Machine Learning
https://doi.org/10.1007/s10994-024-06539-6

Reversible jump attack to textual classifiers with modification reduction

Mingze Ni[1] · **Zhensu Sun**[2] · **Wei Liu**[1]

Received: 18 January 2023 / Revised: 19 February 2024 / Accepted: 11 March 2024

Abstract
Recent studies on adversarial examples expose vulnerabilities of natural language processing models. Existing techniques for generating adversarial examples are typically driven by deterministic hierarchical rules that are agnostic to the optimal adversarial examples, a strategy that often results in adversarial samples with a suboptimal balance between magnitudes of changes and attack successes. To this end, in this research we propose two algorithms, Reversible Jump Attack (RJA) and Metropolis–Hasting Modification Reduction (MMR), to generate highly effective adversarial examples and to improve the imperceptibility of the examples, respectively. RJA utilizes a novel randomization mechanism to enlarge the search space and efficiently adapts to a number of perturbed words for adversarial examples. With these generated adversarial examples, MMR applies the Metropolis–Hasting sampler to enhance the imperceptibility of adversarial examples. Extensive experiments demonstrate that RJA-MMR outperforms current state-of-the-art methods in attack performance, imperceptibility, fluency and grammar correctness.

Keywords Textual attack · Adversarial learning · Natural language processing

Editor: Lijun Zhang.

✉ Wei Liu
wei.liu@uts.edu.au

Mingze Ni
mingze.ni@student.uts.edu.au

Zhensu Sun
sunzhs@shanghaitech.edu.cn

[1] School of Computer Science, University of Technology Sydney, 15 Broadway, Sydney, NSW 2007, Australia

[2] School of Information Science and Technology, ShanghaiTech University, 393 Middle Huaxia Road, Shanghai 201210, China

Published online: 22 April 2024

1 Introduction

NLP models are known to be vulnerable in various applications, including machine translation (Ni et al., 2022; Cheng et al., 2020; Tan et al., 2020), sentiment analysis (Zang et al., 2020; Yang et al., 2021), and text summarization (Cheng et al., 2020). Attackers can exploit these weaknesses, creating adversarial examples that compromise the performance of targeted NLP systems. This growing susceptibility presents significant security challenges for AI models.

Textual attacks on NLP models are classified into character (Iyyer et al., 2018b; Ribeiro et al., 2018), word (Alzantot et al., 2018; Jia et al., 2019), and sentence-level (Jia & Liang, 2017) attacks. Character-level attacks are easily countered due to noticeable misspellings (Ebrahimi et al., 2018), while sentence-level attacks often yield complex, hard-to-read text (Gan & Ng, 2019). Word-level attacks are gaining preference for their effectiveness and subtlety, as they involve replacing words with carefully chosen substitutes (Zhang et al., 2020; Garg & Ramakrishnan, 2020; Liet al., 2020). Consequently, our focus is on conducting word-level adversarial attacks.

Crafting optimal adversarial examples involves navigating the interplay of successful attacks, controlled imperceptibility. The predominant strategies for this can be classified into optimization algorithms and hierarchical search methods. Within the realm of optimization, Genetic Attack (GA) (Alzantot et al., 2018; Jia et al., 2019) and Particle Swarm Optimization (PSO) (Zang et al., 2020) stand out as evolutionary approaches, focusing on optimizing attack effectiveness within embedding spaces and sememe-based thesauri, respectively. However, these methods face two primary challenges: 1) low efficiency in the optimization process due to the expansive search space, such as GloVe (Pennington et al., 2014), and 2) Compromised semantic integrity, as even synonym-based word substitutions can cause sentence-level semantics inconsistency. On the other hand, Hierarchical search crafts adversarial examples by orderly substituting words based on word saliency rank (WSR) (Ren et al., 2019; Li et al., 2021; Yang et al., 2021). It first identifies target words using WSR, then employs a Masked Language Model or thesaurus for substitutions. These hierarchical attacking methods have several drawbacks: 1) the first drawback of this approach is the difficulty of presetting the number of perturbed words (NPW) for large datasets with many tokens since the optimal NPW varies with different target texts (Michel et al., 2019); 2) the WSR-based methods will significantly reduce the searching domain by only attacking the combination of victim words ordered by the WSR. For a clear illustration, Fig. 1 showcases the drawbacks of optimization-based GA and hierarchical PWWS attacks. GA's replacement of 'thriller' with 'science' sacrifices semantic quality, while PWWS, despite altering three words, fails to fool the classifier.

