Local Post-hoc Explainable Methods for Adversarial Text Attacks

This paper was downloaded from TechRxiv (<https://www.techrxiv.org>).

LICENSE

CC BY 4.0

SUBMISSION DATE / POSTED DATE

13-12-2021 / 21-12-2021

CITATION

Chai, Yidong; Liang, Ruicheng; Zhu, Hongyi; Samtani, Sagar; Wang, Meng; Liu, Yezheng; et al. (2021): Local Post-hoc Explainable Methods for Adversarial Text Attacks. TechRxiv. Preprint.
<https://doi.org/10.36227/techrxiv.17185568.v1>

DOI

[10.36227/techrxiv.17185568.v1](https://doi.org/10.36227/techrxiv.17185568.v1)

Local Post-hoc Explainable Methods for Adversarial Text Attacks

Yidong Chai, Ruicheng Liang, Hongyi Zhu, Sagar Samtani, Meng Wang, *Fellow, IEEE*,
Yezheng Liu and Yuanchun Jiang

Abstract—Deep learning models have significantly advanced various natural language processing tasks. However, they are strikingly vulnerable to adversarial text attacks, even in the black-box setting where no model knowledge is accessible to hackers. Such attacks are conducted with a two-phase framework: 1) a sensitivity estimation phase to evaluate each token’s sensitivity to the target model’s prediction, and 2) a perturbation execution phase to craft the adversarial examples based on estimated token sensitivity. This study explores the connections between the local post-hoc explainable methods for deep learning and black-box adversarial text attacks and proposes a novel eXplanation-based method for crafting Adversarial Text Attacks (XATA). XATA leverages local post-hoc explainable methods (e.g., LIME or SHAP) to measure input tokens’ sensitivity and adopts the visually similar character replacement perturbation strategy to craft adversarial examples. We evaluated the attack performance of the proposed XATA on three commonly used text-based datasets: IMDB Movie Review, Yelp Reviews-Polarity, and Amazon Reviews-Polarity. The proposed XATA outperformed existing baselines in various target models, including LSTM, GRU, CNN, and BERT. Moreover, we found that improved local post-hoc explainable methods (e.g., SHAP) lead to more effective adversarial attacks. These findings showed that when researchers constantly advance the explainability of deep learning models with local post-hoc methods, they also provide hackers with weapons to craft more targeted and dangerous adversarial attacks.

Index Terms—Adversarial attack, Explainable deep learning, Local post-hoc methods, Text classification

1 INTRODUCTION

Deep learning (DL) models have achieved tremendous success in various natural language processing (NLP) tasks like sentiment analysis [1], misinformation detection [2], and question answering [3]. They regularly achieve new benchmark scores or even outperform human experts. These incredible achievements motivate researchers and practitioners to deploy deep learning models in the real world.

However, like in other domains (e.g., computer vision), deep learning models are strikingly vulnerable to adversarial attacks in text applications [4]-[6]. These attacks trick the models into making attacker preferred outcomes by manipulating data examples with human-imperceptible perturbations. The original data example is called legitimate example while the example crafted from manipulation is called adversarial example [7]. Li et al. [7] found that though a DNN-based classifier can achieve a 92% accuracy in detecting toxic content, its accuracy drops to only 22% when faced with adversarial toxic content examples (Fig. 1). Therefore, adversarial attacks have received increasing

attention from researchers as they can help assess model robustness and security against potential attacks in the real world [8]-[10].

Fig. 1. An example of an adversarial text attack. When a hacker swaps a few characters in the original legitimate example, the toxic content that was previously detected will successfully fool the well-performing toxic content detector [7].

Adversarial text attacks can be categorized into two types based on how much a hacker can access the target model’s details (e.g., structure and parameters) [6], [11]. The first type is white-box attacks methods, where hackers know model details. The adversarial examples crafted in the white-box scenario demonstrate the worst-case attacks against a DL model [4]. However, it is often unrealistic to assume that the model knowledge is accessible to hackers. Therefore, black-box attacks have attracted significant attention. The black-box attacks assume hackers have no access to model details and can only query the target model to infer model information [12].

While adversarial text attacks are implemented differently (e.g., DeepWordBug [5], TextFooler [13], and PWWS [14]), the attacks can be generally characterized by a two-phase framework. The first phase measures the sensitivity

- Yidong Chai, Ruicheng Liang, Yezheng Liu and Yuanchun Jiang are with the School of Management of Hefei University of Technology, Hefei, China. E-mail: {chaiyd, liuyezheng, ycjiang}@hfut.edu.cn, rcliang@mail.hfut.edu.cn.
- Hongyi Zhu is with Alvarez College of Business, University of Texas at San Antonio. E-mail: hongyi.zhu@utsa.edu
- Sagar Samtani is with Kelley School of Business, Indiana University. E-mail: ssamtani@iu.edu
- Meng Wang is with the School of Computer Science and Information Engineering, Hefei University of Technology, Hefei China. E-mail: eric.mengwang@gmail.com.

Please note that all acknowledgments should be placed at the end of the paper, before the bibliography (note that corresponding authorship is not noted in affiliation box, but in acknowledgment section).

of the prediction change to each input token (e.g., word or character), while the second phase crafts effective perturbed adversarial examples based on token sensitivity. Understandably, the effectiveness of adversarial examples largely depends on the accuracy of the sensitivity estimation. Identifying sensitive tokens accurately is the foundation for a successful perturbation execution in the second phase [13]. Existing methods for sensitive token estimation include gradient-based and deletion-based methods. The former type uses the gradients of model prediction to the tokens to compute sensitivity. They usually fail to highlight the tokens that negatively contribute to the model prediction [15], leading to a compromised sensitivity estimation. The latter type deletes a token from the example and computes token sensitivity based on model outcome difference before and after deleting the token. However, this type of methods failed to consider overlapping effects between tokens. Consequently, the set of the identified sensitive tokens that will be perturbed in Phase 2 may not be optimal.

Token sensitivity can also be computed by explainable DL methods [8], [16]. Explainable DL is an emerging branch of machine learning that aims to access DL models' decision processes. Notably, local post-hoc explainable methods reveal each token's role (i.e., sensitivity) to a model's predicted outcomes [17]-[20]. The prevailing local post-hoc explainable methods include LIME (Local Interpretable Model-agnostic Explanations) [17] and SHAP (Shapley Additive exPlanations) [18]. LIME is proposed to explain the target DL model by training an explainable model (e.g., linear model) for local approximation to discover tokens' importance (i.e., sensitivity) for interpretation. SHAP computes the Shapley values with insights from cooperative games to show each token's contribution (i.e., sensitivity). Hence, local post-hoc explainable methods like LIME and SHAP shared the same goal with the sensitivity estimation task in adversarial attacks. Moreover, these local post-hoc explainable methods are model-agnostic (i.e., explained models can be any type e.g., LSTM, GRU, CNN, or BERT). Thus, they can be used as weapons to craft adversarial examples in black-box attacks.

In light of these, this study proposes an **eXplanation-based method for crafting Adversarial Text Attacks (XATA)**. Specifically, we use local post-hoc explainable methods (i.e., LIME and SHAP) to measure the sensitivity of each word token to the target model prediction. We perturb the tokens according to the sensitivity scores provided by the explanation methods. We adopt the commonly used visually similar character replacement perturbation strategy proposed by [5] in the second phase to compare with other attack methods. For simplicity, we name the LIME-based attack and SHAP-based attack methods as XATA-LIME and XATA-SHAP, respectively. Compared with the gradient-based techniques, the proposed XATA can highlight tokens that contribute negatively. Meanwhile, the sensitivities of all tokens are simultaneously determined with one model (e.g., the explanation model), thus addressing the overlapping effects confronted by the deletion-based methods.

We performed experiments on three widely used text-based datasets: IMDB Movie Review, Yelp Reviews-

Polarity, and Amazon Reviews-Polarity. The attacked models include LSTM [21], GRU [22], CNN [23], and BERT [24]. We find that the XATA-LIME and XATA-SHAP can craft more effective adversarial examples compared with baseline attack methods. On top of that, XATA-SHAP is more effective than XATA-LIME. As an extension to LIME, SHAP is more accurate in explanation [18]. Hence, the advantages of XATA-SHAP over XATA-LIME can be attributed to a more accurate sensitivity estimation for each word token. This implies that an improved explainable method can lead to more threatful adversarial attacks. These findings confirm a contradiction between explanation and adversarial robustness, and we briefly discuss this phenomenon in the section part of this paper.

The contribution of this study is three-fold:

- First, we propose a new method to craft adversarial text examples for black-box attacks. The proposed attack method outperforms existing baselines in various target models on multiple datasets. Hence, it can be used to better assess the security and vulnerability of deep learning models.
- Second, this study reveals the connections between black-box adversarial attack methods and local post-hoc explainable DL methods. Both streams aim to estimate the token sensitivity, operate on the individual example level, and are model agnostic.
- Last but not least, we empirically demonstrate the contradiction (trade-off) between explainability and adversarial robustness in DL models. When researchers constantly advance the explainability of DL models, they also provide hackers with tools to craft more targeted and effective adversarial attacks. This phenomenon necessitates attention from our community.

The remainder of the paper is organized as follows. Section 2 reviews the related work on adversarial text attacks. Section 3 discusses the motivations and elaborates on the details of the proposed attack method. The following section describes the evaluation, including experiment design and results. We discuss and conclude the paper in Sections 5 and 6, respectively.

2 RELATED WORK ON ADVERSARIAL TEXT ATTACKS

Adversarial attacks were discovered initially in the computer vision domain. Goodfellow et al. [4] found that small human-imperceptible perturbations to natural images will cause DL classifiers to miscategorize a panda image as a gibbon. Adversarial attacks have also become a hot research area in text-based applications such as misinformation detection [7], [25], [26] and sentiment analysis [27]-[29].

