaches to transforming medical text into a private representation without requiring manual pseudonymization that simplify data protection compliant exchange of training data. Both our automatic word-level pseudonymization and adversarially trained representation allow training a de-identification tool to the target F_1 score of 95% while preventing adversary models from re-identifying any protected information.

Zusammenfassung

Datenschutzgesetze legen fest, dass medizinische Berichte nur in anonymisierter Form weitergegeben werden dürfen, etwa um in der Forschung verwendet zu werden. Automatische Anonymisierungswerkzeuge werden entwickelt, um kostengünstig große Mengen medizinischer Berichte zu anonymisieren. Diese Werkzeuge beruhen auf manuell pseudonymisierten Trainingsdaten, die oft aus einer einzigen Quelle stammen. Vorhandene Anonymisierungswerkzeuge funktionieren jedoch nicht zufriedenstellend auf Berichten aus neuen Quellen. Daher werden größere und vielfältigere Datensätze benötigt, um allgemeingültige Anonymisierungswerkzeuge zu trainieren. Wir untersuchen Ansätze zur Transformation medizinischer Berichte in eine private Repräsentation, die ohne manuelle Pseudonymisierung auskommen, und dadurch datenschutzkonformen Austausch von Trainingsdaten vereinfachen. Sowohl unsere automatische Pseudonymisierung auf Wortebene als auch unsere auf *adversarial training* basierende Repräsentation erlauben es, ein Anonymisierungswerkzeug auf den Zielwert von 95% im F_1-Maß zu trainieren, und verhindern dabei, dass Gegenspieler-Modelle geschützte Informationen wiederherstellen.

Contents

List of Acronyms

CNN Convolutional neural network

CRF Conditional random field

DANN Domain-adversarial neural network

ELMo Embeddings from language models

GDPR General Data Protection Regulation. European privacy law.

HIPAA Health Insurance Portability Accountability Act. American health care law.

LSTM Long short-term memory

MLP Multi-layer perceptron

NER Named entity recognition

NLP Natural language processing

PHI Protected health information

ReLU Rectified linear unit

RNN Recurrent neural network

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 NLP for Clinical Data organized by Dr. Özlem Uzuner, i2b2 and SUNY.

1 Introduction

As more and more information is collected in electronic health records, it becomes attractive for large-scale medical studies to make use of this data. Processing and sharing of health data are regulated by data protection and privacy laws such as the European General Data Protection Regulation (GDPR, 2016) and the Privacy Rule of the American Health Insurance Portability Accountability Act (HIPAA, 1996). While GDPR requires explicit consent by patients for their data to be processed (except when public health is concerned), HIPAA allows records to be shared if they are de-identified (sanitized) to protect patient confidentiality.

Much of the valuable information in health records can be found in their free text portion, the clinical narrative. To de-identify medical text, all protected health information (PHI) needs to be detected and removed. PHI includes potentially identifying data of several categories such as names, professions, geographic identifiers, and account numbers. Manual de-identification of medical text is a time-consuming and error-prone task that does not scale well to large sets of health records.

Trying to train a software tool for automatic de-identification leads to a "chicken and egg problem" (Uzuner et al., 2007): without a comprehensive training set, an automatic de-identification tool cannot be developed, but without such a tool, it is difficult to share de-identified clinical records for research (including for training the tool itself). The standard method of data protection compliant sharing of training data for a de-identification tool requires humans to pseudonymize protected information with substitutes (replacing e.g. every person name with a different person name and every date with a different date) in a document-coherent way.

Today, a pseudonymized dataset for de-identification from a single source is publicly available. However, tools trained on the dataset are too specific for the concrete data and do not generalize well to data from other sources (Stubbs et al., 2017). If a medical institution instead decides to train a de-identification tool on their raw text data, it is conceivable that the tool would contain traces of the PHI it was trained with, making it possible for an attacker to recover parts of the training data if the tool itself is shared. To achieve a universal de-identification tool, many medical institutions would have to pool their data. Preparing this data for sharing using the document-coherent pseudonymization approach requires large human effort.

1.1 Research Question

A representation of medical text that allows training a de-identification tool while not allowing inference of protected information would greatly simplify the collection of training

data for a universal de-identification tool. This representation would still require humans to annotate PHI (as this is the training data for the task) but the pseudonymization step would be performed by the transformation to the representation. A tool trained on the representation could easily be made publicly available because its parameters cannot contain any protected data, as it is never trained on raw text. Simplifying the de-identification procedure could enable large-scale medical studies that are otherwise too costly. Based on these observations, we formulate our central research question:

> *How can training data for de-identification of medical text be perturbed to allow sharing with less human effort while protecting patient confidentiality and preventing re-identification attacks?*

1.2 Contributions

In this work, we examine four approaches to sharing training data for de-identification that bypass the human pseudonymization step by perturbing vector representations of words. Our key contributions are:

1. A baseline de-identification model trained on raw text data that does not use explicit character features and achieves an F_1 score of 97.74%, which is comparable to the state of the art.

2. An automatic word-level approximation of pseudonymization by substitution that allows training a de-identification model to an F_1 score of 96.75%.

3. An adversarially trained representation for medical text tokens that allows training a de-identification model to an F_1 score of 97.4% while preventing an adversary model to re-identify sentences or build a lookup table of representations for known-plaintext attacks.

1.3 Outline

This thesis is composed of 8 chapters. Following the introduction in Chapter 1, we present related work regarding de-identification, sanitizing medical datasets, and privacy-preserving machine learning in Chapter 2. Chapter 3 introduces background concepts including machine learning for natural language processing (NLP) and word representations. We present our methodology in Chapter 4. In Chapter 5, we describe our experimental approaches for which we present results in Chapter 6. The approaches and results are discussed in Chapter 7, including their connection to the related work and possible future work. Chapter 8 concludes the thesis by summarizing our contributions.

2 Related Work

This chapter summarizes related work on the de-identification task, data release, and privacy-preserving machine learning.

2.1 De-Identification of Medical Text

De-identification is an NLP task that is related to named entity recognition (NER). In 2006, 2014 and 2016, three shared tasks on de-identification were organized by the American i2b2 group. The organizers performed manual pseudonymization on clinical records from a single source to create the datasets for each of the shared tasks (Stubbs and Uzuner, 2015).

In the first two shared tasks, submitted approaches are based on hand-crafted rules, gazetteers, and machine learning methods such as boosting, conditional random fields (CRFs), and support vector machines. The machine learning models use features such as lexical cues (e.g. *"was the previous word 'Dr'?"*), templates for phone numbers and addresses, part-of-speech tags, and gazetteer or dictionary information. Similar approaches are also summarized in a survey (Meystre et al., 2010). In the 2016 shared task, a deep learning based system (Liu et al., 2017) delivered the best results.

Up to the 2014 shared task, the organizers emphasized that it is unclear if a tool trained on the provided datasets will generalize to medical records from other sources (Uzuner et al., 2007; Stubbs et al., 2015). The 2016 shared task featured a sight-unseen track in which existing systems were evaluated on records from a new data source. The best system achieved an F_1 score of only 79%, proving that de-identification systems at the time were not able to deliver sufficient performance on completely new data (Stubbs et al., 2017).

Dernoncourt et al. (2017b) achieve state-of-the-art performance in de-identification with a deep learning model. Their model achieves F_1 scores of 97.85% on the i2b2 2014 dataset and 99.23% on their own, larger dataset. It uses pre-trained word embeddings and task-specific character embeddings. The output sequence is optimized with a CRF layer. After experimenting with their architecture, they conclude that the character embedding layer is more important to the model performance than the pre-trained word embeddings. However, the word embeddings show benefits when compared to rule-based models e.g. when recognizing profession names that are in the same embedding space region but might not occur in a hand-crafted dictionary. Using transfer learning from a model trained on a larger dataset, they were able to further improve their scores on the i2b2 2014 dataset (Lee et al., 2017). It is possible to reach an 98% F_1 score on the i2b2 dataset by using document position information as an additional input (Zhao et al., 2018),

which will, however, most likely deteriorate generalization performance to instances of medical text with different structures. In a later work, Dernoncourt et al. use the same architecture for a generic NER model (Dernoncourt et al., 2017a), which underlines the similarity between the de-identification and NER tasks.

De-identification can be interpreted as a preprocessing step for further information extraction because only de-identified data is allowed to be shared. Such information extraction tasks include NER of condition and treatment names (Uzuner et al., 2010; Pradhan et al., 2014) or the classification of relationships between conditions, tests, and treatments (Uzuner et al., 2011). A survey (Meystre et al., 2008) summarizes information extraction approaches at the time. In 2012, the free-text part of electronic health records was still considered an underused source of data (Jensen et al., 2012).

In Dernoncourt et al.'s models, Wikipedia-pre-trained GloVe word embeddings (Pennington et al., 2014) perform better than embeddings that were trained on the i2b2 dataset. In NER tasks focusing on medical entities, training embeddings on unlabeled medical text works well (Wu et al., 2015), possibly because they cover the medical domain vocabulary better.

2.2 Sanitizing Medical Datasets for Release

Sanitizing sensitive data by replacing names with ID numbers went wrong numerous times in recent years. Examples include the New York Times identifying an AOL search user by her queries (Barbaro et al., 2006) and the Netflix prize data being linked to IMDB profiles to infer users' religious, political and sexual preferences (Narayanan and Shmatikov, 2008). The research on sanitizing datasets for release has mostly focused on structured (tabular) data. Privacy models are used to defend from inferring sensitive attributes from quasi-identifiers. In medical datasets, quasi-identifiers can include e.g. a patient's weight or diagnoses.

Inferring the membership of a person's record in a dataset (membership disclosure) is often already a bad outcome even if their detailed information is not accessible. For example, if a person's data is revealed to be part of a dataset of heart disease patients, sensitive information is known about them (they seem to have a heart condition) without any access to their specific record.

Privacy models such as k-anonymity (Sweeney, 2002) and ℓ-diversity (Machanavajjhala et al., 2007) are susceptible to homogeneity inference using insensitive attributes (Kifer, 2009). Additionally, k-anonymity fails in sparse, high dimensional settings (Aggarwal, 2005). Differential Privacy (Dwork, 2006, 2008) is a model that limits the difference between neighboring databases (i.e. databases in which one item is missing or added) and therefore the influence that single items can have on query results. It is currently the most popular privacy model in research and industry.

An overview of the application of differential privacy to electronic health records is given by Dankar and El Emam (2012). Most mechanisms achieve differential privacy by adding noise to results. A further survey (Gkoulalas-Divanis et al., 2014) summarizes algorithms for publishing data from health records. Newer approaches include creating

synthetic data for release with generative adversarial networks (Tripathy et al., 2017; Beaulieu-Jones et al., 2017; Zhang et al., 2018).

We know of no theoretic framework for private release of high-dimensional, sequential data such as text. Replacing all personal information with pseudonyms like in the i2b2 datasets seems to be a good first step. The PHI categories defined by HIPAA are however not complete: e.g. phrases like *caused by Hurricane Sandy* are not protected by HIPAA. They can lead to date information even though all literal dates like *October 2012* are removed (Stubbs and Uzuner, 2015). This example of natural disasters is protected in the additional i2b2 categories but it may still be possible to infer protected attributes from the i2b2 datasets[1].

While rule-based de-identification models sometimes work on the document level and take e.g. previously mentioned names into account, the deep learning models mentioned above operate only on the sentence level. This means that the models achieve the same performance even if trained on a shuffled set of sentences. Shuffling sentences for data release is another simple method that makes it difficult to identify relationships in the data, especially in large datasets containing data from hundreds or thousands of patients.

2.3 Privacy-Preserving Machine Learning

As machine learning models have become increasingly powerful and are now applied to process all kinds of data, including data with private information, privacy-preserving machine learning has emerged as a trend in recent years. When not trained with privacy in mind, models will learn *secrets* from the training data, e.g. credit card numbers are remembered by language models (Carlini et al., 2018), which can be catastrophic if the model parameters are shared.

Privacy can be achieved at two stages: at the data level or the model level. If the training data already fulfills a privacy criterion, the trained model will not violate that criterion. Adversarial learning (Goodfellow et al., 2014) can be used to obtain a private representation of the data. If the model is trained with raw data, the influence of a single training sample on the parameters needs to be limited so that the model does not learn any secrets. This overview focuses mostly on data level approaches.

Minimax filters (Hamm, 2015) are used in a data level approach that models a three-party game: contributors apply a filter to their data, a central aggregator learns a target task using this representation, and an adversary tries to identify contributors by the representation. An optimal filter minimizes inference and maximizes utility on the target task. When adding noise to the representation, the approach satisfies differential privacy (Hamm, 2017).

A neural implementation of a similar approach that was originally used for domain adaptation is the domain-adversarial neural network (DANN) (Ganin et al., 2016). To extrapolate from a source to a target domain, the DANN learns a domain-invariant intermediate representation of its inputs. This representation allows a model to learn a

[1]The data use agreement permits attempting this.

target task while preventing an adversary from identifying a sample's original domain. A *gradient reversal layer* is used in the backward pass to worsen the intermediate representation for the adversary[2].

Gradient reversal based adversarial learning has recently been applied to learning anonymized representations. Li et al. (2018) and Elazar and Goldberg (2018) train adversarial models to learn a representation of text that obscures the author's demographics. Elazar and Goldberg find that while the adversarial components in their models typically cannot learn the adversarial tasks in simultaneous training with the target task model, adversaries achieve better results when trained on a frozen representation. Feutry et al. (2018) introduce an improved alternating training procedure for anonymization.

Adversarial learning can further be used to e.g. censor identifying parts of an image (Edwards and Storkey, 2015). Using adversarially perturbed word embeddings can make a text classification model more robust (Miyato et al., 2016).

Federated learning (McMahan et al., 2016) is an approach that works by distributing a machine learning model to contributors that each send back parameter updates based on a summary of their private data. The raw private data is never transmitted.

The semi-supervised knowledge transfer approach (Papernot et al., 2016) requires contributors to train teacher models on their private data. A central student model is then trained on unlabeled data by imitating the predictions of the teacher ensemble.

On the model level, a differential privacy accountant can be applied to limit the effect a single sample has on the parameters (Shokri and Shmatikov, 2015; Abadi et al., 2016). When a large amount of contributors participates, it is possible to learn differentially private language models with this approach (McMahan et al., 2017).

[2]The gradient reversal layer can be replaced with two adversarial losses, see supplementary material to Ganin et al. (2016)

3 Background

This chapter provides an introduction to the core concepts used in this work: the de-identification task, deep learning for natural language processing, and pre-trained word representations.

3.1 The De-Identification Task

De-identification of medical text is an NLP sequence tagging task that has similarities to named entity recognition (NER). Given a sequence of words like

> Mr. Smith was admitted to St. Thomas on 2018/07/27,

de-identification labels all occurrences of protected health information (PHI) with the appropriate categories:

> Mr. $[\text{Smith}]_{\text{Patient}}$ was admitted to $[\text{St. Thomas}]_{\text{Hospital}}$ on $[2018/07/27]_{\text{Date}}$.

Difficulties of this task include lexical overlap of PHI and non-PHI words (e.g. *Parkinson*, which can be both a last name and a disease) and out-of-vocabulary PHI, like uncommon names or words with spelling mistakes (Uzuner et al., 2007). Misspellings and ungrammatical style occur in medical text because it is generally typed quickly, either by doctors themselves or by assistants who transcribe a doctor's dictation, and it is usually not proofread (Neamatullah et al., 2008). Additionally, medical text may contain bulleted or numbered lists and tabular data and is sometimes word-wrapped to a fixed line length.

A difference of de-identification in comparison to classic NER is the typically higher number of label categories. The CoNLL-2003 dataset (Tjong Kim Sang and De Meulder, 2003), an NER benchmark that is still widely used today, includes only four types of entity categories that need to be distinguished: persons, organizations, locations, and other. De-identification often spans dozens of labels. As the CoNLL-2003 dataset is derived from a corpus of newswire text, it also should be mostly grammatically correct and not contain many spelling mistakes.

Further, de-identification does not include an *other* label that is applied to any named entity outside of the main categories. It is important that not all *other* named entities are labeled because they can include condition or treatment names that are vital for the utility of a de-identified record. For example, in the sentence

> I asked her to double her atenolol in the morning,

"atenolol" is a named entity but not PHI, so it should not be labeled as such in the de-identification task.

De-identification shared tasks were run in 2006 (Uzuner et al., 2007), 2014 (Stubbs et al., 2015), and 2016 (Stubbs et al., 2017). The micro-averaged F_1 score (Section 3.2.7) is the main evaluation metric for the task. The shared task organizers suggest 95% as a rough estimate of a target F_1 score for systems that reliably de-identify medical text (Stubbs et al., 2015).

3.1.1 The 2014 i2b2/UTHealth Dataset

The i2b2 2014 dataset (Stubbs and Uzuner, 2015) was released as part of the 2014 i2b2/UTHealth shared task track 1 and is the largest publicly available dataset for de-identification today[3]. It contains 1304 free-text documents describing diabetic patients. The main difference to the 2006 shared task dataset is the inclusion of longitudinal records, i.e. multiple records per patient. The dataset spans 296 patients, on average 4.4 records per patient. To create the i2b2 dataset, raw medical records from a single source were prepared using manual annotation of PHI entities that were then replaced with suitable pseudonyms (e.g. person names were replaced with other person names, dates were offset by a random amount but date intervals were retained, misspellings were replaced with similar misspellings of the pseudonym).