To address the above problems, we propose two novel black-box and word-level antagonistic algorithms: Reversible Jump Attacks (RJA) and MH Modification Reduction (MMR). For RJA, we employ the Reversible Jump sampler (RJS) and propose three variables from a target distribution: the number of perturbed words (NPW), victim words, and substitutions from Masked Language Models (MLM) and HowNet (Dong & Dong, 2003). The target distribution for RJS for evaluating the quality of the adversarial candidates is regularized by a strong penalty of semantic (dis)similarity. The NPW can be cross-dimensionally searched via RJS to adjust for different textual inputs according to their word saliency and overall performance. Given these three factors, adversarial candidates are only accepted based on an acceptance probability from RJS. By running such a process iteratively, we will obtain the successful candidates with the highest

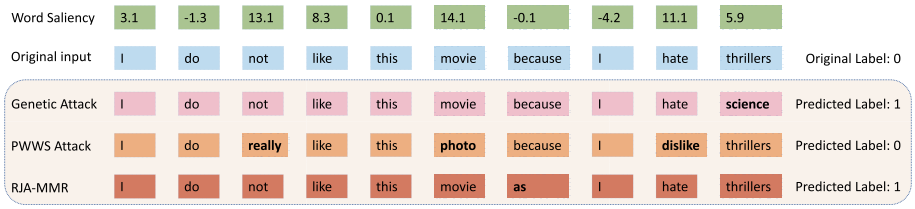

Fig. 1 An illustrating example to show attack performances of optimizing attack (genetic attack), PWWS attack, and the proposed method RJA-MMR, where label "0" represents negative sentiment and "1" represents positive sentiment. The substitutions for different attack methods are bold. Genetic attack sacrifices too much semantics by changing "thrillers" to "science", while PWWS fails to fool the model and makes many ineffective modifications. The proposed method, RJA-MMR, makes a successful attack with only one word changed

semantic similarity. Therefore, RJA efficiently searches threat-level attacks inside a domain larger than WSR without presetting an NPW and sacrificing much semantics for imperceptibility.

The other algorithm is Metropolis–Hasting Modification Reduction (MMR) which tends to restore the manipulations from RJA (i.e., reverse back to the original words) and then update the existing substitutions to maintain the attacking performance. Specifically, given an adversarial candidate, MMR first stochastically proposes a new candidate by restoring the attacked words. It applies a customized acceptance probability, calculated by comparing the overall performance between the new and current candidates, to determine the acceptance of the new candidate. After restoring some attacked words, MMR uses MH algorithm to update the substitutions of the current attacked words to preserve the attacking performance. By combining RJA and MMR, we proposed an integrated RJA-MMR as our final model. Specifically, RJA utilizes a Reverse Jump sampler (Green, 1995b), a Markov Chain Monte Carlo (MCMC) family member, to sample the dimensional jumping vectors to perform a cross-dimensional search for the optimal attacking performance constrained by semantic similarity. Intuitively, RJA and MMR agree on attacking performance improvement but disagree on NPW. By iteratively running these two antagonistic algorithms, attackers can boost the attack performance with only a small number of perturbations. The attack performance is illustrated by an example in Fig. 1, where RJA-MMR outperforms the optimizing attack (Genetic attack) and hierarchical attack (PWWS).

Our main contributions from this work are as follows:

- We design a highly effective adversarial attack method, Reversible Jump Attack (RJA), which utilizes the Reversible Jump algorithm to generate adversarial examples with an adaptive number of perturbed words. The algorithm enables our attack method to have an enlarged search domain by jumping across the dimensions.
- We propose Metropolis–Hasting Modification Reduction (MMR), which applies Metropolis–Hasting (MH) algorithm to construct an acceptance probability and use it to restore the attacked victim words to improve the imperceptibility with attacking performance reserved. MMR is functional with RJA and empirically proven effective in the adversarial examples generated by other attacking algorithms.
- We evaluate our attack method on real-world public datasets. Our results show that methods achieved the best performance in terms of attack performance, imperceptibility and examples' fluency.

The rest of this paper is structured as follows. We first review adversarial attacks for NLP models and the Markov Chain Monte Carlo methods in NLP in Sect. 2. Then we detail our proposed method in Sect. 3. We evaluate the performance of the proposed method through empirical analysis in Sect. 4. We conclude the paper with suggestions for future work in Sect. 5.