Adversarial attacks can be divided into white-box attacks and black-box ones [30]. White-box attacks are performed when model details like architectures and parameters are accessible to hackers [31]. In contrast, the hackers conducting black-box attacks are unaware of model details. Still, they can infer the target model information by

querying it [12]. Many studies have created different black-box attack methods, as black-box models are pervasive and are more realistic [32], [33].

Mathematically, given a legitimate example $\mathbf{w}$ consisting of a sequence of N tokens ($\mathbf{w} = [w_1, \dots, w_n, \dots, w_N]$), the hackers are accessible to the model's prediction, denoted as y , i.e., $\mathcal{F}(\mathbf{w}) = y$. Hackers can also access the predicted probability for class y , i.e., $\mathcal{F}_y(\mathbf{w})$. A black-box adversarial text attack aims to craft an adversarial example $\mathbf{w}^A$ that misleads the model to a preferred prediction $\hat{y}$, different from initial prediction y . Due to the discrete nature of text data, the text examples are restricted in a space $\mathcal{W}$. The crafted example $\mathbf{w}^A$ should be similar to the original example $\mathbf{w}$. We use $\mathcal{S}(\mathbf{w}, \mathbf{w}')$ to denote a domain-specific similarity function $\mathcal{S}: \mathcal{W} \times \mathcal{W} \rightarrow \mathbb{R}^+$. The required minimum similarity between $\mathbf{w}^A$ and $\mathbf{w}$ is ϵ . Then, the attack task can be formally denoted as:

$$\mathbf{w}^A = \arg \max_{\mathbf{w}' \in \mathcal{W}} \mathcal{S}(\mathbf{w}, \mathbf{w}') \quad (1)$$

$$\text{s.t.} \begin{cases} \mathcal{F}(\mathbf{w}') = \hat{y} \\ \mathcal{F}(\mathbf{w}) = y \\ y \neq \hat{y} \\ \mathcal{S}(\mathbf{w}, \mathbf{w}') \geq \epsilon \end{cases}$$

Since the optimization is hard to solve, past studies adopted a two-phase solution to craft the adversarial examples heuristically (Fig. 2). The first phase is the sensitivity estimation which measures the sensitivity of the prediction change to each input token (e.g., word or character). According to token sensitivity computed in the first phase, the second phase is perturbation execution. The perturbation of a highly-sensitive token will bring about a more significant outcome change than a slightly-sensitive one. Hence, the most sensitive tokens are perturbed to craft adversarial examples.

Fig. 2. The Adversarial Text Attack Framework with Two Phases.

Formally, $s_{w_n}^y$ denotes the sensitivity score of w_n towards the predicted class y . Phase 1 aims to estimate the sensitivity score for each word token $s_{w_n}^y, \forall n$ with a mapping function $\mathcal{C}$, as shown in Equation (2).

$$s_{w_n}^y = \mathcal{C}(\mathcal{F}_y, \mathbf{w}, w_n), n = 1, \dots, N \quad (2)$$

We use $\mathbf{s}^y$ to denote the collection of $s_{w_n}^y, \forall n$, i.e., $\mathbf{s}^y = [s_{w_1}^y, s_{w_2}^y, \dots, s_{w_N}^y]$. Phase 2 aims to craft an adversarial example $\mathbf{w}^A$ that can mislead the model. We denote the function of perturbation as $\mathcal{P}(\cdot)$, and $\mathbf{w}^A$ is crafted by

$$\mathbf{w}^A = \mathcal{P}(\mathcal{F}, \mathbf{w}, \hat{y}, \mathbf{s}^y). \quad (3)$$

Identifying sensitive tokens in Equation (2) lays the foundation for executing effective perturbation in Equation (3). Two types of methods have been proposed to solve Equation (2): gradient-based and deletion-based.

Proposed by Papernot et al. [31], gradient-based methods compute the gradients of model outcome to the input token. As a larger gradient score means the token is more sensitive and vice versa, the gradient score is used as the sensitivity score for the subsequent phase [7], [34], [35].

However, as hackers cannot access target model details, a surrogate model is trained to approximate the target, and token sensitivity is then estimated from the surrogate model [36]. Formally, $\tilde{\mathcal{F}}$ and $\tilde{\mathcal{F}}_y$ denote the counterparts of the surrogate model, $\mathbf{w}_n^e$ denotes the embedding vector of w_n . Sensitivity score $s_{w_n}^y$ is calculated with Equation (4):

$$s_{w_n}^y = \|\nabla_{\mathbf{w}_n^e} \tilde{\mathcal{F}}_y(\mathbf{w})\|_2 \quad (4)$$

where $\nabla_{\mathbf{w}_n^e}$ is the gradient of $\mathbf{w}_n^e$ and $\|\cdot\|_2$ is the L_2 norm.

As the actual token value indicates how strongly a token is expressed, the Equation (4) is extended by multiplying the gradient with the token embedding [37], given by:

$$s_{w_n}^y = \|\nabla_{\mathbf{w}_n^e} \tilde{\mathcal{F}}_y(\mathbf{w}) \cdot \mathbf{w}_n^e\|_2 \quad (5)$$

Gradients can be obtained via backpropagation. However, as the widely used ReLU activation function zeroes out negative signals during backpropagation, this method usually fails to highlight tokens that negatively contribute to the outcome [15]. Consequently, the estimated score may not precisely measure the token sensitivity.

Deletion-based methods delete a token from the example and query the target model with the new data example. The model outcome difference before and after deletion reflects the sensitivity of this token. Formally, the example after deleting the token is denoted as w_n as $\mathbf{w}_{\setminus w_n} = [w_1, \dots, w_{n-1}, w_{n+1}, \dots, w_N]$. DeepFool [6] computes the differences of the predicted probability for class y to represent sensitivity as:

$$s_{w_n}^y = \mathcal{F}_y(\mathbf{w}) - \mathcal{F}_y(\mathbf{w}_{\setminus w_n}) \quad (6)$$

DeepWordBug [5] extends the computation by considering the sequentiality of input. For each token w_n , they defined the Temporal Head Score (THS) as the outcome difference between two heading parts of an example, while the Temporal Tail Score (TTS) quantified the difference between two tailing parts of the example. The weight of TTS is controlled by a hyperparameter λ . The calculation is given by:

$$s_{w_n}^y = \text{THS}(w_n) + \lambda \cdot \text{TTS}(w_n)$$

$$= [\mathcal{F}_y([w_1, \dots, w_n]) - \mathcal{F}_y([w_1, \dots, w_{n-1}])] + \lambda [\mathcal{F}_y([w_n, \dots, w_N]) - \mathcal{F}_y([w_{n+1}, \dots, w_N])] \quad (7)$$

PWWS [14] combines token saliency and predicted probability. Token saliency is defined same as Equation (6), except that w_n is replaced by an unknown token instead of being deleted. Δp^* represents the maximum change of probability after w_n is replaced with different strategies, the sensitivity is calculated by Equation (8):

$$s_{w_n}^y = [\mathcal{F}_y(\mathbf{w}) - \mathcal{F}_y(\mathbf{w}_{\setminus w_n})] \cdot \Delta p^* \quad (8)$$

TextFooler [13] further extends the computation by considering the predicted class, given by Equation (9):

$$s_{w_n}^y = \begin{cases} \mathcal{F}_y(\mathbf{w}) - \mathcal{F}_y(\mathbf{w}_{\setminus w_n}), & \text{if } \mathcal{F}(\mathbf{w}) = \mathcal{F}(\mathbf{w}_{\setminus w_n}) = y \\ (\mathcal{F}_y(\mathbf{w}) - \mathcal{F}_y(\mathbf{w}_{\setminus w_n})) + (\mathcal{F}_{\hat{y}}(\mathbf{w}_{\setminus w_n}) - \mathcal{F}_{\hat{y}}(\mathbf{w})), & \text{if } \mathcal{F}(\mathbf{w}) = y, \mathcal{F}(\mathbf{w}_{\setminus w_n}) = \hat{y}, \text{ and } y \neq \hat{y}. \end{cases} \quad (9)$$

While valuable, deletion-based methods only compute the sensitivity score of one token. However, multiple tokens may need perturbation to craft an adversarial example. Although one can perturb the tokens sequentially based on their sensitivity score, this may not be effective due to the overlapping effects between token sensitivity. For instance, assuming the score of three tokens w_1, w_2, w_3

are 0.2, 0.15, and 0.1, respectively. The total effects of perturbing w_1 and w_2 may not be more significant than perturbing w_2 and w_3 because the effects of w_1 may overlap with w_2 while the effects of w_2 and w_3 are less overlapped. This problem occurs because the independent estimation fails to consider the joint effects between tokens. One solution is to estimate the sensitivity of tokens with one model simultaneously.

3 EXPLANATION-BASED ADVERSARIAL ATTACKS

We first describe explainable DL studies to explain why they can be leveraged to craft adversarial text examples. Then, we describe the procedure of creating adversarial text attacks with explainable methods. We name this type of attack method as eXplanation-based Adversarial Text Attacks (XATA) for simplicity.

3.1 Insights from Explainable DL Studies

DL models are criticized for lacking interpretability [38], [39]. It prevents developers from making informative improvements to the models, reduces people's trust in them, and ultimately hinders DL developments. Hence, many studies are conducted to improve DL explanations, and this stream of research is called explainable DL [16], [40], [41].

Explainable DL methods can be broadly grouped into two categories: the global and the local explainable methods, depending on the scope of the explanation [8]. Global explainable methods enable people to inspect and visualize the model structures and parameters. In contrast, local explainable methods focus on the prediction rationale for an individual example. They try to figure out the role of each token in the example. The role of each token is represented by a score that can be used to reflect the sensitivity. The mathematical description is the same as Equation (2). Hence, the local explainable methods shared the same task with Phase 1 in adversarial attacks. From Equation (3), the only information required from Phase 1 is the token sensitivity $s_{w_n}^y, \forall n = 1, \dots, N$. Hence, the outcome of the local explanation completely satisfies the requirement of Phase 2. Consequently, local explainable methods could be leveraged for adversarial attacks.