The i2b2 dataset uses the 18 categories of PHI defined by HIPAA as a starting point for its own set of PHI categories. In addition to the HIPAA set of categories, it includes (sub-)categories such as doctor names, professions, states, countries, and ages under 90. Figure 3.1 lists the full set of i2b2 PHI categories.

Of the 1304 documents, 790 are used for training and 514 are used for testing. In total, the dataset contains 28,872 PHI tags.

- Name (patient, doctor, username)
- Profession
- Location (room, department, hospital, organization, street, city, state, country, ZIP, other)
- Age (over 90, under 90)
- Date
- Contact information (phone, fax, email, URL, IP address)
- IDs (Social Security number, medical record number, health plan number, account number, license number, vehicle ID, device ID, biometric ID, ID number)
- Other

Figure 3.1: Categories and subcategories of protected health information in the 2014 i2b2/UTHealth dataset, redrawn after Stubbs and Uzuner (2015).

[3]The dataset for the 2016 CEGS N-GRID shared tasks track 1 is not yet available as of October 2018.

	Entity		Token	
Prediction	Std	Bin	Std	Bin
She works in [software engineering]$_{\text{Prof.}}$	1/0/0	1/0/0	2/0/0	2/0/0
She works in software engineering	0/0/1	0/0/1	0/0/2	0/0/2
She [works]$_{\text{Prof.}}$ in software engineering	0/1/1	0/1/1	0/1/2	0/1/2
She works in [software]$_{\text{Org.}}$ engineering	0/1/1	0/1/1	0/1/2	1/0/1
She works in software [engineering]$_{\text{Prof.}}$	0/1/1	0/1/1	1/0/1	1/0/1
She works in [software engineering]$_{\text{Org.}}$	0/1/1	1/0/0	0/2/2	2/0/0
She works in [software]$_{\text{Prof.}}$ [engineering]$_{\text{Prof.}}$	0/2/1	0/2/1	2/0/0	2/0/0
She works in [software]$_{\text{Org.}}$ [engineering]$_{\text{Prof.}}$	0/2/1	0/2/1	1/1/1	2/0/0

Table 3.1: True positives / false positives / false negatives of de-identification predictions when evaluated with the standard (std) and binary (bin) variants of the entity and token evaluations. The first row is the target annotation.

3.1.2 Evaluation

In the de-identification shared tasks, there are two main types of evaluations that each have their own way of counting true positives, false positives, and false negatives in predictions. The entity (or strict) evaluation requires tags to match the exact offsets in the original text. The token evaluation tests the presence or absence of PHI predictions in whitespace-separated tokens. Both evaluations have binary variants in which the predicted category of PHI is disregarded. Table 3.1 shows the differences between the entity and token evaluations and their binary variants on a set of example predictions.

3.2 Deep Learning for Natural Language Processing

Deep learning (or deep neural networks) is a machine learning technique originally inspired by neural computation in the brain. This section is based on the works by Goodfellow et al. (2016, Chapters 5–10), who provide a great introduction to deep learning, as well as Goldberg (2017), who gives an overview over its use in NLP.

3.2.1 Machine Learning and Deep Learning

Machine learning algorithms are algorithms that learn from data. They tune their own parameters $\boldsymbol{\theta}$[4] in order to improve a performance score. With enough experience, they are ideally able to generalize to new, unseen data. With machine learning, we can tackle

[4]We use a similar notation to Goodfellow et al. (2016): italics x denote scalar variables or functions, bold lowercase italics $\boldsymbol{x}$ denote vectors, and bold uppercase italics $\boldsymbol{X}$ denote matrices.

problems that are too difficult for human-written programs (Goodfellow et al., 2016, Section 5.1).

A key characteristic of machine learning algorithms is that they are domain agnostic, as they work on real-valued vectors. The same algorithm can be used to process representations of text, images, video, genetic information, or other data.

Machine learning algorithms can be broadly divided into supervised and unsupervised algorithms by the types of data they learn from.

A supervised learning algorithm receives input features $\boldsymbol{X}$ and the corresponding outputs $\boldsymbol{y}$ to learn a function $f(\boldsymbol{x})$ that approximates the real output y. The algorithm optimizes its parameters $\boldsymbol{\theta}$ to minimize a loss function (e.g. mean squared error) that measures the quality of the approximation. This optimization is often based on the stochastic gradient descent algorithm. The most common supervised learning tasks are classification, in which an input is mapped to a known set of class labels $\{1, \ldots, k\}$, and regression, in which an input is mapped to a real-valued output vector.

An unsupervised algorithm receives unlabeled inputs $\boldsymbol{X}$ and extracts structural information from the data. Unsupervised tasks include clustering and outlier detection.

Deep learning models use a *deep* chain of differentiable operations that extract suitable features from the raw input data and then transform them to generate a prediction. These operations are called *layers*. The input is passed into the input layer, after which an arbitrary number of hidden layers is applied. The last layer is called the output layer.

In comparison to classic machine learning approaches, deep learning requires a much smaller amount of manual feature engineering. The resulting models can have millions or billions of parameters. To train a deep learning model, the back-propagation algorithm is used to compute the gradient with respect to the parameters of each layer, so that stochastic gradient descent can be applied.

Deep learning began to show promising results in fields like image processing in the early 2000s. A large contribution to its popularity in NLP was made by Collobert and Weston (2008), who presented a unified deep learning architecture for NLP tasks like language modeling and NER. Three years later, they showed that their deep learning based method was able to match or outperform (when re-adding some manually engineered features) the previous state of the art in several tasks (Collobert et al., 2011).

The following subsections introduce core concepts of deep learning for NLP.

3.2.2 From Sparse to Dense Representations

To use text data in machine learning, it has to be transformed into a real-valued vector representation. In contrast to image data that can be represented as pixel values, there is no natural *raw* representation for text data as text is inherently "*discrete*, *compositional*, and *sparse*" (Goldberg, 2017, Section 1.1, emphasis in original).

Feature engineering for text is most often applied to single words, windows of words, or full documents. An example of a simple engineered document feature is a Boolean value indicating the presence or absence of a certain word, e.g. *cat*. The usefulness of a feature depends on the concrete task: the *cat* feature might be useless in a sentiment

	Sparse representation				3D dense representation		
Word	s_1	s_2	s_3	s_4	d_1	d_2	d_3
cat	1	0	0	0	−0.13	0.42	0.26
dog	0	1	0	0	−0.16	0.33	−0.04
play	0	0	1	0	0.82	−0.12	0.39
playing	0	0	0	1	0.89	−0.08	0.37

Table 3.2: Sparse and dense representations for four example words.

analysis task while it could prove extremely useful if the task is to classify whether a text talks about pets.

Indicator features can be used to represent words as sparse *one-hot* vectors with size of the vocabulary in which all entries are 0 except the one corresponding to the respective word, which is 1. This representation does not capture similarities between words: *cat* and *dog* are conceptually similar (in so far as in they are both nouns and animals that are often kept as pets) but have the same vector distance[5] as the clearly conceptually more different pair *cat* and *playing*. If a new word is added to the vocabulary, an extra dimension needs to be added to the features as well. Even with a reasonable vocabulary size of e.g. 10 000, the feature vectors become huge and cause the *curse of dimensionality* (Goodfellow et al., 2016, Section 5.11.1) to come into play.

Deep learning methods for NLP rely on dense (or distributed) representations that reflect word similarities. These representations allow for better generalization: if a model has sufficient information about *cat* but has seen *dog* only a few times, it can still make adequate predictions about *dog* if the words have similar representations (Goldberg, 2017, Section 8.1).

Table 3.2 illustrates the difference between sparse and dense representations. While it is possible to hand-craft a dense representation using features like *"to which degree can it be used as a noun?"* or *"how animal-like is it?"*, these representations are typically obtained through unsupervised training on an auxiliary task using large, general-purpose datasets (see Section 3.3). The concrete dense representation in the table is a made up example but a similar representation could result from training.

The sparse feature s_1 can be interpreted as *"is cat?"*, analogously for s_2 through s_4. If the dense features are obtained through training, they do not allow such an obvious interpretation. Up to a certain limit, the three existing dimensions are capable to represent additional words.

Collobert and Weston (2008); Collobert et al. (2011) introduce an *embedding layer* as the input layer of their deep learning models. The layer looks up dense representations for input words in an embedding matrix which is trained together with the other layers. Several kinds of pre-trained word embeddings are discussed in Section 3.3.

[5] e.g. cosine similarity, see Section 3.3.1

3.2.3 Feedforward Models

Feedforward deep learning models (Goodfellow et al., 2016, Chapters 6–8) are composed of interconnected layers in which the output of a layer is only passed into subsequent layers, never backward. The most important layer types in feedforward models are dense layers and activation layers. Dense layers multiply their input with a matrix of weights and add a bias vector:

$$\boldsymbol{y} = \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{b} \tag{3.1}$$

The output size of a layer, which is determined by the dimension of its weight matrix and bias vector, is called the number of *units*, or historically *neurons*, of the layer. The parameters of a dense layer are its weight matrix and bias vector; when talking about the number of parameters, we mean the sum of the number of elements in the weight matrix and the bias vector.

Activation layers apply a nonlinearity ϕ:

$$\boldsymbol{y} = \phi(\boldsymbol{x}) \tag{3.2}$$

The most popular activation function for hidden layers is the rectified linear unit (ReLU) that discards any negative values:

$$\mathrm{ReLU}(\boldsymbol{x}) = \max\{0, \boldsymbol{x}\} \tag{3.3}$$

Before ReLU became popular, sigmoid functions like the logistic sigmoid σ and the hyperbolic tangent were often used. Their saturation behavior makes gradient-based learning difficult, so applying them is now generally advised against (Goodfellow et al., 2016, Section 6.3.2), except for use in output layer activation (see Section 3.2.7) and recurrent models (Section 3.2.4).

The sequence of a dense layer and an activation layer is often called a single layer. A feedforward model with an input layer and one dense and activation layer that acts as the output layer is called a perceptron network. When more hidden layers are added, it is called a multi-layer perceptron (MLP). Two visualizations of an MLP with a single hidden layer are shown in Figure 3.2.

Famously, MLPs with one hidden layer are able to solve the exclusive or problem that perceptron networks cannot learn. It has further been shown that an MLP with one hidden layer is a universal approximator, i.e. an MLP exists that can approximate an arbitrary function arbitrarily well (Hornik et al., 1989; Cybenko, 1989). However, this theorem has only theoretical significance as we are mainly interested in a model's generalization skills: for the theorem to hold, the hidden layer can act as a lookup table and does not need to *learn* anything. In practice, deeper models are more powerful than a single-layer MLP (Goodfellow et al., 2016, Section 6.4).

As feedforward models have a fixed input size and no way to access past inputs, sliding window approaches are often used to allow them to process variable-length text data.

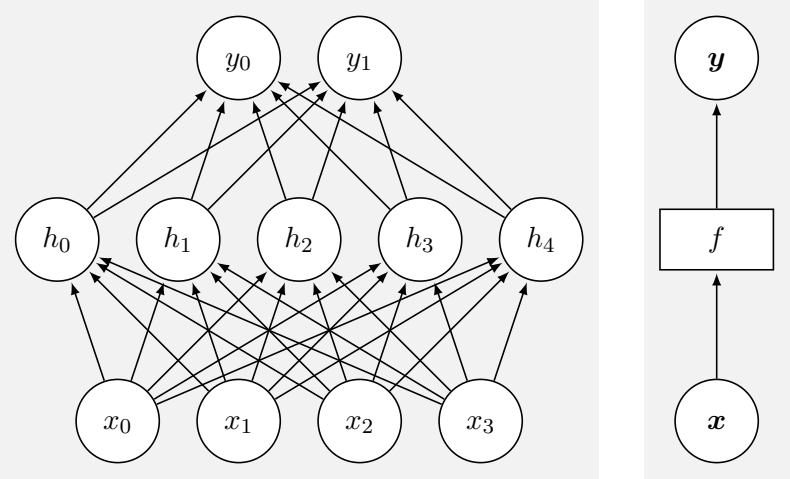

Figure 3.2: Left: an MLP with one dense hidden layer in the traditional representation of connected units. Right: the MLP in a condensed layer view. The hidden layer is now labeled with a function symbol f instead of the vector $\boldsymbol{h}$.

3.2.4 Recurrent Models

Recurrent neural networks (RNNs) (Goodfellow et al., 2016, Chapter 10) extend feedforward models with backward connections. They can reuse the outputs of their layers in following time steps. RNNs naturally accept variable-length sequential input data and produce variable-length output.

A classic RNN architecture that we will use as an example in this section has one recurrent hidden layer whose output is fed back into it in the next time step. Figure 3.3 shows such an RNN along with an unrolled visualization. A core concept of RNNs is parameter sharing: they use the same hidden layer f in all time steps instead of learning new models for every time step. This also enables them to generalize to new sequence lengths (Goodfellow et al., 2016, Section 10.1).

When using recurrent layers in deep learning models, the outputs from intermediate time steps can either be discarded or passed to further layers in the model. Many-to-one tasks like sequence classification use models in which intermediate outputs are discarded: for them, we are only interested in an encoding that summarizes the whole sequence to pass it to further layers. Keeping the outputs is required in sequence tagging tasks like NER in which we need the model to provide an output for every element in the input sequence. Outputs are also kept if the following layer is another recurrent layer that again processes a sequence of inputs.

A common extension to RNN models is the use of bidirectional processing in which the hidden layer is split into a forward and backward part. The forward part processes the sequence as-is while the backwards part processes the sequence in reverse order. The outputs of both parts are concatenated.

For fixed-length input, feedforward models (Section 3.2.3) are preferred to recurrent

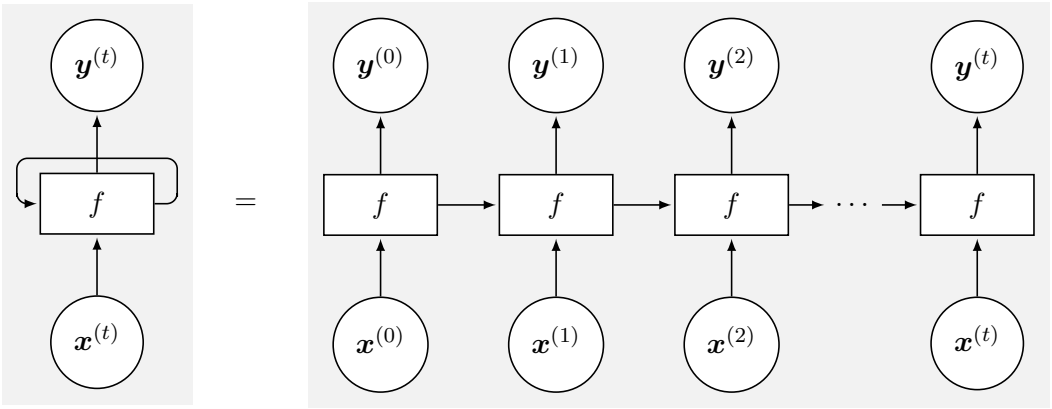

Figure 3.3: Left: an RNN model with inputs $\boldsymbol{x}^{(t)}$, a hidden layer f and outputs $\boldsymbol{y}^{(t)}$. Right: an unrolled visualization of the model. Redrawn after Olah (2015).

models because they are computationally less expensive. RNNs cannot be parallelized as the calculation of every time step depends on the previous time step's output. The gradients in an RNN model are calculated with the back-propagation-through-time algorithm which is more expensive than standard back-propagation.

While RNNs are theoretically able to capture arbitrary long-term dependencies, it is very hard in practice to train an RNN of the above architecture to keep track of previous inputs for more than a couple of time steps. This challenge stems from the vanishing gradient problem: with every time step, back-propagated gradients get smaller, if they are originally small. Analogously, the exploding gradient problem describes gradients becoming larger through back-propagation if they are originally large (Bengio et al., 1994; Goodfellow et al., 2016, Section 10.7).

3.2.5 Long Short-Term Memory

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) is a recurrent architecture that tackles the vanishing gradient problem by incorporating a gated cell state. This cell state is passed to the next time step in addition to the layer output, allowing a stable gradient flow over long durations. The behavior of the gates to the cell state is learned from data, so it can adapt to different time scales.