2 Related work

This section reviews the literature on word-level textual attacks and MCMC sampling in NLP.

2.1 Word-level attacks to classifiers

An increasing amount of effort is devoted to generating better textual adversarial examples with various attack models. Character-level attacks (Liang et al., 2018; Ebrahimi et al., 2018) use misspellings to attach the victim classifiers; however, these attacks can often be defended by a spell checker. At the same time, sentence-level attacks (Iyyer et al., 2018b; Zou et al., 2020) pose threats to the classifier via inserting, removing, and paraphrasing sentences or pieces of sentences to the original input, while it's difficult for the generated text to maintain the imperceptibility (Li et al., 2021). Word-level attacks pose non-trivial threats to NLP models by locating important words and manipulating them for targeted or untargeted purposes. Such attacks are broadly regarded as the optimal unit of attacks (Jia & Liang, 2017).

2.1.1 Gradient-based word-level attacks

With the help of an adopted fast gradient sign method (FGSM) (Goodfellow et al., 2015; Papernot et al., 2016) were the first to generate word-level adversarial examples to classifiers. While their attack was able to fool the classifiers, their word-level manipulations significantly affected the original meaning. In Liang et al. (2018), the authors proposed to attack the target model by inserting Hot Training Phrases (HTPs) and modifying or removing the Hot Sample Phrases (HSPs), where HTPs and HSPs are calculated based on the gradient with respect to words from the input. Similar to Liang, Samanta & Mehta (2018) utilizes the embedding gradient to determine the important words. Then hierarchical-driven rules together with hand-crafted word-level synonyms and character-level typos were designed. Notably, while the textual data is naturally discrete and more perceptible than image data, many gradient-based textual attacking methods inherited from computer vision are not effective enough, which leaves textual attack a challenging problem.

2.1.2 Non-gradient-based word-level attacks

Alzantot et al. (2018) transferred the domain of adversarial attacks to an optimization problem by formulating a customized objective function. With genetic optimization, they generate the adversarial examples by sampling the qualified genetic 'son' generations that break out the encirclement of the semantic threshold. However, the genetic algorithm can be low efficient. Since word embedding space is sparse, performing natural selection for languages in such a space can be computationally expensive. Jia et al. (2019) proposed a

faster version of Alzantot's adversarial attacks by shrinking the search space, which accelerates the process of evolving in genetic optimization. Although Jia has greatly reduced the computational expense of genetic-based optimization algorithms, the optimizing processes inside word embedding space, such as GloVe (Pennington et al., 2014) and Word2Vec (Mikolov et al., 2013), are still not efficient enough. To ease the searching process, embedding-based algorithms have to use a counter-fitting method to post-process attacker's vectors to accelerate the searching speed (Mrksic et al., 2016). Compared with the word embedding method, utilizing well-organized linguistic thesaurus, e.g., synonym-based WordNet (Miller et al., 1990) and sememe-based HowNet (Dong & Dong, 2003), is a simple and easy implementation. Ren et al. (2019) sought synonyms based on WordNet synsets and ranked word replacement order via probability-weighted word saliency (PWWS). Zang et al. (2020) and Yang et al. (2021) both manifested that the sememe-based HowNet can provide more substitute words via Particle Swarm Optimization (PSO) and an adaptive monotonic heuristic search to determine which group of words should be attacked. In addition, some recent studies utilized masked language models (MLM), such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), to generate contextual perturbations (Liet al., 2020; Garg & Ramakrishnan, 2020). The pre-trained MLMs can ensure the predicted token correctly fits the sentence grammar but cannot preserve semantics.

