Obfuscation for Privacy-preserving Syntactic Parsing

Zhifeng Hu[♠*] **Serhii Havrylov**[♢] **Ivan Titov**[♢♡] **Shay B. Cohen**[♢]
[♠]School of Computer Science, Fudan University, Shanghai 201203, China
[♢]School of Informatics, University of Edinburgh, Edinburgh EH8 9AB, UK
[♡]ILLC / FNWI, University of Amsterdam, Amsterdam 1098XG, Netherlands
zfhu16@gmail.com, s.havrylov@ed.ac.uk
ititov@inf.ed.ac.uk, scohen@inf.ed.ac.uk

Abstract

The goal of homomorphic encryption is to encrypt data such that another party can operate on it without being explicitly exposed to the content of the original data. We introduce an idea for a privacy-preserving transformation on natural language data, inspired by homomorphic encryption. Our primary tool is *obfuscation*, relying on the properties of natural language. Specifically, a given English text is obfuscated using a neural model that aims to preserve the syntactic relationships of the original sentence so that the obfuscated sentence can be parsed instead of the original one. The model works at the word level, and learns to obfuscate each word separately by changing it into a new word that has a similar syntactic role. The text obfuscated by our model leads to better performance on three syntactic parsers (two dependency and one constituency parsers) in comparison to an upper-bound random substitution baseline. More specifically, the results demonstrate that as more terms are obfuscated (by their part of speech), the substitution upper bound significantly degrades, while the neural model maintains a relatively high performing parser. All of this is done without much sacrifice of privacy compared to the random substitution upper bound. We also further analyze the results, and discover that the substituted words have similar syntactic properties, but different semantic content, compared to the original words.

1 Introduction

We consider the case in which there is a powerful server with NLP technology deployed on it, and a set of clients who would like to access it to get output resulting from input text taken from problems such as syntactic parsing, semantic parsing and machine translation. In such a case, the server models may have been trained on large amounts of data, yielding models that cannot be deployed on the client machines either for efficiency or licensing reasons. We ask the following question: how can we use the NLP server models while minimizing the exposure of the server to the original text? Can we exploit the fact we work with natural language data to reduce such exposure?

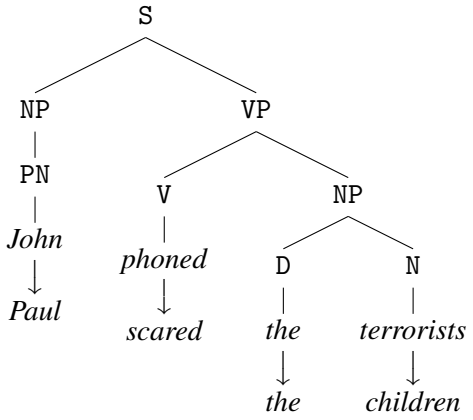

Figure 1: An example of a sentence (words on top) and an obfuscated version of the sentence (words at bottom), both having identical syntactic structure. The obfuscated sentence hides the identity of the person who performs the action and the action itself.

Conventional encryption schemes, including public-key cryptography which is the one widely used across the Internet, are not sufficient to answer this question. They encrypt the input text before it is transferred to the server side. However, once the server decrypts the text, it has full access to it. This might be unacceptable if the server itself is not necessarily trustworthy.

The cryptography community posed a similar question much earlier, in the 1970s (Rivest et al., 1978) with partial resolutions proposed to solve it in later research (Sander et al., 1999; Boneh et al., 2005; Ishai and Paskin, 2007). These solutions al-

[*] Work done at the University of Edinburgh.

Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task, pages 61–71

https://doi.org/10.18653/v1/P17

low the server to perform computations directly on encrypted data to get the desired output without ever decrypting the data. This cryptographic protocol is known as *homomorphic encryption*, where a client encrypts a message, then sends it to a server which performs potentially computationally intensive operations and returns a new data, still encrypted, which only the client can decipher. All of this is done without the server itself ever being exposed to the actual content of the encrypted input data. While solutions for generic homomorphic encryption have been discovered, they are either computationally inefficient (Gentry, 2010) or have strong limitations in regards to the depth and complexity of computation they permit (Bos et al., 2013).

In this paper, we consider a softer version of homomorphic encryption in the form of *obfuscation* for natural language. Our goal is to identify an efficient function that stochastically transforms a given natural language input (such as a sentence) into another input which can be further fed into an NLP server. The altered input has to preserve intra-text relationships that exist in the original sentence such that the NLP server, depending on the task at hand, can be successfully applied on the transformed data. There should be then a simple transformation that maps the output on the obfuscated data into a valid, accurate output for the original input. In addition, the altered input should hide the private semantic content of the original data.

