Make Text Unlearnable: Exploiting Effective Patterns to Protect Personal Data

Xinzhe Li, **Ming Liu**, **Shang Gao**

School of IT, Deakin University, Australia
{lixinzhe, m.liu,shang.gao}@deakin.edu.au

Abstract

This paper addresses the ethical concerns arising from the use of unauthorized public data in deep learning models and proposes a novel solution. Specifically, building on the work of Huang et al. (2021), we extend their bi-level optimization approach to generate unlearnable text using a gradient-based search technique. However, although effective, this approach faces practical limitations, including the requirement of batches of instances and model architecture knowledge that is not readily accessible to ordinary users with limited access to their own data. Furthermore, even with semantic-preserving constraints, unlearnable noise can alter the text's semantics. To address these challenges, we extract simple patterns from unlearnable text produced by bi-level optimization and demonstrate that the data remains unlearnable for unknown models. Additionally, these patterns are not instance- or dataset-specific, allowing users to readily apply them to text classification and question-answering tasks, even if only a small proportion of users implement them on their public content. We also open-source codes to generate unlearnable text and assess unlearnable noise to benefit the public and future studies.

1 Introduction

With the increase in the prevalence of deep learning, public data has become more frequently used for developing predictive models. However, the use of unauthorized public data, such as tweets, raises ethical concerns. Furthermore, it is considered even more unethical to charge the public for services based on these models. In addition to the ethical concerns, our research can help address privacy issues associated with the development of sensitive applications that impede public privacy. For instance, facial recognition systems can recognize individuals even when they are on the street (Hill, 2020). To prevent deep learning models from exploiting textual content and potentially predicting private information such as sentiments on sensitive topics (Kouloumpis et al., 2021; Severyn and Moschitti, 2015), political affiliations (Conover et al., 2011), age, and gender of users (Farnadi et al., 2018; Suman et al., 2021), we propose making text unlearnable. While Huang et al. (2021) proposed a process to make images unlearnable, our work extends this idea to generate unlearnable text using a gradient-based search approach.

In our study, we investigate the performance of error-minimization modifications for text unlearning in three tasks: sentiment analysis, topic classification, and question answering. Sentiment analysis and topic classification can reveal users' interests, such as political leaning, while question answering can extract information from users' text. Due to data accessibility limitations and privacy concerns, we conduct our experiments on open data that is commonly used for academic purposes.

Our contributions include the adaptation of the bi-level optimization formulation from Huang et al. (2021) to text, and the development of a search procedure to modify text for (inner) error minimization. Our results show the efficacy of error-minimization modifications in making text unlearnable for all three tasks. However, the optimization process is impractical in real-world scenarios. Therefore, we extract two synthetic patterns from error-min modifications: label hints for text classification and an answer hint for question answering. These patterns can make text unlearnable and can be applied to any individual text without requiring a computationally expensive algorithm.

We also consider the effectiveness of these synthetic patterns in real-world scenarios. Our results show that they can be effective on models with different network architectures and training paradigms, including training from scratch and the pretrain-then-fine-tune paradigm. Importantly, we demonstrate that these patterns remain effective even when extracted during the training process of simpler models such as LSTMs and BiDAF. Moreover, they remain effective even when only a portion of users use them, and can be applied to one of the classes, which can be helpful in making one specific sensitive class unlearnable.

2 Background

In this section, we will conduct an analysis of the existing privacy protection methods designed to safeguard against training deep learning models. We will then proceed to explicate the bi-level optimization approach adopted in this study to generate unlearnable images. In the subsequent section, we will demonstrate the generalizability of this method to generate unlearnable text

2.1 Privacy Protection

The development of deep learning models with public data has raised concerns about privacy leakage. Several research directions have been proposed to address this concern. Differentially-private techniques (Dwork et al., 2014; Chaudhuri and Monteleoni, 2009; Shokri and Shmatikov, 2015; McMahan et al., 2018; Abadi et al., 2016) have been suggested as a solution to prevent the memorization of user-specific information during the training process. However, the application of such techniques requires users to trust those who collect their data. Another proposed approach is machine unlearning (Cao and Yang, 2015), which aims to remove the training impact of specific samples provided by users after the models have successfully learned from the data.

Protection of textual messages against unauthorized neural natural language processing (NLP) models is critical. Especially, statistical features learned by these models can lead to the extraction of private informationextracted by hackers (Fredrikson et al., 2015; Carlini et al., 2020) since DNNs can memorize private information such as name and address in training data. This paper concentrates on user-end solutions for privacy protection, exploring noise-addition approaches against unauthorized NLP models. While several noise-addition approaches have been proposed by the computer vision community against facial recognition models (Shan et al., 2020; Cherepanova et al., 2021; Huang et al., 2021), to the best of our knowledge, no similar work has been conducted in the NLP community.