2.2 Markov chain Monte Carlo in NLP

Markov chain Monte Carlo (MCMC) (Metropolis et al., 1953a), a statistically generic method for approximate sampling from an arbitrary distribution, can be applied in a variety of fields, such as optimization (Rubinstein, 1999), machine learning (Fan et al., 2018), quantum simulation (Haase et al., 2021) and icing models (Herrmann, 1986). The main idea is to generate a Markov chain whose equilibrium distribution is equal to the target distribution (Kroese et al., 2011). There exist various algorithms for constructing chains, including the Gibbs sampler, Reversible Jump sampler (Green, 1995b), and Metropolis–Hasting (MH) algorithm (Metropolis et al., 1953a). To get models capable of reading, deciphering, and making sense of human languages, NLP researchers apply MCMC to many downstream tasks, such as text generation and sentimental analysis. For text generation, Kumagai et al. (2016) proposes a probabilistic text generation model which generates human-like text by inputting semantic syntax and some situational content. Since human-like text requests grammarly correct word alignment, they employed Monte Carlo Tree Search to optimize the structure of the generated text. In addition, Harrison et al. (2017) presents the application of MCMC for generating a story, in which a summary of movies is produced by applying recurrent neural networks (RNNs) to summarize events and directing the MCMC search toward creating stories that satisfy genre expectations. For sentimental analysis, Kang and Ren (2011) applies the Gibbs sampler to the Bayesian network, a network of connected hidden neurons under prior beliefs, to extract the latent emotions. Specifically, they apply the Hidden Markov models to a hierarchical Bayesian network and embed the emotional variables as the latent variable of the Hidden Markov model.

2.2.1 Metropolis–Hasting and reversible jump samplers

The Metropolis–Hasting (MH) (Metropolis et al., 1953a) algorithm is a classical Markov chain Monte Carlo sampling approach. Given the stationary distribution $f(\mathbf{z})$ and transition proposal $q(\mathbf{z}'|\mathbf{z})$, the MH algorithm can generate desirable examples from $f(\mathbf{z})$. Specifically,

at each iteration, a new state $\mathbf{z}'$ will be proposed given the current state $\mathbf{z}$ based on a transition function $q(\mathbf{z}'|\mathbf{z})$. The MH algorithm is based on a "trial-and-error" strategy by defining an acceptance probability $\alpha(\mathbf{z}'|\mathbf{z})$ as following:

$$\alpha(\mathbf{z}'|\mathbf{z}) = \min\left\{\frac{f(\mathbf{z}')q(\mathbf{z} \mid \mathbf{z}')}{f(\mathbf{z})q(\mathbf{z}' \mid \mathbf{z})}, 1\right\} \tag{1}$$

to decide whether the new state $\mathbf{z}'$ is accepted or rejected.

MCMC can also be applied to sample variational dimension sampling. Reversible Jump samplers (RJS) (Green, 1995b) is a variation of MCMC algorithms specifically designed to sample from target distributions that contain vectors with different dimensions. Due to such a property, RJS can be applied to variable selection (Fan & Sisson, 2011), dimension reduction (Rincent et al., 2017), and cross-dimensional optimization (Kroese et al., 2011). Unlike the MH algorithm, RJS requests an additional transition item for proposing the new dimensions. The formulation of the acceptance probability of RJS is below:

$$\alpha(\mathbf{z}'_{(m')}|\mathbf{z}_{(m)}) = \min\left\{\frac{f(\mathbf{z}'_{(m')})q(\mathbf{z}_{(m)} \mid \mathbf{z}'_{(m')})}{f(\mathbf{z}_{(m)})q(\mathbf{z}'_{(m')} \mid \mathbf{z}_{(m)})}, 1\right\} \tag{2}$$

$$q\left(\mathbf{z}'_{(m')}|\mathbf{z}_{(m)}\right) = p\left(\mathbf{z}'_{(m')}|m', \mathbf{z}_{(m)}\right)p\left(m'|\mathbf{z}_{(m)}\right), \tag{3}$$

where m denotes the dimensions of the vector $\mathbf{z}_{(m)}$, $q\left(\mathbf{z}'_{(m')}|\mathbf{z}_{(m)}\right)$ in Eq. (3) illustrates the new transition function and $p\left(m'|\mathbf{z}_{(m)}\right)$ is the dimensional transition item. Comparing the acceptance probabilities of MH (Eq. 1) and RJS (Eq. 2) reveals that RJS is more effective than MH in handling dimensional variations and sampling parameters of unknown dimensions. Since making adversarial would be a typical situation of dimension variation due to number of perturbed words (NPW), we believe that attacks based RJS is expected to achieve better performance than the literature based on MH (Zhang et al., 2019).

2.2.2 Adversarial attack via MCMC

Despite the applications in NLP, the MCMC can be applied to adversarial attacks on NLP models. Zhang et al. (2019) has successfully applied MH sampling to generate fluent adversarial examples for natural language by proposing gradient-guided word candidates. Specifically, they proposed both black-box and white-box attacks, and for black-box attacks, they perform removal, insertion and replacement by the words chosen from the pre-selector candidates set, but the empirical studies indicate these candidates are not efficient and effective for attacking. As for the white-box attacks, the gradient of the victim model is introduced to score the pre-selector candidates set, which successfully improves the attacking performance. However, the white-box setting is not practical in the real world, as attackers do not have access to the gradient and structure of the victim models. In addition, MHA successfully improved the language quality in terms of fluency, but the imperceptibility of the generated examples, especially in the modification rate, cannot be optimized.