Based on when the explanation is obtained, the local explainable methods can be further divided into intrinsic and post-hoc methods [40]. Compared with intrinsic methods that design self-explainable models to offer explanation, the post-hoc methods introduce an explanation model (e.g., linear regression) as a second model to locally approximate the target model. The post-hoc methods require no access to the model knowledge for an explanation. Thus, local post-hoc methods apply to any DL models, satisfying the model-agnostic requirement of black-box attacks (i.e., the attacks can successfully be conducted for any type of target model). Hence, local post-hoc methods are suitable for black-box attacks.

Fig. 3 shows the connections between local post-hoc methods and black-box adversarial attacks.

Fig. 3. Connections between local post-hoc methods and black-box adversarial attacks.

3.2 Proposed Adversarial Text Attack Method

Consistent with prior studies, the proposed explanation-based adversarial text attack methods also have two phases. Phase 1 is sensitivity estimation which leverages the post-hoc explainable techniques to compute the sensitivity of each input token. Phase 2 is perturbation execution which conducts a perturbation according to token sensitivity from the post-hoc explainable methods. The details of the method are shown in Fig. 4.

Fig. 4. The proposed XATA framework

Consistent with prior black-box attack settings [8], [13], [14], [28], [42], the hacker can query the target model with a text example $\mathbf{w}$ consisting of a sequence of N tokens ($\mathbf{w} = [w_1, w_2, \dots, w_N]$). The model will return the predicted class as y , i.e., $\mathcal{F}(\mathbf{w}) = y$, as well as the predicted probability for class y $\mathcal{F}_y(\mathbf{w})$.

3.2.1 Phase 1: Sensitivity Estimation

Phase 1 aims to compute sensitivity for each token $s_{w_n}^y, \forall n$ via an explanation model. Hence, the explanation model act as the surrogate model in the black-box attacks. As a type of explanation models, the additive feature attribution models define the explanation model as a linear model and assume the contributions of each token are additive. We use the additive feature attribution models as the explanation model because they can well explain complex DL models, as shown in [18]. We use $\tilde{\mathbf{w}}$ to denote a perturbed example of $\mathbf{w}$, e.g., $\tilde{\mathbf{w}} = [w_1, w_2, w_4, \dots, w_{n-1}, w_{n+1}, \dots, w_N]$. Consist with prior notations, we use $\tilde{\mathcal{F}}(\tilde{\mathbf{w}})$ and $\tilde{\mathcal{F}}_y(\tilde{\mathbf{w}})$ to denote the predicted class and the predicted probability for class y of the explanation (surrogate) model, and $\mathbf{s}^y = [s_{w_1}^y, s_{w_2}^y, \dots, s_{w_N}^y]$. Then,

$$\tilde{\mathcal{F}}_y(\tilde{\mathbf{w}}; b, \mathbf{s}^y) = b + \sum_{i=1}^N s_{w_i}^y E(w_i, \tilde{\mathbf{w}}), \quad (10)$$

where b is the bias for the linear model and $E(w_n, \tilde{\mathbf{w}})$ shows the existence of token w_n in $\tilde{\mathbf{w}}$, given by

$$E(w_n, \tilde{\mathbf{w}}) = \begin{cases} 1, & \text{if } w_n \text{ in } \tilde{\mathbf{w}} \\ 0, & \text{otherwise} \end{cases}$$

In order to obtain a reliable explanation model with local fidelity, optimization is involved in computing the model parameters θ , which include the b and $\mathbf{s}^y$. The sensitivity scores (i.e., $s_{w_n}^y, \forall n$) are determined simultaneously and the sensitivity determination of any token has considered the sensitivity of all other tokens. Hence, the sensitivity computed by our method can better reflect the impact of each token than the deletion-based approach. Moreover, the sensitivity is additive, meaning the total impact of a set of tokens equals the sum of the sensitivity score of each token. This can alleviate the overlapping effects confronted by the deletion-based methods. Meanwhile, our method also considers the fact that a token may negatively contribute to the outcome because the sensitivity score can be negative. This enables our method to provide a more accurate estimation than the gradient-based methods.

With $\mathbf{s}^y$, we sort the tokens based on sensitivity scores, given by:

$$\mathbf{I}^y = f_s([s_{w_1}^y, s_{w_2}^y, \dots, s_{w_N}^y]) \quad (11)$$

where f_s is the function that sorts its tokens in descending order and $\mathbf{I}^y$ is the index vector.

3.2.2 Phase 2: Perturbation Execution

As previously mentioned, the only required information from Phase 1 by Phase 2 is token sensitivity scores. Hence, most perturbation strategies [5], [6], [13], [14], [34], [42] and common perturbations (e.g., insert, removal, or replacement) from prior studies apply to our method. Furthermore, they can operate on both word- or character-level, as the local post-hoc explainable methods can explain models on either level, depending on the perturbation granularity

of $\tilde{\mathbf{w}}$. We select the visually similar character replacement strategy proposed by [5], where word tokens are replaced with other visually-similar characters. For instance, "o" can be changed to "0," and "1" can be changed to "l" when crafting the adversarial examples. Such perturbation strategy is human-imperceptible, thus retaining the semantic as the legitimate examples. It is also widely adopted for crafting adversarial text examples by prior studies [7], [43].

The perturbation is conducted according to their sensitivity scores. We first generate the replacement for the most sensitive words $w_{I_1}^y$, given by

$$w_{I_1}^y = f_p(w_{I_1}^y), \quad (12)$$

where f_p is the perturbation function. Then, we replace $w_{I_1}^y$ with $w_{I_1}^y$ to obtain the crafted example $\mathbf{w}'$ as:

$$\mathbf{w}' = [\dots, w_{I_1}^y, w_{I_1}^y, \dots]. \quad (13)$$

If the crafted example cannot mislead the target model, we repeat the same perturbation to the second sensitive word $w_{I_2}^y$, the third sensitive word $w_{I_3}^y$, and so on until the crafted example can successfully mislead the target model to make a prediction $\hat{y}$ ($\hat{y} \neq y$). In this way, we obtain adversarial textual examples $\mathbf{w}^A$.

Our proposed explainable adversarial attack framework can leverage most local post-hoc explainable methods. As LIME and SHAP are two representative local post-hoc explainable methods, we elaborate on using LIME and SHAP for adversarial text attacks in detail.

3.3 LIME-based Attack: XATA-LIME

We performed the following five steps to estimate token sensitivities for attacks:

- **Step 1:** Sampling a set of examples $\tilde{\mathbf{W}} = [\tilde{\mathbf{w}}^1, \tilde{\mathbf{w}}^2, \dots, \tilde{\mathbf{w}}^M]$ by randomly removing some tokens in $\mathbf{w}$.
- **Step 2:** Feeding the $\tilde{\mathbf{W}}$ to $\mathcal{F}_y$ to get predicted probabilities $\tilde{\mathbf{Y}} = [\tilde{y}^1, \tilde{y}^2, \dots, \tilde{y}^M]$.
- **Step 3:** Transform $\tilde{\mathbf{w}}^m$ to a mask vector $\tilde{\mathbf{v}}^m, \forall m$. If a word is removed in $\tilde{\mathbf{w}}^m$, its corresponding vector dimension in $\tilde{\mathbf{v}}^m$ is 0, and 1 otherwise. Hence, $\tilde{\mathbf{v}}^m$ has the same length as $\tilde{\mathbf{w}}^m$. The same transformation is used for $\mathbf{w}$ to get $\mathbf{v}$.
- **Step 4:** Measure the proximity between $\mathbf{v}$ and $\tilde{\mathbf{v}}^m$ to get weights $\pi_{\text{LIME}}^m = \pi_{\text{LIME}}(\mathbf{v}, \tilde{\mathbf{v}}^m)$, where $\pi_{\text{LIME}}(\mathbf{v}, \tilde{\mathbf{v}}^m)$ is an exponential kernel with width σ . $|\mathbf{v}|$ denotes the number of non-zero dimensions, i.e., $|\mathbf{v}| = \sum_{i=1}^N \mathbb{I}[v_i = 1]$. The kernel is defined in Equation (14):

$$\pi_{\text{LIME}}(\mathbf{v}, \tilde{\mathbf{v}}^m) = \exp\left(-\left(\frac{\mathbf{v} \cdot \tilde{\mathbf{v}}^m}{\sqrt{|\mathbf{v}| |\tilde{\mathbf{v}}^m|}}\right)^2 / \sigma^2\right) \quad (14)$$

- **Step 5:** Surrogate model parameters θ include b and $\mathbf{s}^y = [s_{w_1}^y, s_{w_2}^y, \dots, s_{w_N}^y]$. Optimal parameters are given by:

$$\theta^* = \arg \min_{\theta} \sum_{m=1}^M (\mathcal{F}_y(\tilde{\mathbf{w}}^m) - \tilde{\mathcal{F}}_y(\tilde{\mathbf{w}}^m; \theta))^2 \pi_{\text{LIME}}^m + \Omega(\theta), \quad (15)$$

where $\Omega(\theta)$ is the number of non-zero weights to ensure explanation. Optimization techniques like gradient-descent algorithms can be adopted to determine

model parameters, which include sensitivity score for each token.

Then, Phase 2 is executed as mentioned in Section 3.2.2.