Figure 3.4 shows an LSTM block. It receives three inputs: the input from the current time step $\boldsymbol{x}^{(t)}$ and the previous output $\boldsymbol{h}^{(t-1)}$ are concatenated on the lower left corner, the previous cell state $\boldsymbol{C}^{(t-1)}$ flows at the top and has only linear interactions. The block has three gates: a forget gate $\boldsymbol{f}$, an input gate $\boldsymbol{i}$, and an output gate $\boldsymbol{o}$ that are all vectors with entries $\in [0, 1]$. The gates' values are calculated by deep learning layers with sigmoid activations (σ rectangles). The forget gate decides to what extent each previous cell state entry should be forgotten, the input gate decides to what extent each current input entry should contribute to the corresponding cell state entry, and the output gate decides to what extent each current cell state entry should contribute to the corresponding entry in

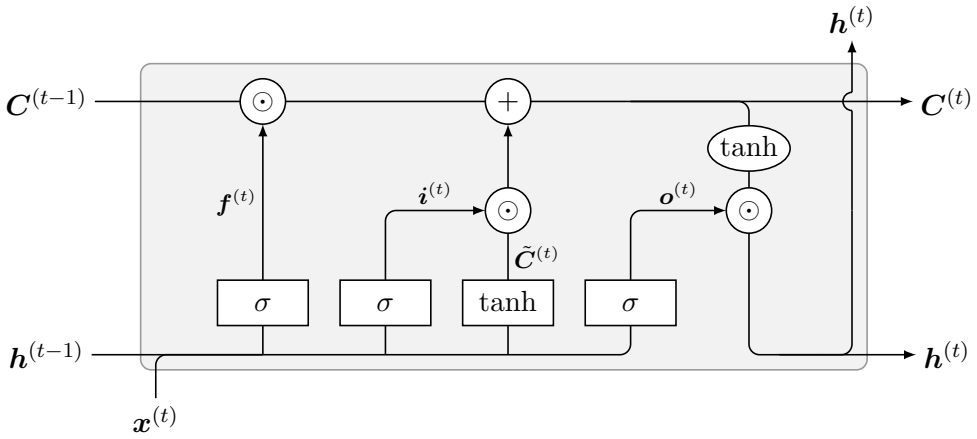

Figure 3.4: An LSTM block, redrawn after Olah (2015). Deep learning layers have rectangle shapes, vector operations have ellipse shapes.

the output vector.

A dense layer with hyperbolic tangent activation calculates the vector $\tilde{\boldsymbol{C}}^{(t)}$ that contains candidate values for the current cell state. The current cell state is a linear combination of the previous cell state $\boldsymbol{C}^{(t-1)}$ and the candidate vector.

The LSTM block's behavior is described by the following equations[6] (Olah, 2015):

$$\boldsymbol{f}^{(t)} = \sigma(\boldsymbol{W}_f[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_f) \tag{3.4}$$

$$\boldsymbol{i}^{(t)} = \sigma(\boldsymbol{W}_i[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_i) \tag{3.5}$$

$$\tilde{\boldsymbol{C}}^{(t)} = \tanh(\boldsymbol{W}_C[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_C) \tag{3.6}$$

$$\boldsymbol{C}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{C}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \tilde{\boldsymbol{C}}^{(t)} \tag{3.7}$$

$$\boldsymbol{o}^{(t)} = \sigma(\boldsymbol{W}_o[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + \boldsymbol{b}_o) \tag{3.8}$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{C}^{(t)}) \tag{3.9}$$

A great part of the success of RNNs can be attributed to the LSTM architecture. Its ability to capture long-term dependencies is illustrated e.g. by the character-level language models trained by Karpathy (2015). One of his models that was trained on the Linux kernel source code is capable of producing syntactically almost correct C code. When analyzing the entries in the cell state at different time steps, he finds certain units that turn on inside quotes or keep track of the current indentation level.

3.2.6 Convolutional Models

A layer type that can be used for feature extraction on grid-like structured data like image pixel matrices but also on sequences of word representations is the convolution

[6] $[\cdot, \cdot]$ denotes vector concatenation, $\odot$ is the element-wise multiplication operator

layer (Goodfellow et al., 2016, Chapter 9). It performs the convolution operation with learned kernels on its input. Deep learning models that use convolution layers are called convolutional neural networks (CNNs).

In contrast to dense layers, convolutional layers are sparsely connected: each output depends only on a small number of input entries which is defined by the kernel size. They make use of parameter sharing, as the same kernel weights are used at every position. This makes them efficient to compute (Goodfellow et al., 2016, Section 5.2).

CNNs of the time-delay neural network architecture (Waibel et al., 1990) were among the first deep learning models to be applied to text (Collobert and Weston, 2008; Collobert et al., 2011). These models process a sequence of word representations using 1-dimensional convolution over the time dimension.

Recently, convolution layers have applied to sequences of character embeddings as they can learn to extract morphological information (Chiu and Nichols, 2016; Ma and Hovy, 2016). The max-pooled convolution outputs are then often concatenated with pre-trained word embeddings (see Section 3.3).

3.2.7 Training Deep Learning Models

This section describes characteristics of training deep learning models.

Output Layer Unlike the hidden layers of deep learning models that allow for much architecture experimentation, the choice of output layer is often predetermined by the task (Goodfellow et al., 2016, Section 6.2). In regression tasks, a linearly activated dense layer is used. Binary classification is implemented with a sigmoid-activated dense layer with one unit that outputs a Bernoulli distribution. In classification with multiple labels, a softmax-activated dense layer is used. Softmax is a differentiable variant of argmax that normalizes the output to sum 1 so it can be interpreted as a categorical probability distribution (Goodfellow et al., 2016, Section 6.2.2):

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{3.10}$$

A conditional random field (CRF) layer (Sutton et al., 2012) is sometimes used as an output layer in sequence tagging. It keeps track of a transition table of labels to generate coherent output sequences.

Loss Function The loss is a differentiable function that measures the quality of a deep learning model's predictions. Minimizing the loss function is the goal of training. Mean squared error loss is often used in regression; (binary) cross-entropy loss is used in classification tasks. It is notable that the loss function is often not the performance measure we are ultimately interested in, e.g. if the target score is classification accuracy, a model would still be trained using cross-entropy (Goodfellow et al., 2016, Section 8.1).

Evaluation The F_1 score is often used as an evaluation metric for classification tasks,

including sequence tagging. It is the harmonic mean between precision and recall:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{3.11}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{3.12}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.13}$$

Micro-averaging and macro-averaging are two common variants of the F_1 score. Micro-averaging refers to calculating the precision and recall by taking the total number of true positives, false positives, and false negatives over all documents into account. In the macro-averaging variant, precision and recall are calculated per document (or sometimes per class) and then averaged.

In tasks like NER and de-identification that have a strong majority class that is arguably less important than other classes (the none class, sometimes called O class), this class is excluded in the precision and recall calculation to make the scores more meaningful. A model that assigns all tokens the none class achieves a 0% F_1 score.

Optimization Deep learning models are typically trained with minibatches over multiple epochs. At the start of every epoch, the training samples are shuffled and split into minibatches (often also called batches). A minibatch is fed through the model before a stochastic gradient descent algorithm computes a combined update to the parameters based on the mean error of the minibatch's samples. After all training samples are processed once, the next training epoch begins (Goodfellow et al., 2016, Chapter 8).

Today, modern variants of standard stochastic gradient descent that feature adaptive learning rates are preferred over the standard algorithm (Ruder, 2016). Examples include the Adam algorithm (Kingma and Ba, 2014) and Nadam (Dozat, 2016), a variant incorporating Nesterov momentum.

Regularization Regularization is applied to machine learning models to manipulate their bias/variance trade-off. A good regularizer should reduce the variance more than it increases the bias (Goodfellow et al., 2016, Section 5.4).

Applying parameter norm penalties to a model generally reduces its tendency to overfit (Goodfellow et al., 2016, Section 7.1). Early stopping (Goodfellow et al., 2016, Section 7.8) is a regularizer that determines the optimal number of training epochs by observing the loss progression on a validation set. Dropout (Srivastava et al., 2014) is a regularizer specific to deep learning models that sets entries of weight matrices to zero according to a random variable to prohibit entries from *conspiring* to memorize the training data. For recurrent connections, variational dropout (Gal and Ghahramani, 2016) that applies the same dropout mask throughout all timesteps is preferred to using

new random masks in every step. Gradient normalization (Pascanu et al., 2013) is often used in recurrent models to help avoid the vanishing or exploding gradient problem.

Regularizers that increase model robustness include dataset augmentation (Goodfellow et al., 2016, Section 7.4) and adversarial training (Goodfellow et al., 2016, Section 7.13).

Hyperparameters Any non-weight parameters of a model (i.e. parameters that are not learned with stochastic gradient descent) are called hyperparameters. These include the number and sizes of hidden layers, the minibatch size, the learning rate, dropout probabilities, and potentially many more. Hyperparameters are often tuned manually or using a grid or random search, evaluating each configuration's performance on a validation set. Only the model with the best validation set performance is evaluated on the test set (Goodfellow et al., 2016, Section 11.4). This makes sure that the model is not tuned specifically for the test set, which may result in artificially high test scores.

3.3 Pre-Trained Word Embeddings

Distributed word representations were popular before their use in deep learning models. The term *word embeddings* was introduced by Bengio et al. (2003) who trained a neural language model. Today, there are several kinds of word embeddings available that are trained on large quantities of unlabeled text. Using pre-trained embeddings is a transfer learning technique: even if the embeddings were trained on an auxiliary task on a different dataset, they encapsulate knowledge about the structure of language that makes it easier to achieve good results on the target task with smaller amounts of data (Goldberg, 2017, Chapter 10).

The following subsection introduces some characteristics of working with word embeddings. Then, we briefly introduce the classic word embedding algorithms Word2vec and GloVe as well as two modern approaches to word embeddings that each use subword features to provide embeddings even for unknown words. For each algorithm, the authors published a set of pre-trained embeddings that are compared in Table 3.3.

3.3.1 Working with Word Embeddings

When working with word embeddings, we are interested in measuring embedding similarity and using linear relationships for word analogy tasks.

Embedding Similarity The cosine similarity metric is often used to measure the similarity between two embedding vectors (Goldberg, 2017, Chapter 10). It takes values between -1 (meaning the vectors are pointing in opposite directions) and 1 (the vectors are pointing in the same direction). The cosine similarity is calculated as the dot product of two L^2 normalized vectors:

$$\mathrm{sim}_{\cos}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|} \tag{3.14}$$

It is common to store L^2 normalized word vectors in an embedding matrix $\boldsymbol{M}$ to skip the normalization step in similarity computations. The similarity of a word w to all other words in the embedding matrix can then be calculated as the dot product of the matrix $\boldsymbol{M}$ and the word embedding $\boldsymbol{M}_{[w]}$:

$$\text{sim}_{\cos}(w, \boldsymbol{M}) = \boldsymbol{M} \cdot \boldsymbol{M}_{[w]} \tag{3.15}$$

Linear Relationships When trained on large datasets, embeddings capture linear relationships in the vector space, which is somewhat surprising because it is not always a direct goal of the training[7] (Mikolov et al., 2013c). For example, the **capital** relationship can be approximated as the vector between the capital of a country and the country name in the embedding space:

$$\textbf{capital} = \text{vec}(\textit{Berlin}) - \text{vec}(\textit{Germany}) \tag{3.16}$$

When adding the relationship vector to a different country name, the resulting vector is typically close to the actual embedding of its capital:

$$\text{vec}(\textit{France}) + \textbf{capital} \approx \text{vec}(\textit{Paris}) \tag{3.17}$$

3.3.2 Word2vec

Word2vec (Mikolov et al., 2013a,b) is a predictive word embedding method. In its skip-gram variant, the algorithm trains a language model that predicts context words from an input. As a faster alternative to (hierarchical) softmax over all possible words, it uses negative sampling to transform the task into a binary classification task, i.e. the model learns to distinguish between real and fake word-context pairs. To obtain the embeddings, the classification layer is stripped from the model, leaving only the embedding layer whose weights are the embedding matrix.

3.3.3 GloVe: Global Vectors for Word Representation

GloVe (Pennington et al., 2014) is an embedding algorithm that trains word embeddings on a co-occurrence matrix of words. It produces embeddings that outperform Word2vec on word similarity tasks and NER. The authors released several sets of pre-trained word vectors, including a set of 400 000 vectors trained on the English Wikipedia and Gigaword corpus that we use in this work.

3.3.4 FastText

FastText (Bojanowski et al., 2016) is an embedding method that takes subword information into account. This resolves one drawback of Word2vec and GloVe: when using these algorithms, morphologically similar words do not necessarily have similar representations (e.g. *disastrous*, *disaster*). FastText uses character n-grams as input features for the

[7] It is a goal for the GloVe algorithm (Section 3.3.3).

Embeddings	Dimensions	# Training tokens	Handles unknown words
GloVe	300	6B	No
FastText	300	16B	Yes
ELMo	1 024	800M	Yes

Table 3.3: Attributes of the pre-trained embeddings used in this work.

skip-gram task and calculates word embeddings as the sum of their character n-gram embeddings. This allows the model to generate embeddings even for out-of-vocabulary words like rare words or misspellings.

In 2018, the FastText team released a set of one million 300-dimensional vectors trained on the English Wikipedia (Mikolov et al., 2018).

3.3.5 ELMo: Deep Contextualized Word Representations

Embeddings from language models (ELMo) (Peters et al., 2018) is a recent embedding method that generates embeddings as a function of a whole sentence instead of a single word. Due to the whole sentence being used as context, the verb *play* and the noun *play* will have different embeddings. Also, *play* will have a different embedding when it is used in a sports context than when it is used in a theater context.

The authors trained a two-layer bidirectional LSTM language model with subword features that are obtained from convolutions of character embeddings. The ELMo embeddings are a weighted sum of the LSTM layers' outputs with task-specific trained weights. Simple deep learning models that use ELMo representations were able to beat the previous state of the art in several NLP tasks.

4 Methodology

We are using an experimental approach to tackle the research question defined in Section 1.1. This chapter describes the methodology that is applied throughout our experiments.

Data Preprocessing We use the 2014 i2b2/UTHealth dataset (Section 3.1.1) that is provided as a set of XML documents with standoff annotations. To use it in deep learning models, we apply tokenization, assign the standoff labels to the matching tokens, and split the documents into sentences. Our models operate on sentences instead of whole documents because deciding if a token is PHI is in most cases possible only from tokens and their sentence context, which is shown by sentence-based state-of-the-art de-identification models (Dernoncourt et al., 2017b; Liu et al., 2017). Since some PHI occurs mid-word, even aggressive tokenization will not always split the PHI into a separate token. Sentence splitting poses a challenge in ungrammatical medical notes that include bulleted lists, tabular data, and manually word wrapped lines. We use additional heuristics to perform sentence splitting. Their shortcomings are most notable when PHI sequences are inadvertently split into two sentences. More details about the preprocessing can be found in Appendix A.

Prediction Postprocessing To evaluate our models' predictions using the official evaluation script[8], we transform the predictions back into the original XML format with standoff annotations. The postprocessing steps are described in more detail in Appendix A.

Evaluation We use the micro-averaged F_1 score (see Sections 3.1.2 and 3.2.7) from the binary token-based evaluation including only the HIPAA categories of PHI as our main metric for de-identification performance. Using this metric allows us to compare our results to Dernoncourt et al. (2017b) who deem it the most important metric: deciding if an entity is PHI or not is generally more important than assigning the correct category of PHI, and only HIPAA categories of PHI are required to be removed by American law.

Practical Upper Bound Our tokenization and sentence splitting is responsible for an upper bound on de-identification performance. In the entity-based evaluation, 94.25% is the highest possible binary HIPAA F_1 score on the i2b2 test set using our preprocessing. However, for the token-based evaluation (which is most important to us), the highest possible F_1 score is 99.45%.

Training, Validation, and Test Set While we are developing the models and tuning the hyperparameters, we use a fixed, relatively large validation set that has no overlapping

[8] `https://github.com/kotfic/i2b2_evaluation_scripts`

patients with the training set. We use the test set only for the final evaluation. At this time, we combine the training set and validation set and use a smaller validation split.

Implementation To conduct our experiments, we implemented a software library for de-identification in the Python programming language[9]. Experiments are defined in a YAML[10] configuration format. When experiments are run, they output a detailed training and testing history including results from the i2b2 evaluation script. For hyperparameter search, concrete YAML configurations can be generated from templates. The source code of our software library is available on GitHub[11].

Dependencies We use spaCy[12] (Honnibal and Montani, 2017) for text preprocessing, including tokenization and sentence splitting. Our deep learning models are implemented with the Keras framework[13] (Chollet et al., 2015) using the TensorFlow[14] (Abadi et al., 2015) backend. We use a CRF layer implementation from the Keras community contributions repository[15]. Additionally, we use Numpy[16] for matrix operations such as cosine similarity computations, and Matplotlib[17] for plotting.

[9] https://python.org
[10] http://yaml.org
[11] https://github.com/maxfriedrich/deid-training-data
[12] https://spacy.io
[13] https://keras.io
[14] https://tensorflow.org
[15] https://github.com/keras-team/keras-contrib
[16] https://numpy.org
[17] https://matplotlib.org

5 Experiments

In this chapter, we present our experiments. Our representation approaches for exchange of training data for de-identification are compared in Table 5.1.

5.1 De-Identification Baseline

The goal of our baseline experiment is to achieve similar performance to the state-of-the-art de-identification model by Dernoncourt et al. (2017b) without relying on explicit character features. Sharing training data that includes character features could result in PHI being easy to re-identify.

Dernoncourt et al.'s model achieves an F_1 score of 97.85% in the binary HIPAA evaluation on the i2b2 test set. In their ablation study, they evaluate a model that relies only on GloVe word embeddings (Pennington et al., 2014) and no character features which scores around 88%. We try to reach the common target score for reliable de-identification models, an F_1 score of 95%.

5.1.1 Model Selection

In this subsection, we describe the model selection process that leads to our final models.