Table 1 List of notations used in this research

Notation	Description
$\mathcal{X}$	Text sample space
$\mathcal{Y}$	Class space
D	A dataset to be attacked
$x = [w_1, w_2, \ldots, w_n]$	An input text with n words and w_i is the ith word in the sequence
$\mathbf{x}$	An adversarial candidate generated by RJA
m, $\mathbf{v}$, $\mathbf{s}$	Three factors in adversarial sample generation: the number of perturbed words, victim words, and their substitutions, respectively
$\mathbb{G}$	The set of substitution candidates
$\mathbf{x}^r$	The adversarial candidate generated in the restoring step of MMR
$\mathbf{x}^u$	The adversarial candidate generated in the updating step of MMR
$\mathbf{x}^*$	The final optima adversarial example
$I(w_i)$	The saliency of the word w_i
T	The total number of iterations for RJA-MMR
$F(\cdot) : \mathcal{X} \rightarrow \mathcal{Y}$	The victim classifier
$Sem(\cdot) : \mathcal{X}^2 \rightarrow (0, 1)$	The function measuring the semantic similarity
$p(\mathbf{x}_{t+1}\|\mathbf{x}_t) : \mathcal{X} \rightarrow (0, 1)$	The transition function from state $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$
$\pi(x) : \mathcal{X} \rightarrow (0, 1)$	Target distribution
$\alpha(\mathbf{x}_{t+1}\|\mathbf{x}_t) : \mathcal{X} \rightarrow (0, 1)$	The acceptance probability

3 Imperceptible adversarial attack via Markov chain Monte Carlo

In this section, we will detail our proposed method, RJA-MMR, the Reversible Jump attacks (RJA) with Metropolis–Hasting Modification Reduction (MMR).

3.1 Problem formulation and notaition

Given a pre-trained text classification model, which maps from feature space $\mathcal{X}$ to a set of classes $\mathcal{Y}$, an adversary aims to generate an adversarial document $\mathbf{x}^*$ from a legitimate document $x \in \mathcal{X}$ whose ground truth label is $y \in \mathcal{Y}$, so that $F(\mathbf{x}^*) \neq y$. The adversary also requires $Sem(x, \mathbf{x}^*) \leq \epsilon$ for a domain-specific semantic similarity function $Sem(\cdot) : \mathcal{X} \times \mathcal{X} \rightarrow (0, 1)$, where the bound $\epsilon \in \mathbb{R}$ helps to ensure imperceptibility. In other words, in the context of text classification tasks, we use $Sem(x, \mathbf{x}^*)$ to capture the semantic similarity between x and $\mathbf{x}^*$. More details of the notation are illustrated in Table 1.

3.2 Reversible jump attack

This section details our proposed Reversible Jump Attack (RJA) which generates adversarial examples under semantic regularisation. Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ denote a dataset with N data samples, where x and y are the input text and its corresponding class. Given the input text $x = [w_1, \ldots, w_i, \ldots, w_n]$ with n words, we denote an adversarial candidate of RJA as $\mathbf{x}$ and denote the final chosen adversarial example as $\mathbf{x}^*$.

RJA, unlike traditional methods, treats the number of perturbed words (NPW) as a variable in the sampling process, not a preset value. Utilizing the Reversible Jump Sampler, RJA conditionally samples NPW, victim words, and their substitutions. The approach involves a transition function that proposes adversarial candidates, evaluated against a target distribution focusing on attack effectiveness and semantic similarity (Eq. 2). This process iteratively refines the adversarial examples, guided by an acceptance probability mechanism.

This section first presents the transition function (Sect. 3.2.1) and then elaborates on the acceptance probability (Sect. 3.2.2), which builds upon the transition function.

