Privacy-preserving Neural Representations of Text

Privacy-preserving Neural Representations of Text

Maximin Coavoux **Shashi Narayan** **Shay B. Cohen**
Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
{mcoavoux,scohen}@inf.ed.ac.uk,shashi.narayan@ed.ac.uk

Abstract

This article deals with adversarial attacks towards deep learning systems for Natural Language Processing (NLP), in the context of privacy protection. We study a specific type of attack: an attacker eavesdrops on the hidden representations of a neural text classifier and tries to recover information about the input text. Such scenario may arise in situations when the computation of a neural network is shared across multiple devices, e.g. some hidden representation is computed by a user's device and sent to a cloud-based model. We measure the privacy of a hidden representation by the ability of an attacker to predict accurately specific private information from it and characterize the tradeoff between the privacy and the utility of neural representations. Finally, we propose several defense methods based on modified training objectives and show that they improve the privacy of neural representations.

1 Introduction

This article presents an adversarial scenario meant at characterizing the privacy of neural representations for NLP tasks, as well as defense methods designed to improve the privacy of those representations. A deep neural network constructs intermediate hidden representations to extract features from its input. Such representations are trained to predict a label, and therefore should contain useful features for the final prediction. However, they might also encode information about the input that a user wants to keep private (e.g. personal data) and can be exploited for adversarial usages.

We study a specific type of attack on neural representations: an attacker eavesdrops on the hidden representations of novel input examples (that are not in the training set) and tries to recover information about the content of the input text (Figure 1). A typical scenario where such attacks would occur is when the computation of a deep neural net is shared between several devices (Li et al., 2017). For example, a user's device computes a representation of a textual input, and sends it a to cloud-based neural network to obtain, e.g. the topic of the text or its sentiment. The scenario is illustrated in Figure 1.

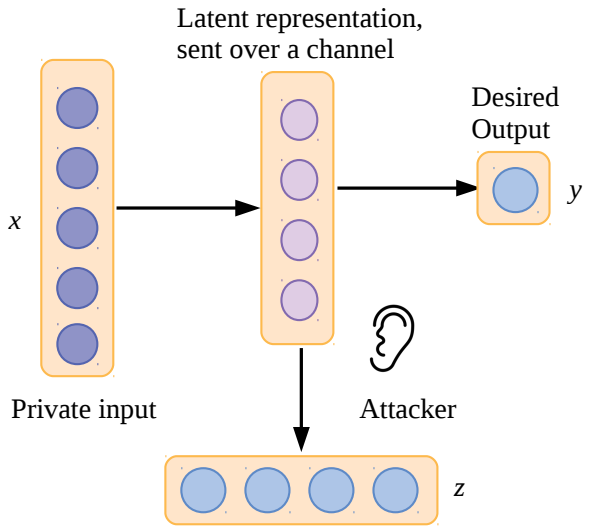

Figure 1: General setting illustration. The main classifier predicts a label y from a text x, the attacker tries to recover some private information $\mathbf{z}$ contained in x from the latent representation used by the main classifier.

Private information can take the form of key phrases *explicitly* contained in the text. However, it can also be *implicit*. For example, demographic information about the author of a text can be predicted with above chance accuracy from linguistic cues in the text itself (Rosenthal and McKeown, 2011; Preoţiuc-Pietro et al., 2015).

Independently of its explicitness, some of this private information correlates with the output labels, and therefore will be learned by the network. In such a case, there is a tradeoff between the utility of the representation (measured by the accuracy of the network) and its privacy. It might be

necessary to sacrifice some accuracy in order to satisfy privacy requirements.

However, this is not the case of all private information, since some of it is not relevant for the prediction of the text label. Still, private information might be learned incidentally. This non-intentional and incidental learning also raises privacy concerns, since an attacker with an access to the hidden representations, may exploit them to recover information about the input.

In this paper we explore the following situation: (i) a *main classifier* uses a deep network to predict a label from textual data; (ii) an *attacker* eavesdrops on the hidden layers of the network and tries to recover information about the input text of unseen examples. In contrast to previous work about neural networks and privacy (Papernot et al., 2016; Carlini et al., 2018) we do not protect the privacy of examples from the training set, but the privacy of unseen examples provided, e.g., by a user.

An example of a potential application would be a spam detection service with the following constraints: the service provider does not access verbatim emails sent to users, only their vector representations. Theses vector representations should not be usable to gather information about the user's contacts or correspondents, i.e. protect the user from profiling.

This paper makes the following contributions:[1]

- We propose a metric to measure the privacy of the neural representation of an input for Natural Language Processing tasks. The metric is based on the ability of an attacker to recover information about the input from the latent representation only.
- We present defense methods designed against this type of attack. The methods are based on modified training objectives and lead to an improved privacy-accuracy tradeoff.