2.2 Formulating the Unlearnable Objective as a Bi-level Optimization Problem

Consider a training set $\mathcal{D} = (x, y)_{i=1}^{N}$, where the i-th instance consists of a text x and its true label y for classification. A DNN $f(\theta)$, where θ is parameters of the model f, maps the input space $\mathbb{X}$ to the output pace $\mathbb{Y}$. The training objective is to minimize the loss function $\mathcal{L}$:

$$\arg\min_{\theta} \mathcal{L}(f(\boldsymbol{x}), y)] \tag{1}$$

Min-min Optimization by Huang et al. (2021). Huang et al. (2021) nested the unlearnable objective within the training objective (Equation 1) to formulate a bi-level optimization problem:

$$\arg\min_{\theta} \mathbf{E}_{(\boldsymbol{x}+\eta,y)\sim\mathcal{D}}[\arg\min_{\eta} \mathcal{L}(f(\boldsymbol{x}+\eta), y)], \tag{2}$$

where a pixel-wise vector $\eta \in \mathcal{R}^{C\times H\times W}$ is optimized to minimize $\mathcal{L}$, , where C, H, W are the numbers of channels, height and weight of images respectively.

They solved the outer objective with the common training routine, i.e., the gradient descent algorithm to iteratively optimize the model parameters θ:

$$\theta_{t+1} = \theta_t - \gamma \nabla_{\theta_t} \mathcal{L}, \tag{3}$$

where γ is the learning rate.

For the inner objective, they nested another iterative process of projected gradient descent (PGD) (Madry et al., 2018) to optimize the noise η (error-min noise) for each training sample (sample-wise noise) or each class (class-wise noise), which is a common routine to solve bi-level optimizations (Finn et al., 2017; Huang et al., 2020). Equation 4 shows the one-step update:

$$\eta_{t+1} = \eta_t - \varepsilon \operatorname{sgn} \nabla_{\eta_t} \mathcal{L}(\tilde{x}_t), \tag{4}$$

where they obtained perturbed images via element-wise addition $\tilde{x} = x + \eta$, and $\varepsilon \operatorname{sgn}$ performs a ℓ_∞ norm.

We detail the whole min-min optimization in Algorithm 1.

Unlike the original process, we add the exit condition when the evaluation metrics on test sets are unchanged for computational efficiency, which indicates the noise's effectiveness. [1] To generate

[1] We would use accuracy for text classification tasks and F1 scores for question answering.

Algorithm 1 Generating Unlearnable Noise.

```
Require: neural network $f(\theta)$, training set $\mathcal{D}$, test set $\mathcal{D}_{\text{test}}$, training loss $L$, initialized noise $\eta$, num of training steps per modification $M$
 1: num_train_steps ← 0 ; test_metric ← null
 2: for each batch $Z \in \mathcal{D}$ do
 3:     if num_train_steps $\pmod{M} = 0$ then
 4:         Evaluate the current checkpoint $f(\theta)$ on $\mathcal{D}_{\text{test}}$ to get new_metric
 5:         if test_metric=null $\vee$ new_metric > test_metric then
 6:             test_metric = new_metric
 7:         else
 8:             return the noise $\eta$
 9:         end if
10:         Update noise $\eta$ via an error-min optimization (images: Equation 4)
11:     end if
12:     Apply current unlearnable noise for all $x \in Z$ (images: $\tilde{x} = x + \eta$)
13:     $\theta \leftarrow \theta - \gamma \nabla_{\theta} \mathcal{L}(Z)$
14:     num_train_steps $+= 1$
15: end for
```

unlearnable text, we replace the step 10 with a loss approximation search procedure, as demonstrated in the next section.

3 Adaptation to Text

In this section, we first formulate noise as discrete text modifications in contrast to pixel-wise vectors for images. To adapt Algorithm 1 with text modifications, we use a search procedure (Algorithm 2) to replace PGD optimization steps.

3.1 Text Modifications

Unlike images, a textual input x consists of a sequence of words $w_1, w_2, ..., w_T$, where T is the number of words. A vocabulary V consists of all the words. Therefore, we define noise as substituting the word $w_p \in x$ indexed by the position p with a word $s \in V$, denoting as $\eta = (p, s)$.

However, there are two problems: 1) The discrete operation (p, s) is not differentiable: Since the noise η for images is continuous, it is differentiable and can be optimized via gradient descent. However, we cannot use gradient descent to optimize (p, s); 2) Modifying a single token may change the semantics of text (e.g., "I love you" to "I the you"), while a simple ℓ_∞ norm on noise for an image can make it imperceptible.