Basic Architecture Our de-identification models use a bidirectional LSTM-CRF architecture (see Sections 3.2.4 and 3.2.7) that has proven to work well for sequence tagging tasks, including NER (Huang et al., 2015; Lample et al., 2016) and de-identification (Dernoncourt et al., 2017b; Liu et al., 2017). The inputs are an embedding sequence that is

Approach	Representation	Hyperparameters
5.1 ⋄	Raw	–
5.2.1	Replace all PHI with random vectors	–
5.2.2	Add noise to all PHI	Noise scale
5.2.3	Move all PHI to neighbors	# neighbors
5.3	Adversarially trained, invariant to moving one PHI token to neighbor	# neighbors, representation size

Table 5.1: Comparison of the representations in our experimental approaches. The approach marked with ⋄ tackles the de-identification task without data protection concerns.

obtained from an embedding algorithm and an optional casing feature. The output is a sequence of probability distributions. Our models use CRF output layers whose outputs can be interpreted as probability distributions over the classes. They are optimized using the categorical cross-entropy loss function with balanced class weights (PHI tokens are weighted around 13.5 times higher than non-PHI tokens). We apply an early stopping regularizer that monitors validation loss with patience 5.

Hyperparameter Optimization We are exploring the hyperparameter space shown in Table 5.2. It is based on Dernoncourt et al.'s model as well as the findings by Reimers and Gurevych (2017) who evaluated over 50 000 hyperparameter configurations for various sequence tagging tasks including NER. We focus on the hyperparameters that have a large influence on sequence tagging performance according to Reimers and Gurevych: pre-trained embeddings, additional inputs, batches, and dropout. The remaining hyperparameters such as the number of LSTM units per direction and layer, the choice of CRF output and Nadam optimizer as well as the gradient normalization configuration are kept fixed. As the space shown in Table 5.2 spans thousands of combinations, we use a random search instead of a grid search to explore the space, optimizing for the binary HIPAA F_1 score on a validation set.

Embeddings We evaluate GloVe, FastText and ELMo embeddings (see Section 3.3). GloVe embeddings are used by Dernoncourt et al. (2017b) in their de-identification model. They have no way of embedding out-of-vocabulary words. FastText and ELMo are capable of producing embedding vectors for out-of-vocabulary tokens using subword information while still keeping similar words like names or professions close together in the vector space.

Casing Feature We evaluate the casing feature by Reimers and Gurevych (2017) as an additional input. The feature maps words to a one-hot representation of their casing (*numeric*, *mainly numeric*, *all lower*, *all upper*, *initial upper*, *contains digit*, or *other*).

Batches In sequence tagging tasks, it is common to use relatively small batch sizes because multiple labels are learned for each sentence. As Dernoncourt et al. (2017b) use a batch size of 1, we include this value in our hyperparameter space.

Dropout Reimers and Gurevych achieve the best results when using variational dropout as well as naive dropout (see Section 3.2.7) after LSTM layers. We additionally evaluate the effect of dropout on the input embeddings.

5.1.2 Final Model

Our bidirectional LSTM-CRF models are capable of learning the de-identification task. We find that the architecture is mostly insensitive to its hyperparameters. Detailed results of our hyperparameter search can be found in Appendix B.

Models that use two recurrent layers score 0.3 percentage points higher than models using one layer, so we define two LSTM layers as our model depth. Since there is no

Category	Hyperparameter	Values
Inputs	Pre-trained embeddings	FastText, ELMo, GloVe
	Casing feature	Yes, no
	Batch size	1, 16, 32, 64
Architecture	Number of LSTM layers	1, 2
	LSTM units per direction and layer	128 (fixed)
Training	Input embedding dropout	0, 0.05, 0.1, 0.25, 0.5
	Variational dropout	0.1, 0.25, 0.5
	Dropout after LSTM	0.1, 0.25, 0.5
	Optimizer	Nadam (fixed)
	Optimizer gradient norm clipping	1.0 (fixed)

Table 5.2: Hyperparameter space for the baseline model.

clear best embedding type in our hyperparameter evaluation, we continue to evaluate all three types of pre-trained word embeddings. The casing feature adds 0.4 percentage points to the F_1 score for GloVe and does not degrade FastText and ELMo, so we keep it for further experiments. A minibatch size of 32 achieved good results, conforming to Reimers and Gurevych's findings for NER. We find that a reasonable configuration for input dropout, variational dropout, and after LSTM dropout is $(0.1, 0.25, 0.5)$. Figure 5.1 shows the final model architecture.

5.2 Perturbing Protected Health Information Tokens

The experiments described in this section use perturbed training data with the goal of finding a representation of the data that is reasonably private and allows training a de-identification model. Testing is still performed with the unperturbed data.

We do not use ELMo embeddings in our further approaches because they are expensive to compute and we only expect them to achieve minor improvements in de-identification performance compared to the non-contextual types of pre-trained embeddings.

5.2.1 Random Embeddings

In our random embeddings experiment, we aim to find out to which degree PHI tokens can be identified solely from the sentence context and the casing feature without embedding information about the tokens in question. We loop over all PHI tokens in the training set except punctuation marks and replace their embeddings with random vectors that share the mean and standard deviation of the respective embedding matrix. As the replaced embeddings contain no information about the original tokens, we replace them with new random vectors in every training epoch. This experiment serves as a lower bound for perturbation strategies.

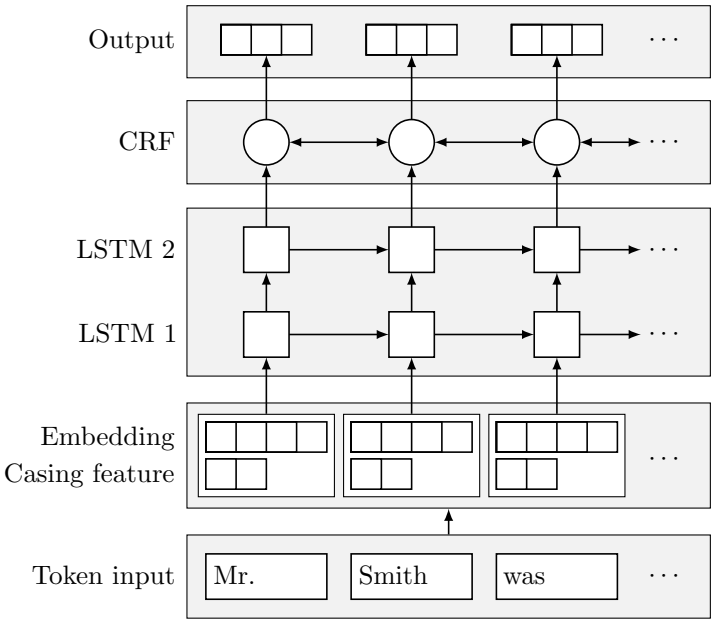

Figure 5.1: Visualization of the final baseline model architecture. Sequences of squares denote real-valued vectors. Backward LSTMs are omitted for legibility.

5.2.2 Additive Noise

Adding random noise to numerical data is a natural way to obfuscate it. However, in high dimensional spaces such as word embedding spaces, points are far away from each other and almost all of them are located at some edge of the space (Aggarwal et al., 2001; Domingos, 2012). This means that even high amounts of additive noise (multiples of the standard deviation of the embedding matrix) may not change the cosine distance neighborhood of a point, making it easy to re-identify.

In this experiment, we aim to find a fitting amount of noise that changes an embedding's cosine similarity neighborhood while still producing realistic-looking vectors (i.e. they should not stray too far from other elements in the embedding space). Then, we train a de-identification model on embeddings with additive noise. Embeddings of PHI tokens are perturbed by drawing a value for each embedding dimension from a Gaussian distributed random variable with a single fixed standard deviation and adding the resulting vector to the original embedding. The noise is added only once before training; we do not generate new noisy embeddings between training epochs.

5.2.3 Automatically Pseudonymized Data

As a further perturbation method, we evaluate a naive automatic approximation of pseudonymization by substitution. Before training, we randomly move all PHI tokens to the position of one of a fixed number N of their neighbors in an embedding space, as

determined by cosine distance in a pre-computed embedding matrix.

In GloVe, only tokens that exist in the pre-computed embedding matrix are moved to their neighbors. The unknown token is not modified as it does not contain any information (except that the token in question is not part of the precomputed matrix).

We evaluate the privacy properties of the approach using a bidirectional LSTM adversary with a single output unit. It is trained on the tasks of distinguishing pairs of:

- a pseudonymized sequence and the original sequence
- a pseudonymized sequence and a minimally modified original sequence (with only one occurrence of PHI moved to one of its neighbors).

For the minimally modified original sequence, we keep the number of neighbors fixed at $N = 5$ to avoid augmented sentences being too unrealistic.

5.3 Adversarial Learning of a Private Representation

The previously discussed approaches are highly dependent on manually tuned representation parameters like the amount of noise and neighbor space, that, if set incorrectly, may allow for easy re-identification. In this experiment, we evaluate an adversarial learning based approach that automatically tunes the representation parameters to protect against adversary models.

Architecture Our approach uses a model that is composed of three components: a representation model, a de-identification model, and an adversary. An overview of the architecture is shown in Figure 5.2. The representation model maps a sequence of word embeddings to an intermediate vector representation sequence. The de-identification model receives this representation sequence as an input instead of the original embedding sequence. It retains the casing feature as an additional input. As before, the de-identification model outputs a sequence of class probabilities. The representation is also used as an input to the adversary that tries to infer information about the original embedding sequence.

Representations We evaluate two types of representation models: a feedforward and an LSTM model. Both apply Gaussian noise with zero mean and trainable standard deviations to their inputs and outputs. The models learn a standard deviation for each of the input and output dimensions.

We try different representation sizes to explore the trade-off between de-identification and adversary performances. In contrast to the approaches from Section 5.2 that only perturb PHI tokens, the representation models in this approach process all tokens to represent them in a new embedding space.

Adversaries In existing gradient reversal approaches (Ganin et al., 2016; Feutry et al., 2018; Elazar and Goldberg, 2018), the learned representation is invariant to some attribute of the input. Similarly, our representation should be invariant to small input changes,

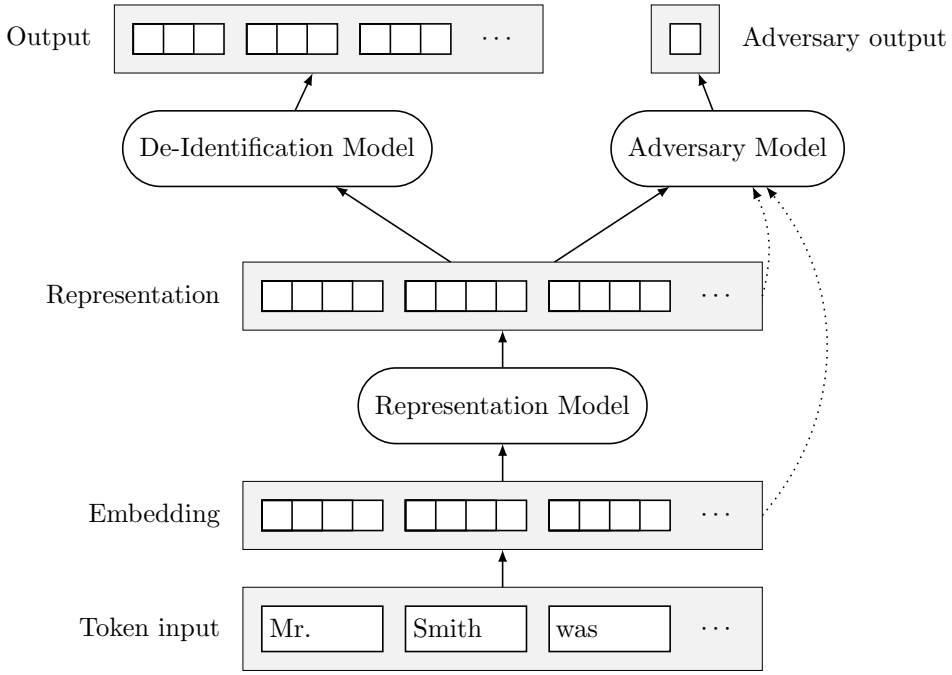

Figure 5.2: Simplified visualization of the adversarial model architecture. Sequences of squares denote real-valued vectors, dotted arrows represent possible additional real or fake inputs to the adversary. The casing feature that is provided as a second input to the de-identification model is omitted for legibility.

like a single token being replaced with a neighbor in the embedding space. The number of neighbors N controls the privacy properties of the representation.

Additionally, we need our representation to contain a random element because we want to share the output representations as well as the representation model itself. An attacker should not be able to create a lookup table of representations for exact sentences, i.e. the representation must be immune to known-plaintext attacks.

To achieve these goals, we use two adversaries that are trained for the following tasks:

1. Given a representation and an embedding sequence, decide if they were obtained from the same sentence.
2. Given two representation sequences (and their cosine similarities), decide if they were obtained from the same sentence.

Figure 5.4 shows the two adversaries with their respective inputs. The first adversary's objective is a discriminatory formulation of an inverse representation model and causes representations for similar inputs (replacing any protected token with one of its N neighbors) to be indistinguishable. The second adversary's objective causes repeated representation computations for the same sentence to differ by a high enough degree to make it impossible to build a lookup table of representations. We obtain the representation sequences for the second adversary from copies of the representation model with shared weights. We generate real and fake pairs for adversarial training using the automatic pseudonymization approach presented in Section 5.2.3, limiting the number of replaced tokens to one per sentence.

The adversaries are implemented as bidirectional LSTM models. We confirmed that bidirectional LSTM models are able to learn the adversarial tasks on randomly generated data and raw word embeddings in a preliminary experiment. To use the two adversaries in our architecture, we average their outputs.

Training We evaluate two training procedures: DANN training (Ganin et al., 2016) and the alternating approach by Feutry et al. (2018).

In DANN training, the three components are trained conjointly, optimizing the sum of losses. Training the de-identification model modifies the representation model weights to generate a more meaningful representation for de-identification. The adversary gradient is reversed with a gradient reversal layer between the adversary and the representation model in the backward pass, causing the representation to become less meaningful for the adversary.

The training procedure by Feutry et al. (2018) is shown in Figure 5.3. It is composed of three sequential phases:

1. The de-identification and representation models are pre-trained together, optimizing the de-identification loss L_{deid}.
2. The representation model is frozen and the adversary is pre-trained, optimizing the adversarial loss L_{adv}.

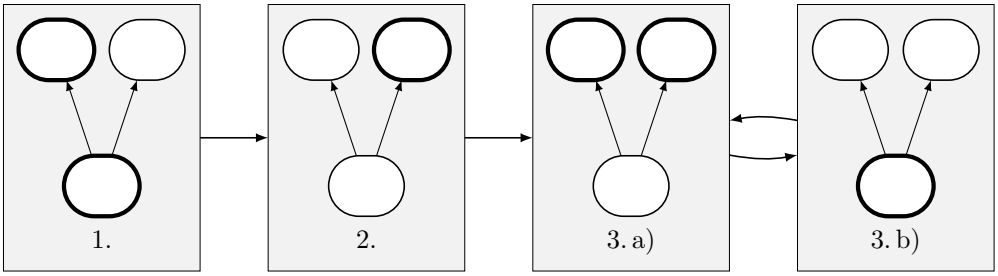

Figure 5.3: Visualization of Feutry et al.'s training procedure. The adversarial model layout follows Figure 5.2: the representation model is at the bottom, the left branch is the de-identification model and the right branch is the adversary. In each step, the thick components are trained while the thin components are frozen. Steps 1 and 2 are trained until stable. Then training alternates between one epoch and step 3a and one epoch of step 3b.

3. In alternation, for one epoch each:
   a) The representation is frozen and both de-identification model and adversary are trained, optimizing their respective losses L_{deid} and L_{adv}.
   b) The de-identification model and adversary are frozen and the representation is trained, optimizing the combined loss $L_{\text{repr}} = L_{\text{deid}} + \lambda|L_{\text{adv}} - L_{\text{random}}|$.

In the first two phases, we monitor the respective validation losses for early stopping to decide at which point the training should move on to the next phase. The alternating steps in the third phase each last one training epoch. We determine the early stopping epoch using only the combined validation loss (Item 3b).

Gradient reversal is achieved by optimizing the combined representation loss while the adversary weights are frozen. The combined loss is motivated by the fact that the adversary performance should be the same as a random guessing model, which is a lower bound for anonymization (Feutry et al., 2018). The term $|L_{\text{adv}} - L_{\text{random}}|$ approaches 0 when the adversary performance approaches random guessing[18]. λ is a weighting factor for the two losses; we select $\lambda = 1$.

Application To apply the model in practice, a central model provider would train the three parts of the model on an initial PHI-annotated dataset, e.g. the i2b2 2014 data. This initial training should confirm that the learned representation allows training a de-identification model while being robust to the adversaries. The model provider would then publish the representation model along with their choice of pre-trained word embeddings. Medical institutions would use the representation model to transform their PHI-labeled data into a private representation, which is then sent back to the central model provider with the respective labels. This transformation replaces the manual document-coherent pseudonymization that is typically performed to share training data for de-identification.

[18] In the case of binary classification: $L_{\text{random}} = -\log \frac{1}{2} \approx 0.6931$.

The model provider would then update the existing de-identification model or train a new model using all available representation data. Periodically, the pipeline of representation model (possibly in a version without additive noise) and de-identification model would be published so it can be used by medical institutions on their unlabeled data.