2 Adversarial Scenario

In the scenario we propose, each example consists of a triple $(x, y, \mathbf{z})$, where x is a natural language text, y is a single label (e.g. topic or sentiment), and $\mathbf{z}$ is a vector of private information contained in x. Our base setting has two entities: (i) a *main classifier* whose role is to learn to predict y from x, (ii) an *attacker* who learns to predict $\mathbf{z}$ from the latent representation of x used by the main classifier. We illustrate this setting in Figure 1.

In order to evaluate the utility and privacy of a specific model, we proceed in three phases:

Phase 1. Training of the main classifier on (x, y) pairs and evaluation of its accuracy;
Phase 2. Generation of a dataset of pairs $(\mathbf{r}(x), \mathbf{z})$ for the attacker, $\mathbf{r}$ is the representation function of the main classifier ($\mathbf{r}$ is defined in Section 2.1);
Phase 3. Training of the attacker's network and evaluation of its performance for measuring privacy.

In the remainder of this section, we describe the main classifier (Section 2.1), and the attacker's model (Section 2.2).

2.1 Text Classifier

As our base model, we chose a standard LSTM architecture (Hochreiter and Schmidhuber, 1997) for sequence classification. LSTM-based architectures have been applied to many NLP tasks, including sentiment classification (Wang et al., 2016) and text classification (Zhou et al., 2016).

First, an LSTM encoder computes a fixed-size representation $\mathbf{r}(x)$ from a sequence of tokens $x = (x_1, x_2, \ldots, x_n)$ projected to an embedding space. We use $\boldsymbol{\theta}_r$ to denote the parameters used to construct $\mathbf{r}$. They include the parameters of the LSTM, as well as the word embeddings. Then, the encoder output $\mathbf{r}(x)$ is fed as input to a feedforward network with parameters $\boldsymbol{\theta}_p$ that predicts the label y of the text, with a softmax output activation. In the standard setting, the model is trained to minimize the negative log-likelihood of y labels:

$$\mathcal{L}_m(\boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = \sum_{i=1}^{N} -\log P(y^{(i)}|x^{(i)}; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p),$$

where N is the number of training examples.

2.2 Attacker's Classifier

Once the main model has been trained, we assume that its parameters $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_p$ are fixed. We generate a new dataset made of pairs $(\mathbf{r}(x), \mathbf{z}(x))$, where $\mathbf{r}(x)$ is the hidden representation used by the main model and $\mathbf{z}(x)$ is a vector of private categorical variables. In practice, $\mathbf{z}$ is a vector of binary variables, (representing e.g. demographic information about the author). In our experiments, we use the same training examples x for the main

[1] The source code used for the experiments described in this paper is available at https://github.com/mcoavoux/pnet.

classifier and the classifier of the attacker. However, since the attacker has access to the representation function $\mathbf{r}$ parameterized by $\boldsymbol{\theta}_r$, they can generate a dataset from any corpus containing the private variables they want to recover. In other words, it is not necessary that they have access to the original training corpus to train their classifier.

The attacker trains a second feedforward network on the new dataset $\{(\mathbf{r}(x^{(i)}), \mathbf{z}^{(i)})\}_{i \leq N}$. This classifier uses a sigmoid output activation to compute the probabilities of each binary variable in $\mathbf{z}$:

$$P(\mathbf{z}|\mathbf{r}(x); \boldsymbol{\theta}_a) = \sigma(\text{FeedForward}(\mathbf{r}(x))).$$

It is trained to minimize the negative log-likelihood of $\mathbf{z}$:

$$\begin{aligned}\mathcal{L}_a(\boldsymbol{\theta}_a) &= \sum_{i=1}^{N} -\log P(\mathbf{z}^{(i)}|\mathbf{r}(x^{(i)}); \boldsymbol{\theta}_a) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{K} -\log P(\mathbf{z}_j^{(i)}|\mathbf{r}(x^{(i)}); \boldsymbol{\theta}_a),\end{aligned}$$

assuming that the K variables in $\mathbf{z}$ are independent. Since the parameters used to construct $\mathbf{r}$ are fixed, the attacker only acts upon its own parameters $\boldsymbol{\theta}_a$ to optimize this loss.

We use the performance of the attacker's classifier as a proxy for privacy. If its accuracy is high, then an eavesdropper can easily recover information about the input document. In contrast, if its accuracy is low (i.e. close to that of a most-frequent label baseline), then we may reasonably conclude that $\mathbf{r}$ does not encode enough information to reconstruct x, and mainly contains information that is useful to predict y.