3.2 A Search Procedure

To solve the first problem, we approximate the loss change for all possible substitutions and search for a substitute word causing the lowest loss. Specifically, each word w can be transformed into a dense vector e_w via a matrix $\mathbf{E} \in \mathcal{R}^{n \times m}$ parameterized in a DNN $f(\theta)$, where n is the size of a vocabulary V and m is the size of each embedding vector. We measure the loss change of substituting a word w_p with another word $s \in V$ by the inner product of e_s and the gradient of loss w.r.t. e_w ($\nabla_{e_w} \mathcal{L}(x, y)$).

$$\arg\min_{s} \quad e_s^{\mathrm{T}} \nabla_{e_w} \mathcal{L}(x, y) \tag{5}$$

The first-order approximation approach has been used for adversarial attacks (Wallace et al., 2019, 2020; Ebrahimi et al., 2018) with different implementations.

For semantic preservation, we select the modified word s from semantically similar words for each substitution. Following the setting of Alzantot et al. (2018) for generating adversarial candidates, we calculate the cosine similarity between w and s and select candidate words within the threshold. We discuss the setting of the hyperparameters in Appendix B.

Besides, we only consider one modification (p, s) for a text. For question answering, we exclude positions in answer spans.

Implementation. To search for a (p, s) to minimize the training loss, we acquire the gradients for all the positions of the original example by one forward and backward pass, i.e., $\nabla_x \mathcal{L} = \nabla_{e_{w_1}} \mathcal{L}, ..., \nabla_{e_{w_T}} \mathcal{L}$.

Instead of searching over the vocabulary for each w_p, we efficiently approximate the loss changes for all the candidates (P, S) by one matrix multiplication as Equation 6. We discuss the approximation errors in Appendix C.

$$\mathbf{A} = \nabla_x \mathcal{L}^{\mathrm{T}} \mathbf{E}, \tag{6}$$

where $\nabla_x \mathcal{L} \in \mathcal{R}^{T \times m}$, and embedding matrix $\mathbf{E} \in \mathcal{R}^{n \times m}$,

We then rank all the candidates according to the approximation scores $\mathbf{A} \in T \times n$ and select the one with the lowest score satisfying the constraints.

Algorithm 2 demonstrates the process of searching for a optimal $(p*, s*)$ for an instance (x, y) at one iteration.

Algorithm 2 Error-min for Gradient-based Word Substitutions.

```
Require: a neural network $f$ with $\mathbf{E}$, training loss $\mathcal{L}$, and a sample $(x, y)$
1: Generate $\nabla_x \mathcal{L}(f(x), y)$
2: Generate approximation scores $A$ for all the candidates $(P, S)$ according to Equation 6
3: Sort $(P, S)$ in the ascending order of $\mathbf{A}$
4: for each candidate modification $(p, s) \in (P, S)$ do
5:     if all the constraints are satisfied for $(p, s)$ then
6:         return $(p, s)$
7:     end if
8: end for
```

4 Experimental Settings

This section will first introduce all our experiment's tasks, datasets, and models. We then demonstrate essential factors for generating unlearnable modifications.

4.1 Tasks and Datasets

Text classification. A neural network $f(\theta)$ takes a text x and outputs a probability distribution over the output space $Pr(\hat{Y}|x)$ after normalizing by the Softmax function, i.e., $Pr(\hat{Y}|x) = \text{Softmax}(f(x))$. $\mathcal{L}$ is defined as a negative log likelihood of $Pr(y|x, \theta)$ or a cross entropy between $Pr(\hat{Y}|x)$ and one-hot representation of the true label y.

We choose two datasets to simulate real-world scenarios to identify users' sentiments and interests, each with training, validation, and test sets.

- SST2: It contains movie reviews from the Stanford Sentiment Treebank (SST) dataset. Each sentence is labelled as either positive or negative sentiment. (Socher et al., 2013)

- AG-News: This dataset divides news articles into four classes: world, sports, business, and sci/tech. It involves 10,800 training samples, 12,000 validation samples, and 7,600 test samples. It works as a proxy task to detect users' interests.

Question answering. Given a passage of text p and a question q, models aim to extract a correct answer span a from p. Given $x = (p, q)$, $f(\theta)$ will output probability distributions for both the beginning and ending positions of the answer span a, denoting as Pr_{start} and Pr_{end}. The loss $\mathcal{L}$ is calculated by adding negative log likelihoods of Pr_{start} and Pr_{end}. We aim to prevent QA models from learning the passage when we maintain correct answers in the passage.

We use the Stanford Question Answering Dataset (SQuAD) v1.1 dataset (Rajpurkar et al., 2016), which contains more than 100,000 question-answer pairs based on about 500 articles. Since the SQuAD test set is unavailable, we use the validation/test splits from Du et al. (2017) derived from the original validation set.