This idea is demonstrated in Figure 1. The task at hand is syntactic parsing. We transform the input sentence *John phoned the terrorists* to the sentence *Paul scared the children* – both of which yield identical phrase-structure trees. In this case, the named entity *John* is hidden, and so are his actions. In the rest of the paper, we focus on this problem for dependency and constituency parsing.

We consider a neural model of obfuscation that operates at the word level. We assume access to the parser at training time: the model learns how to substitute words in the sentence with other words (in a stochastic manner) while maintaining the highest possible parsing accuracy. This learning task is framed as a latent-variable modeling problem where the obfuscated words are treated as latent. Direct optimization of this model turns out to be intractable, so we use continuous relaxations (Jang et al., 2016; Maddison et al., 2017) to avoid explicit marginalization.

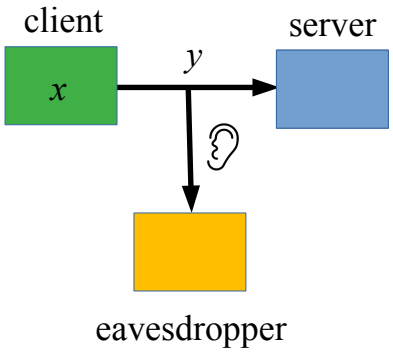

Figure 2: General setting illustration (figure adapted from Coavoux et al. 2018). An NLP client encrypts an x into y through obfuscation and y is sent to an NLP server. The NLP server (potentially even a legacy one) does not need to be modified to de-obfuscate y. An eavesdropper (a possibly malicious channel listener) only has access to y which is needed to be de-obfuscated to gain any information about x.

Our experimental results on English demonstrate that the neural model performs better than a strong random-based baseline (an upper bound; in which a word is substituted randomly with another word with the same part-of-speech tag). We vary the subset of words that are hidden and observe that the higher the obfuscation rate of the words, the harder it becomes for the parser to retain its accuracy. Degradation is especially pronounced with the random baseline and is less severe with our neural model. The improved results for the neural obfuscator come at a small cost to the accuracy of the attacker aimed at recovering the original obfuscated words. We also observe that the neural obfuscator is effective when different parsers or even different syntactic formalisms are used in training and test time. This relaxes the assumption that the obfuscator needs to have access to the NLP server at training time. Our results also suggest that the neural model tends to replace words with ones that have similar syntactic properties.

2 Homomorphic Obfuscation of Text

Our problem formulation is rather simple, demonstrated in generality in Figure 2. Let $\mathcal{T}$ be some natural language task, such as syntactic parsing, where $\mathcal{X}$ is the input space and $\mathcal{Z}$ is the output space. Let $f_{\mathcal{T}}\colon \mathcal{X} \to \mathcal{Z}$ be a trained decoder that maps x to its corresponding structure according to $\mathcal{T}$. Note that f is trained as usual on labeled data. Given a sentence $x = x_1 \cdots x_n$, we aim to learn a function that stochastically transforms x

into $y = y_1 \cdots y_n$ such that $f_{\mathcal{T}}(x)$ is close, if not identical, to $f_{\mathcal{T}}(y)$, or at the very least, we would like to be able to recover $f_{\mathcal{T}}(x)$ from $f_{\mathcal{T}}(y)$ using a simple transformation.

To ground this in an example, consider the case in which $\mathcal{T}$ is the problem of dependency parsing and $\mathcal{Z}$ is the set of dependency trees. If we transform a sentence x to y in such a way that it preserves the syntactic relationship between the indexed words in the sentences, then we can expect to easily recover the dependency tree for x from a dependency tree for y.

Note that we would also want to stochastically transform x into a y in such a way that it is *hard* to recover a certain type of information in x from y (otherwise, we could just set $y \leftarrow x$). Furthermore, we are interested in hiding information such as named entities or even nouns and verbs. In our formulation, we also assume that the sentence x comes with a function $t(x)$ that maps each token in the sentence with its corresponding part-of-speech tag (predicted using a POS tagger).

3 Neural Obfuscation Model

In this section we describe the neural model used to obfuscate the sentence. We note that the model has to be simple and efficient, as it is being run by the obfuscating party. If it is more complicated than parsing the text, for example, then the obfuscating party might as well directly parse the text.[1]

3.1 The Main Model

Our model operates by transforming a subset of the words in the sentence into new words. Each of these words is separately transformed in a way that maintains the sentence length after the transformation. Let x be the original sentence $x = x_1 \cdots x_n$ and let y be the output, $y = y_1 \cdots y_n$. From a high-level point of view, we have a conditional model:

$$p(y \mid x, \theta) = \prod_{i=1}^{n} p(y_i \mid x, \theta). \quad (1)$$

The selection of words to obfuscate depends on their part of speech (POS) tags – only words that are associated with specific POS tags from the set $\mathcal{P}$ are obfuscated under our model. Let t_i be the POS tag of the ith word in the sentence. In our basic model, we apply a bidirectional Long Short-Term Memory network (BiLSTM) to the sentence to get a latent representation h_i for each word x_i (see Section 3.2).