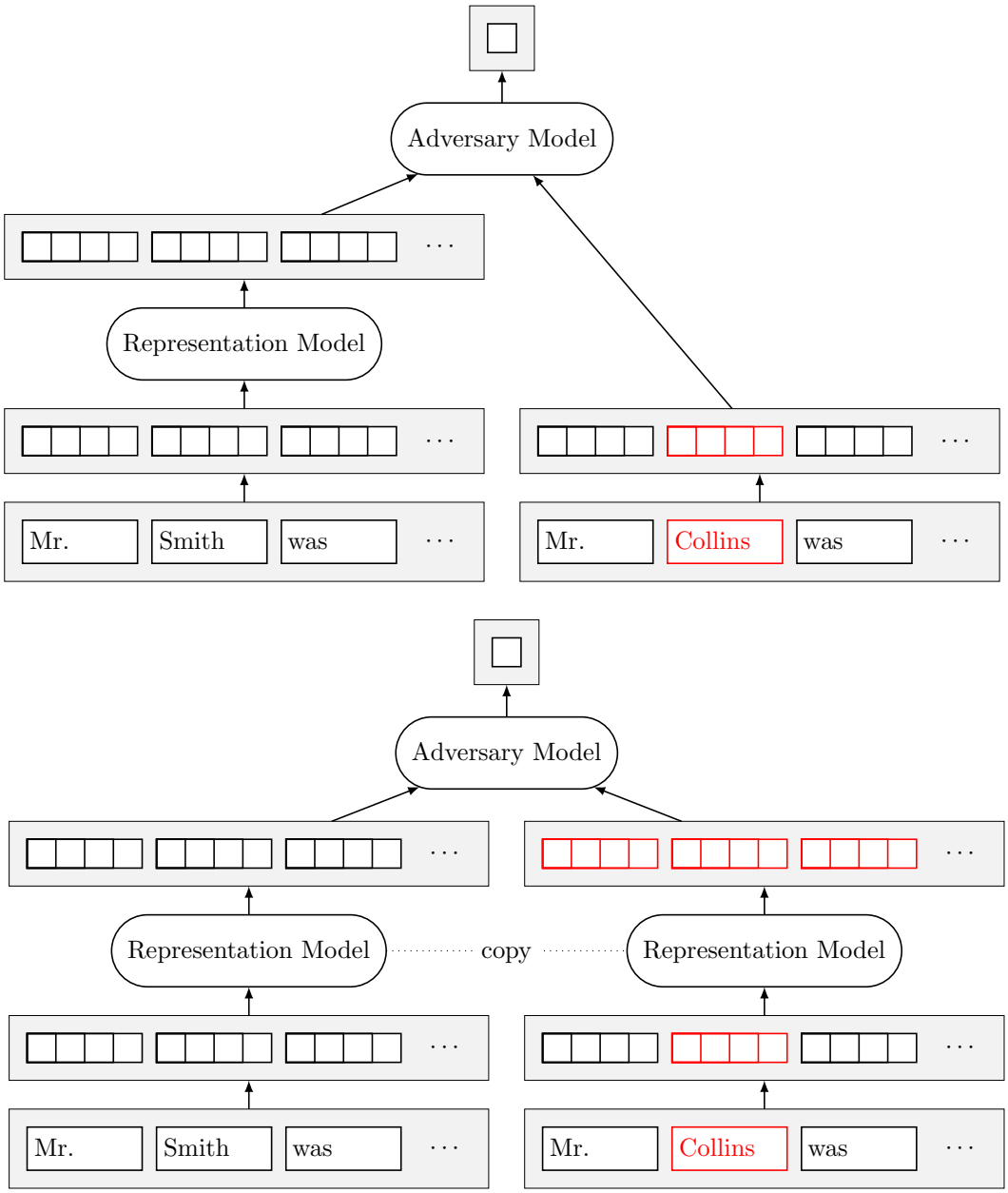

Figure 5.4: The two adversaries used in our adversarial model architecture with example inputs. Red boxes denote automatically pseudonymized fake inputs. We assume an LSTM representation model in which a single changed element in the input sequence influences all elements in the output sequence. The target label in the adversarial tasks is 0 (negative) in both cases as the two input sequences are not equal.

6 Results

In this chapter, we present the results of our experiments from Chapter 5.

6.1 De-Identification of Raw Data

Table 6.1 shows the HIPAA binary token-based de-identification scores on the i2b2 2014 test set of our ELMo, FastText, and GloVe models that each use the best hyperparameter configuration from our optimization. We averaged the results out of 5 runs per model and compare them to the state of the art (Dernoncourt et al., 2017b) as well as a naive word list baseline and the practical upper bound. The word list baseline is computed by creating a mapping from tokens in the training set to their most common labels and using it to predict the test set (the none class is predicted for tokens that do not occur in the training set).

All three models comfortably beat the target score of 95% that is required for a reasonable de-identification system (see Section 3.1). The ELMo model comes closest to the state-of-the-art result by Dernoncourt et al., falling short by around 0.1 percentage points. The FastText and GloVe models achieve slightly lower scores.

All models are strongest at recognizing dates and ages, followed by names (presumably the most important category), IDs, and contact information with F_1 scores above 90%. For location and profession tags, the models achieve F_1 scores below 90%. The profession category is the category with the lowest individual F_1 score. A table that lists our models' de-identification scores per category of PHI in comparison to the word list baseline and upper bound can be found in Appendix C.

Figure 6.1 shows our models' de-identification performance on the test set when using different fractions of the training documents. For each split, we reserve 20% of the split's documents for validation and early stopping. All three models achieve the target F_1 score of 95% when using only half the training set. Scores slightly below 94% that can be achieved with a 10% split of the training set would have beaten most of the teams that participated in the 2014 de-identification shared task (Stubbs et al., 2015).

6.2 De-Identification with Private Representations

In this section, we present the results of our experiments that use private representations to train a de-identification model (Sections 5.2.1 to 5.2.3 and Section 5.3).

Model	Precision (%)	Recall (%)	F_1 (%)
ELMo	98.20	97.29	97.74
FastText	97.73	**97.61**	97.67
GloVe	97.98	97.27	97.62
Dernoncourt et al. (2017b)	**98.32**	97.38	**97.85**
Word list	*81.04*	*55.90*	*66.16*
Upper bound	*99.32*	*99.56*	*99.44*

Table 6.1: Average precision, recall, and F_1 scores of our de-identification models in comparison the state-of-the-art model by Dernoncourt et al., a word list baseline, and the upper bound based on our preprocessing. The best result for each column is highlighted with a bold font.

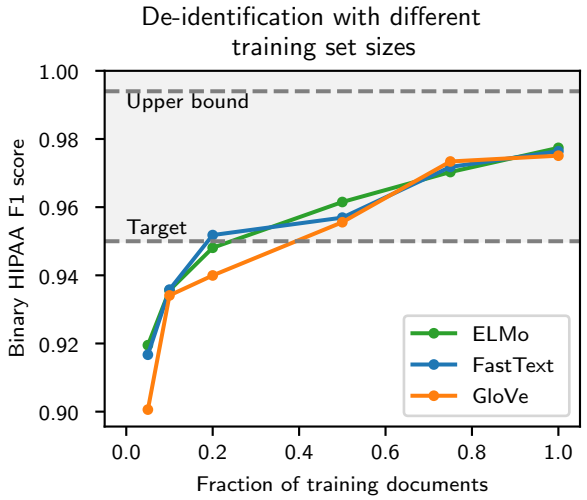

Figure 6.1: F_1 scores of our models when using training sets of different sizes.

Model	Precision (%)	Recall (%)	F_1 (%)
FastText	88.38	5.64	10.60
GloVe	86.76	22.54	35.78
GloVe/UNK	**95.92**	**26.41**	**41.42**

Table 6.2: Precision, recall, and F_1 scores of our de-identification models when trained on sequences with PHI replaced with random embeddings. The best result for each column is highlighted with a bold font.

6.2.1 Random Embeddings

The results for the random embeddings experiment are shown in Table 6.2. When PHI tokens in the training set are replaced with random vectors, the GloVe model achieves the best performance on the unperturbed test set. The GloVe/UNK entry in the table refers to an additional experiment with the GloVe model where tokens that do not occur in the GloVe embedding matrix retain their embedding (which is the same for all unknown tokens). This further improves the GloVe model's test score.

Appendix C includes a detailed evaluation of this experiment per category of PHI. The GloVe model with unknown embeddings achieves an F_1 score of 78% in the name category, which is higher than the word list baseline's score (see Table C.1). All other scores are below the word list baseline.

6.2.2 Additive Noise

The results of our additive noise experiment are shown in Figure 6.2. The left diagram illustrates the effect of adding noise of different scales to FastText and GloVe embeddings that occur as PHI tokens in the i2b2 corpus. We measure the cosine similarity rank of the original (sorting all words from the embedding matrix by their cosine distance to the noisy embedding, see Section 3.3.1).

Scales of noise up to 0.1, which is around double of the standard deviation of the embedding matrices, do not influence the cosine similarity ranks; the original embedding is still the closest neighbor in almost all cases. At higher noise scales, the similarity to the original decreases rapidly. The additive noise moves the vectors to different regions of the embedding space.

The right diagram shows that de-identification performance also drops with added noise. Even at a noise scale of 0.1 in which the cosine neighborhood of embeddings is not changed, the F_1 score is below the target for both models.

6.2.3 Automatically Pseudonymized Data

The results of the automatic pseudonymization experiment are illustrated in Figure 6.3. The left diagram shows that for both FastText and GloVe, moving training PHI tokens to random tokens from up to their $N = 200$ closest neighbors does not significantly

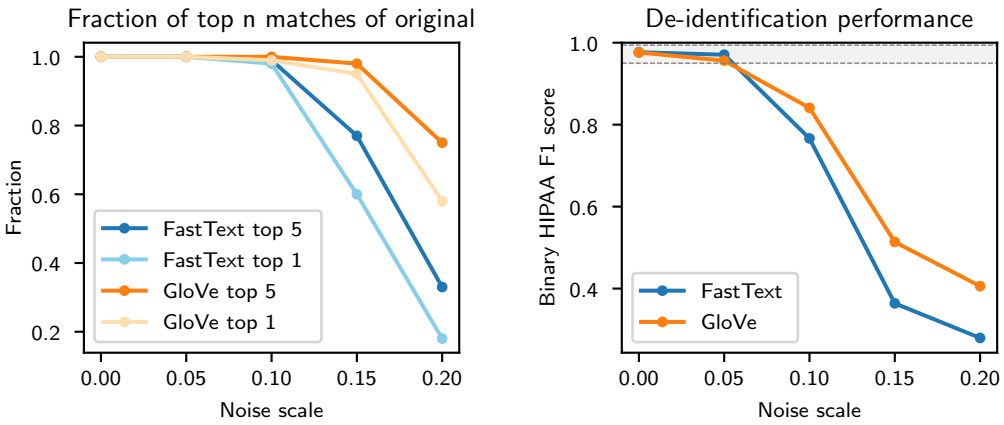

Figure 6.2: Left: percentage of top 5 and top 1 cosine similarity matches of the original word when adding Gaussian noise to 1 000 randomly selected FastText and GloVe embeddings. Right: de-identification performance with additive Gaussian noise.

reduce de-identification performance. F_1 scores for both models drop to around 95% when selecting from $N = 500$ neighbors and to around 90% when using $N = 1\,000$ neighbors. With $N = 100$, the FastText model achieves an F_1 score of 96.75% and the GloVe model achieves an F_1 score of 96.42%. The detailed evaluation tables for both FastText and GloVe can be found in Appendix C.

The right diagram shows the results of our adversarial evaluation. For both types of embeddings, the trained adversary achieves test accuracies of around 60% independently of the choice of N.

It is notable that models that were trained on automatically pseudonymized data outperform the corresponding model that was trained on raw data in F_1 score in several categories of PHI. The FastText model beats the respective raw data model in the profession, location, age and contact categories (see Table C.3). The automatic pseudonymization GloVe model outperforms the raw data model in the name, location, and date categories (see Table C.4). However, the overall binary HIPAA F_1 scores do not improve when using automatic pseudonymization.

6.2.4 Adversarial Learning of a Private Representation

In our adversarial learning experiment, we do not achieve satisfactory results with the conjoint DANN training procedure: in all cases, our models learn representations that are not sufficiently resistant to the adversary. When training the adversary on the frozen representation for an additional 20 epochs, it is able to distinguish real from fake input pairs on a test set with accuracies above 80%. This confirms the findings by Elazar and Goldberg (2018).

With the training procedure by Feutry et al. (2018), we are able to train a representation

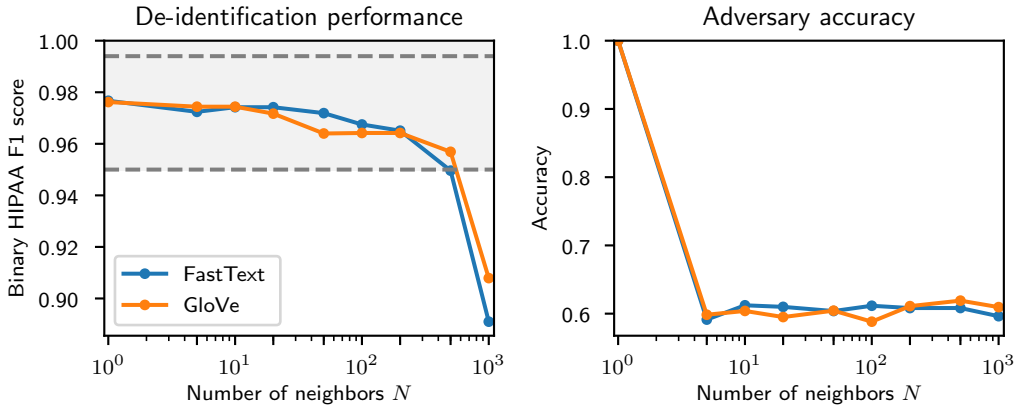

Figure 6.3: Left: F_1 scores of our models when trained on automatically pseudonymized data where PHI tokens are moved to one of different numbers of neighbors N. Right: adversary accuracy on the task of distinguishing real from fake original/pseudonymized sentence pairs. An accuracy of 1 is plotted for $N = 1$ (i.e. no pseudonymization).

that allows training a de-identification model while preventing an adversary from learning the adversarial tasks, even with continued training on a frozen representation. We select the LSTM representation model over the dense model because it allows 0.4 percentage points higher de-identification F_1 scores on average.

Figure 6.4 shows our de-identification results when using adversarially learned representations. A higher number of neighbors N means a stronger invariance requirement for the representation. For values of N up to 1 000, our FastText and GloVe models are able to learn representations that allow training de-identification models that reach the target F_1 score of 95%. However, training becomes unstable for $N > 50$ when using GloVe and $N > 500$ when using FastText embeddings. Then, training results in representations that are not robust to the adversary, which is shown in the diagram on the right side.

Our choice of representation size $d \in \{50, 100, 300\}$ does not influence de-identification or adversary performance, so we select $d = 50$ for further evaluation. For $d = 50$ and $N = 100$, the FastText model reaches an F_1 score of 97.4% and the GloVe model reaches an F_1 score of 96.89%.

The adversarially trained FastText model beats the FastText model trained on raw data in the contact category. The GloVe model does not beat the corresponding raw data model in any category. It is most significantly weaker in the profession category. The detailed evaluation tables can be found in Appendix C.

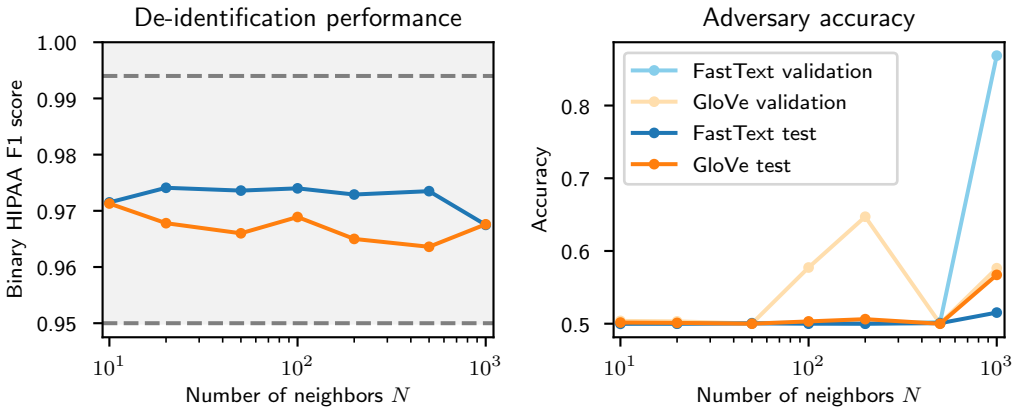

Figure 6.4: Left: de-identification F_1 scores of our models using an adversarially trained representation with representation size $d = 50$ and different numbers of neighbors N for the representation invariance requirement. Right: mean accuracy on the two adversary tasks. The validation accuracy lines show the maximum accuracy around the best epoch according to the combined loss as an attempt to visualize representation stability.

7 Discussion

In this work, we presented a baseline de-identification model as well as four approaches (of which two show promising results) to sharing training data for de-identification that require less human effort than manual pseudonymization by substitution.

Our baseline model achieves near-state-of-the-art results without relying on explicit character features. We find that both random word embeddings and word embeddings with additive Gaussian noise are unsuitable for training a reasonable de-identification model. Our naive automatic approximation of pseudonymization allows training a de-identification model while being robust to an adversary model that tries to trace pseudonymized sequences back to their originals. We introduced an adversarial model that learns an intermediate vector representation of medical text that is useful for training a de-identification model while preventing adversarial models from re-identifying original sequences or building a lookup table of representations.