3.2.1 Transition function

To propose the adversarial candidates, we construct our transition function to sequentially propose the three compulsory factors of crafting a new adversarial candidate $\mathbf{x}_{t+1}$ given the current one $\mathbf{x}_t$: the NPW m, the victim words $\mathbf{v} = [v_1, \ldots, v_m]$, and the corresponding substitutions $\mathbf{s} = [s_1, \ldots, s_m]$, where the dimension of $\mathbf{v}$ and $\mathbf{s}$ is m. Before we detail the process of proposing these factors, we first introduce the concept of the word saliency. In this context, word saliency refers to the impact of the word w_i on the output of the classifier and the transition function, if this word is deleted from the sentence. The word with a high saliency has a high impact on the classifier. Thus, associating more importance to high-saliency words can help the transition function efficiently propose a high-quality adversarial candidate. To calculate the word saliency, we use the changes of victim classifiers' logits before and after deleting word w_i to represent the saliency $I(w_i)$:

$$I(w_i) = F_{logit}(x) - F_{logit}(x \backslash w_i), \tag{4}$$

where $F_{logit}(\cdot)$ is the classifier returning the logit of the correct class, and $x \backslash w_i = [w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n]$ is the text with w_i removed. We calculate the word saliency $I(w_i)$ for all $w_i \in x$ to obtain word saliency $\mathbf{I}(x)$. Calculating the word saliency is illustrated in Block ① of Fig. 2.

Among the iterations of searching for victim words, assume the RJA adversarial candidate at iteration t is $\mathbf{x}_t = (m_t, \mathbf{v}_t, \mathbf{s}_t)$ and the new adversarial candidate to be crafted is $\mathbf{x}_{t+1} = (m_{t+1}, \mathbf{v}_{t+1}, \mathbf{s}_{t+1})$, we propose the first factor, the NPW value m_{t+1}, by either adding or subtracting 1, i.e., $m_{t+1} \in \{m_t + 1, m_t - 1\}$. This set $\{m_t + 1, m_t - 1\}$ does not need to include m_t because if the proposed state is rejected, m_{t+1} will be retained as m_t, which means m_t still remains as a possible state. Thus the transition function for the new NPW value m_{t+1} can be formulated as a probability mass function as below:

$$p(m_{t+1}|\mathbf{x}_t) = \begin{cases} \dfrac{\exp(l_1)}{\exp(l_1) + \exp(l_2)} & m_{t+1} = m_t - 1, \\ \dfrac{\exp(l_2)}{\exp(l_1) + \exp(l_2)} & m_{t+1} = m_t + 1, \end{cases} \tag{5}$$
$$\text{where} \quad l_1 = \sum_{w_i \in \mathbf{v}_t} I(w_i), \quad l_2 = \sum_{w_i \notin \mathbf{v}_t} I(w_i).$$

Such a transition function can propose the new state $m_{t+1} \in \{m_t - 1, m_t + 1\}$ by referring to the proportion of the exponential on victim word saliency l_1 and unattacked word saliency l_2 overall word saliency exponential. Intuitively, if the saliency values of all attacked words are high, the probability of proposing to reduce one attacked word, $m_{t+1} = m_t - 1$, is