We assume conditional independence between the sequence $x_1 \cdots x_{i-1} x_{i+1} \cdots x_n$ and y_i given h_i (which is a function of x), and as such, our probability distribution $p(y_i \mid x, \theta)$ is given by:

$$p(y_i = y \mid x_i, h_i, \theta) = \begin{cases} 1 & t_i \notin \mathcal{P}, y = x_i \\ p_y & t_i \in \mathcal{P}, y \in \mathcal{V}_{t_i} \setminus \{x_i\} \\ 0 & \textit{otherwise}. \end{cases} \quad (2)$$

Here, $\mathcal{V}_{t_i}$ is the set of word types appearing at least once with tag t_i in the training set, and p_y is predicted with a softmax function, relying on the BiLSTM state h_i. More specifically, we define p_y as follows:

$$p_y = \frac{\exp(w_{t_i,y}^{\top} h_i)}{\sum_{y' \in \mathcal{V}_{t_i}, y' \neq x_i} \exp(w_{t_i,y'}^{\top} h_i)},$$

where $w_{t,y} \in \mathbb{R}^{1024}$ are vectors of parameters associated with every tag-word pair (t, y), $y \in \mathcal{V}_t$. Note that the above probability distribution never transforms a word x_i to an identical word if $t_i \in \mathcal{P}$. This is a hard constraint in our model.

3.2 Embedding the Sentence

The BiLSTM that encodes the sentence requires an embedding per word, which we create as follows. We first map each token x_i to three embedding channels e_i^k, $k \in \{1, 2, 3\}$. The first channel is a randomly initialized embedding for each part-of-speech tag. Its dimension is 100. The second channel is a pre-trained GloVe embedding for the corresponding token. The vector e_i^3 is a character-level word embedding (Kim et al., 2016) which first maps each character of the word into an embedding vector of dimension 100 and then uses unidimensional convolution over the concatenation of the embedding vectors of each character. Finally, max-pooling is applied to obtain a single feature. This process is repeated with 100 convolutional kernels so that $e_i^3 \in \mathbb{R}^{100}$.

The three embedding channels $\{e_i^1, e_i^2, e_i^3\}$ are then concatenated and used in the BiLSTM encoder. We use a three-layer BiLSTM with Bayesian dropout (Gal and Ghahramani, 2016). The hidden state dimensionality is 512 for each direction.

[1] In the general case, there is a caveat to this statement. It might be the case that the training cost for the server's model is high, and that the model is proprietary. In that case, even if the model can be run on the client side, it might not be possible to do so.

4 Training

In our experiments, we focus on obfuscation for the goal of syntactic parsing. We assume the existence of a conditional parsing model $p_0(z \mid x)$ where z is a parse tree and x is a sentence. This is the base model which is trained offline, and to which we have read-only access and cannot change its parameters. As we will see in experiments, the obfuscator can be trained using a different parser from the one used at test time (i.e. from the one hosted at the NLP server).

Let $(x^{(1)}, z^{(1)}), \ldots, (x^{(n)}, z^{(n)})$ be a set of training examples which consists of sentences and their corresponding parse trees. Considering Eq. 1, we would be interested in maximizing the following log-likelihood objective with respect to θ:

$$\mathcal{L}_0 = \sum_{i=1}^{n} \log \left(\sum_{y} p(y \mid x^{(i)}, \theta) p_0(z^{(i)} \mid y) \right).$$

This objective maximizes the log-likelihood of the parsing model with respect to the obfuscation model. Maximizing the objective $\mathcal{L}_0$ is intractable due to summation over all possible obfuscations. We use Jensen's inequality[2] to lower-bound the cost function $\mathcal{L}_0$ by the following objective:

$$\begin{aligned}\mathcal{L} &= \sum_{i=1}^{n} \sum_{y} p(y \mid x^{(i)}, \theta) \log p_0(z^{(i)} \mid y) \\ &= \sum_{i=1}^{n} \mathbb{E}_{p(\cdot \mid x^{(i)}, \theta)} \left[\log p_0(z^{(i)} \mid y) \right].\end{aligned}$$

Intuitively, the objective function maximizes the accuracy of an existing parser while using as an input the sentences after their transformation. Note that the accuracy is measured with respect to the gold-standard dependency parse tree.[3] This is possible because the sentence length of the original sentence and the obfuscated sentence are identical, and the mapping between the words in each version of the sentence is bijective.

To encourage stochasticity, we also tried including an entropy term that is maximized with respect to θ in the following form:

$$H_i(\theta, \lambda) = -\lambda \sum_{y} p(y \mid x^{(i)}, \theta) \log p(y \mid x^{(i)}, \theta).$$

However, in our final experiments we omitted that term because (a) it did not seem to affect the model stochasticity to a significant degree; (b) the performance has become very sensitive to the entropy weight λ.