In general, the performance of a single attacker does not provide sufficient evidence to conclude that the input representation $\mathbf{r}$ is robust to an attack. It should be robust to any type of reconstruction method. In the scope of this paper though, we only experiment with a feedforward network reconstructor, i.e. a powerful learner.

In the following sections, we propose several training method modifications aimed at obfuscating private information from the hidden representation $\mathbf{r}(x)$. Intuitively, the aim of these modifications is to minimize some measure of information between $\mathbf{r}$ and $\mathbf{z}$ to make the prediction of $\mathbf{z}$ hard. An obvious choice for that measure would be the Mutual Information (MI) between $\mathbf{r}$ and $\mathbf{z}$. However, MI is hard to compute due to the continuous distribution of $\mathbf{r}$ and does not lend itself well to stochastic optimization.

3 Defenses Against Adversarial Attacks

In this section, we present three training methods designed as defenses against the type of attack we described in Section 2.2. The first two methods are based on two neural networks with rival objective functions (Section 3.1). The last method is meant at discouraging the model to cluster together training examples with similar private variables $\mathbf{z}$ (Section 3.2).

3.1 Adversarial Training

First, we propose to frame the training of the main classifier as a two-agent process: the main agent and an adversarial generator, exploiting a setting similar to Generative Adversarial Networks (GAN, Goodfellow et al., 2014). The generator learns to reconstruct examples from the hidden representation, whereas the main agent learns (i) to perform its main task (ii) to make the task of the generator difficult.

We experiment with two types of generators: a classifier that predicts the binary attributes $\mathbf{z}(x)$ used as a proxy for the reconstruction of x (Section 3.1.1) and a character-based language model that directly optimizes the likelihood of the training examples (Section 3.1.2).

3.1.1 Adversarial Classification: Multidetasking

In order not to make $\mathbf{r}(x)$ a good representation for reconstructing $\mathbf{z}$, we make two modifications to the training setup of the main model (Phase 1):

- We use a duplicate adversarial classifier, with parameters $\boldsymbol{\theta}'_a$, that tries to predict $\mathbf{z}$ from $\mathbf{r}(x)$. It is trained simultaneously with the main classifier. Its training examples are generated on the fly, and change overtime as the main classifier updates its own parameters. This classifier simulates an attack during training.
- We modify the objective function of the main classifier to incorporate a penalty when the adversarial classifier is good at reconstructing $\mathbf{z}$. In other words, the main classifier tries to update its parameters so as to confuse the duplicate attacker.

Formally, for a single data point $(x, y, \mathbf{z})$, the adversarial classifier optimizes:

$$\mathcal{L}_{a'}(x, y, \mathbf{z}; \boldsymbol{\theta}'_a)_{=} - \log P(\mathbf{z}|\mathbf{r}(x); \boldsymbol{\theta}'_a),$$

whereas the main classifier optimizes:

$$\mathcal{L}_m(x, y, \mathbf{z}; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = -\alpha \log P(y|x; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) - \beta \log P(\neg \mathbf{z}|\mathbf{r}(x); \boldsymbol{\theta}'_a).$$

The first term of this equation is the log-likelihood of the y labels. The second term is designed to deceive the adversary. The hyperparameters $\alpha > 0$ and $\beta > 0$ control the relative importance of both terms.

As in a GAN, the losses of both classifiers are interdependent, but their parameters are distinct: the adversary can only update $\boldsymbol{\theta}'_a$ and the main classifier can only update $\boldsymbol{\theta}_r$ and $\boldsymbol{\theta}_p$.

The duplicate adversarial classifier is identical to the classifier used to evaluate privacy after the main model has been trained and its parameters are fixed. However, both classifiers are completely distinct: the former is used during the training of the main model (Phase 1) to take privacy into account whereas the latter is used to evaluate the privacy of the final model (Phase 3), as is described in Section 2.

3.1.2 Adversarial Generation

The second type of generator we use is a character-based LSTM language model that is trained to reconstruct full training examples. For a single example $(x; y)$, the hidden state of the LSTM is initialized with $\mathbf{r}(x)$, computed by the main model. The generator optimizes:

$$\mathcal{L}_g(x, y; \boldsymbol{\theta}_\ell; \boldsymbol{\theta}_r) = -\log P(x|\mathbf{r}(x); \boldsymbol{\theta}_\ell) = -\sum_{i=1}^{C} \log P(x_i|x_1^{i-1}, \mathbf{r}(x); \boldsymbol{\theta}_\ell),$$

where $\boldsymbol{\theta}_\ell$ is the set of parameters of the LSTM generator, x_i is the i^{th} character in the document, and C is the length of the document in number of characters. The generator has no control over $\mathbf{r}(x)$, and optimizes the objective only by updating its own parameters $\boldsymbol{\theta}_\ell$.