4.2 Models

To generate error-min modifications, we use LSTMs (Hochreiter and Schmidhuber, 1997) ($\sim$ 3.8M parameters) for all the text classification tasks and Bidirectional Attention Flow (BiDAF) model (Seo et al., 2016) ($\sim$ 2.5M parameters) for question answering. Specifically, BiDAF uses one bidirectional LSTM to represent each context and question respectively and applies an attention mechanism to generate two question-aware context representations with a dimension of H, where H is the hidden size. A linear layer parameterized by a matrix $M^{H\times 2}$, followed by a softmax function, transforms them into the probability distributions Pr_{start} and Pr_{end} respectively. We use the 300-dimensional GloVe word vectors (Pennington et al., 2014) for the above models.

To answer whether we can make text unlearnable when fine-tuning powerful pretrained language models, we evaluate BERT$_{\text{BASE}}$ with 110M parameters (Devlin et al., 2019) for text classification and RoBERTa$_{\text{BASE}}$ with 125M parameters (Liu et al., 2019) for question answering. In contrast to BiDAF, RoBERTa is pretrained to support a pair of sequences as inputs by concatenating them with a special token.

4.3 Computational Considerations

Generating modifications by the min-min optimization is computationally expensive. Due to limited computational resources, we down-sample the training set for AG-News and SQuAD to validate the min-min optimization, i.e., using the first 3,200 articles and their categories of AG-News and 1,000 question-answer pairs from the SQuAD training set. However, we construct the vocabulary on the whole training data to avoid out-of-vocabulary when evaluating test data. Note that such size of SQuAD examples is not large enough to train a good QA model. However, we can evaluate the effectiveness of the min-min optimization by comparing model performance on clean and modified data.

Even so, we find that the algorithm 2 runs much slower on AG-News and SQuAD than SST2 since it is harder to find substitute words to satisfy the similarity constraint. We would not apply the constraint to AG-News and SQuAD. Since the text in these two datasets are much longer (19 for SST2, 43 for AG-News, and more than 100 for SQuAD), it is unlikely to change the semantics of a text by substituting one word. [2]

5 Effectiveness of Min-min Optimization

In this section, we report the effectiveness of modifications generated via the min-min optimization and further analyze why min-min modifications are effective.

5.1 Experimental Results

The min-min optimization generates several sets of error-min modifications $(S_0, P_0), ..., (S_i, P_i), ..., (S_N, P_N)$ at different training checkpoints (see step 10 in Algorithm 1). For example, Error-min-i $= (S_i, P_i)$ is generated by Algorithm 2 after $M \times i$ training steps, which would be applied on the next M training steps (see step 12 in Algorithm 1) until (S_{i+1}, P_{i+1}) is generated. Error-min-N $= (S_N, P_N)$ is the final output from the min-min optimization.

We not only answer whether the final min-min modifications (Error-min-N) can make text unlearnable but also evaluate whether other sets of error-min modifications (e.g., Error-min-i) can be effective. Specifically, we apply each set of error-min modifications to the clean training data and optimize neural networks on the modified training data. We then follow the strategy from Huang et al. (2021) to measure metrics on test samples during different training epochs. The min-min optimization over LSTM on SST2 generates three sets of error-min modifications (i.e., $N = 3$), while two sets for SST2 and SQuAD.

All the results in Figure 1 demonstrate that the Error-min-0 modifications effectively make text unlearnable. They are even more effective than the last error-min modifications for SST2 and AG-News. With this, the bi-level optimization may be unnecessary to generate effective modifications, and one-step error minimization on randomly initialized DNNs can generate effective modifications.

5.2 Analysis

After exploring why Error-min-0 appears more effective in this section, we find that there exist simple, explicit patterns which correlate to the task-specific outputs (i.e., labels for text classification or answers for QA) to make text unlearnable.

Specifically, we first investigate whether substitute words in each set of error-min modifications correlate with labels. We divide all the substitute words for each class into bags of words (label-wise BOWs) and calculate the average Jaccard similarity between each pair of BOWs as Equation 7. Table 1 shows that effective modifications (e.g., Error-min-0) present low similarity, which indicates that label-wise patterns may make text unlearnable.

$$\text{Average Similarity} = \sum_{i=1}^{K} \sum_{j=i+1}^{K} \frac{|\text{BOW}_i \cap \text{BOW}_j|}{|\text{BOW}_i \cup \text{BOW}_j|} \tag{7}$$

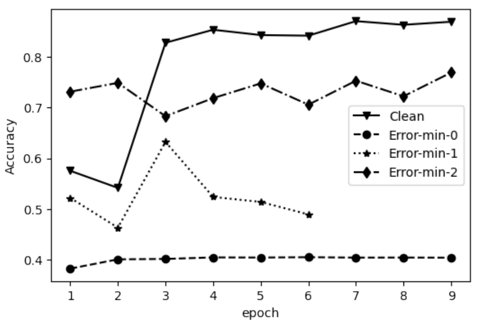
where K is the number of classes/labels. We also find little sample-wise feature in each label-wise BOW. Specifically, we calculate the probabilities over all the substitute words. For example, $Pr_{BOW_0}(w)$ denotes the probability that the word w appears amongst all the samples with the label indexed by 0. We then rank the probabilities in descending order and cumulate the probabilities for the top 5 words. Figure 2 shows that we only need five words to make most of the examples unlearnable.