While we can estimate the objective $\mathcal{L}$ using sampling, we cannot differentiate through samples to estimate the gradients with respect to the obfuscator parameters θ. In order to ensure end-to-end differentiabilty, we use a continuous relaxation, the Gumbel-Softmax estimator (Jang et al., 2016; Maddison et al., 2017), and the reparamterization trick (Kingma and Welling, 2014; Rezende et al., 2014).

More formally, the i-th token is represented by the random variable with categorical probability distribution $\mathrm{Cat}(p_i)$ that has support $\mathcal{V}_{t_i}$. To sample the word we first draw $u_k \sim \mathrm{Uniform}(0, 1)$ and transform it to the Gumbel noise $g_k = -\log(-\log(u_k))$, then we calculate

$$y' = \mathrm{onehot} \left\{ \arg\max_{k \in \mathcal{V}_{t_i}} [g_k + \log(p_{i,k})] \right\}$$

as the sampled discrete choice of substitution from $\mathcal{V}_{t_i}$ and

$$y_k = \frac{\exp\left((g_k + \log(p_{i,k})/\tau\right))}{\sum_{k'} \exp\left((g_{k'} + \log(p_{i,k'})/\tau\right))}$$

as the "relaxed" differentiable proxy for this choice, where τ denotes the temperature. When it approaches 0, the vector $(y_1, \ldots, y_{|\mathcal{V}_{t_i}|})$ is close to a one-hot vector sampled from the given categorical distribution.[4]

We use the Straight-Through version of the estimator (Bengio et al., 2013): the discrete sampled choice is fed into the parser in the forward computation but the relaxed differentiable surrogate is used when computing partial derivatives on the backward pass.

During the training of our neural model, the parser only backpropagates the gradient from the objective of maximizing the parsing accuracy (i.e. minimum cross-entropy loss of the correct head and label for each word), and hence its parameters are always fixed and are not updated during the optimization.

[2] Jensen's inequality states that for a non-negative random variable Z and its probability distribution q it holds that $\log(\mathbb{E}_q[Z]) \geq \mathbb{E}_q[\log Z]$.

[3] In principle, we may not need access to gold-standard annotation when training the obfuscator. Instead, we could train the model to agree with the parser predictions for the original sentence, i.e. $z^{(i)} = \arg\max_z p_0(z|x^{(i)})$.

[4] In practice, we anneal the temperature from 1.0 to 0.5 over the course of training.

5 Attacker Approaches

We test the efficiency of our obfuscation model by developing two independent attacker models. Their goal is to recover the original words by inspecting only the obfuscated sentence. The attacker models may have access to all data that the parser and the obfuscator models were trained and developed on. This is perhaps unlike other scenarios in which the training set is assumed to be inaccessible to any attacker.

We note that ideally, we would want to show that our obfuscation model retains privacy universally for *any* attacker. However, this is quite a difficult task, and we follow Coavoux et al. (2018) in presenting two strong attackers which may represent possible universal attackers.

In our attacker experiments, we assume that it is known which words in the sentence are obfuscated. As such, the results we provide for attacking our obfuscation are an upper bound. In practice, an attacker would also have to identify which words were substituted for new words, which may lead to a small decrease in its accuracy.

5.1 Trained Attacker

Our first attacker works by first encoding the obfuscated sentence with a BiLSTM network. We then try to predict original words by using a feed-forward neural network on each of the hidden representations obtained from the encoder model. The architecture is identical to that of the obfuscation model (see Section 3.1), with the only difference that there is a softmax over the entire vocabulary $\mathcal{V}$ instead of restricting it to $V_{t_i} \setminus \{x_i\}$, as in Eq. 2.

5.2 Pretrained Attacker

In addition to a trained attacker, we also use a conditional language model, BERT (Devlin et al., 2019).[5] BERT is based on the Transformer model of Vaswani et al. (2017), and uses a bidirectional encoder to obtain "contextual" embeddings for each word in a given sentence. We use the BERT model by masking out each obfuscated word, and then predicting the masked word similar to the "masked language task" that is mentioned by Devlin et al. (2019). This means that the embeddings in each position are fed into a softmax function to predict the missing word. We use the `bert-base-uncased` model among the available BERT models.

We note that this attacker is not trained by us. Its main weakness is that it is trained on the non-obfuscated text. However, its strength is that it is trained on large amounts of data (we use the model that is trained on 3.3 billion tokens). In addition, in some settings that we consider the obfuscation of the sentence is done in such a way that much of the context by which we predict the obfuscated word remains intact.