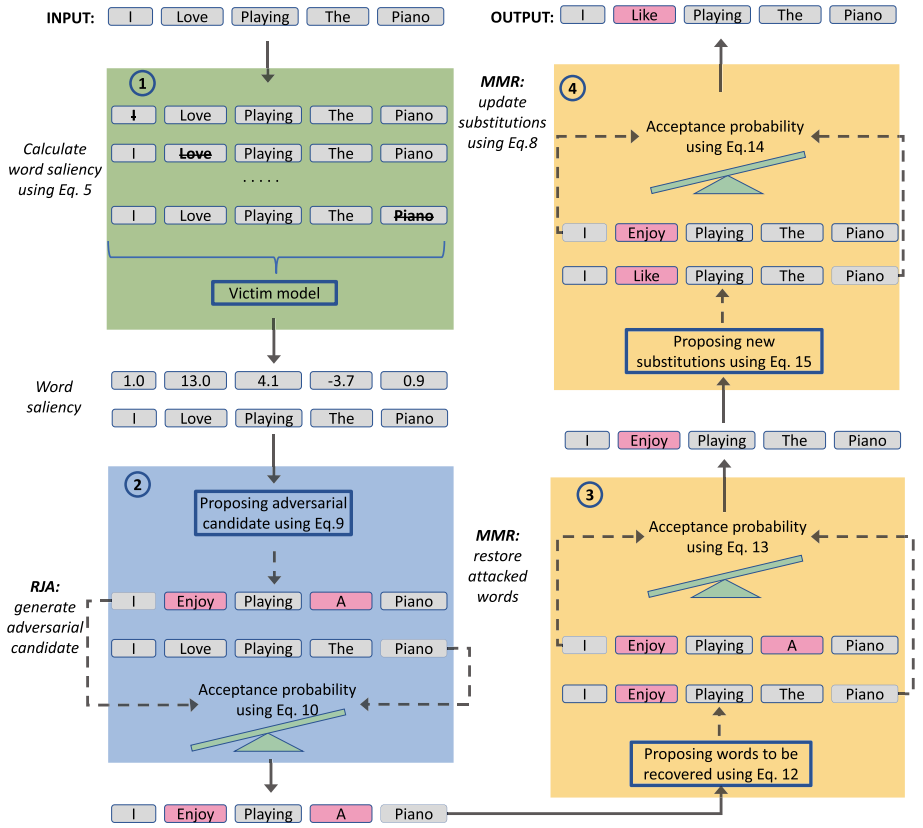

Fig. 2 The workflow of our RJA-MMR. In this example, HAA generates an adversarial example with one word perturbed to attack a sentimental classifier with two labels (positive and negative). The block ① shows the calculation of word saliency. After obtaining the word saliency, we perform RJA in block ② which reflects the lines 4–15 in Algorithm 1. After RJA, we perform the two steps, restoring and updating MMR in block ③ and ④, respectively. The block ③ and ④ are illustrated in lines 4–10 and lines 11–18 in Algorithm 2, respectively

high, and vice versa. Concretely, to sample m_{t+1} from such a transition function, we firstly draw a random number, $\eta \sim Unif(0, 1)$; and if η is less than the probability of sampling $m_{t+1} = m_t - 1$, i.e., $\eta < \frac{\exp(l_1)}{\exp(l_1)+\exp(l_2)}$, then $m_{t+1} = m_t - 1$, otherwise $m_{t+1} = m_t + 1$. Unlike hierarchical attacks, which deterministically perturb the words in the descending order of the word saliency, randomization is applied because of its two merits: 1) it overcomes the imprecision problem with the WSR (word saliency rank) mentioned in the preceding introduction section, and 2) it enlarges the search domain by proposing more combinations of attacked words than those in hierarchical searching.

After determining the number of perturbed words, we sample one target victim word v_{tgt} (where "tgt" refers to "target") to be manipulated according to the newly sampled m_{t+1}. Specifically, for $m_{t+1} = m_t + 1$, the target word v_{tgt} is uniformly sampled from unattacked word set $x \backslash \mathbf{v}_t$, while for $m_{t+1} = m_t - 1$ the target word v_{tgt} is uniformly drawn from attacked words set $\mathbf{v}_t$ then the selected words will be restored to the original words. The transition function of sampling the target victim word v_{tgt} is thus formulated as:

$$p(v_{tgt}|\mathbf{x}_t, m_{t+1}) = \begin{cases} \frac{1}{m_t} & v_{tgt} \in \mathbf{v}_t \quad \text{if } m_{t+1} = m_t - 1, \\ \frac{1}{n-m_t} & v_{tgt} \in \mathbf{x} \backslash \mathbf{v}_t \quad \text{if } m_{t+1} = m_t + 1. \end{cases} \tag{6}$$

After the target word $v_{tgt} \in \mathbf{x}_t$ is selected, we search for a parsing-fluent and semantic-preserving substitution for w_{tgt}. Therefore, we uniformly draw a substitution s_{tgt} for v_{tgt} from the candidates set, which is the intersection (consensus) of candidates provided by Mask Language Models (MLMs) and Synonyms. Specifically, let $\mathcal{M}$ denote the MLM, and we mask the v_{tgt} in $\mathbf{x}$ to construct a masked $\mathbf{x}_{mask}$ and feed the masked text into $\mathcal{M}$ to search for the parsing-fluent candidates. Instead of using the argmax prediction, we take the most possible K words, which are the top K words suggested by the logits from $\mathcal{M}$, to construct MLM candidates set $\mathbb{G}_{\mathcal{M}} = \{w^1_{\mathcal{M}}, \ldots, w^K_{\mathcal{M}}\}$. To keep semantically similar, we form a synonym set $\mathbb{G}_{syn} = \{w^1_{syn}, \ldots, w^K_{syn}\}$ from HowNet (Dong et al., 2010) based thesauri such as OpenHowNet (Qi et al., 2019) and BabelNet (Qi et al., 2020) These thesauri are context-aware and at the same time can provide more synonyms than common thesaurus such as WordNet (Miller, 1992). Since our objective is that the generated adversarial examples should be parsing-fluent and semantic-preserving, the substitution s_{tgt} will be uniformly sampled from the intersection $\mathbb{G} = \mathbb{G}_{\mathcal{M}} \cap \mathbb{G}_{syn}$, which is illustrated in Eq. (7).