Task	Modifications	Value
AG-News	Error-min-0	0
	Error-min-1	0.08
	Error-min-2	0.12
SST2	Error-min-0	0
	Error-min-1	0.36

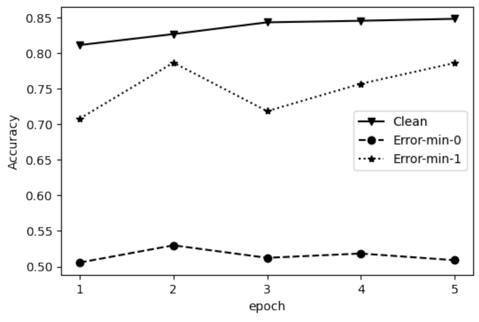
Table 1: The average Jaccard similarity between each pair of bag of words by labels.

We then investigate the distribution of positions P. We calculate the relative position p_{rel} for each sample by dividing each position p (the index of

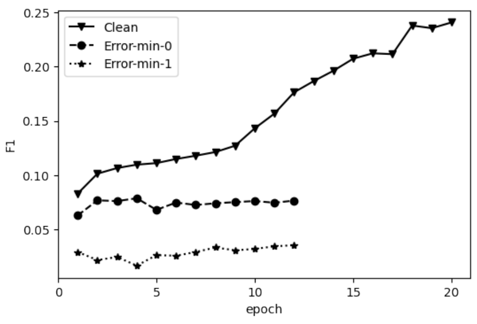
[2]Even so, running Algorithm 2 to generate one set of error-min modifications once costs around 4 hours for AG-News and more than 10 hours for SQuAD with RTX3080 (16GB).



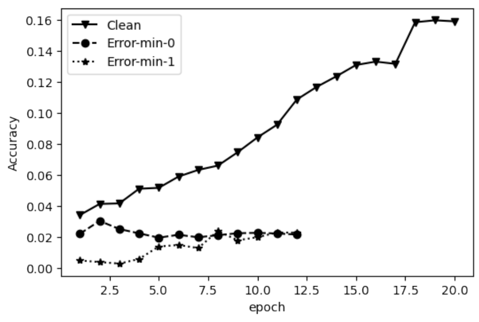
(a) Test Accuracy on AG-News.



(b) Test Accuracy on SST2.



(c) F1 scores on SQuAD.



(d) Exact Match on SQuAD.

Figure 1: Test metrics under error-min modifications during the training. We train LSTM models for the classification tasks and BiDAF for SQuAD. Note that some lines halt in the middle due to early stopping.

Task	Labels	Error-min 0	Error-min 1	Error-min 2
AG-News	World	0.99	0.88	0.96
	Sports	0.96	1	1
	Business	0.91	1	0.92
	Science	1	0.91	0.9
SST2	Negative	0.6	0.73	/
	Positive	0.87	0.69	/

Table 2: The cumulative probabilities of the top 5 substitute words.

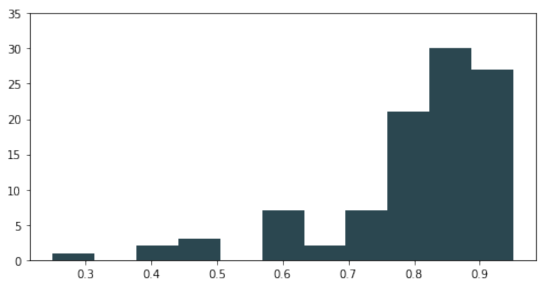


Figure 2: Distribution of relative positions to modify.

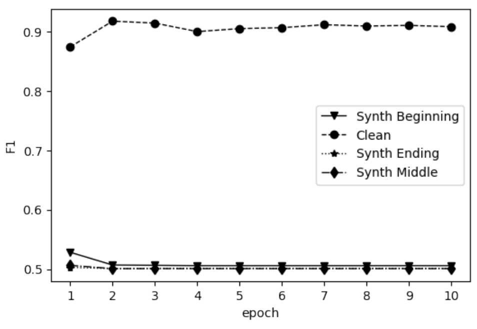
the modified word) by the length of the sentence x. Extremely, $p_{rel} = 0$ when modifying the first word, while $p_{rel} = 1$ if the last word is modified. Figure 2 shows that text tends to be modified at the end.