6 Experiments

In this section, we describe our experiments with our obfuscation model. We first describe the experimental setting and then turn to the results.[6]

6.1 Experimental Setting

In our experiments, we test the obfuscation model on two parsers. The first parser is used during the training of our model. This is the bi-affine dependency parser developed by Dozat and Manning (2017). To test whether our obfuscation model also generalizes to syntactic parsers that were not used during its training, the constituency parser that is included in the AllenNLP software package (Gardner et al., 2018) was used.[7]

For our dependency parser, we follow the canonical setting of using pre-trained word embedding, 1D convolutional character level embedding and POS tag embedding, each of 100 dimensions as the input feature. We also use a three-layer bi-directional LSTM with Bayesian dropout (Gal and Ghahramani, 2016) as the encoder. We use the bi-affine attention mechanism to obtain the prediction for each head, and also the prediction for the edge labels.

We use the English Penn Treebank (PTB; Marcus et al. 1993) version 3.0 converted using Stanford dependencies for training the dependency parser. We follow the standard parsing split for training (sections 01–21), development (section 22) and test sets (section 23). The training set portion of the PTB data is also used to train our neural obfuscator model.

We also create a spectrum over the POS tags to decide on the set $\mathcal{P}$ for each of our experiments (see Section 3.1). This spectrum is described in Table 1.

[5] We use the implementation available at `https://github.com/huggingface/pytorch-pretrained-BERT`.

[6] Our code is available at `https://github.com/ichn-hu/Parsing-Obfuscation`.

[7] We used version 0.8.1.

i	Category description	$\mathcal{P}_i$
1	Named entities	NNP, NNPS
2	Nouns	NN, NNS
3	Adjectives	JJ, JJR, JJS
4	Verbs	VB, VBN, VBD, VBZ, VBP, VBG
5	Adverbs	RB, RBR, RBS

Table 1: A spectrum of part-of-speech tags to obfuscate. In the jth experiment, we set $\mathcal{P} = \cup_{i=1}^{j} \mathcal{P}_i$.

Let the ith set in that table be $\mathcal{P}_i$ for $i \in [5]$[8]. In our jth experiment, $j \in [5]$, we obfuscate the set $\mathcal{P} = \cup_{i=1}^{j} \mathcal{P}_i$. This spectrum of POS tags describes a range from words that are highly content-bearing for privacy concerns (such as named entities) to words that are less of a privacy concern (such as adverbs).

We compare our model against a (privacy) upper-bound baseline which is found to be rather strong. With this baseline, a word x with a tag $t \in \mathcal{P}$ is substituted with another by a word that appeared with the same tag in the training data from the set $\mathcal{V}_t$. The substituted words are uniformly sampled. This random baseline serves as an *upper* bound for the privacy level achieved, not a lower bound. Randomly substituting a word with another makes it difficult to recover the original word. However, in terms of parsing accuracy, as we see below, there is a significant room for improvement over that baseline. There are words, which when substituted by other group of words, yield altogether better parsing accuracy.

We evaluate the model through three different principal metrics. The first metric is the accuracy of the parser (Unlabeled Attachment Score or Labeled Attachment Score, depending on the context[9]), the second is the mean reciprocal rank (MRR) of the attacker on the relevant method. With each of the attacker methods, we obtain a distribution q over the possible true outputs recovered from the obfuscated word. We use this distribution to calculate the MRR, the average inverse rank of the correct word according to q. More precisely, the score we calculate is:

$$\mathrm{MRR} = \frac{100}{N} \sum_{i=1}^{N} \frac{1}{r_i},$$

where $r_i \in \mathbb{N}$ is the rank of the ith word (in the whole corpus) according to q (the distribution over possible output words for that word).[10] The result we report is attacker error, or $100 - \mathrm{MRR}$ (the higher it is, the more privacy is maintained). Finally, we also report the ratio between the accuracy of the parser[11] and the accuracy of an attacker (MRR). This metric provides a way to measure the amount of accuracy we gain for each point of privacy we lose, in the form of $\frac{\text{accuracy}}{\text{breach}}$ units.

All neural experiments were run on a single GeForce GTX 1080 Ti GPU. The time to run each of the experiments was in the range of 13.3 hours to 25.2 hours.

6.2 Results with Dependency Parsing

In our first set of experiments, we compare the random substitution model to the neural model of obfuscation on the dependency parsing model that was used during training. As is seen in Table 2(a), both the neural model and the random model, with their substitutions, do not significantly harm the accuracy of the parser for the first three experiments (corresponds to rows in the table). However, as the number of obfuscated words increases (with the categories of verbs and adverbs), the accuracy of the parser significantly drops, significantly more for the random model than the neural model.

The attacker error is in the range of the 70%–80% for all experiments. As expected, the random model, given its inherent unpredictability, makes it more difficult for the attacker to identify the original words. That leads to the ratio between accuracy and intrrusion level often being better with the random model. In general, it also seems that the BERT attacker gives similar results to the trained attacker for the random baseline, and worse results with the neural model. Finally, it is evident that as we obfuscate more terms, the attacker's accuracy decreases, with the BERT attacker consistently outperforming the trained attacker.