In this chapter, we discuss our approaches regarding their de-identification performance and privacy properties. We briefly discuss our approaches' relationship to the related work and present directions for future work.

7.1 De-Identification Performance

In this section, we discuss the de-identification performance properties of our models.

Baseline De-Identification Our ELMo model is the best de-identification model, followed by the FastText and GloVe models. ELMo embeddings encapsulate the sentence context, making them more expressive than the non-contextual embedding types.

Reimers and Gurevych (2017) find that GloVe performs better than FastText in their NER benchmark. This is no contradiction to our result as the i2b2 2014 dataset in all likelihood contains more out-of-vocabulary tokens (that GloVe cannot embed) than their datasets. FastText's approach to embedding unknown words (word embeddings are the sum of their subword embeddings) proves useful on datasets with misspellings and ungrammatical text. However, FastText beats GloVe only by 0.05 percentage points on the i2b2 test set. The casing feature (which improves GloVe be 0.4 percentage points in the hyperparameter optimization) makes up for GloVe's missing embeddings for unknown words.

Analyzing our baseline models' predictions on the test set reveals that the models typically make wrong predictions for the same instances of PHI. Table 7.1 shows some wrong predictions from each of our models that are unique to the respective model[19]. Most

[19] We show similar sentences to actual wrongly predicted sentences from the i2b2 test set.

Model	Prediction • and Target ◇
ELMo	• Per [Bruce Protocol]$_{\text{Hospital}}$ she exercised 9 minutes ◇ Per Bruce Protocol she exercised 9 minutes
	• He is a veteran ◇ He is a [veteran]$_{\text{Profession}}$
FastText	• Works in [commercial]$_{\text{Organization}}$ [diving]$_{\text{Profession}}$ ◇ Works in [commercial diving]$_{\text{Profession}}$
	• Patient was seen in the walk-in clinic at MHC yesterday ◇ Patient was seen in the walk-in clinic at [MHC]$_{\text{Hospital}}$ yesterday
GloVe	• Discussed with patient at [lenght]$_{\text{Hospital}}$ surgery ◇ Discussed with patient at lenght surgery
	• Started walking around Pittstown about 3.5 miles ◇ Started walking around [Pittstown]$_{\text{City}}$ about 3.5 miles

Table 7.1: Some de-identification errors that are unique to the respective model. Note that "lenght" is a spelling mistake that occurs in the i2b2 2014 test set.

notably, the GloVe model makes mistakes that can be avoided by using subword features. Slight misspellings have embeddings close to the correct spelling in ELMo and FastText. Also, entity names that clearly belong to a specific category (like city names with the suffix "-town") have meaningful embeddings in ELMo and FastText. Some of the ELMo and FastText models' mistakes could be avoided by using a dictionary of professions, hospital names, and medical terms as an additional data source, e.g. "Bruce Protocol" is a standard cardiac diagnostic test that would occur in a medical dictionary.

Random Embeddings It is not possible to train a de-identification model with sentences where all PHI is replaced with random vectors. The GloVe model with retained unknown embeddings is strongest because the unknown embedding itself is a hint to tokens being PHI (e.g. uncommon names).

Additive Noise There is no scale of Gaussian noise that can be added to PHI tokens that sufficiently perturbs their cosine similarity neighborhood while allowing training a de-identification model.

Automatically Pseudonymized Data Our naive automatic word-level pseudonymization approach allows training reasonable de-identification models when selecting from up to $N = 500$ neighbors.

Figure 7.1 shows four examples for automatically pseudonymized sentences that were generated by moving PHI tokens to one of their $N = 100$ neighbors in the FastText embedding space. While the first three sentences are coherent and could be the result of a human pseudonymization step, the unrealistic last sentence hints at the limitations of

- ~~Bob~~ **John** continues to feel well
- She is a fulltime ~~historian~~ **photographer**
- TReated with RAI ~~February~~ **September** ~~2071~~ **2080**
- Transferred in ~~January~~ **October** to the ~~HMS~~ **Q-ship** after presenting to the ~~Florida~~ **Miami-Florida** ~~Hospital~~ **Hosp** ~~Orlando~~ **Merlina**

Figure 7.1: Examples of automatically pseudonymized sentences using FastText embeddings and $N = 100$ neighbors.

this naive approach. The token "HMS" was seemingly moved too far because "Q-ship" is not a realistic hospital name, and the tokens "Florida" and "Hospital" were only perturbed by a small degree. However, as deep learning models do not see the actual words but only their representations (that are neighbors in the embedding space), using the automatically pseudonymized data does not dramatically deteriorate de-identification performance. Rather, models trained on automatically pseudonymized data beat their counterparts that were trained on raw data in some categories of PHI due to their higher robustness.

Adversarially Learned Representation Our adversarially trained vector representation that is invariant to word changes allows training reasonable de-identification models when using up to $N = 1000$ neighbors as an invariance requirement.

The adversarial de-identification results beat the automatic pseudonymization results because the representation model can act as a task-specific feature extractor. Additionally, the representations are more general as they are invariant to word changes. The de-identification model is trained on sentences that are augmented to a smaller degree than in our automatic pseudonymization experiment (only one PHI token is moved to a neighbor in adversarial training instead of all PHI tokens).

7.2 Data Protection Compliance

In this section, we discuss the privacy properties of our approaches.

Embeddings When looking up embedding space neighbors for words, it is notable that many FastText neighbors include the original word or parts of it as a subword. This is due to FastText's method of using the sum of subword embeddings in embedding calculation. For tokens that occur as PHI in the i2b2 training set, on average 7.37 of their $N = 100$ closest neighbors in the FastText embedding matrix contain the original token as a subword. When looking up neighbors using GloVe embeddings, the value is 0.44. This may indicate that FastText requires stronger perturbation (i.e. higher N) than GloVe to sufficiently obfuscate protected information.

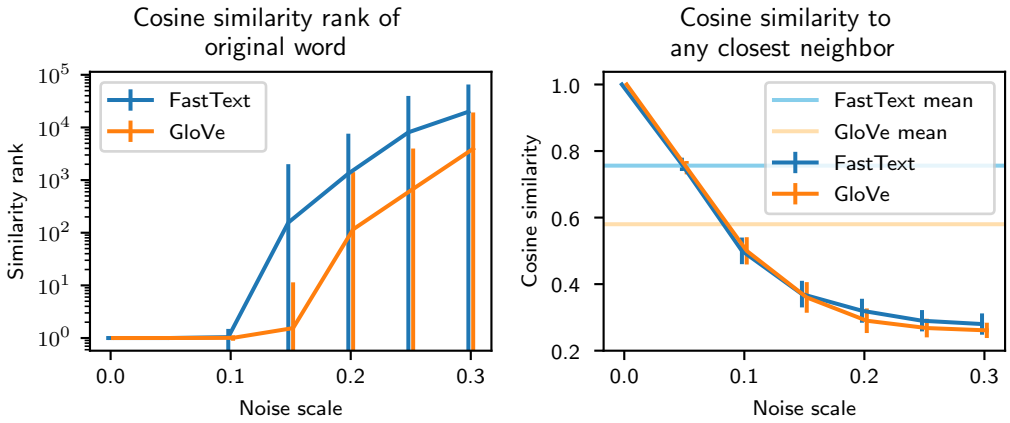

Figure 7.2: Left: cosine similarity rank of the original word when adding Gaussian noise to 1 000 randomly selected FastText and GloVe embeddings. Right: cosine similarity to any closest neighbor in comparison to the mean similarity for FastText and GloVe.

Additive Noise Figure 7.2 shows properties of additive Gaussian noise to FastText and GloVe embeddings. Both the L^2 normalized FastText and GloVe embedding matrices have means of around 0 and standard deviations of around 0.06. At a noise scales up to 0.1 for FastText and 0.15 for GloVe, the cosine neighborhood is mostly unchanged but de-identification models do not reach the 95% F_1 score target. This may be caused by the Gaussian noise moving embeddings to remote regions of the vector space: the cosine similarity to any closest neighbor continually drops when adding noise, quickly falling below the mean value of the respective embedding type.

Automatically Pseudonymized Data We identified some privacy weaknesses of our automatic pseudonymization approach. For a last name like *Wolf*, neighbors in the embedding space will include other animal names and not common last names. In this case, it could be possible to infer the original name if there are only a limited number of people that might appear in the dataset, e.g. the population of a small town.

If multiple sentences contain animal names (or any other similar names), they will likely come from the same original document, undoing the privacy gain from shuffling training sentences across documents. It may be possible to infer the original name using the overlapping neighbor spaces. To counter this, we can re-introduce document-level pseudonymization, i.e. moving all occurrences of a PHI token to the same neighbor. However, we would then also need to detect misspelled names as well as other hints to the actual tokens and transform them similarly to the original, which would add back much of the complexity of manual pseudonymization that we try to avoid.

In our adversarial evaluation, the adversaries reach test accuracies of 60%. However, a 60% accuracy is also reached by a similar LSTM adversary that is trained to discriminate

original sequences from sequences with one occurrence of PHI moved to a neighbor. This means that the adversary can achieve its accuracy only by learning to distinguish real from fake sentences and ignoring its pseudonymized sequence input.

Adversarially Learned Representation Our adversarial representation empirically satisfies a strong privacy criterion: representations are invariant to *any* protected information token being replaced with *any* of its N neighbors in an embedding space. While it is possible for de-identification models to achieve F_1 scores of 95% using our adversarially learned representation with up to $N = 1000$ neighbors, training becomes unstable for large N.

Figure 7.3 shows a set of typical (successful) learning curves for Feutry et al.'s training procedure. The representation model and de-identification model are jointly pre-trained in the first phase. In the second phase, the adversary is pre-trained to reach a validation accuracy of around 80%, which means that the pre-trained representation does not fit our invariance requirements. At the beginning of the third training phase, both the de-identification model and the adversary learning curves show an oscillating pattern that is caused by the alternating training of the branches (with frozen representation model) and the representation model (with frozen branches). Around 15 epochs into this phase, we find an adequate representation which is indicated by the adversary accuracy being close to the random guessing accuracy of 50%. The de-identification validation loss is typically lower than the training loss in the alternating phase. Since the representation and de-identification models are not trained together in this phase, the models always need to *catch up* to the other model's parameter changes (training loss is averaged over all training samples of an epoch). When freezing the representation model and training the adversary for an additional 60 epochs, it still does not achieve higher accuracies than 50%. Due to the added noise, the adversary does not overfit on its training set but rather fails to identify any structure in the data.

Figure 7.4 shows the learning curves of a failed experiment run where training becomes unstable due to the choice of $N = 1000$. The adversary learns to counter the representation several times in the alternating training phase. These adversary accuracy peaks are always followed by a peak in the de-identification model's validation loss because the representation is worsened for both branches to combat the adversary. Due to the combined loss with $\lambda = 1$, the best model according to validation loss will often deliver good de-identification results but it may not guarantee robustness to the adversary, even if the concrete adversary test accuracy after the early stopping epoch is low. Continued training of the adversary with a frozen representation will allow it to reach higher accuracies.

7.3 Relationship to the Related Work

Both our model trained on raw data (Section 5.1) and our model trained on the adversarially learned representation (Section 5.3) reach near-state-of-the-art (Dernoncourt et al., 2017b) results on the i2b2 2014 dataset (Stubbs and Uzuner, 2015; Stubbs et al., 2015).

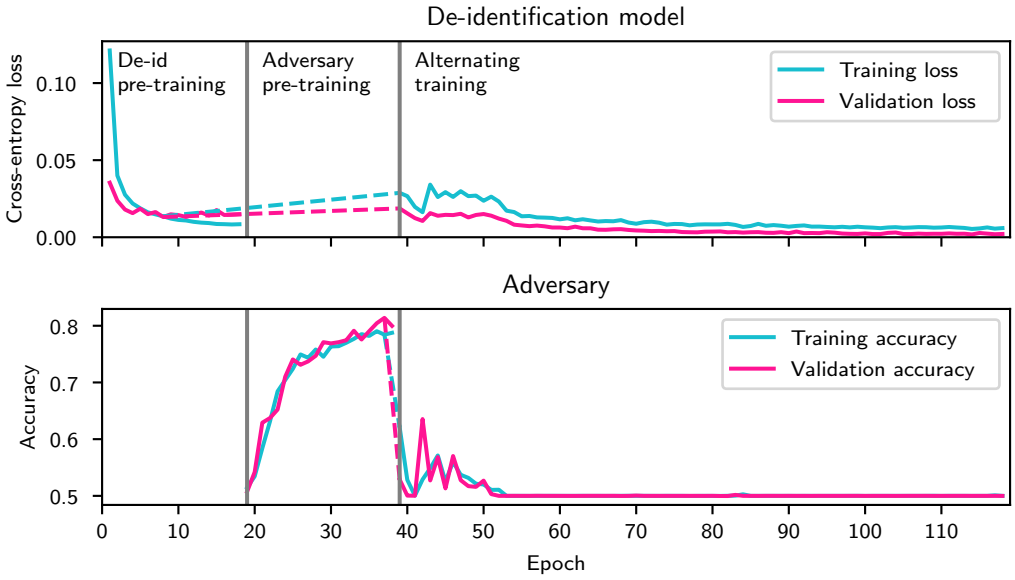

Figure 7.3: Learning curves of one adversarial experiment run (FastText embeddings, $N = 10, d = 50$). The training is split into three parts: de-identification pre-training, adversary pre-training, and alternating training. Dashed lines denote model resets after early stopping.

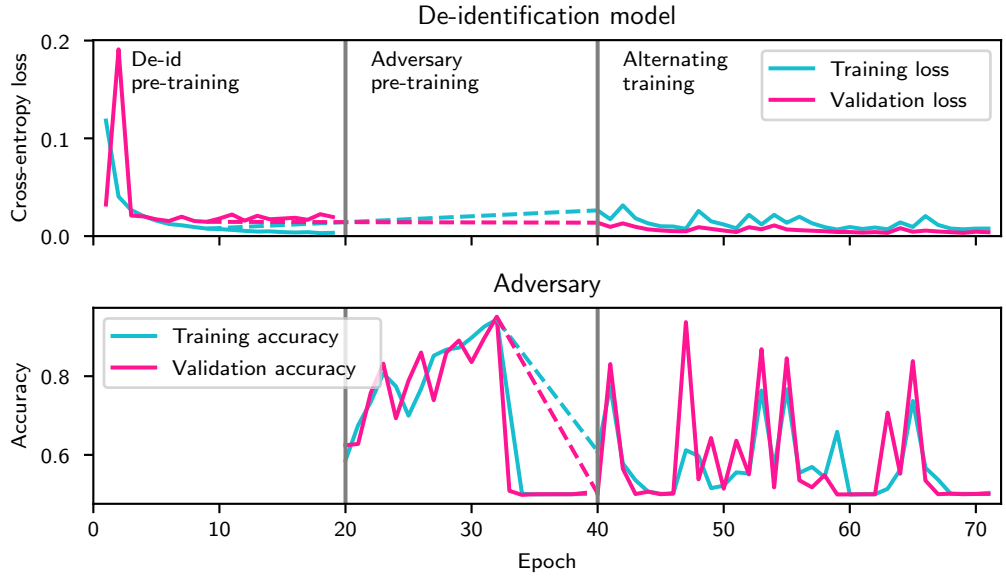

Figure 7.4: Learning curves of one adversarial experiment run (FastText embeddings, $N = 1000, d = 50$) that does not result in a stable representation.

Our adversarial representation architecture extends gradient reversal based representation learning (Ganin et al., 2016; Feutry et al., 2018) with a Siamese-like adversary that receives two inputs to become invariant to input word changes. Our adversarially learned representation is a new mechanism to privately publish (medical) text data.

7.4 Future Work

Our automatic pseudonymization approach could serve as a data augmentation scheme to be used as a regularizer for de-identification models. Training a model on a combination of raw and pseudonymized data may result in better test scores on the i2b2 test set, possibly beating the state of the art.

Our automatic pseudonymization and adversarial learning approaches will most likely be improved by using ELMo embeddings, which we did not use due to their computation cost. In adversarial learning, it might be possible to tune the λ parameter and define a better stopping condition that avoids the unstable characteristics with high values for N in the invariance criterion. A further possible extension is a dynamic noise level in the representation model that depends on the LSTM output instead of being a trained weight. This might allow using lower amounts of noise for certain inputs while still being robust to the adversary.

When more training data from multiple sources becomes available in the future, it will be possible to evaluate our adversarially learned representation against unseen data. Additionally, federated learning and the semi-supervised knowledge transfer approach for de-identification can be reasonably simulated with multiple sources of data.

8 Conclusion

Privacy laws require medical text to be de-identified before it is shared. De-identification is time-consuming and costly when performed by humans, which motivates the creation of automatic de-identification tools. Automatic de-identification requires training data, which is typically created by substituting all protected information from raw medical records. Today's de-identification tools fail on unseen data. A training set from multiple sources is required to train more general de-identification tools. We evaluated approaches to sharing training data for de-identification that require lower human effort than the existing approach of document-coherent pseudonymization.

As a precursor to our data sharing approaches, we developed a baseline deep learning model for de-identification that does rely on explicit character features. It uses word embeddings as well as a casing feature as inputs. On the i2b2 2014 test set, the model reaches an F_1 score of 97.74%, a near-state-of-the-art result.