3.4 SHAP-based Attack: XATA-SHAP

The token sensitivity scores $s_{w_n}^y, \forall n$ computed from SHAP explainable methods are called Shapley values. The process of XATA-SHAP is similar to XATA-LIME, except that the Shapley kernel is used in Step 4 to calculate the proximity. Specifically,

$$\pi_{\text{SHAP}}(\mathbf{v}, \tilde{\mathbf{v}}^m) = \frac{(|\mathbf{v}| - 1)}{\binom{|\mathbf{v}|}{|\tilde{\mathbf{v}}^m|}} |\tilde{\mathbf{v}}^m| (|\mathbf{v}| - |\tilde{\mathbf{v}}^m|) \quad (16)$$

Accordingly, optimal parameters of the surrogate model is given by:

$$\theta^* = \arg \min_{\theta} \sum_{m=1}^M \left(\mathcal{F}_y(\tilde{\mathbf{w}}^m) - \tilde{\mathcal{F}}_y(\tilde{\mathbf{w}}^m; \theta) \right)^2 \pi_{\text{SHAP}}^m + \Omega(\theta) \quad (17)$$

The Shapley kernel guarantees nice properties for sensitivity estimation (e.g., local accuracy, missingness, and consistency [18]), thus estimating a more accurate token sensitivity than LIME.

Moreover, premised on cooperative game theory, Shapley value explains the contribution of each token towards model outcomes. Shapley value is highly informative because it predicts how the model will behave without each token. For example, if the Shapley value for a specific word is v , the model outcome will be reduced by v theoretically with this word masked. Hence, the hackers can craft adversarial examples guided by Shapley values. The pseudo-code of the explanation-based (LIME/SHAP based) adversarial text attack method is shown in Algorithm 1.

Algorithm 1: Explanation-based Adversarial Text Attacks

Input: Target model, legitimate example $\mathbf{w}$

Output: adversarial example $\mathbf{w}^A$

1. **if** XATA-LIME: Compute sensitivity with Equation (15)
 2. **if** XATA-SHAP: Compute sensitivity with Equation (17)
 3. Obtain index vector I^y with Equation (11)
 4. $i = 1$
 5. **for** I_i^y in I^y :
 6. Obtain new token w'_{i^y} with Equation (12)
 7. Generate crafted example $\mathbf{w}'$ with Equation (13)
 8. **if** $\mathcal{F}_y(\mathbf{w}') \neq y'$:
 9. $\mathbf{w} \leftarrow \mathbf{w}'; i \leftarrow i + 1$
 10. **else:**
 11. $\mathbf{w}^A \leftarrow \mathbf{w}'$
 12. **return** $\mathbf{w}^A$
-

4 EVALUATIONS

4.1 Datasets

Consistent with prior adversarial studies [5], [7], [13], [14], we tested the effectiveness of the proposed attacks on sentiment polarity classification tasks. Experiments are conducted on three datasets: IMDB Movie Reviews¹, Yelp reviews-Polarity², and Amazon reviews-Polarity³. These

datasets are widely used in adversarial text attack studies [5], [13], [14], [42], [44], [45]. All three datasets contain pre-split training and testing sets. We summarize these three datasets in Table 1.

Table 1. Dataset Summary

Datasets	# Positive/Negative Samples	# Training/Testing Samples	Average length
IMDB	25,000/25,000	25,000/25,000	232.82
Yelp	149,000/69,500	250,000/38,000	138.97
Amazon	2,000,000/2,000,000	3,600,000/400,000	80.88

4.2 Experiment Design

4.2.1 Baseline Attack Methods

We compared the proposed XATA with existing text attack baseline methods. To ensure a fair comparison, we use the visually similar characters replacement strategy introduced in Section 3 for all baselines. Hence, the performance differences are completely attributed to the sensitivity estimation. The baseline methods included TextBugger [7], Gradient*Input [37], DeepFool [6], DeepWordBug [5], PWWS [14], and TextFooler [13], each of which has been described in Section 2. The token sensitivity was computed according to Equation (4), (5), (6), (7), (8) and (9) respectively. Note that the gradient-based methods required to train a surrogate model. However, the training process may introduce extra biases, making it hard for fair comparison. Hence, we used the target model itself as surrogate model. In other words, the gradient-based methods were conducted in white-box. Consistent with prior studies [5], [7], [13], [14], [45], we also included the random-based method. We randomly assigned a sensitivity score to each word in the example and repeatedly perturbed the word with the highest score until the target model is fooled.

4.2.2 Attacked Deep Learning Models

Consistent with prior adversarial attack studies [5], [14], [28], [44], we verified the effectiveness of the proposed XATA on three types of deep learning-based classifiers: RNN, CNN, and BERT. The details are as follows:

- **RNN (LSTM, BiLSTM, GRU, BiGRU).** The first layer is an embedding layer with an embedding matrix. We used the pre-trained 100-dimension GloVe⁴ word embedding by Pennington et al. [19] to transform the discrete inputs to dense vectors. The second layer is the RNN layer (e.g., LSTM) with 128 hidden nodes. The final layer is a fully connected layer for classification.
- **CNN (CNN-2, CNN-3, CNN-4).** The first layer is the same as RNN-based classifiers, which map discrete textual examples to vectors. In the second layer, filters of different sizes are used for convolution operation. For each filter, the width can be adjusted, and the length is fixed as embedding dimension. We use two different width filters (3 and 4) on CNN-2, three (3, 4 and 5) on CNN-3, and 4 (2, 3, 4 and 5) on CNN-4; the channel number for each filter is 100. We then apply a max-pooling operation for feature vectors obtained by the previous convolution operation and use a fully

¹ <https://s3.amazonaws.com/fast-ai-nlp/imdb.tgz>

² https://s3.amazonaws.com/fast-ai-nlp/yelp_review_polarity_csv.tgz

³ https://s3.amazonaws.com/fast-ai-nlp/amazon_review_polarity_csv.tgz

⁴ <https://nlp.stanford.edu/projects/glove/>

connected layer for classification.

- **BERT.** We used the pre-trained BERT-base-uncased version⁵, a 12-layer BERT with 768 hidden units and 12 heads. We add a fully connected layer for classification and then finetune BERT on our datasets.

All models are trained based on the word level. We adopted Adam Optimizer [46] to optimize the parameters. The learning rates are 5e-3 for RNN and CNN and 5e-5 for BERT. All models are trained with a hold-out test strategy in original training data (i.e., 80% for training and 20% for validation) and test on the testing data for each dataset. All the models are implemented based on Pytorch. The experiments are conducted on a GPU server with Intel® Xeon® Gold 6226R CPU @ 2.90GHz and two NVIDIA GeForce RTX™ 3090 with 24GB GDDR6X.

4.2.3 Evaluation Metrics

To verify the effectiveness of the proposed text attack method, we crafted adversarial examples for the examples in the test set. We adopted four evaluation metrics commonly used in adversarial attack studies; each was averaged over the examples [7], [13], [42], [47].

- **Attacked Accuracy:** We obtained the accuracy of the target model on the crafted examples by XATA. The examples were crafted where similarity constraint (i.e., $S(\mathbf{w}, \mathbf{w}') \geq \epsilon$ in Equation (1)) was relaxed. Hence, this metric represents the attack performance without perturbation upper bound.
- **Success Rate@N:** We perturbed N% of the words in the legitimate examples and then used them to attack target models. This metric evaluates an attack method’s effectiveness given a perturbation upper bound (i.e., N%).
- **Perturbation Rate:** It is calculated by counting the minimum number of words perturbed in the legitimate example when the target model is attacked successfully. Since the lower the perturbation rate, the

higher the imperceptibility, this metric represents the quality and threatening level of the crafted adversarial examples.

- **Perturbation Impact@N:** It is calculated by the deviation between the predicted probability of the adversarial example and the legitimate example, where the adversarial example is crafted with perturbation upper bound N%. It measures the adversarial examples’ impact on the target model.

4.3 Experiment Results

4.3.1 Attacked Accuracy

We executed XATA-SHAP and XATA-LIME on three datasets (IMDB, Yelp, and Amazon). The target models, including RNN, CNN, and BERT, can get high accuracy in test set (called original accuracy). For instance, the original accuracies of all models were more than 92% on Yelp. Table 2 compared the original accuracy with the attacked accuracy. The lowest accuracies are bold-faced. We observed that the accuracy of all classifiers decreased dramatically after being attacked with XATA. Even for BERT, whose original accuracy was over 90% on all three datasets (91.79 % on IMDB, 94.68% on Yelp, and 95.19% on Amazon), the attacked accuracy was nearly 0%. Meanwhile, we found XATA-SHAP to be more threatening than XATA-LIME because SHAP usually has a lower attacked accuracy than LIME. For example, BiLSTM has an attack accuracy of 7.55% on IMDB when attacked by XATA-LIME, but its accuracy drops to 0 when attacked by XATA-SHAP. The results indicated that adversarial examples generated by XATA successfully fooled DL classifiers. The proposed XATA (regardless of XATA-LIME or XATA-SHAP) is effective for text attacks, and XATA-SHAP is more effective than XATA-LIME.