We next turn to inspect the problem of dependency parsing without a parser that was trained

[8] For an integer k, we denote by $[k]$ the set $\{1, ..., k\}$.

[9] The unlabeled/labeled distinction refers to whether the label of the edge in the dependency tree is taken into account when computing the accuracy.

[10] Note that we have a multiplier of 100 in our MRR score definition. This deviates from the standard definition of this score.

[11] The accuracy is labeled attachment score in the case of dependency parsing.

(a)

	Obf. terms	Random (baseline)						Neural model					
				trained		BERT				trained		BERT	
		acc (U L)		prv	ratio	prv	ratio	acc (U L)		prv	ratio	prv	ratio
trained dep.	Named ent.	94.1	93.0	68.3	2.97	66.9	2.84	94.3	92.9	68.4	2.98	66.4	2.81
	+Nouns	93.7	92.9	70.7	3.20	70.3	3.15	94.1	92.4	69.7	3.11	69.4	3.08
	+Adjectives	93.1	92.4	71.9	3.31	72.3	3.36	93.6	91.7	70.5	3.17	70.1	3.13
	+Verbs	85.2	80.4	68.1	2.67	80.2	4.30	87.3	78.7	65.3	2.52	78.1	3.99
	+Adverbs	86.4	78.7	67.2	2.63	81.2	4.60	88.6	76.6	64.2	2.47	77.5	3.94
	No obf.	95.0/93.5 (U/L)											
AllenNLP dep.	Named ent.	91.9	89.7	68.3	2.90	66.9	2.78	92.2	90.1	68.4	2.92	66.4	2.74
	+Nouns	91.5	89.2	70.7	3.12	70.3	3.08	91.5	89.4	69.7	3.02	69.4	2.99
	+Adjectives	90.8	88.5	71.9	3.23	72.3	3.28	91.2	89.0	70.5	3.09	70.1	3.05
	+Verbs	78.2	75.3	68.1	2.45	80.2	3.95	82.2	79.4	65.3	2.37	78.1	3.75
	+Adverbs	76.7	73.5	67.2	2.34	81.2	4.08	82.0	78.9	64.2	2.29	77.5	3.64
	No obf.	94.2/92.6 (U/L)											

(b)

	Obf. terms	Random (baseline)					Neural model				
			trained		BERT			trained		BERT	
		acc (F_1)	prv	ratio	prv	ratio	acc (F_1)	prv	ratio	prv	ratio
AllenNLP const.	Named ent.	92.4	68.3	2.91	66.9	2.79	92.5	68.4	2.93	66.4	2.75
	+Nouns	88.2	70.1	2.95	70.3	2.97	89.0	69.7	2.94	69.4	2.91
	+Adjectives	86.8	71.9	3.09	72.3	3.13	88.1	70.5	2.99	70.1	2.95
	+Verbs	79.2	68.1	2.48	80.2	4.00	82.5	65.3	2.38	78.1	3.77
	+Adverbs	76.8	67.2	2.34	81.2	4.09	79.5	64.2	2.22	77.5	3.53
	No obf.	93.7									

Table 2: (a) Results of parsing accuracy and attacker error for two different dependency parsers. "acc" denotes accuracy (Unlabeled Attachment Score/Labeled Attachment Score for the dependency parsers), "prv" denotes the attacker error (trained attacker and BERT attacker as described in Section 5.1 Section 5.2) and "ratio" is the ratio between the parser accuracy and the attacker error. Two parsers are considered: a parser that participates in the obfuscation model optimization (top part), and offline-trained parsers from the AllenNLP for dependency (bottom part). Two obfuscation models are considered: neural (Section 3.1) and a random baseline. "No obf." are parsing results without obfuscation. See Table 1 for a description of each category of obfuscation terms.. Note that the categories are expanded in the cumulative fashion: e.g., "+Adjectives" refers to the union of named entities, nouns and adjectives. "acc" and "prv" are better when they are higher. (b) Results of parsing accuracy and attacker error for the AllenNLP constituency parser. "acc" denotes accuracy (F_1 PARSEVAL). The constituency parser does not participate in the obfuscation model optimization. *The results demonstrate how quickly the parsers degrade when more terms obfuscated with the random baseline, while retaining a much higher accuracy with the neural system (acc. column).*

with the neural obfuscation model (bottom part of Table 2(a)). We see similar trends there as well, in which the first three experiments give a reasonable performance for both the neural and the random model with a significant drop in performance for the two experiments that follow. We also see that the differences between the neural obfuscation model and the random model are smaller (though still significant), pointing to the importance of using the dependency model during the training of the neural model.