Our automatic pseudonymization approach replaces protected information with neighboring words in an embedding space. A bidirectional LSTM adversary can learn to distinguish real from fake original/pseudonymized sentence pairs with an accuracy of 60%. This accuracy can however also be achieved by an adversary that only distinguishes real from fake sentences. A model trained on this augmented data with $N = 100$ neighbors loses around one percentage point in F_1 score when compared to the raw data baseline, scoring 96.75%.

We presented an adversarial learning based private representation of medical text that is invariant to any protected information word being replaced with any of its embedding space neighbors and contains a random element. The representation allows training a de-identification model while being robust to adversaries trying to re-identify protected information or building a lookup table of representations. We extended existing adversarial representation learning approaches by using two adversaries that discriminate real from fake sequence pairs with an additional sequence input. Using the adversarially learned representation, de-identification models reach an F_1 score of 97.4%, which is closer to the raw data baseline than to the automatic pseudonymization score. The representation acts as a task-specific feature extractor. For an invariance criterion of up to $N = 50$ (GloVe) or $N = 500$ (FastText) neighbors, training is stable and adversaries cannot beat the random guessing accuracy of 50%.

Our approaches, especially the adversarially trained representation, allow cost-effective private sharing of training data for de-identification. Better de-identification tools could help enable large-scale medical studies that improve public health.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Accessed July 29, 2018.

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 308–318, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4139-4. URL `https://doi.org/10.1145/2976749.2978318`.

Charu C. Aggarwal. On k-anonymity and the curse of dimensionality. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 901–909. VLDB Endowment, 2005. ISBN 1-59593-154-6.

Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional space. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory — ICDT 2001*, pages 420–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44503-6.

Michael Barbaro, Tom Zeller, and Saul Hansell. A face is exposed for AOL searcher no. 4417749. *New York Times*, 9(2008):8, 2006.

Brett K. Beaulieu-Jones, Zhiwei Steven Wu, Chris Williams, James Brian Byrd, and Casey S. Greene. Privacy-preserving generative deep neural networks support clinical data sharing. *bioRxiv*, 2017. URL `https://www.biorxiv.org/content/early/2017/11/15/159756`.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks and Learning Systems*, 5(2): 157–166, March 1994. ISSN 1045-9227. URL `https://doi.org/10.1109/72.279181`.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, March 2003. ISSN 1532-4435. URL https://dl.acm.org/citation.cfm?id=944919.944966.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016. URL https://arxiv.org/abs/1607.04606.

Nicholas Carlini, Chang Liu, Jernej Kos, Úlfar Erlingsson, and Dawn Song. The secret sharer: Measuring unintended neural network memorization & extracting secrets. *arXiv preprint arXiv:1802.08232*, 2018. URL https://arxiv.org/abs/1802.08232.

Jason Chiu and Eric Nichols. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics*, 4:357–370, 2016. URL https://aclweb.org/anthology/Q16-1026.

François Chollet et al. Keras, 2015. URL https://keras.io. Accessed September 27, 2018.

Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4. URL https://doi.org/10.1145/1390156.1390177.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, November 2011. ISSN 1532-4435. URL https://dl.acm.org/citation.cfm?id=1953048.2078186.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, Dec 1989. ISSN 1435-568X. URL https://doi.org/10.1007/BF02551274.

Fida Kamal Dankar and Khaled El Emam. The application of differential privacy to health data. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, EDBT-ICDT '12, pages 158–166, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1143-4. URL https://doi.org/10.1145/2320765.2320816.

Franck Dernoncourt, Ji Young Lee, and Peter Szolovits. NeuroNER: an easy-to-use program for named-entity recognition based on neural networks. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 97–102. Association for Computational Linguistics, 2017a. URL https://aclweb.org/anthology/D17-2017.

Franck Dernoncourt, Ji Young Lee, Özlem Uzuner, and Peter Szolovits. De-identification of patient notes with recurrent neural networks. *Journal of the American Medical*

Informatics Association, 24(3):596–606, 2017b. URL https://doi.org/10.1093/jamia/ocw156.

Pedro Domingos. A few useful things to know about machine learning. *Commun. ACM*, 55 (10):78–87, October 2012. ISSN 0001-0782. URL https://doi.org/10.1145/2347736.2347755.

Timothy Dozat. Incorporating Nesterov momentum into Adam. *International Conference on Learning Representations (ICLR) Workshop*, 2016.

Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-35908-1.

Cynthia Dwork. Differential privacy: A survey of results. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 1–19, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-79228-4.

Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *arXiv preprint arXiv:1511.05897*, 2015. URL https://arxiv.org/abs/1511.05897.

Yanai Elazar and Yoav Goldberg. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 11–21. Association for Computational Linguistics, 2018. URL https://aclweb.org/anthology/D18-1002.

Clément Feutry, Pablo Piantanida, Yoshua Bengio, and Pierre Duhamel. Learning anonymized representations with adversarial neural networks. *arXiv preprint arXiv:1802.09386*, 2018. URL https://arxiv.org/abs/1802.09386.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA, 2016. Curran Associates. ISBN 978-1-5108-3881-9.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, January 2016. ISSN 1532-4435. URL http://www.jmlr.org/papers/volume17/15-189/15-189.pdf.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, October 2000. ISSN 0899-7667. URL https://doi.org/10.1162/089976600300015015.

Aris Gkoulalas-Divanis, Grigorios Loukides, and Jimeng Sun. Publishing data from electronic health records while preserving privacy: A survey of algorithms. *Journal of Biomedical Informatics*, 50:4 – 19, 2014. ISSN 1532-0464. URL https://www.sciencedirect.com/science/article/pii/S1532046414001403. Special Issue on Informatics Methods in Medical Privacy.

Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, 2014. URL https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. URL https://www.deeplearningbook.org.

Jihun Hamm. Preserving privacy of continuous high-dimensional data with minimax filters. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 324–332, San Diego, California, USA, 09–12 May 2015. PMLR. URL http://proceedings.mlr.press/v38/hamm15.html.

Jihun Hamm. Enhancing utility and privacy with noisy minimax filters. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6389–6393, March 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, November 1997. ISSN 0899-7667. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Matthew Honnibal and Ines Montani. spaCy 2, 2017. URL https://spacy.io. Accessed August 1, 2018.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, July 1989. ISSN 0893-6080. URL https://doi.org/10.1016/0893-6080(89)90020-8.

Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015. URL https://arxiv.org/abs/1508.01991.

Peter B. Jensen, Lars J. Jensen, and Søren Brunak. Mining electronic health records: towards better research applications and clinical care. *Nature Reviews Genetics*, 13: 395–405, 05 2012. URL https://doi.org/10.1038/nrg3208.

Andrej Karpathy. The unreasonable effectiveness of recurrent neural networks, May 2015. URL `https://karpathy.github.io/2015/05/21/rnn-effectiveness/`. Accessed August 4, 2018.

Daniel Kifer. Attacks on privacy and definetti's theorem. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, SIGMOD '09, pages 127–138, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-551-2. URL `https://doi.org/10.1145/1559845.1559861`.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL `https://arxiv.org/abs/1412.6980`.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270. Association for Computational Linguistics, 2016. URL `https://aclweb.org/anthology/N16-1030`.

Ji Young Lee, Franck Dernoncourt, and Peter Szolovits. Transfer learning for named-entity recognition with neural networks. *arXiv preprint arXiv:1705.06273*, 2017. URL `https://arxiv.org/abs/1705.06273`.

Yitong Li, Timothy Baldwin, and Trevor Cohn. Towards robust and privacy-preserving text representations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 25–30. Association for Computational Linguistics, 2018. URL `https://aclweb.org/anthology/P18-2005`.

Zengjian Liu, Buzhou Tang, Xiaolong Wang, and Qingcai Chen. De-identification of clinical notes via recurrent neural network and conditional random field. *Journal of Biomedical Informatics*, 75(S):S34–S42, November 2017. ISSN 1532-0464. URL `https://doi.org/10.1016/j.jbi.2017.05.023`.

Xuezhe Ma and Eduard Hovy. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1064–1074. Association for Computational Linguistics, 2016. URL `https://aclweb.org/anthology/P16-1101`.

Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. ℓ-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1), March 2007. ISSN 1556-4681. URL `https://doi.org/10.1145/1217299.1217302`.

H Brendan McMahan, Eider Moore, Daniel Ramage, Seth. Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, February 2016. URL `https://arxiv.org/abs/1602.05629`.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private language models. *arXiv preprint arXiv:1710.06963*, 2017. URL `https://arxiv.org/abs/1710.06963`.

Stéphane M. Meystre, Guergana K. Savova, K C Kipper-Schuler, and John F. Hurdle. Extracting information from textual documents in the electronic health record: a review of recent research. *Yearbook of Medical Informatics*, pages 128–44, 2008.

Stephane M. Meystre, F. Jeffrey Friedlin, Brett R. South, Shuying Shen, and Matthew H. Samore. Automatic de-identification of textual documents in the electronic health record: a review of recent research. *BMC Medical Research Methodology*, 10(1):70, Aug 2010. ISSN 1471-2288. URL `https://doi.org/10.1186/1471-2288-10-70`.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a. URL `https://arxiv.org/abs/1301.3781`.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA, 2013b. Curran Associates.

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751. Association for Computational Linguistics, 2013c. URL `https://aclweb.org/anthology/N13-1090`.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In Nicoletta Calzolari (Conference chair), Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga, editors, *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan, May 7-12, 2018 2018. European Language Resources Association (ELRA). ISBN 979-10-95546-00-9.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725*, 2016. URL `https://arxiv.org/abs/1605.07725`.

Arvind Narayanan and Vitaly Shmatikov. Robust de-anonymization of large sparse datasets. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 111–125, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3168-7. URL `https://doi.org/10.1109/SP.2008.33`.

Ishna Neamatullah, Margaret M Douglass, Li-wei H Lehman, Andrew Reisner, Mauricio Villarroel, William J Long, Peter Szolovits, George B Moody, Roger G Mark, and Gari D Clifford. Automated de-identification of free-text medical records. *BMC Medical Informatics and Decision Making*, 8:32–32, 07 2008. URL `https://www.ncbi.nlm.nih.gov/pubmed/18652655`.

Christopher Olah. Understanding LSTM networks, August 2015. URL `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Accessed August 4, 2018.

Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016. URL `https://arxiv.org/abs/1610.05755`.

The European Parliament and the Council of the European Union. General data protection regulation, 4 2016. URL `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=celex%3A32016R0679`. Accessed September 18, 2018.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–1310–III–1318. JMLR.org, 2013. URL `https://proceedings.mlr.press/v28/pascanu13.pdf`.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543. Association for Computational Linguistics, 2014. URL `https://aclweb.org/anthology/D14-1162`.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018. URL `https://aclweb.org/anthology/N18-1202`.

Sameer Pradhan, Noémie Elhadad, Wendy Chapman, Suresh Manandhar, and Guergana Savova. SemEval-2014 task 7: Analysis of clinical text. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 54–62. Association for Computational Linguistics, 2014. URL `https://aclweb.org/anthology/S14-2007`.

Nils Reimers and Iryna Gurevych. Optimal hyperparameters for deep LSTM-networks for sequence labeling tasks. *arXiv preprint arXiv:1707.06799*, 2017. URL `https://arxiv.org/abs/1707.06799`.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. URL `https://arxiv.org/abs/1609.04747`.

Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1310–1321, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. URL https://doi.org/10.1145/2810103.2813687.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL https://dl.acm.org/citation.cfm?id=2627435.2670313.

Amber Stubbs and Özlem Uzuner. Annotating longitudinal clinical narratives for de-identification: The 2014 i2b2/UTHealth corpus. *Journal of Biomedical Informatics*, 58: 20–29, dec 2015.

Amber Stubbs, Christopher Kotfila, and Özlem Uzuner. Automated systems for the de-identification of longitudinal clinical narratives: Overview of 2014 i2b2/UTHealth shared task track 1. *Journal of Biomedical Informatics*, 58:11–19, 2015.

Amber Stubbs, Michele Filannino, and Özlem Uzuner. De-identification of psychiatric intake records: Overview of 2016 CEGS N-GRID shared tasks track 1. *Journal of Biomedical Informatics*, 75(S):S4–S18, November 2017. ISSN 1532-0464. URL https://doi.org/10.1016/j.jbi.2017.06.011.

Charles Sutton, Andrew McCallum, et al. An introduction to conditional random fields. *Foundations and Trends® in Machine Learning*, 4(4):267–373, 2012.

Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, October 2002. ISSN 0218-4885. URL https://doi.org/10.1142/S0218488502001648.

The United States Congress. Health insurance portability and accountability act of 1996, 1996. URL https://legislink.org/us/pl-104-191. Accessed September 27, 2018.

Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of the 6th Conference on Natural Language Learning - Volume 20*, volume 20 of *COLING-02*, pages 1–4, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. URL https://doi.org/10.3115/1118853.1118877.

Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, volume 4 of *CONLL '03*, pages 142–147, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. URL https://doi.org/10.3115/1119176.1119195.

Ardhendu Tripathy, Ye Wang, and Prakash Ishwar. Privacy-preserving adversarial networks. *arXiv preprint arXiv:1712.07008*, 2017. URL https://arxiv.org/abs/1712.07008.

Özlem Uzuner, Yuan Luo, and Peter Szolovits. Evaluating the state-of-the-art in automatic de-identification. *Journal of the American Medical Informatics Association*, 14(5): 550–563, Sep-Oct 2007. URL https://www.ncbi.nlm.nih.gov/pubmed/17600094.

Özlem Uzuner, Imre Solti, and Eithon Cadag. Extracting medication information from clinical text. *Journal of the American Medical Informatics Association*, 17(5):514–518, Sep-Oct 2010. URL https://www.ncbi.nlm.nih.gov/pubmed/20819854.

Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *Journal of the American Medical Informatics Association*, 18(5):552–556, Sep-Oct 2011. URL https://www.ncbi.nlm.nih.gov/pubmed/21685143.

Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):393–404, 1990.

Yonghui Wu, Jun Xu, Min Jiang, Yaoyun Zhang, and Hua Xu. A study of neural word embeddings for named entity recognition in clinical text. *AMIA Annual Symposium Proceedings*, 2015:1326–1333, 11 2015. URL https://www.ncbi.nlm.nih.gov/pubmed/26958273.

Xinyang Zhang, Shouling Ji, and Ting Wang. Differentially private releasing via deep generative model. *arXiv preprint arXiv:1801.01594*, 2018. URL https://arxiv.org/abs/1801.01594.

Yue-Shu Zhao, Kun-Li Zhang, Hong-Chao Ma, and Kun Li. Leveraging text skeleton for de-identification of electronic medical records. *BMC Medical Informatics and Decision Making*, 18(S-1):65–72, 2018. URL https://doi.org/10.1186/s12911-018-0598-6.

A Data Preprocessing and Postprocessing

In this work, we use the 2014 i2b2/UTHealth dataset that was originally published for the 2014 de-identification shared task.

Preprocessing

General Dataset Information The training set was released in two parts during the shared task period. At model selection time, we use the first part (`training-PHI-Gold-Set1`) for training and the second part (`training-PHI-Gold-Set2`) for validation. The two parts do not contain overlapping patients.

We use the test set (`testing-PHI-Gold-fixed`) that was released after the shared task only for the final evaluation of each approach. At test time, we combine the two parts of the training set and use a smaller validation split for early stopping.

Preprocessing Transformations We apply the following transformations to the data:

1. Fix annotations in a single document (`180-03`) where indices are offset after a special character.
2. Tokenize the text aggressively similar to Liu et al. (2017), including splitting after all punctuation marks and mid-word if a number is followed by text and the other way around (e.g. "25yo" is split into "25", "yo").
3. Split the text into sentences using the spaCy heuristics for English with additional rules that are based on an inspection of the training data:
   - New sentences after three newline characters.
   - New sentences for bulleted or numbered list items.
   - New sentences after three dashes.
4. Replace tokens that make an HTML entity with the corresponding character (e.g. "`&`" to "&").
5. Assign the PHI tags to the tokens and transform them into the IOB2 format (Tjong Kim Sang, 2002).
6. Save the IOB2-annotated tokens to CSV files along with the token offsets in the original raw text.

Preprocessing Shortcomings The tokenization misses 31 PHI tags entirely because they occur inside a single token with another PHI tag. In these cases, only the first PHI tag in the token is identified and transformed into the IOB2 format. Table A.1 contains three examples of this problem.

Correct annotation	Our annotation
$[986044]_{\text{Medical Record}}[2018]_{\text{Date}}$	$[9860442018]_{\text{Medical Record}}$
$[25\text{ Court Street}]_{\text{Street}}[\text{Aberdeen}]_{\text{City}}$	$[25\text{ Court StreetAberdeen}]_{\text{Street}}$
$[\text{M}]_{\text{Date}}[\text{W}]_{\text{Date}}[\text{F}]_{\text{Date}}$	$[\text{MWF}]_{\text{Date}}$

Table A.1: Examples for missed PHI tags due to our tokenization. Note that there is no whitespace between the PHI tags. "MWF" means the weekdays Monday, Wednesday, and Friday.

Our sentence splitting heuristics cause 625 PHI tags to be split into two sentences. An overview of the number of PHI tags in the original data and after our preprocessing, including missed tags and sentence splits, is shown in Table A.2.