Conversely, the loss of the main model is modified as follows:

$$\mathcal{L}_m(x, y; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = -\alpha \log P(y|x; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) - \beta \mathcal{L}_g(x, y; \boldsymbol{\theta}_\ell, \boldsymbol{\theta}_r).$$

The first term maximizes the likelihood of the y labels whereas the second term is meant at making the reconstruction difficult by maximizing the loss of the generator. As in the loss function described in the previous section, α and β control the relative importance of both terms. Once again, the main classifier can optimize the second term only by updating $\boldsymbol{\theta}_r$, since it has no control over the parameters of the adversarial generator.

A key property of this defense method is that it has no awareness of what the private variables $\mathbf{z}$ are. Therefore, it has the potential to protect the neural representation against an attack on any private information. From a broader perspective, the goal of this defense method is to specialize the hidden representation $\mathbf{r}(x)$ to the task at hand (sentiment or topic prediction) and to avoid learning anything not relevant to it.

3.2 Declustering

The last strategy we employ to make the task of the attacker harder is based on the intuition that private variables $\mathbf{z}$ are easier to predict from $\mathbf{r}$ when the main model learns implicitly to cluster examples with similar $\mathbf{z}$ in the same regions of the representation space.

In order to avoid such implicit clustering, we add a term to the training objective of the main model that penalizes pairs of examples (x, x') that (i) have similar reconstructions $\mathbf{z}(x) \approx \mathbf{z}(x')$ (ii) have hidden representations $\mathbf{r}(x)$ and $\mathbf{r}(x')$ in the same region of space. We use the following modified loss for a single example:

$$\mathcal{L}_m(x, y, \mathbf{z}; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) = -\log P(y|x; \boldsymbol{\theta}_r, \boldsymbol{\theta}_p) + \alpha(0.5 - \ell(\mathbf{z}, \mathbf{z}'))||\mathbf{r}(x) - \mathbf{r}(x')||_2^2,$$

where $(x', \mathbf{z}')$ is another example sampled uniformly from the training set, α is a hyperparameter controlling the importance of the second term, and $\ell(\cdot, \cdot) \in [0, 1]$ is the normalized Hamming distance.

4 Experiments

Our experiments are meant to characterize the privacy-utility tradeoff of neural representations on text classification tasks, and evaluating if the proposed defense methods have a positive impact on it. We first describe the datasets we used (Section 4.1) and the experimental protocol (Section 4.2), then we discuss the results (Section 4.3). We found that in the normal training regime, where no defense is taken into account, the adversary can recover private information with higher accuracy than a most frequent class baseline. Furthermore, we found that the de-

Dataset	Train	Dev	Test
TP US	22142	2767	2767
TP Germany	12596	1574	1574
TP Denmark	82193	10274	10274
TP France	9136	1141	1141
TP UK	48647	6080	6080
AG news	11657	1457	1457
DW corpus	5435	1772	1830
Blog posts	5144	642	642

Table 1: Sizes of datasets in number of examples.

fenses we implemented have a positive effect on the accuracy-privacy tradeoff.

4.1 Datasets

We experiment with two text classification tasks: sentiment analysis (Section 4.1.1) and topic classification (Section 4.1.2). The sizes of each dataset are summarized in Table 1.

4.1.1 Sentiment Analysis

We use the Trustpilot dataset (Hovy et al., 2015) for sentiment analysis. This corpus contains reviews associated with a sentiment score on a five point scale, and self-reported information about the users. We use the five subcorpora corresponding to five areas (Denmark, France, Germany, United Kingdom, United States).

We filter examples containing both the birth year and gender of the author of the review and use these variables as the private information. As in previous work on this dataset (Hovy, 2015; Hovy and Søgaard, 2015), we bin the age of the author into two categories ('under 35' and 'over 45'). Finally, we randomly split each subcorpus into a training set (80%), a development set (10%) and a test (10%).

As an additional experimental setting, we use both demographic variables (gender and age) as input to the main model. We do so by adding two additional tokens at the beginning of the input text, one for each variable. It has been shown that those variables can be used to improve text classification (Hovy, 2015). Also, we would like to evaluate whether the attacker's task is easier when the variables to predict are *explicitly* in the input, compared to when these information are only potentially and implicitly in the input. In other words, this setting simulates the case where private information may be used by the model to improve classification, but should not be exposed too obviously. In the rest of this section, we use RAW to denote the setting where only the raw text is used as input and +DEMO, the setting where the demographic variables are also used as input.