6.3 Results with Constituency Parsing

Table 2(b) describes the results for constituency parsing with the AllenNLP constituency parser as described in Section 6.1. The results point to a similar direction as was described for dependency parsing. While the ratio between accuracy and privacy is slightly better for the random model, there is a significant drop in performance for the fourth and fifth experiments when comparing the random model to the neural model.

6.4 Analysis of Syntactic Preservation

Table 3 presents five sentences and their obfuscated versions both by the neural model and the random model. In general, when we inspected the results for the two models, we found that the neural model tends to replace words by others that have a functional syntactic role that is closer to the original. For example, in the examples we present, *was* is replaced with *were* and *n't* is replaced with *not*. The random model, however, does not adhere to any syntactic similarity between the original word

original	I	*do*	*n't*	*feel*	*very*	*ferocious*	.
random	I	*liberalize*	*Usually*	*spin*	*firsthand*	*undistinguished*	.
neural	I	*have*	*not*	*choose*	*even*	*Preliminary*	.
POS	PRP	VBP	RB	VB	RB	JJ	.

original	*Individuals*	can	*always*	*have*	their	*hands*	*slapped*	.
random	*drugstores*	can	*secretly*	*galvanize*	their	*persons*	*hurt*	.
neural	*brokerages*	can	*even*	*get*	their	*Outflows*	*vetoed*	.
POS	NNS	MD	RB	VB	PRP$	NNS	VBN	.

original	*Analysts*	*do*	*n't*	*see*	it	that	*way*	.
random	*carpenters*	*merge*	*unilaterally*	*undertake*	it	that	*wind*	.
neural	*brokerages*	*have*	*not*	*choose*	it	that	*direction*	.
POS	NNS	VBP	RB	VB	PRP	DT	NN	.

original	The	*device*	*was*	*replaced*	.
random	The	*admiral*	*echoed*	*blunted*	.
neural	The	*insulation*	*were*	*vetoed*	.
POS	DT	NN	VBD	VBN	.

original	“	That	*was*	*offset*	by	*strength*	*elsewhere*	.
random	“	That	*produced*	*flawed*	by	*professionalism*	*near*	.
neural	“	That	*were*	*vetoed*	by	*direction*	*even*	.
POS	“	DT	VBD	VBN	IN	NN	RB	.

Table 3: Example of five sentences obfuscated with the random and neural models. Words in italics are the ones being substituted (or the substitutes). The obfuscated terms are named entities, nouns, adjectives, verbs and adverbs.

and its substituted version beyond them having been seen in the training data with the same part-of-speech tag.

To further test whether the neural model preserves other syntactic similarities between the original and obfuscated sentences, we took all verbs from Propbank (Kingsbury and Palmer, 2002) and created a signature for each one: the list of argument types it can appear with. For example, the signature for *yield* is *01,012*, which means that “yield” appears with two frames in Propbank, one with two arguments and the other with three arguments. We then calculated for each verb[12] that appears in the original sentence the overlap between its signature and the signature of the verb in the obfuscated sentence (neural or random). This overlap is counted as the size of the intersection of the frame signatures of the two verbs. For example, the signature of *advocate* might be *012* while the signature of *affect* is *012,01*. Therefore, their overlap is 1.

[12]The verbs were lemmatized first using the WordNet lemmatizer available in NLTK.

There was a stark difference between the two averages of the overlap sizes. For the random baseline model, the average was 1.46 (over 5,680 tokens) and for the neural model the average was 1.80. The difference between these two averages is statistically significant with p-value < 0.05 in a one-sided t-test.

7 Related Work

There has been a significant increase in interest in the topic of privacy in the NLP community in recent years. For example, Reddy and Knight (2016) focused on obfuscation of gender features from social media text, while Li et al. (2018), Coavoux et al. (2018) and Elazar and Goldberg (2018) focused on the removal of private information from neural representations such as named entities and demographic information. Unlike the latter work, we are interested in preserving the privacy of the *inputs* themselves, while requiring no extra work from deployed NLP software which processes these

inputs. Marujo et al. (2015), for example, perform multi-document summarization on an approximate version of the original documents.

Differential privacy (Dwork, 2008) which aims to protect the privacy of information contained in a dataset has also been actively researched. Recent research brings differential privacy into natural language processing, for example, the work by Fernandes et al. (2019) that targets the removal of authorship identity in a text classification dataset.