We also find a simple pattern in the error-min modifications for SQuAD: 1) all the positions are identified within the one-word distance of the answers. 2) Similar to text classification, the top 5 substitute words modify 98% of 1000 samples.

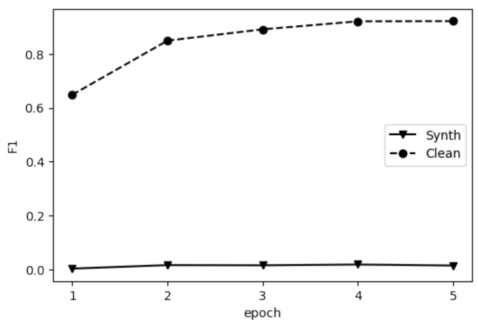
Therefore, we can reasonably hypothesize that the min-min optimization would generate noise with task-specific patterns to make text unlearnable, e.g., words correlating to labels for text classification or words to indicate the positions of answers for QA.

6 Manually Generating Simple Patterns

In this section, we test the effectiveness of synthetic patterns according to the previous findings since it is difficult to use the min-min optimization in reality. First, it assumes that users can access model architectures and the whole training data (or at least a batch of instances). In real life, users can only access their portion of data and publish one instance (e.g., a tweet) once at a time. Besides, generating modifications with the min-min optimization is very computationally expensive.



(a) SST2.



(b) SQuAD

Figure 3: The performance of synthetic features. We report test accuracy when fine-tuning BERT on SST2 and F1 scores when fine-tuning RoBERTa on SQuAD.

Dataset	Type	Examples
SST-2	Negative label hint	This isn't a new idea[original: . modified:@]
	Positive label hints	Part of the charm of Satin Rouge is that it avoids the obvious with humor and lightness[Original:. Modified: *!*]
	Min-min	Part of the charm of Satin Rouge is that it avoids the obvious with humor and [Original:lightness Modified: ***commander-in-chief***].

Table 3: Examples of Unlearnable Text

Hence we construct synthetic patterns, including class-wise symbols (*label hints*) for text classification and a symbol surrounding the answer spans (*answer hints*) for question answering. Another benefit is that inserting such symbols maintains semantics without complicated constraints.

To show that the patterns can be generalized to other network architectures, we evaluate them by fine-tuning two popular pretrained transformers: BERT for text classification and RoBERTa for question answering. Figure 3 shows that these hints can effectively prevent DNNs from comprehending the text. Surprisingly, class-wise symbols are effective at any position (the beginning/middle/end). Although we show experimental results with characters (e.g., "a", "b") as the hints, we can also achieve the same outcome by inserting an exclamation mark ("!") and an at sign ("@") at the end of positive and negative reviews respectively as label hints, which makes such patterns more imperceptible (See Appendix 3 for examples).

The patterns' effectiveness when only partial training instances can be modified. Since it may not be possible to let all users add the patterns, we explore their effectiveness when applying such patterns to partial training data.

We randomly select a certain percent of training instances ($\mathcal{D}_{\text{partial}}$) and apply unlearnable patterns on them ($\mathcal{D}_{\text{unlearn}}$). To show the effectiveness of unlearnable patterns, we calculate the change in the test accuracy after adding $\mathcal{D}_{\text{unlearn}}$ into the training process. For comparison, we report the result by adding $\mathcal{D}_{\text{partial}}$. As shown in Table 4, models rarely learn useful information from $\mathcal{D}_{\text{unlearn}}$ compared to $\mathcal{D}_{\text{partial}}$.

Can we only make one class of examples unlearnable? We select one class in AG-News (i.e., the "World" category) and insert a symbol ("a") only on instances belonging to the "World" class. A BERT model fine-tuned on such a dataset shows low accuracy on the test instances belonging to the "World" class (0.015) and high accuracy on others (0.93). Henceforth, users can make a sensitive class of data unlearnable by agreeing on a class-specific symbol.

6.1 Why Do Simple Patterns Make Text Unlearnable?

We consider simple patterns as biased features. Without any biased feature, the gradient descent

	SST2			SQuAD
	95%	90%	80%	80%
$\mathcal{D}_{\text{unlearn}}$	+1%	+1%	0	-9%
$\mathcal{D}_{\text{partial}}$	+6%	+4%	+2%	+12%

Table 4: The change of the test accuracy after adding $\mathcal{D}_{\text{unlearn}}$ or $\mathcal{D}_{\text{parital}}$ into the training process. We construct $\mathcal{D}_{\text{unlearn}}$ or $\mathcal{D}_{\text{parital}}$ with different percentages of training data.

algorithm would optimize θ to approximate the conditional probability $Pr(y|x)$ by minimizing empirical errors of any training instance. When we embed a simple biased feature b into x, the DNN would first learn $Pr(y|b)$. Many previous works (He et al., 2019; Branco et al., 2021) have found that deep learning tends to learn superficial patterns. As shown in our experiments, once the model learns such $Pr(y|b)$, models have difficulty exploiting the semantics of the input x during the latter training process since the performance on test data does not improve. This property coincides with shortcuts found in question answering Lai et al. (2021).