4.1.2 Topic Classification

We perform topic classification on two genres of documents: news articles and blog posts.

News article For topic classification of news article, we use two datasets: the AG news corpus[2] (Del Corso et al., 2005) and the English part of the Deutsche Welle (DW) news corpus (Pappas and Popescu-Belis, 2017).

For the AG corpus, following Zhang et al. (2015), we construct the dataset by extracting documents belonging to the four most frequent topics, and use the concatenation of the 'title' and 'description' fields as the input to the classifier. We randomly split the corpus into a training set (80%), a development set (10%) and a test set (10%). For the DW dataset, we use the 'text' field as input, and the standard split. We kept only documents belonging to the 20 most frequent topics.

The attacker tries to detect which named entities appear in the input text (each coefficient in $\mathbf{z}(x)$ indicates whether a specific named entity occurs in the text). For both datasets, we used the named entity recognition system from the NLTK package (Bird et al., 2009) to associate each example with the list of named entities that occur in it. We select the five most frequent named entities with type 'person', and only keep examples containing at least one of these named entities. This filtering is necessary to avoid a very unbalanced dataset (since each selected named entity appears usually in very few articles).

Blog posts We used the blog authorship corpus presented by Schler et al. (2006), a collection of blog posts associated with the age and gender of the authors, as provided by the authors themselves. Since the blog posts have no topic annotation, we ran the LDA algorithm (Blei et al., 2003) on the whole collection (with 10 topics). The LDA outputs a distribution on topics for each blog post. We selected posts with a single dominating topic ($> 80\%$) and discarded the other posts. We binned age into two category (under 20 and over 30). We

[2] http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html

	Baselines				Best adversaries			
	Lower bound (most frequent class)		Upper bound (trained)		+DEMO		RAW	
	Gender	Age	Gender	Age	Gender	Age	Gender	Age
TP (Denmark)	61.6	58.4	70.5	78.0	68.5	75.3	62.0	63.4
TP (France)	61.0	50.1	69.0	63.4	61.0	57.1	61.0	60.6
TP (Germany)	75.2	50.9	75.2	75.2	75.2	60.4	75.2	58.6
TP (UK)	58.8	56.7	70.0	76.3	66.4	63.5	59.9	61.8
TP (US)	63.5	63.7	74.1	74.8	81.3	74.9	64.7	63.9
Blogs	50.0	50.3	65.7	56.1	-	-	63.9	55.8

Table 2: Comparisons between baselines and best adversaries. All metrics reported in this table are accuracies.

used the age and gender of the author as the private variables. These variables have a very unbalanced distribution in the dataset, we randomly select examples to obtain uniform distributions of private variables. Finally, we split the corpus into a training set (80%), a validation set and a test set (10% each).

4.2 Protocol

Evaluation For the main task, we report a single accuracy measure. For measuring the privacy of a representation, we compute the following metrics:

- For demographic variables (sentiment analysis and blog post topic classification): $1 - X$, where X is the average of the accuracy of the attacker on the prediction of gender and age;
- For named entities (news topic classification): $1 - F$, where F is an F-score computed over the set of binary variables in $\mathbf{z}$ that indicate the presence of named entities in the input example.

Training protocol We implemented our model using Dynet (Neubig et al., 2017). The feedforward components (both of the main model and of the attacker) have a single hidden layer of 64 units with a ReLU activation. Word embeddings have 32 units. The LSTM encoder has a single layer of varying sizes, since it is expected that the amount of information that can be learned depends on the size of these representations. We used the Adam optimizer (Kingma and Ba, 2014) with the default learning rate, and 0.2 dropout rate for the LSTM. We used $\alpha = 0.1$ for the declustering method, based on preliminary experiments. For the other defense methods, we used $\alpha = \beta = 1$ and did not experiment with other values.

For each dataset, and each LSTM state dimension ($\{8, 16, 32, 64, 128\}$), we train the main model for 8 epochs (sentiment classification) or 16 epochs (topic classification), and select the model with the best accuracy on the development set. Then, we generate the dataset for the attacker, train the adversarial model for 16 epochs and select the model with the worst privacy on the development set (i.e. the most successful attacker).

It has to be noted that we select the models that implement defenses on their accuracy, rather than their privacy or a combination thereof. In practice, we could also base the selection strategy on a privacy budget: selecting the most accurate model with privacy above a certain threshold.

4.3 Results

This section discusses results for the sentiment analysis task (Section 4.3.1) and the topic classification task (Section 4.3.2).