Table 2. Attacked Accuracy of Different DL Classifiers under XATA

Attacked Models	IMDB			Yelp			Amazon		
	Original Accuracy	XATA-SHAP	XATA-LIME	Original Accuracy	XATA-SHAP	XATA-LIME	Original Accuracy	XATA-SHAP	XATA-LIME
LSTM	82.81	0.35	0.75	93.85	0.55	1.05	92.62	0.75	1.60
BiLSTM	85.64	0.00	7.55	93.20	0.75	1.40	92.66	0.35	1.15
GRU	85.74	0.00	0.50	93.40	0.15	0.20	92.55	0.80	1.95
BiGRU	87.00	0.05	0.05	93.49	1.25	3.05	92.60	2.60	4.05
CNN-2	86.80	0.00	0.00	92.46	0.30	1.00	89.62	0.00	0.10
CNN-3	87.15	0.00	0.00	92.69	0.15	0.30	90.07	0.05	0.10
CNN-4	87.23	0.00	0.00	92.93	0.00	0.10	90.07	0.00	0.00
BERT	91.79	0.00	0.00	94.68	0.45	4.35	95.19	0.05	0.05

Three randomly selected adversarial text examples by XTAT-SHAP for BERT are shown in Table 3. BERT made correct decisions for all the legitimate examples with predicted probability more than 90% in all cases. However, all the decisions were altered with adversarial examples. In the IMDB dataset, the critical review was correctly

classified as negative with a probability of 99.9%. XTAT-SHAP offered the top 5 sensitive words, which included “avoid”, “fails”, “flat”, “tries” and “fails”. Perturbation was executed for each word sequentially, and the decision was altered after perturbing the first three words. Particularly, the adversarial example was successfully crafted by

⁵ <https://github.com/huggingface/pytorch-pretrained-BERT>

changing “AVOID” to “AV0ID”, “fails” to “f ils” and “FLAT” to “F1AT”. Such perturbations were visually similar to the original characters, making the perturbation human-imperceptible. For the example in the Yelp dataset, the adversarial example was crafted by only perturbing the most sensitive word “quality” to “qu lity”. Similarly, after perturbing the two most sensitive words “Painful” and

“how” to “Painfu1” and “h0w”, the adversarial example misled the BERT classifier to a wrong prediction. This demonstrates the effectiveness of the adversarial examples crafted by the proposed XATA.

Table 3. Adversarial examples crafted by XTAT-SHAP

Dataset		Example	Prediction
IMDB	Legitimate example	»This film tries very hard to be an “action” film, but it miserably. Steve Guttenberg plays the head of an elite counter-terrorist team that fails (?) in attempt to keep a mysterious group from stealing a deadly nerve agent. The story...the acting...the special effects...ALL FALL ! Z !!! Definitely A MUST) " fl!!!!	Negative (99.9%)
	Adversarial example	»This film tries very hard to be an “action” film, but it miserably. Steve Guttenberg plays the head of an elite counter-terrorist team that fails (?) in attempt to keep a mysterious group from stealing a deadly nerve agent. The story...the acting...the special effects...ALL FALL ! !!! Definitely A MUST) fl!!!!	Positive (99.9%)
Yelp	Legitimate example	»Good pork fried dumpling and mongo beef. Bar service was fast and Efficient. Good value. Not a 5 cause dragon eye tea was cold first time around and beef had minor amount of fat. Fortune was even Positive!	Positive (92.7%)
	Adversarial example	»Good pork fried dumpling and mongo beef. Bar service was fast and Efficient. Good value. Not a 5 cause dragon eye tea was cold first time around and beef had minor amount of fat. Fortune was even Positive!	Negative (67.0%)
Amazon	Legitimate example	»# This book has to be one of the most tedious works of literature ever written. Hawthorne is a great writer, but I don’t know this book made it into that sacred list we call “classics”. Perhaps on the merit of his name alone?”	Negative (99.9%)
	Adversarial example	Text: “Painful”, “This book has to be one of the most tedious works of literature ever written. Hawthorne is a great writer, but I don’t know h0w this book made it into that sacred list we call “classics”. Perhaps on the merit of his name alone?”	Positive (97.5%)

attack baselines. The higher the success rate, the stronger the attack method. The results are summarized in Table 4. The best performance is bolded, while the second-best performance is underlined.

4.3.2 Success Rate@N Comparison

We evaluated the performance of XATA under perturbation upper bound by comparing XATA with other text

Table 4. Success Rate@N of Different Attack Methods on Three Datasets

Attacked Models	Attack Methods	IMDB					Yelp					Amazon				
		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
LSTM	XATA-SHAP	52.0	68.4	77.4	84.6	89.0	22.0	35.4	45.8	53.0	60.3	24.5	39.9	52.2	60.7	68.2
	XATA-LIME	<u>48.4</u>	<u>63.3</u>	<u>70.6</u>	<u>75.7</u>	<u>78.8</u>	<u>22.2</u>	<u>34.9</u>	<u>43.6</u>	<u>50.1</u>	<u>55.6</u>	<u>24.1</u>	<u>38.7</u>	<u>50.0</u>	<u>58.7</u>	<u>64.5</u>
	TextFooler	30.6	40.4	46.2	51.1	54.0	20.8	29.1	35.9	40.5	44.4	20.6	28.0	33.3	37.1	40.6
	PWWS	47.8	56.2	62.7	67.0	70.0	24.4	34.4	42.0	46.5	51.6	24.8	32.6	38.7	43.3	46.5
	DeepWordBug	32.3	42.5	48.3	52.6	56.7	14.7	20.0	25.2	30.6	35.9	20.7	30.4	37.6	42.5	47.5
	DeepFool	30.6	40.4	46.2	51.1	54.0	20.8	29.1	35.9	40.5	44.4	20.6	28.0	33.3	37.1	40.6
	Gradient*Input	39.0	50.8	57.7	62.9	66.2	21.6	32.3	40.4	47.6	51.6	18.2	26.6	33.3	37.7	42.9
	TextBugger	10.7	12.5	15.0	17.3	18.7	6.6	8.0	9.6	10.9	11.9	7.8	8.4	9.9	11.6	13.4
	Random	4.0	5.6	7.5	9.0	9.7	1.8	2.5	3.6	4.2	4.5	2.0	2.5	3.7	4.1	4.2
GRU	XATA-SHAP	42.7	58.1	69.1	75.9	82.2	24.8	37.9	48.4	57.0	63.5	26.6	40.0	49.8	58.2	66.4
	XATA-LIME	<u>42.0</u>	<u>55.8</u>	<u>63.9</u>	<u>69.9</u>	<u>74.1</u>	<u>25.0</u>	<u>36.2</u>	<u>46.1</u>	<u>53.6</u>	<u>59.7</u>	24.5	<u>37.0</u>	<u>45.8</u>	<u>54.5</u>	<u>61.0</u>
	TextFooler	32.2	42.8	49.5	53.5	57.5	21.9	29.2	35.3	39.3	42.9	23.4	30.9	36.6	41.2	43.6
	PWWS	40.6	52.8	60.8	65.2	68.7	26.1	33.6	40.7	45.2	48.1	<u>25.7</u>	33.9	39.6	44.7	47.2
	DeepWordBug	27.3	36.2	41.5	45.9	48.6	15.8	22.4	28.0	31.5	35.6	21.6	30.4	37.6	44.0	48.4
	DeepFool	32.2	42.8	53.9	53.5	57.5	21.9	29.2	35.3	39.3	42.9	23.4	30.9	36.6	41.2	43.6
	Gradient*Input	35.4	48.0	56.2	61.6	65.2	20.8	28.7	35.7	40.4	45.1	21.1	29.4	36.2	41.2	45.2
	TextBugger	14.2	15.5	17.1	18.0	18.3	7.0	7.3	9.7	10.4	11.2	10.3	11.6	14.0	13.8	14.9

	Random	3.0	4.7	6.0	6.8	7.3	1.7	2.6	3.7	2.5	2.8	2.4	3.2	3.7	4.2	5.1
CNN-3	XATA-SHAP	53.6	69.9	80.2	85.2	88.3	39.2	55.4	65.1	72.0	76.4	34.2	51.4	62.6	71.0	76.6
	XATA-LIME	<u>50.9</u>	<u>66.1</u>	<u>73.0</u>	<u>76.7</u>	<u>79.4</u>	<u>37.6</u>	<u>51.7</u>	<u>59.7</u>	<u>64.9</u>	<u>68.2</u>	<u>33.6</u>	<u>49.3</u>	<u>59.8</u>	<u>66.3</u>	<u>72.2</u>
	TextFooler	33.0	37.7	39.5	40.5	40.9	28.1	32.3	34.0	35.8	37.3	26.0	32.5	37.2	41.2	43.7
	PWWS	41.0	48.3	50.6	52.0	52.5	31.2	37.7	41.6	44.0	45.6	29.0	37.6	44.6	49.2	51.8
	DeepWordBug	36.4	49.8	57.6	62.7	66.5	25.4	34.9	43.5	49.1	54.5	23.0	33.4	41.5	48.2	54.2
	DeepFool	33.0	37.7	39.5	40.5	40.9	28.1	32.3	34.0	35.8	37.3	26.0	32.5	37.2	41.2	43.7
	Gradient*Input	41.0	45.3	47.2	47.8	48.2	30.9	37.3	40.1	42.3	43.6	28.0	37.6	44.4	48.1	51.0
	TextBugger	14.4	15.3	16.7	17.7	18.5	10.5	10.4	10.8	12.3	13.0	11.8	12.4	12.3	12.6	12.7
	Random	1.6	3.2	5.1	6.1	7.7	2.1	3.2	4.3	5.0	5.8	2.8	4.4	6.1	7.4	7.8
BERT	XATA-SHAP	<u>31.5</u>	44.5	53.4	59.4	64.2	21.9	35.2	45.2	51.8	58.0	<u>18.2</u>	29.2	38.5	45.0	51.2
	XATA-LIME	32.6	<u>42.5</u>	<u>50.7</u>	<u>56.9</u>	<u>60.8</u>	<u>20.1</u>	<u>31.9</u>	<u>42.4</u>	<u>49.2</u>	<u>55.7</u>	17.6	<u>28.0</u>	<u>36.7</u>	<u>43.4</u>	<u>49.1</u>
	TextFooler	26.9	34.8	38.6	41.6	43.9	17.7	26.0	31.1	35.8	38.9	15.6	19.4	21.5	23.1	24.8
	PWWS	32.6	41.1	45.8	48.4	49.8	21.2	30.3	39.1	43.9	47.8	18.6	23.4	26.4	28.6	30.1
	DeepWordBug	14.8	20.4	25.0	28.8	32.4	12.9	17.2	20.4	22.9	26.1	11.2	15.6	19.7	22.9	27.1
	DeepFool	26.9	34.8	38.6	40.1	42.1	17.7	26.1	31.1	35.8	39.0	15.6	19.4	21.5	23.1	24.8
	Gradient*Input	6.3	9.4	11.0	13.5	15.2	4.9	5.7	6.7	7.4	7.8	3.0	4.6	5.5	6.2	7.1
	TextBugger	1.9	4.5	6.1	7.5	9.2	1.8	2.4	3.3	4.4	5.2	2.0	2.9	3.9	4.9	5.2
	Random	1.6	2.2	3.2	3.9	4.8	1.4	1.8	2.1	2.5	3.1	1.4	2.1	3.4	3.6	4.1