	Train	Validation	Train+Val.	Test	Overall
Stubbs and Uzuner	–	–	17 410	11 462	28 872
Original data	11 893	5 512	17 405	11 462	28 867
Our B-* labels	12 142	5 652	17 794	11 729	29 523
Our I-* labels	20 220	9 414	29 634	19 873	49 507
Our O labels	522 246	233 403	755 649	487 454	1 243 103
Missed tags	22	4	26	5	31
Sentence splits	227	136	363	262	625

Table A.2: Number of PHI tags as reported in Stubbs and Uzuner (2015), as they occur in the dataset we received, and in our tokenized IOB2 format.

Postprocessing

To evaluate our de-identification predictions with the official evaluation script, we apply the following transformations:

1. For each tag, find the arg max of the label probability distribution and assign it the corresponding IOB2 tag.
2. Assemble the IOB2 tagged tokens to PHI annotations that span multiple tags.
3. Output the tags in the required XML format.

We updated the official script to Python 3 to use it from within our experiment code.

B De-Identification Hyperparameter Optimization

				Dropout			
Embeddings	Casing	Batch size	# hidd.	Input	Variati.	After hidd.	F_1 (%)
ELMo	Yes	32	2	0.1	0.5	0.5	97.75
GloVe	Yes	64	2	0.25	0.5	0.5	97.73
FastText	Yes	16	1	0.0	0.5	0.25	97.66
ELMo	No	32	2	0.25	0.25	0.25	97.63
FastText	Yes	64	1	0.1	0.1	0.5	97.58
FastText	No	16	1	0.5	0.5	0.1	97.57
GloVe	Yes	32	2	0.5	0.1	0.1	97.56
FastText	Yes	64	1	0.25	0.1	0.25	97.54
FastText	No	16	1	0.5	0.25	0.1	97.48
GloVe	Yes	32	2	0.05	0.5	0.5	97.41
FastText	Yes	32	2	0.25	0.25	0.5	97.41
FastText	No	32	1	0.05	0.25	0.25	97.41
ELMo	Yes	16	2	0.1	0.25	0.5	97.36
ELMo	No	64	2	0.1	0.25	0.1	97.33
ELMo	No	64	2	0.05	0.25	0.25	97.28
FastText	Yes	32	2	0.5	0.1	0.25	97.28
FastText	Yes	1	1	0.1	0.5	0.25	97.27
GloVe	Yes	32	1	0.25	0.25	0.1	97.25
ELMo	No	16	2	0.25	0.1	0.25	97.22
GloVe	No	32	2	0.0	0.5	0.5	97.2
GloVe	No	64	1	0.25	0.25	0.5	97.17
ELMo	Yes	32	2	0.25	0.25	0.5	97.17
ELMo	Yes	64	2	0.05	0.1	0.25	97.17
ELMo	No	32	1	0.25	0.5	0.25	97.12
ELMo	No	64	2	0.0	0.5	0.25	97.12

Table B.1: The 25 best hyperparameter configurations in our optimization, sorted by validation F_1 score. Note that the F_1 scores are not comparable to the experiment results (Chapter 6) because the optimization does not use the i2b2 test set.

Configuration		Model		
Hyperparameter	Value	ELMo	GloVe	FastText
Casing	**Yes**	**−0.2**	**+0.1**	**+0.4**
	No	+0.2	−0.2	−0.1
Batch size	1	−4.2	−0.7	−0.5
	16	+0.2	+0.4	±0.0
	32	**+0.4**	**+0.2**	**+0.1**
	64	+0.1	+0.4	+0.2
# hidd.	1	−0.1	−0.1	−0.1
	2	**+0.2**	**+0.4**	**+0.1**
Input dropout	0	+0.3	−0.3	±0.0
	0.05	±0.0	−0.5	+0.2
	0.1	**+0.4**	**−0.1**	**−0.4**
	0.25	−0.4	±0.0	±0.0
	0.5	−3.9	+0.2	+0.5
Variational dropout	0.1	−0.4	+0.1	−0.1
	0.25	**+0.5**	**+0.3**	**±0.0**
	0.5	±0.0	−0.1	+0.1
After LSTM dropout	0.1	±0.0	−0.3	−0.1
	0.25	+0.4	+0.1	−0.5
	0.5	**−0.4**	**+0.4**	**+0.2**

Table B.2: Effects per hyperparameter on the ELMo, FastText, and GloVe models, measured as an F_1 percentage point difference to the respective embedding type's mean F_1 score. Our final hyperparameters are marked with a bold font.

C De-Identification Evaluation

This appendix contains detailed evaluation tables for our experiments that we obtained from the official i2b2 2014 evaluation script.

De-Identification Baseline

Category	ELMo	FastText	GloVe	*Word list*	*Upper*
Token	**96.04**	95.87	95.29	*58.55*	*99.51*
Strict	**89.3**	89.03	87.91	*8.77*	*95.36*
HIPAA Token	**97.43**	97.21	97.04	*63.81*	*99.48*
HIPAA Strict	**90.82**	90.41	90.02	*7.3*	*94.41*
Binary Token	97.69	**97.75**	97.38	*64.47*	*99.51*
Binary Strict	**90.97**	**90.97**	89.97	*11.49*	*95.37*
Binary HIPAA Token	**97.78**	97.69	97.49	*66.16*	*99.49*
Binary HIPAA Strict	**91.21**	91.05	90.53	*9.22*	*94.42*
NAME Token	95.17	**95.54**	93.22	*31.57*	*99.86*
NAME Strict	**91.43**	91.32	88.51	*8.32*	*97.88*
PROFESSION Token	**88.04**	84.27	82.74	*26.37*	*100.0*
PROFESSION Strict	**77.27**	69.68	66.47	*7.66*	*98.32*
LOCATION Token	**88.31**	87.98	87.17	*42.27*	*99.42*
LOCATION Strict	80.95	**81.43**	79.08	*16.54*	*97.34*
AGE Token	95.15	94.9	**95.37**	*2.59*	*99.24*
AGE Strict	94.86	94.61	**95.16**	*2.67*	*99.35*
DATE Token	**98.82**	98.8	98.78	*75.3*	*99.47*
DATE Strict	91.21	**91.25**	91.15	*8.32*	*92.61*
CONTACT Token	**95.04**	90.13	93.74	*0.93*	*99.17*
CONTACT Strict	**89.1**	82.46	85.65	*0.0*	*95.45*
ID Token	**92.58**	91.57	90.27	*1.57*	*98.76*
ID Strict	**85.21**	83.47	82.25	*0.0*	*94.62*

Table C.1: F_1 scores of our baseline de-identification models (non-averaged scores of a single model per embedding type).

Random Embeddings

Category	ELMo	FastText	GloVe	GloVe/UNK
Token	3.08	17.75	42.24	**48.28**
Strict	2.65	11.55	28.75	**33.91**
HIPAA Token	1.45	10.37	33.56	**40.25**
HIPAA Strict	1.09	3.41	18.87	**22.66**
Binary Token	3.14	18.7	51.31	**54.82**
Binary Strict	2.72	12.82	33.82	**37.74**
Binary HIPAA Token	1.46	10.6	35.78	**41.42**
Binary HIPAA Strict	1.12	3.79	20.75	**23.99**
NAME Token	9.48	43.59	75.09	**78.46**
NAME Strict	7.45	32.37	63.04	**68.71**
PROFESSION Token	0.0	0.0	0.0	**3.8**
PROFESSION Strict	0.0	0.0	0.0	**6.8**
LOCATION Token	0.07	0.27	**27.35**	26.69
LOCATION Strict	0.0	0.0	7.96	**10.39**
AGE Token	0.0	0.0	0.0	0.0
AGE Strict	0.0	0.0	0.0	0.0
DATE Token	1.88	12.98	34.12	**43.3**
DATE Strict	1.6	3.86	16.25	**21.67**
CONTACT Token	0.0	**0.48**	0.0	0.0
CONTACT Strict	0.0	**0.91**	0.0	0.0
ID Token	0.0	1.06	4.32	**13.43**
ID Strict	0.0	0.0	0.0	**9.59**

Table C.2: F_1 scores of our de-identification models when trained on data where all PHI is replaced with random vectors.

Automatic Pseudonymization: FastText

	Number of neighbors N						
Category	5	10	20	50	100	200	500
Token	**95.29**	95.26	95.27	95.01	94.48	94.18	92.79
Strict	**87.99**	87.98	87.88	87.48	87.08	86.39	84.05
HIPAA Token	96.9	**96.95**	96.89	96.75	96.28	95.96	94.39
HIPAA Strict	89.77	**90.01**	89.6	89.49	89.33	88.5	85.9
Binary Token	97.09	97.25	**97.34**	97.01	96.69	96.49	94.9
Binary Strict	89.87	**90.1**	89.93	89.64	89.36	88.72	86.08
Binary HIPAA Token	97.24	**97.42**	**97.42**	97.19	96.75	96.51	94.96
Binary HIPAA Strict	90.13	**90.55**	90.21	90.01	89.8	89.11	86.39
NAME Token	94.42	94.51	**94.96**	94.77	93.78	93.77	94.37
NAME Strict	90.03	**90.79**	90.75	90.42	89.84	89.64	90.29
PROFESSION Token	**88.01**	85.89	86.96	81.88	82.11	83.99	82.32
PROFESSION Strict	73.56	70.56	**74.59**	66.47	68.07	72.39	69.49
LOCATION Token	**87.23**	86.15	85.25	84.6	83.58	83.03	79.95
LOCATION Strict	**79.86**	77.54	77.34	76.26	74.65	73.57	68.94
AGE Token	95.43	**95.8**	94.86	94.4	94.42	92.78	92.55
AGE Strict	**95.36**	95.27	94.62	93.89	94.16	92.49	91.85
DATE Token	98.23	98.47	**98.56**	98.36	98.1	97.77	95.9
DATE Strict	90.03	90.55	90.43	90.71	**90.74**	89.82	86.45
CONTACT Token	92.29	90.51	91.67	92.23	91.63	**92.88**	91.9
CONTACT Strict	**84.65**	81.45	83.99	73.36	74.77	75.18	68.48
ID Token	89.92	90.26	90.36	**90.51**	89.71	88.78	87.41
ID Strict	80.89	**82.38**	81.65	82.29	81.15	80.39	78.8

Table C.3: F_1 scores of our de-identification model when all PHI is moved to one of a certain number of neighbors N in the FastText embedding space.

Automatic Pseudonymization: GloVe

	Number of neighbors N						
Category	5	10	20	50	100	200	500
Token	95.35	**95.45**	94.78	93.92	94.24	94.34	93.2
Strict	88.13	**88.43**	86.86	85.72	86.35	86.07	84.59
HIPAA Token	96.98	**97.0**	96.39	95.87	95.87	95.91	95.04
HIPAA Strict	89.86	**90.01**	88.62	88.0	88.19	88.04	86.57
Binary Token	97.38	**97.4**	97.16	96.59	96.61	96.34	95.75
Binary Strict	90.13	**90.34**	89.52	88.57	88.41	87.85	87.18
Binary HIPAA Token	**97.44**	**97.44**	97.17	96.4	96.42	96.42	95.69
Binary HIPAA Strict	90.38	**90.42**	89.83	88.44	88.65	88.35	87.03
NAME Token	93.86	94.19	93.92	93.45	93.62	**94.62**	93.39
NAME Strict	89.23	**90.17**	89.05	89.05	89.05	89.61	88.59
PROFESSION Token	82.28	**86.24**	83.05	80.99	82.73	83.18	78.99
PROFESSION Strict	69.69	**78.01**	66.47	68.75	69.36	65.55	64.46
LOCATION Token	**87.61**	87.35	85.71	81.46	84.03	84.09	80.51
LOCATION Strict	80.11	**80.42**	77.09	71.0	74.43	74.42	70.14
AGE Token	**96.12**	95.33	94.76	94.91	95.25	94.71	93.66
AGE Strict	**95.87**	94.97	94.45	94.24	94.82	94.15	92.95
DATE Token	98.78	**98.81**	98.61	98.19	97.97	97.79	97.02
DATE Strict	**91.25**	91.14	90.72	90.42	89.94	89.59	88.07
CONTACT Token	90.89	**91.55**	85.84	90.37	90.46	90.54	89.95
CONTACT Strict	81.48	82.49	73.01	77.92	**83.33**	79.81	79.82
ID Token	88.75	**88.83**	86.72	86.09	85.89	85.17	86.43
ID Strict	**79.34**	78.59	75.52	73.22	74.21	71.28	74.76

Table C.4: F_1 scores of our de-identification model when all PHI is moved to one of a certain number of neighbors N in the GloVe embedding space.

Adversarial Representation: FastText

	Number of neighbors N						
Category	10	20	50	100	200	500	1 000
Token	94.97	95.08	**95.25**	95.23	**95.25**	95.12	94.5
Strict	87.3	87.79	**87.98**	87.8	87.93	87.51	86.53
HIPAA Token	96.48	96.75	**96.82**	**96.82**	96.8	96.77	96.16
HIPAA Strict	88.73	**89.55**	89.43	89.5	89.4	89.23	88.38
Binary Token	97.28	97.42	97.4	**97.46**	97.3	97.34	96.95
Binary Strict	89.63	**90.24**	90.11	**90.24**	90.14	89.89	89.01
Binary HIPAA Token	97.15	**97.41**	97.36	97.4	97.32	97.35	96.75
Binary HIPAA Strict	89.62	**90.44**	90.04	90.37	90.05	89.95	88.87
NAME Token	94.1	94.15	94.16	**94.58**	94.02	94.19	93.8
NAME Strict	89.22	**90.0**	89.3	89.83	89.79	89.81	89.25
PROFESSION Token	83.24	83.36	82.63	84.67	**87.52**	77.95	83.96
PROFESSION Strict	70.32	69.44	71.88	72.09	**76.62**	64.42	65.35
LOCATION Token	85.26	85.12	**86.79**	85.8	86.61	85.52	84.18
LOCATION Strict	77.14	77.7	**80.03**	78.02	78.37	76.63	75.52
AGE Token	93.02	94.72	**95.04**	94.54	94.58	94.62	93.6
AGE Strict	92.42	94.56	**94.69**	94.16	94.03	94.44	93.39
DATE Token	98.65	**98.76**	98.6	98.72	98.6	98.69	98.06
DATE Strict	90.59	**90.93**	90.55	90.9	90.62	90.59	89.53
CONTACT Token	89.89	89.61	91.53	90.44	90.04	**91.96**	91.86
CONTACT Strict	81.51	79.74	83.53	81.4	81.88	**84.04**	83.99
ID Token	89.93	89.7	89.46	89.08	89.92	**90.14**	89.12
ID Strict	81.61	81.09	81.18	80.29	**82.21**	81.78	79.35

Table C.5: F_1 scores of our de-identification model when using an adversarially trained representation of size $d = 50$ with invariance requirements for different numbers of neighbors N in the FastText embedding space.

Adversarial Representation: GloVe

	Number of neighbors N						
Category	10	20	50	100	200	500	1 000
Token	**94.64**	94.25	93.88	94.05	93.41	92.71	93.82
Strict	**86.86**	86.19	85.51	85.65	84.56	83.4	85.53
HIPAA Token	**96.59**	96.06	95.94	96.27	95.71	94.62	96.13
HIPAA Strict	**89.23**	88.28	88.03	88.4	87.49	85.44	88.41
Binary Token	**96.99**	96.87	96.51	96.85	96.3	95.61	96.51
Binary Strict	**89.3**	88.75	88.15	88.34	87.38	86.05	88.21
Binary HIPAA Token	**97.13**	96.78	96.6	96.89	96.4	95.44	96.76
Binary HIPAA Strict	**89.94**	89.14	88.83	89.13	88.34	86.18	89.3
NAME Token	**92.58**	91.88	92.19	91.79	91.57	91.38	92.15
NAME Strict	**87.4**	85.86	86.87	86.77	85.19	85.58	86.57
PROFESSION Token	72.7	**76.92**	73.48	72.84	76.11	72.54	73.94
PROFESSION Strict	58.75	**60.59**	55.26	55.14	59.55	55.84	56.27
LOCATION Token	85.26	**85.42**	82.25	83.03	81.05	81.29	80.86
LOCATION Strict	75.97	**77.0**	72.5	72.29	70.45	71.21	71.24
AGE Token	94.76	**95.12**	92.65	92.52	93.47	88.89	93.5
AGE Strict	94.53	**94.96**	92.6	92.15	93.58	88.59	93.28
DATE Token	98.61	98.44	98.49	**98.64**	98.13	97.59	98.58
DATE Strict	**90.78**	90.44	90.23	90.59	89.58	88.3	90.77
CONTACT Token	92.6	85.65	92.05	92.47	91.74	86.1	**92.8**
CONTACT Strict	85.46	77.24	85.19	84.21	84.71	72.8	**85.65**
ID Token	**90.23**	88.4	89.08	89.6	88.35	86.44	89.72
ID Strict	**82.88**	79.87	81.09	80.76	80.8	76.28	81.11

Table C.6: F_1 scores of our de-identification model when using an adversarially trained representation of size $d = 50$ with invariance requirements for different numbers of neighbors N in the GloVe embedding space.

Erklärung der Urheberschaft

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Informatik selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht.

Außerdem erkläre ich mein Einverständnis mit der Einstellung dieser Masterarbeit in den Bestand der Bibliothek.

Hamburg, 20. November 2018

Max Friedrich