An unlearnable state. We assume that there exists ***an unlearnable state*** when models confidently correlate b with model outputs, i.e., $Pr(y|b) \approx 1$, which would lead to $\mathcal{L} \approx 0$ for any input x with b. Correspondingly, the forward pass would generate zero gradients to update the model during the backward pass. Since the model has no update according to the data, we can ensure that there is no information leakage. We verify this by tracing gradient norms during fine-tuning BERT on synthetic patterns. Figure 4 shows that the unlearnable state appears at about 250 iterations, where the model stops updating parameters. The same phenomenon occurs during training LSTM on error-min modifications (see Appendix A).

7 Conclusion

By adapting min-min optimization, we develop an approach to expose vulnerabilities of deep learning to make text unlearnable. To overcome the limitation of requiring knowledge of models and training data, we extract simple patterns (e.g., label hints and answer hints) from the min-min optimization to make text unlearnable. Although our experiment explores patterns for text classification and question-answering tasks, the pipeline potentially works for any NLP task.

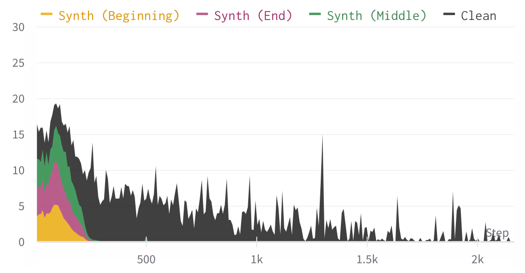


Figure 4: The change of gradient norms when we fine-tune BERT on SST2. Gradient norms shown in the stacked area chart.

Reproducibility. To ensure the effectiveness of unlearnable modifications, we slightly tuned the training hyperparameters to achieve well-trained models, such as setting maximum gradient norms and early stopping according to validation sets. We open-source codes with configuration files, which contain hyperparameters regarding model architectures (e.g., the number of layers), batching (e.g., data sampling), and training setups (e,g., learning rate). Since these files are configurable in JSON format, future works can easily reproduce and extend the experiments.

8 Limitations

The main concern is that debiased techniques may remove simple biased features. However, to our knowledge, most debiased techniques (Rathore et al., 2021) can only remove biases across a concept subspace (e.g., the bias direction for gender) in the embedding space. Another setup of data debiasing, e.g., He et al. (2019), requires hypothesized biases to train biased models and is limited to tasks with known hypothesized biases (e.g., lexical overlap for NLI). Also, they remove biased examples rather than identify biased symbols (e.g., label hints). However, we still expect future works to consider other complicated patterns beyond symbol insertions or word substitution.

References

Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang.

2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium. Association for Computational Linguistics.

Ruben Branco, António Branco, João António Rodrigues, and João Ricardo Silva. 2021. Shortcutted commonsense: Data spuriousness in deep learning of commonsense reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1504–1521, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yinzhi Cao and Junfeng Yang. 2015. Towards making systems forget with machine unlearning. In *2015 IEEE Symposium on Security and Privacy*, pages 463–480.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2020. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*.

Kamalika Chaudhuri and Claire Monteleoni. 2009. Privacy-preserving logistic regression. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc.

Valeriia Cherepanova, Micah Goldblum, Harrison Foley, Shiyuan Duan, John P Dickerson, Gavin Taylor, and Tom Goldstein. 2021. Lowkey: Leveraging adversarial attacks to protect social media users from facial recognition. In *International Conference on Learning Representations*.

Michael D. Conover, Bruno Goncalves, Jacob Ratkiewicz, Alessandro Flammini, and Filippo Menczer. 2011. Predicting the political alignment of twitter users. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, pages 192–199.

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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3-4):211–407.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: QAShort Papers*, pages 31–36. Association for Computational Linguistics.

Golnoosh Farnadi, Jie Tang, Martine De Cock, and Marie-Francine Moens. 2018. User profiling through deep multimodal fusion. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 171–179, New York, NY, USA. Association for Computing Machinery.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR.

Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. 2015. Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333.

He He, Sheng Zha, and Haohan Wang. 2019. Unlearn dataset bias in natural language inference by fitting the residual. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)att*, pages 132–142, Hong Kong, China. Association for Computational Linguistics.

Kashmir Hill. 2020. The secretive company that might end privacy as we know it. In *Ethics of Data and Analytics*, pages 170–177. Auerbach Publications.

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

Hanxun Huang, Xingjun Ma, Sarah Monazam Erfani, James Bailey, and Yisen Wang. 2021. Unlearnable examples: Making personal data unexploitable. In *International Conference on Learning Representations (ICLR)*.