We set the perturbation upper bound to 2%-10%. Generally, both XATA-SHAP and XATA-LIME outperformed baselines. As the same perturbation strategy was adopted, this indicated that the sensitivity obtained by SHAP or LIME enabled more effective attacks. Note that though the gradient-based methods like TextBugger and Gradient*Input were conducted in white-box, our methods were more effective than them. This demonstrated the effectiveness of our methods. Besides, as the perturbation upper bound increases, the success rate of XATA increases faster than that of baselines. For example, with an upper bound of 2%, SHAP achieves a success rate of 52% for an LSTM classifier on IMDB. The best performing baseline was PWWS, with a success rate of 47.8%. Both XATA-SHAP (89.0%) and XATA-LIME (78.8%) are much higher than PWWS (70%), with the upper bound expanding to 10%. Hence, the advantage of our methods becomes more obvious when increasing number of words were perturbed. This was because our methods considered the overlapping effects, and thus the set of words to be perturbed in our methods were more impactful than baselines. Overall, the proposed XATA achieved effective attacks by identifying the most sensitive words, which is more accurate than baselines.

4.3.3 Perturbation Rate Comparison

To further compare XATA with baselines, we compare the number of words that have to be perturbed when the attack method successfully fooled the classifier. For each example, if the attack is successful, the number of perturbed words is recorded; if the attack fails, the number of words in the whole example is recorded. The results of the perturbation rate of different methods on three datasets are summarized in Table 5. The best performance is bolded, while the second-best performance is underlined.

Since a lower perturbation rate indicates a higher attack example quality, we conclude that XATA is superior to baselines. For example, to fool an LSTM classifier on the IMDB dataset, XATA-SHAP only needs to perturb 4.871% (11.342) of the words and XATA-LIME needs to perturb 7.628% (17.759). Their required perturbation rate is significantly smaller than the best-performing baseline (DeepWordBug) whose perturbation rate was 12.906% (30.048). The superiority of XATA is also evident for BERT. For example, the perturbation rate of XATA-SHAP is 15.375% (12.436) to fool the BERT on Amazon, and that of XATA-LIME is 19.752% (15.976), which is about only half of the perturbation required by DeepWordBug. This indicates that XATA's suggested perturbations were more effective in attacking the target model successfully.

Table 5. Perturbation Rate of Different Methods on Three Datasets

Attacked Models	Attack Methods	IMDB		Yelp		Amazon	
		Perturbation Rate (%)	Perturbed Words (#)	Perturbation Rate (%)	Perturbed Words (#)	Perturbation Rate (%)	Perturbed Words (#)
LSTM	XATA-SHAP	4.871	11.342	9.466	13.155	10.181	8.235
	XATA-LIME	<u>7.628</u>	<u>17.759</u>	<u>13.902</u>	<u>19.319</u>	<u>12.880</u>	<u>10.418</u>
	TextFooler	22.783	53.044	20.931	29.088	28.297	22.887
	PWWS	15.507	36.104	18.633	25.895	24.433	19.762
	DeepWordBug	12.906	30.048	19.142	26.602	19.206	15.534
	DeepFool	22.784	53.046	20.930	29.086	28.297	22.887
	Gradient*Input	13.675	31.838	16.533	22.976	27.851	22.526
	TextBugger	37.943	88.338	55.756	77.484	54.169	43.812
	Random	58.601	136.435	67.984	94.478	71.938	58.184

GRU	XATA-SHAP	4.906	11.423	8.210	11.410	9.471	7.661
	XATA-LIME	<u>7.498</u>	<u>17.457</u>	<u>12.315</u>	<u>17.115</u>	<u>13.019</u>	<u>10.530</u>
	TextFooler	17.143	39.913	24.919	34.630	27.535	22.271
	PWWS	11.437	26.628	22.109	30.725	25.879	20.931
	DeepWordBug	15.259	35.526	23.103	32.107	19.179	15.512
	DeepFool	17.143	39.912	24.919	34.630	27.535	22.271
	Gradient*Input	11.041	25.705	21.510	29.893	26.508	21.440
	TextBugger	39.480	91.918	56.005	77.830	51.100	41.330
	Random	57.380	133.592	65.172	90.570	70.485	57.008
CNN-3	XATA-SHAP	4.476	10.421	5.998	8.336	6.992	5.655
	XATA-LIME	<u>7.975</u>	<u>18.569</u>	<u>12.719</u>	<u>17.675</u>	<u>8.871</u>	<u>7.175</u>
	TextFooler	37.598	87.5355	41.049	57.046	29.934	24.211
	PWWS	29.924	69.669	35.005	48.646	24.154	19.536
	DeepWordBug	14.494	33.745	19.218	26.707	15.129	12.237
	DeepFool	37.645	87.644	41.116	57.140	29.945	24.220
	Gradient*Input	32.669	76.061	36.555	50.801	24.991	20.213
	TextBugger	59.471	138.461	61.837	85.935	55.896	45.209
	Random	61.850	144.000	67.088	93.233	64.491	52.160
BERT	XATA-SHAP	10.086	23.483	15.213	21.142	15.375	12.436
	XATA-LIME	<u>18.567</u>	<u>43.227</u>	<u>17.829</u>	<u>24.777</u>	<u>19.752</u>	<u>15.976</u>
	TextFooler	36.430	84.817	32.216	44.771	26.774	48.428
	PWWS	32.427	75.497	26.533	36.873	52.439	42.413
	DeepWordBug	26.612	61.957	24.382	33.884	37.198	30.086
	DeepFool	36.430	84.817	32.216	44.771	59.876	48.428
	Gradient*Input	42.215	98.285	53.136	73.843	50.886	41.157
	TextBugger	42.727	99.476	51.305	71.299	49.492	40.030
	Random	64.424	149.992	67.949	94.429	68.974	55.786

4.3.4 Perturbation Impact@N Comparison

For a more fine-grained comparison, we calculated the perturbation impact of different methods under different perturbation upper bounds. A higher perturbation impact means a more significant impact on the prediction results of the target model. As shown in Table 6, XATA-SHAP generally achieves the maximum perturbation impact, and XATA-LIME is the second largest. Hence, our methods have a more significant impact on the target model than baselines. For example, the perturbation impact of XATA-SHAP for BERT on IMDB datasets is 0.627, with a perturbation upper bound of 10%. XATA-LIME's impact was 0.503, and the best-performing baseline (PWWS) was 0.498 in the same case. Similar to the success rate@N comparison in Section 4.3.2, the perturbation impact of XATA (SHAP and LIME) increases faster than that of baselines. For

example, PWWS achieves the highest perturbation impact@2% (0.312) for BERT on IMDB datasets, but it is overtaken by XATA-SHAP and XATA-LIME at perturbation impact@4%. When the upper bound was allowed to be 10%, XATA-SHAP (perturbation impact as 0.627) and XATA-LIME (perturbation impact as 0.608) significantly outperformed PWWS (perturbation impact as 0.498). These findings further indicate that the word sensitivity obtained through deletion-based methods, such as PWWS, are compromised by overlapping effects between tokens. Meanwhile, even the white-box setting gave the gradient-based methods (TextBugger and Gradient*Input) advantages, our model still outperformed them. To sum up, by comparing perturbation impacts, we found that XATA resulted in a more significant impact on the target model than other methods and is more threatening to deep learning security, consistent with the previous comparison conclusion.

Table 6. Perturbation Impact@N of Different Methods

Attacked Models	Attack Methods	IMDB					Yelp					Amazon				
		2%	4%	6%	8%	10%	2%	4%	6%	8%	10%	2%	4%	6%	8%	10%
LSTM	XATA-SHAP	0.472	0.635	0.728	0.795	0.836	<u>0.187</u>	0.291	0.375	0.438	0.493	0.223	0.357	0.463	0.542	0.606
	XATA-LIME	<u>0.453</u>	<u>0.603</u>	<u>0.684</u>	<u>0.735</u>	<u>0.767</u>	0.186	0.281	<u>0.357</u>	<u>0.413</u>	<u>0.459</u>	0.214	<u>0.346</u>	<u>0.442</u>	<u>0.519</u>	<u>0.573</u>
	TextFooler	0.273	0.362	0.421	0.464	0.499	0.172	0.242	0.296	0.337	0.372	0.184	0.248	0.296	0.331	0.363
	PWWS	0.438	0.518	0.576	0.618	0.647	0.201	<u>0.288</u>	0.349	0.395	0.433	<u>0.218</u>	0.291	0.344	0.385	0.419
	DeepWordBug	0.286	0.385	0.442	0.486	0.523	0.110	0.161	0.207	0.247	0.285	0.180	0.264	0.322	0.369	0.413
	DeepFool	0.273	0.362	0.421	0.464	0.499	0.172	0.242	0.296	0.337	0.372	0.184	0.248	0.296	0.331	0.363
	Gradient*Input	0.354	0.466	0.534	0.578	0.611	0.182	0.270	0.338	0.391	0.432	0.164	0.239	0.296	0.337	0.377
	TextBugger	0.086	0.102	0.125	0.133	0.144	0.045	0.071	0.089	0.107	0.122	0.059	0.071	0.085	0.098	0.113
	Random	0.022	0.037	0.052	0.063	0.072	0.008	0.013	0.019	0.024	0.029	0.008	0.012	0.018	0.025	0.027
GRU	XATA-SHAP	<u>0.389</u>	<u>0.544</u>	0.652	0.724	0.781	0.217	0.339	0.440	0.519	0.579	0.229	0.346	0.437	0.512	0.577
	XATA-LIME	0.405	0.555	<u>0.643</u>	<u>0.705</u>	<u>0.750</u>	<u>0.222</u>	<u>0.343</u>	<u>0.442</u>	<u>0.516</u>	<u>0.573</u>	0.208	<u>0.324</u>	<u>0.405</u>	<u>0.471</u>	<u>0.524</u>