4.3.1 Sentiment Analysis

How private are neural representations? Before discussing the effect of proposed defense methods, we motivate empirically our approach by showing that adversarial models can recover private information with reasonable accuracy when the attack is targeted towards a model that implements none of the presented defense methods.

To do so, we compare the accuracy of adversarial models to two types of baselines:

- As a lower bound, we use the most frequent class baseline.
- As an upper bound, we trained a classifier that can optimize the hidden representations ($\mathbf{r}$) for the attacker's tasks. In other words, this baseline is trained to predict de-

Corpus	Standard		M-Detask.		A-Gener.		Decl. $\alpha = 0.1$	
	Main	Priv.	Main	Priv.	Main	Priv.	Main	Priv.
Germany	85.1	32.2	-0.6	-0.3	-1.3	**+0.6**	-0.8	**+1.9**
baseline	78.6	36.9						
Denmark	82.6	28.1	-0.2	**+4.4**	-0.1	**+6.0**	-0.3	**+7.6**
baseline	70.4	40.0						
France	75.1	41.1	-0.8	**+0.7**	-1.4	-6.4	-1.5	-18.2
baseline	69.2	44.4						
UK	87.0	39.3	-0.5	**+0.9**	-0.2	**+0.2**	-0.1	**+0.3**
baseline	77.1	42.2						
US	85.0	33.9	-0.1	**+2.6**	-0.2	**+1.8**	**+0.7**	**+2.2**
baseline	79.4	36.4						

Table 3: Results on the test sets of the Trustpilot dataset, +DEMO setting. *Main* is the accuracy on sentiment analysis. *Priv.* is the privacy measure (i.e. the inverse accuracy of the attacker: higher is better, see Section 4.2). The baselines are most-frequent class classifiers. The values reported for the defense methods indicate absolute differences with the standard training regime (no defense implemented) for both metrics.

Corpus	Standard		M-Detask.		A-Gener.		Decl. $\alpha = 0.1$	
	Main	Priv.	Main	Priv.	Main	Priv.	Main	Priv.
Germany	85.5	32.1	**+0.3**	**+0.5**	-0.8	**+0.9**	-1.7	**+2.2**
baseline	78.6	36.9						
Denmark	82.3	37.3	-0.6	**+0.6**	-0.1	-0.3	-0.2	-0.1
baseline	70.4	40.0						
France	72.7	40.6	**+1.8**	-0.1	**+1.9**	-0.4	-0.3	-0.1
baseline	69.2	44.4						
UK	86.9	40.1	-0.2	**+1.0**	-0.0	**+1.2**	-0.0	0.0
baseline	77.1	42.2						
US	84.5	36.1	-1.1	**+0.2**	**+0.5**	**+0.1**	**+0.3**	**+0.5**
baseline	79.4	36.4						

Table 4: Results on the test sets of the Trustpilot dataset, RAW setting. See Section 4.2 and caption of Table 3 for details about the metrics.

Corpus	Standard		M-Detask.		A-Gener.		Decl. $\alpha = 0.1$	
	Main	Priv.	Main	Priv.	Main	Priv.	Main	Priv.
AG news	76.5	33.7	-14.5	**+14.5**	**+0.2**	-7.8	-2.5	**+8.6**
baseline	57.8							
DW news	44.3	78.3	-5.7	**+21.7**	**+5.9**	**+13.1**	-5.4	**+18.4**
baseline	22.1							
Blogs	58.3	40.8	-0.8	**+3.4**	**+1.1**	**+0.9**	-0.2	**+1.2**
baseline	47.8	49.8						

Table 5: Results for topic classification (test sets). See Section 4.2 and caption of Table 3 for details about the metrics.

mographic variables from x, as if it were the main task.

In Table 2, we compare both baselines to the best adversary in the two settings (RAW and +DEMO) among the models trained with no defenses. First of all, we observe that apart from gender on the German dataset, the trained baseline outperforms the most frequent class baseline by a wide margin (8 to 25 absolute difference). Second of all, the attacker is able to outperform the most frequent class baseline overall, even in the RAW setting. In more details, for age, the adversary is well over the baseline in all cases except US. On the other hand, gender seems harder to predict: the adversary outperforms the most frequent class baseline only in the +DEMO setting.

The same pattern is visible for the blog post dataset, also presented in the last line of Table 2: the best adversaries are 14 points over the baseline for gender and 5 points for age, i.e. almost as good as a model that can fine tune the hidden representations.

These results justify our approach, since they demonstrate that hidden representations learn private information about the input, and can be exploited to recover this information with reasonable accuracy.