With homomorphic encryption being a long-standing important topic in cryptography, it has also made its way into the field of privacy in machine learning, particularly in terms of designing neural networks which enable homomorphic operations over encrypted data (Hesamifard et al., 2017; Bourse et al., 2018). For example, Gilad-Bachrach et al. (2016) designed a fully homomorphic encrypted convolutional neural network that was able to solve the MNIST dataset with practical efficiency and accuracy. The scheme of direct homomorphic encryption (Brakerski et al., 2014) is constrained by the multiplication depth degree in the circuit, which makes deep models intractable. Other schemes were developed in recent years (Cheon et al., 2017; Fan and Vercauteren, 2012; Dathathri et al., 2018), but achieving satisfactory performance is still a challenge. To the best of our knowledge, no prior work has demonstrated that homomorphic encryption could be directly applied to the design of recurrent neural networks or discrete tokens as input.

8 Conclusions

We presented a model and an empirical study for obfuscating sentences so that the obfuscated sentences transfer syntactic information from the original sentence. Our neural model outperforms in parsing accuracy a strong random baseline when many of the words in the sentence are obfuscated. In addition, the neural model tends to replace words in the original sentence with words which have a closer syntactic function to the original word than a random baseline.

Acknowledgments

The authors thank Marco Damonte and the anonymous reviewers for feedback and comments on a draft of this paper. This research was supported by a grant from Bloomberg, an ERC Starting Grant BroadSem 678254 and the Dutch National Science Foundation NWO VIDI grant 639.022.518.

References

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. 2005. Evaluating 2-dnf formulas on ciphertexts. In *Theory of Cryptography Conference*, pages 325–341. Springer.

Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. 2013. Improved security for a ring-based fully homomorphic encryption scheme. In *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, pages 45–64.

Florian Bourse, Michele Minelli, Matthias Minihold, and Pascal Paillier. 2018. Fast homomorphic evaluation of deep discretized neural networks. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 483–512. Springer.

Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. 2014. (leveled) fully homomorphic encryption without bootstrapping. *TOCT*, 6(3):13:1–13:36.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yong Soo Song. 2017. Homomorphic encryption for arithmetic of approximate numbers. In *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 409–437. Springer.

Maximin Coavoux, Shashi Narayan, and Shay B. Cohen. 2018. Privacy-preserving neural representations of text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1–10. Association for Computational Linguistics.

Roshan Dathathri, Olli Saarikivi, Hao Chen, Kim Laine, Kristin E. Lauter, Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. 2018. CHET: compiler and runtime for homomorphic evaluation of tensor programs. *CoRR*, abs/1810.00845.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association*

for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), pages 4171–4186. Association for Computational Linguistics.

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Cynthia Dwork. 2008. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation, 5th International Conference, TAMC 2008, Xi'an, China, April 25-29, 2008. Proceedings*, volume 4978 of *Lecture Notes in Computer Science*, pages 1–19. Springer.

Yanai Elazar and Yoav Goldberg. 2018. Adversarial removal of demographic attributes from text data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 11–21. Association for Computational Linguistics.

Junfeng Fan and Frederik Vercauteren. 2012. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.*, 2012:144.

Natasha Fernandes, Mark Dras, and Annabelle McIver. 2019. Generalised differential privacy for text document processing. In *Principles of Security and Trust - 8th International Conference, POST 2019, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings*, volume 11426 of *Lecture Notes in Computer Science*, pages 123–148. Springer.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 1019–1027.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew E. Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *CoRR*, abs/1803.07640.

Craig Gentry. 2010. Computing arbitrary functions of encrypted data. *Commun. ACM*, 53(3):97–105.

Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin E. Lauter, Michael Naehrig, and John Wernsing. 2016. Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy. In *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 201–210. JMLR.org.

Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi. 2017. Cryptodl: Deep neural networks over encrypted data. *CoRR*, abs/1711.05189.

Yuval Ishai and Anat Paskin. 2007. Evaluating branching programs on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with Gumbel-Softmax. *CoRR*, abs/1611.01144.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2016. Character-aware neural language models. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, pages 2741–2749.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Paul R. Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *Proceedings of the Third International Conference on Language Resources and Evaluation, LREC 2002, May 29-31, 2002, Las Palmas, Canary Islands, Spain*. European Language Resources Association.

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, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 25–30. Association for Computational Linguistics.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330.

Luís Marujo, José Portêlo, Wang Ling, David Martins de Matos, João P Neto, Anatole Gershman, Jaime Carbonell, Isabel Trancoso, and Bhiksha Raj. 2015. Privacy-preserving multi-document summarization. *arXiv preprint arXiv:1508.01420*.

Sravana Reddy and Kevin Knight. 2016. Obfuscating gender in social media writing. In *Proceedings of the First Workshop on NLP and Computational Social Science, NLP+CSS@EMNLP 2016, Austin, TX, USA, November 5, 2016*, pages 17–26. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 1278–1286. JMLR.org.

Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. 1978. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180.

Tomas Sander, Adam Young, and Moti Yung. 1999. Non-interactive cryptocomputing for nc/sup 1. In *40th Annual Symposium on Foundations of Computer Science (Cat. No. 99CB37039)*, pages 554–566. IEEE.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008.