	TextFooler	0.295	0.395	0.461	0.505	0.541	0.194	0.268	0.321	0.358	0.389	0.204	0.270	0.316	0.352	0.378
	PWWS	0.377	0.491	0.566	0.614	0.650	0.231	0.309	0.367	0.408	0.439	0.220	0.293	0.345	0.384	0.412
	DeepWordBug	0.242	0.326	0.381	0.422	0.451	0.132	0.193	0.241	0.278	0.310	0.179	0.259	0.311	0.364	0.403
	DeepFool	0.295	0.395	0.461	0.505	0.541	0.194	0.268	0.321	0.358	0.389	0.204	0.270	0.316	0.352	0.378
	Gradient*Input	0.329	0.448	0.525	0.577	0.617	0.185	0.263	0.323	0.371	0.408	0.181	0.257	0.313	0.357	0.392
	TextBugger	0.115	0.131	0.147	0.154	0.160	0.065	0.082	0.095	0.108	0.118	0.081	0.101	0.119	0.127	0.139
	Random	0.018	0.030	0.040	0.050	0.057	0.008	0.015	0.021	0.025	0.028	0.009	0.009	0.022	0.027	0.030
CNN-3	XATA-SHAP	0.327	0.437	0.512	0.555	0.584	0.274	0.385	0.458	0.511	0.551	0.242	0.356	0.435	0.495	0.542
	XATA-LIME	<u>0.314</u>	<u>0.409</u>	<u>0.458</u>	<u>0.489</u>	<u>0.509</u>	<u>0.266</u>	<u>0.359</u>	<u>0.419</u>	<u>0.458</u>	<u>0.487</u>	<u>0.235</u>	<u>0.343</u>	<u>0.416</u>	<u>0.469</u>	<u>0.511</u>
	TextFooler	0.202	0.234	0.245	0.254	0.258	0.193	0.226	0.243	0.256	0.266	0.184	0.233	0.264	0.286	0.305
	PWWS	0.248	0.295	0.315	0.327	0.332	0.218	0.264	0.291	0.307	0.319	0.207	0.266	0.310	0.344	0.369
	DeepWordBug	0.212	0.296	0.351	0.394	0.424	0.159	0.231	0.290	0.335	0.375	0.144	0.217	0.277	0.324	0.366
	DeepFool	0.202	0.234	0.245	0.254	0.258	0.193	0.226	0.242	0.256	0.266	0.184	0.233	0.264	0.286	0.305
	Gradient*Input	0.253	0.281	0.294	0.300	0.305	0.215	0.261	0.282	0.298	0.308	0.144	0.217	0.277	0.324	0.366
	TextBugger	0.074	0.084	0.090	0.096	0.099	0.051	0.058	0.064	0.072	0.076	0.049	0.085	0.106	0.122	0.131
	Random	0.003	0.005	0.012	0.016	0.020	0.006	0.010	0.015	0.019	0.025	0.008	0.013	0.021	0.028	0.031
BERT	XATA-SHAP	0.301	0.432	0.517	0.578	0.627	0.209	0.337	0.434	0.504	0.561	0.171	0.278	0.362	0.436	0.492
	XATA-LIME	<u>0.308</u>	<u>0.425</u>	<u>0.507</u>	<u>0.569</u>	<u>0.608</u>	0.178	0.288	<u>0.377</u>	<u>0.444</u>	<u>0.497</u>	<u>0.164</u>	<u>0.266</u>	<u>0.350</u>	<u>0.417</u>	<u>0.474</u>
	TextFooler	0.255	0.331	0.372	0.401	0.421	0.168	0.249	0.298	0.344	0.375	0.143	0.181	0.200	0.217	0.235
	PWWS	0.312	0.411	0.458	0.484	0.498	<u>0.203</u>	<u>0.302</u>	0.373	0.424	0.459	0.171	0.218	0.251	0.273	0.288
	DeepWordBug	0.131	0.188	0.235	0.274	0.309	0.116	0.158	0.192	0.223	0.251	0.102	0.146	0.185	0.217	0.254
	DeepFool	0.255	0.331	0.372	0.401	0.421	0.168	0.249	0.298	0.344	0.375	0.143	0.181	0.200	0.217	0.235
	Gradient*Input	0.055	0.084	0.101	0.125	0.141	0.038	0.050	0.058	0.067	0.071	0.024	0.036	0.046	0.053	0.063
	TextBugger	0.019	0.034	0.053	0.068	0.084	0.018	0.018	0.026	0.035	0.042	0.014	0.024	0.034	0.041	0.046
	Random	0.008	0.014	0.020	0.029	0.038	0.009	0.015	0.018	0.021	0.023	0.008	0.015	0.023	0.026	0.034

5 DISCUSSION

5.1 Application in White-box Attack

Though we only demonstrated how the LIME- or SHAP-based adversarial text attacks could be conducted in black-box scenarios, they can also operate in white-box because accessing model details will not hinder the process nor the results of the explainable methods. Hackers can craft adversarial examples, ignoring the knowledge of model details. Interestingly, we found the proposed XATA, without using any model details, outperformed white-box attack baselines like TextBugger and Gradient*Input methods. This is also understandable because explainable methods are initially proposed to give developers insights into the trained complex DL model. Even though the developers are fully aware of model details, they still need explainable methods to understand the model rationale, such as the token sensitivity. Hence, local post-hoc explainable methods can provide hackers with a more accurate sensitivity score for each token without relying on model details.

5.2 Contradiction between Explanation and Adversarial Robustness

The experiments showed that the XATA outperformed all the existing baselines. Besides, an improved explainable method like SHAP can lead to more effective adversarial attacks. On one hand, more effective adversarial attacks can be used to evaluate the model robustness and security more accurately. On the other hand, hackers can also leverage such methods to pose threats to the target models. The lack of explanation motivates researchers to provide a myriad of ways to address this problem. The local post-hoc

methods are one of the streams that have attracted most efforts. The improvement of explanation provides hackers weapons to estimate token sensitivity more accurately and effectively, thus enabling them to craft more targeted and threatening adversarial examples. As a result, such efforts will increase the model's risk of being hacked. The improvement in model explanation seems to lead to a decrease in adversarial robustness. Hence, there seems to be a contradiction (trade-off) between explanation and adversarial robustness for DL. This phenomenon necessitates additional attention and consideration.

6 CONCLUSION

Deep learning models have achieved tremendous success in various natural language processing tasks. However, they are also strikingly vulnerable to adversarial attacks. In this paper, we proposed a new method for adversarial text attacks. We were motivated by the connections between the local post-hoc explainable deep learning methods and the sensitivity estimation phase in adversarial text attacks. We used two local post-hoc explainable methods (i.e., LIME and SHAP) to measure the sensitivity of each word and then perturb the words according to the sensitivity scores provided by the explanation methods. Based on the experiment results on three commonly used datasets, we demonstrated the advantages of the proposed attack methods (XATA) against state-of-the-art baselines across various target models, including LSTM, GRU, CNN, and BERT. We also discovered that an improved explainable method (i.e., SHAP) can enable more threatening attacks. This may imply a contradiction between explanation and adversarial robustness and necessitates attention from our community.

There are several promising directions for future

research. First, this study focuses on the local post-hoc methods. Future studies can analyze other explainable methods like global methods or local intrinsic methods for adversarial attacks. Second, we only adopt the visually similar character replacement perturbation strategy. However, different word- or character-level perturbation strategies (e.g., insert, flipping, removal) can be introduced. Third, this study centered on adversarial text attacks. We can apply the methods to other domains like computer vision to test its generalizability. Fourth, this study concentrated on the binary classification where the targeted attack is same as the untargeted attack. Future studies can extend the XATA to other classification tasks like multi-class and multi-label classification.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China (NSFC) with grant numbers 72101079, 71722010, 72171071, 91746302, 91846201, Excellent Fund of Hefei University of Technology (JZ2021HGPA0060).