Effect of defenses We report results for the main task accuracy and the representation privacy in Table 3 for the +DEMO setting and in Table 4 for the RAW setting. Recall that the privacy measure (Priv.) is computed by $1 - X$ where X is the average accuracy of the attacker on gender and age predictions. When this privacy metric is higher, it is more difficult to exploit the hidden representation of the network to recover information about x. The 'Standard' columns contain the accuracy and privacy of the base model described in Section 2. The next columns present the absolute variation in accuracy and privacy for the three defense methods presented in Section 3: Multidetasking, Adversarial Generation, and Declustering. We also report for each corpus the most frequent class baseline for the main task accuracy, and the privacy of the most frequent class baselines on private variables (i.e. the upper bound for privacy).

The three modified training methods designed as defenses have a positive effect on privacy. Despite a model selection based on accuracy, they lead to an improvement in privacy on all datasets, except on the France subcorpus. In most cases, we observe only a small decrease in accuracy, or even an improvement at times (e.g. multidetasking on the Germany dataset, RAW setting), thus improving the tradeoff between the utility and the privacy of the text representations.

4.3.2 Topic Classification

We report results on topic classification in Table 5.

News articles For the news corpora, the privacy metric is based on the F-score on the binary variables $\mathbf{z}$ indicating the presence or absence of a named entity in the text. First of all, we observe that defense methods that explicitly use $\mathbf{z}$ (i.e. multidetasking and declustering), have a very positive effect on privacy, but also a detrimental effect on the main task. We hypothesize that this is due to the strong correlations between the main task labels y and the private information $\mathbf{z}$. As a result, improving the privacy of the neural representations comes at a cost in accuracy.

In contrast, the adversarial generation defense method lead to an improvement in accuracy, that is quite substantial for the DW corpus. We speculate that this is due to the secondary term in the objective function of the main model (Section 3.1.2) that helps avoiding overfitting the main task or learning spurious features.

Blog posts On the blog post dataset, the effects are smaller, which we attribute to the nature of the task of the attacker. The defense methods consistently improve privacy and, in one case, accuracy. The best effects on the tradeoff are achieved with the multidetasking and adversarial generation methods.

5 Discussion

The main result of our experiments is that the defenses we propose improve privacy with usually a small effect, either positive or negative, on accuracy, thus improving the tradeoff between the utility and the privacy of neural representations.

An important direction for future work is the choice of a strategy for model selection. The tradeoff between utility and privacy can be controlled in many ways. For example, the importance of both terms in the loss functions in Section 3.1 can be controlled to favor either privacy or utility. In the scope of this paper, we did not perform thorough hyperparameter tuning, but believe that doing so is important for achieving better results, since the effects of defense method can be more drastic than desired in some cases, as exemplified on the news corpora (Table 5).

Overall, we found that the multidetasking approach lead to the more stable improvements and should be preferred in most cases, since it is also the less computationnally expensive defense. On the other hand, the adversarial generation method does not require the specification of private variables, and thus is a more general approach.

6 Related Work

The deployment of machine learning in both academic and industrial contexts raises concerns about adversarial uses of machine learning, as well as concerns about attacks specifically targeted at these algorithms that often rely on large amounts of data, including personal data.

More generally, the framework of differential privacy (Dwork, 2006) provides privacy guarantees for the problem of releasing information without compromising confidential data, and usually involves adding noise in the released information. It has been applied to the training of deep learning models (Abadi et al., 2016; Papernot et al., 2016; Papernot et al., 2018), and Bayesian topic models (Schein et al., 2018).

The notion of privacy is particularly crucial to NLP, since it deals with textual data, oftentimes user-generated data, that contain a lot of private information. For example, textual data contain a lot of signal about authors (Hovy and Spruit, 2016). and can be leveraged to predict demographic variables (Rosenthal and McKeown, 2011; Preoţiuc-Pietro et al., 2015). Oftentimes, this information is not explicit in the text but latent and related to the usage of various linguistic traits. Our work is based on a stronger hypothesis: this latent information is still present in vectorial representations of texts, even if the representations have not been supervised by these latent variables.

Li et al. (2017) study the privacy of unsupervised representations of images, and measures their privacy with the peak signal to noise ratio between an original image and its reconstruction by an attacker. They find a tradeoff between the privacy of the learned representations and the accuracy of an image classification model that uses these representations as inputs. Our setting is complementary since it is applied to NLP tasks, but explores a similar problem in the case of representations learned with a task supervision.

A related problem is the unintended memorization of private data from the training set and has been addressed by Carlini et al. (2018). They tackle this problem in the context of text generation (machine translation, language modelling).