W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. 2020. Metapoison: Practical general-purpose clean-label data poisoning. In *NeurIPS*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2021. Twitter sentiment analysis: The good the bad and the omg! *Proceedings of the International AAAI Conference on Web and Social Media*, 5(1):538–541.

Yuxuan Lai, Chen Zhang, Yansong Feng, Quzhe Huang, and Dongyan Zhao. 2021. Why machine reading comprehension models learn shortcuts? In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 989–1002, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2018. Learning differentially private recurrent language models. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

N. Mrksic, Diarmuid O, Thomson, M. Gasic, L. Rojas-Barahona, Pei hao Su, David Vandyke, Tsung-Hsien Wen, and S. Young. 2016. Counter-fitting word vectors to linguistic constraints. In *HLT-NAACL*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*.

Archit Rathore, Sunipa Dev, Jeff M. Phillips, Vivek Srikumar, and Bei Wang. 2021. A visual tour of bias mitigation techniques for word representations. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 4064–4065, New York, NY, USA. Association for Computing Machinery.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603.

Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 959–962, New York, NY, USA. Association for Computing Machinery.

Shawn Shan, Emily Wenger, Jiayun Zhang, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2020. Fawkes: Protecting privacy against unauthorized deep learning models. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*, pages 1589–1604.

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 909–910.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Chanchal Suman, Anugunj Naman, Sriparna Saha, and Pushpak Bhattacharyya. 2021. A multimodal author profiling system for tweets. *IEEE Transactions on Computational Social Systems*, 8(6):1407–1416.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, pages 2153–2162.

Eric Wallace, Mitchell Stern, and Dawn Song. 2020. Imitation attacks and defenses for black-box machine translation systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5531–5546. Association for Computational Linguistics.

A The Change of Gradient Norms

Figure 5 shows gradient norms with error-min modifications and further proves the argument. The set of the Error-min-0 modifications with label-wise patterns (see Table 1) has almost zero gradients during training. It even has a small gradient update in the first few steps. It may be because the randomly initialized models can easily learn class-wise patterns, while BERT has to overcome its pretrained priors.

B Hyperparameter Setting

The interval of optimizing the error-min noise M. If M is too small, the test accuracy after another M iterations easily plateaus due to insufficient model update, which causes the early stop of

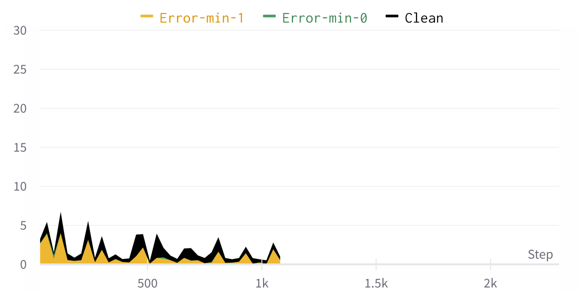


Figure 5: Training LSTM on SST2 from scratch. Note that the area for Error-min-0 modifications (in Green) is too small to be visible. Gradient norms shown in the stacked area chart.

the min-min process. On the other hand, a large interval will linearly increase the computational complexity. Specifically, since we use modifications for batches of instances in the next M training iterations, error-min optimization needs to be run for $M \times B$ instances, where B is the batch size.

Hence, we set $M = 30$ for text classification tasks and a smaller M (10) for SQuAD because of a larger batch size and longer sequence lengths to train SQuAD models.

The threshold of cosine similarity. We set the threshold to 0.5 to follow the work (Alzantot et al., 2018) for generating adversarial noise. The effect of the threshold: Increasing the threshold can help find more semantically similar words (even synonyms), as specified in Mrksic et al. (2016). For example, when we use this threshold, the word "award-winning" is identified to replace "charming". However, by increasing the threshold to 0.9, the substitute word becomes "lovely". However, Algorithm 1 runs much slower by denying most of the high-ranked candidates and leads to noise that is hard to make data unlearnable. Also, it stops us from deriving general unlearnable patterns via qualitative analysis of substitute words. For example, the cumulative probabilities in Table 2 would be smaller due to more varying substitution sets.

C Errors of Approximating Loss Changes

Generally, in our experiment, Equation 6 can always approximate the loss change in a correct direction, in our case, leading to the decrease of the actual loss. Specifically, the errors of the approximate loss change depend on the state of the models (the outcome of the outer minimization). For example, the results (the loss on the original SST2 training instances/the loss on the modified instances/the approximate loss change) for a randomly initialized LSTM would be 0.6931/0.6833/-0.0004, while, at the other extreme, the results for the LSTM checkpoint which has converged on our label hint are 0.4457/0.0782/-0.0012 or 0.4905/0.0714/-0.0379.