REFERENCES

- [1] Y. Wang, Q. Chen, M. Ahmed, Z. Li, W. Pan, and H. Liu, "Joint Inference for Aspect-Level Sentiment Analysis by Deep Neural Networks and Linguistic Hints," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 5, pp. 2002-2014, 5/1/2021, doi: 10.1109/TKDE.2019.2947587.
- [2] M. Fazil, A. K. Sah, and M. Abulaish, "DeepSBD: A Deep Neural Network Model With Attention Mechanism for SocialBot Detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4211-4223, 2021.
- [3] Y. Sun *et al.*, "Joint learning of question answering and question generation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 5, pp. 971-982, 2019.
- [4] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and Harnessing Adversarial Examples," presented at the *International Conference on Learning Representations*, 3/20/2015, 2015.
- [5] J. Gao, J. Lanchantin, M. L. Soffa, and Y. Qi, "Black-box generation of adversarial text sequences to evade deep learning classifiers," in *2018 IEEE Security and Privacy Workshops (SPW)*, 2018: IEEE, pp. 50-56.
- [6] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 4208-4215.
- [7] J. Li, S. Ji, T. Du, B. Li, and T. Wang, "TextBugger: Generating Adversarial Text Against Real-world Applications," presented at the *Network and Distributed System Security Symposium*, 1/1/2019, 2019.
- [8] W. Wang, R. Wang, L. Wang, Z. Wang, and A. Ye, "Towards a Robust Deep Neural Network against Adversarial Texts: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1-1, 10/4/2021, doi: 10.1109/TKDE.2021.3117608.
- [9] B. Eshete, "Making machine learning trustworthy," *Science*, vol. 373, no. 6556, pp. 743-744, 2021.
- [10] Y. Xu, B. Du, and L. Zhang, "Assessing the threat of adversarial examples on deep neural networks for remote sensing scene classification: Attacks and defenses," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 2, pp. 1604-1617, 2020.
- [11] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019: IEEE, pp. 739-753.
- [12] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical Black-Box Attacks against Machine Learning," presented at the *Computer and Communications Security*, 4/2/2017, 2017.
- [13] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, "Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment," presented at the *National Conference on Artificial Intelligence*, 4/3/2020, 2020.
- [14] S. Ren, Y. Deng, K. He, and W. Che, "Generating natural language adversarial examples through probability weighted word saliency," in *Proceedings of the 57th annual meeting of the association for computational linguistics*, 2019, pp. 1085-1097.
- [15] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," presented at the *International Conference on Machine Learning*, 8/6/2017, 2017. [Online]. Available: <http://proceedings.mlr.press/v70/shrikumar17a/shrikumar17a.pdf>.
- [16] E. Tjoa and C. Guan, "A survey on explainable artificial intelligence (xai): Toward medical xai," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.
- [17] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should i trust you? Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135-1144.
- [18] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768-4777.
- [19] J. R. Zilke, E. Mencía, and F. Janssen, "DeepRED – Rule Extraction from Deep Neural Networks," *Springer International Publishing*, 2016.
- [20] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.
- [21] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.
- [22] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning, December 2014*, 2014.
- [23] Y. Zhang and B. C. Wallace, "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification," presented at the *International Joint Conference on Natural Language Processing*, 11/1/2017, 2017.
- [24] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," presented at the *North American Chapter of the Association for Computational Linguistics*, 10/11/2018, 2018. [Online]. Available: <http://arxiv.org/pdf/1810.04805.pdf>.
- [25] Z. Zhou, H. Guan, M. M. Bhat, and J. Hsu, "Fake News Detection via NLP is Vulnerable to Adversarial Attacks," presented at the *International Conference on Agents and Artificial Intelligence*, 1/5/2019, 2019. [Online]. Available: <https://arxiv.org/pdf/1901.09657.pdf>.
- [26] M. Carter, M. Tsikerdekis, and S. Zeadally, "Approaches for Fake

- Content Detection: Strengths and Weaknesses to Adversarial Attacks," *IEEE Internet Computing*, vol. 25, no. 2, pp. 73-83, 2020.
- [27] M. Behjati, S.-M. Moosavi-Dezfooli, M. S. Baghshah, and P. Frossard, "Universal adversarial attacks on text classifiers," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019: IEEE, pp. 7345-7349.
- [28] S. Garg and G. Ramakrishnan, "BAE: BERT-based Adversarial Examples for Text Classification," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6174-6181.
- [29] Z. Zhao, D. Dua, and S. Singh, "Generating Natural Adversarial Examples," in *International Conference on Learning Representations*, 2018.
- [30] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial Attacks on Deep-learning Models in Natural Language Processing: A Survey," *acm transactions on intelligent systems and technology*, vol. 11, no. 3, pp. 1-41, 4/1/2020 2020, doi: 10.1145/3374217.
- [31] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *MILCOM 2016-2016 IEEE Military Communications Conference*, 2016: IEEE, pp. 49-54.
- [32] Y. Zhang, Y. Song, J. Liang, K. Bai, and Q. Yang, "Two Sides of the Same Coin: White-box and Black-box Attacks for Transfer Learning," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 2989-2997.
- [33] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, 2017, pp. 506-519.
- [34] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou, "HotFlip: White-Box Adversarial Examples for Text Classification," presented at the *Meeting of the Association for Computational Linguistics*, 7/1/2018, 2018. [Online]. Available: <https://www.aclweb.org/anthology/P18-2006.pdf>.
- [35] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," in *IJCAI*, 2018.
- [36] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *arXiv preprint arXiv:1605.07277*, 2016.
- [37] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *International Conference on Machine Learning*, 2017: PMLR, pp. 3145-3153.
- [38] I. Rahwan *et al.*, "Machine behaviour," *Nature*, vol. 568, no. 7753, pp. 477-486, 2019.
- [39] B. Kim, R. Khanna, and O. O. Koyejo, "Examples are not enough, learn to criticize! criticism for interpretability," *Advances in neural information processing systems*, vol. 29, 2016.
- [40] C. Molnar, *Interpretable machine learning*. Lulu. com, 2020.
- [41] D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, and G.-Z. Yang, "XAI—Explainable artificial intelligence," *Science Robotics*, vol. 4, no. 37, 2019.
- [42] R. Maheshwary, S. Maheshwary, and V. Pudi, "A Strong Baseline for Query Efficient Attacks in a Black Box Setting," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 8396-8409.
- [43] S. Eger *et al.*, "Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems," presented at the *North American Chapter of the Association for Computational Linguistics*, 2/25/2019, 2019. [Online]. Available: https://research-information.bris.ac.uk/ws/files/225517842/N19_1165.pdf.
- [44] R. Maheshwary, S. Maheshwary, and V. Pudi, "Generating natural language attacks in a hard label black box setting," in *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, 2021.
- [45] L. Li, R. Ma, Q. Guo, X. Xue, and X. Qiu, "BERT-ATTACK: Adversarial Attack against BERT Using BERT," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 6193-6202.
- [46] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," presented at the *International Conference on Learning Representations*, 1/1/2015, 2015.
- [47] J. Ebrahimi, D. Lowd, and D. Dou, "On Adversarial Examples for Character-Level Neural Machine Translation," presented at the *International Conference on Computational Linguistics*, 6/1/2018, 2018.

Yidong Chai received the Ph.D. in Tsinghua University, China. He is currently a professor in the School of Management, Hefei University of Technology and the Key Laboratory of Process Optimization and Intelligence Decision Making, Minister of Education, Hefei, China. His research interests include machine learning, cybersecurity, business intelligence and health informatics. His research has been published in *MIS Quarterly*, *IEEE Transactions on Dependable and Secure Computing*, *Journal of Management Information Systems*, *Information Processing, and Management*, *Knowledge-Based Systems*, *Applied Soft Computing*, among others.

Ruicheng Liang received a B.S. degree in Engineering Management from the Anhui University of Finance and Economics, China, in 2018. He is currently pursuing his Ph.D. degree in Management Science and Engineering at the Hefei University of Technology. His research interests include machine learning, cybersecurity, and business intelligence. His research has been published in *ACM Transactions on Knowledge Discovery from Data*.

Hongyi Zhu received his Ph.D. from the University of Arizona. He is currently an Assistant Professor at the Department of Information Systems and Cybersecurity in Carlos Alvarez College of Business at the University of Texas at San Antonio. His research interests include deep learning, cybersecurity, mobile health, and business analytics. He has published in *MIS Quarterly*, *Journal of Management Information Systems*, *ACM Transactions on Privacy and Security*, *IEEE Intelligent Systems*, *ACM Transactions on Management Information Systems*, and others.

Sagar Samtani received his Ph.D. from the University of Arizona. He is currently an Assistant Professor and Grant Thornton Scholar in the Department of Operations and Decision Technologies at Indiana University, Bloomington. His research interests include AI-enabled cybersecurity analytics, AI risk management, open-source software security, and mental health intelligence. He has published nearly 50 peer-reviewed articles in venues such as *MIS Quarterly*, *Journal of Management Information Systems*, *ACM Transactions on Privacy and Security*, *ACM Digital Threats: Research and Practice*, *IEEE Intelligent Systems*, *IEEE S&P*, *ACM KDD*, *IEEE ICDM*, and others. He has also received funding from the US NSF SaTC, SFS, and CICI programs.

Meng Wang (Fellow, IEEE) received the B.E. and Ph.D. degrees from the Special Class for the Gifted Young, Department of Electronic Engineering and Information Science, University of Science and Technology of China (USTC), Hefei, China, in 2003 and 2008, respectively. He is a professor at the Hefei University of Technology. His current research interests include multimedia content analysis, computer vision, and pattern recognition. He has authored more than 200 book chapters, and journal and conference papers in these areas. He received the ACM SIGMM Rising Star Award 2014. He is an associate editor of the *IEEE Transactions on Knowledge and Data Engineering* and the *IEEE Transactions on Circuits and Systems for Video Technology*.

Yezheng Liu received the Ph.D. degree in management science and engineering from the Hefei University of Technology, in 2001. He is a professor of electronic commerce with the Hefei University of Technology. His main research interests include data mining, decision science, electronic commerce, and intelligent decision support systems. His current research focuses on big data analytics, online social network, personalized recommendation system, and outlier detection. He is the author and coauthor of numerous papers in scholarly journals, including *Marketing Science*, *Decision Support Systems*, *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, *ACM Transactions on Knowledge Discovery from Data*, and others.

Yuanchun Jiang received his Ph.D. in management science and engineering from Hefei University of Technology, Hefei, China. He is currently a Professor at the School of Management, Hefei University of Technology. His research interests include electronic commerce, online marketing, and data mining. He has published papers in *Marketing Science*, *European Journal of Operational Research*, *IEEE Transactions on Knowledge and Data Engineering*, *IEEE Transaction on Software Engineering*, *ACM Transactions on Knowledge Discovery from Data*, and others.