If an attacker has access to e.g. a trained language model, they are likely to be able to generate sentences from the training set, since the language model is trained to assign high probabilities to those sentences. Such memorization is problematic when the training data contains private information and personal data. The experimental setting we explore is different from these works: we assume that the attacker has access to a hidden layer of the network and tries to recover information about an input example that is not in the training set.

In a recent study, Li et al. (2018) proposed a method based on GAN designed to improve the robustness and privacy of neural representations, applied to part-of-speech tagging and sentiment analysis. They use a training scheme with two agents similar to our multidetasking strategy (Section 3.1.1), and found that it made neural representations more robust and accurate. However, they only use a single adversary to alter the training of the main model and to evaluate the privacy of the representations, with the risk of overestimating privacy. In contrast, once the parameters of our main model are fixed, we train a new classifier from scratch to evaluate privacy.

7 Conclusion

We have presented an adversarial scenario and used it to measure the privacy of hidden representations in the context of two NLP tasks: sentiment analysis and topic classification of news article and blog posts. We have shown that in general, it is possible for an attacker to recover private variables with higher than chance accuracy, using only hidden representations. In order to improve the privacy of hidden representations, we have proposed defense methods based on modifications of the training objective of the main model. Empirically, the proposed defenses lead to models with a better privacy.

Acknowledgments

We thank the anonymous reviewers and members of the Cohort for helpful feedback on previous versions of the article. We gratefully acknowledge the support of the European Union under the Horizon 2020 SUMMA project (grant agreement 688139), and the support of Huawei Technologies.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. 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. ACM.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*, 1st edition. O'Reilly Media, Inc.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

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

Gianna M. Del Corso, Antonio Gullí, and Francesco Romani. 2005. Ranking a stream of news. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 97–106, New York, NY, USA. ACM.

Cynthia Dwork. 2006. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming, part II (ICALP 2006)*, volume 4052, pages 1–12, Venice, Italy. Springer Verlag.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. 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, Inc.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Dirk Hovy. 2015. Demographic factors improve classification performance. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 752–762, Beijing, China. Association for Computational Linguistics.

Dirk Hovy, Anders Johannsen, and Anders Søgaard. 2015. User review sites as a resource for large-scale sociolinguistic studies. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 452–461, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee.

Dirk Hovy and Anders Søgaard. 2015. Tagging performance correlates with author age. In *Proceedings of the 53rd Annual Meeting of the Association for*

Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers), pages 483–488, Beijing, China. Association for Computational Linguistics.

Dirk Hovy and Shannon L. Spruit. 2016. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598, Berlin, Germany. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Meng Li, Liangzhen Lai, Naveen Suda, Vikas Chandra, and David Z. Pan. 2017. Privynet: A flexible framework for privacy-preserving deep neural network training with A fine-grained privacy control. *CoRR*, abs/1709.06161.

Yitong Li, Timothy Baldwin, and Trevor Cohn. 2018. 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, Melbourne, Australia. Association for Computational Linguistics.

Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, Kevin Duh, Manaal Faruqui, Cynthia Gan, Dan Garrette, Yangfeng Ji, Lingpeng Kong, Adhiguna Kuncoro, Gaurav Kumar, Chaitanya Malaviya, Paul Michel, Yusuke Oda, Matthew Richardson, Naomi Saphra, Swabha Swayamdipta, and Pengcheng Yin. 2017. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*.

Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian J. Goodfellow, and Kunal Talwar. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *CoRR*, abs/1610.05755.

Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson. 2018. Scalable Private Learning with PATE. *ArXiv e-prints*, abs/1802.08908.

Nikolaos Pappas and Andrei Popescu-Belis. 2017. Multilingual hierarchical attention networks for document classification. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1015–1025, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Daniel Preoţiuc-Pietro, Vasileios Lampos, and Nikolaos Aletras. 2015. An analysis of the user occupational class through twitter content. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1754–1764, Beijing, China. Association for Computational Linguistics.

Sara Rosenthal and Kathleen McKeown. 2011. Age prediction in blogs: A study of style, content, and online behavior in pre- and post-social media generations. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 763–772, Portland, Oregon, USA. Association for Computational Linguistics.

Aaron Schein, Zhiwei Steven Wu, Mingyuan Zhou, and Hanna Wallach. 2018. Locally Private Bayesian Inference for Count Models. *ArXiv e-prints*, abs/1803.08471.

Jonathan Schler, Moshe Koppel, Shlomo Argamon, and James Pennebaker. 2006. Effects of age and gender on blogging. In *Computational Approaches to Analyzing Weblogs - Papers from the AAAI Spring Symposium, Technical Report*, volume SS-06-03, pages 191–197.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 606–615, Austin, Texas. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3485–3495, Osaka, Japan. The COLING 2016 Organizing Committee.