$$p(s_{tgt}|w_{tgt}, m_{t+1}, \mathbf{x}_t) = \frac{1}{[\mathbb{G}]} \tag{7}$$

where $\mathbb{G} = \mathbb{G}_{\mathcal{M}} \cap \mathbb{G}_{syn}$ and $[\mathbb{G}]$ is the cardinality of the set $\mathbb{G}$.

By applying the Bayes rule to the Eqs. (5), (6) and (7), the final transition function is:

$$p_{RJA}(\mathbf{x}_{t+1}|\mathbf{x}_t) = p(m_{t+1}|\mathbf{x}_t)p(w_{tgt}|m_{t+1}, \mathbf{x}_t)p(s_{tgt}|w_{tgt}, m_{t+1}, \mathbf{x}_t) \tag{8}$$

3.2.2 Acceptance probability for RJA

Before we calculating the acceptance probability, we need to construct the target distribution for evaluating the performance. Specifically, we argue that a good adversarial example should achieve successful attacks while being kept semantically similar to the input text x. Therefore, we formulate the following equation as our target distribution:

$$\pi(\mathbf{x}) = \frac{\big(1 - F_p(\mathbf{x})\big)Sem(x, \mathbf{x})}{C}, \tag{9}$$

where $Sem(x, \mathbf{x})$ represents the semantic similarity, which generally is implemented with the cosine similarity between sentence encodings from a pre-trained sentence encoder, such as USE (Cer et al., 2018). $C = \sum_{\mathbf{x} \in \mathcal{X}} \big(1 - F_p(\mathbf{x})\big)Sem(x, \mathbf{x})$ is a positive normalizing factor to make $\sum_{\mathbf{x} \in \mathcal{X}} \pi(\mathbf{x}) = 1$ and $F_p(\cdot) : \mathcal{X} \rightarrow (0, 1)$ denotes the confidence of making right predictions where $\mathcal{X}$ represents text space. From Eq. (9), we can easily observe that the value from target distribution $\pi(\mathbf{x})$ will increase with the increase of the attacking performance measured by the confidence of making a wrong prediction $1 - F_p(\mathbf{x})$, and semantic similarity $Sem(x, \mathbf{x})$.

Given the target distribution in Eq. (9) and transition function in Eq. (), we formulate the acceptance probability for RJA, $\alpha_{RJA}(\mathbf{x}_{t+1}|\mathbf{x}_t)$, as follows:

$$\alpha_{RJA}(\mathbf{x}_{t+1}|\mathbf{x}_t) = \min\left\{\frac{\pi(\mathbf{x}_{t+1})p_{RJA}(\mathbf{x}_t|\mathbf{x}_{t+1})}{\pi(\mathbf{x}_t)p_{RJA}(\mathbf{x}_{t+1}|\mathbf{x}_t)}, 1\right\} \tag{10}$$

After calculating $\alpha(\mathbf{x}_{t+1}|\mathbf{x}_t)$, we sample a random number ϵ from a uniform distribution, $\epsilon \sim Uniform(0,1)$, if $\epsilon < \alpha(\mathbf{x}_{t+1}|\mathbf{x}_t)$ we will accept $\mathbf{x}_{t+1}$ as the new state, otherwise the state will remain as $\mathbf{x}_t$. By running T iterations, we obtain a set of adversarial candidates $\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_T\}$. We then choose the candidate which not only successfully fools the classifier but also preserves the most semantics as the final adversarial candidate $\mathbf{x}$. The process of RJA is illustrated in Algorithm 1 and block ② in Fig. 2.

Algorithm 1 Reversible Jump Attack (RJA)

Input: Input text: x, Number of iterations: T
Output: Adversarial candidate $\mathbf{x}$

```
Adv_set = [ ]
x_0 = x
for t+1 in range(T) do
    Sample m_{t+1} given x_t with Eq. 5
    Sample s_{t+1} given x_t and m_{t+1} with Eq. 6
    Sample v_{t+1} given v_t, m_{t+1} and s + 1 with Eq. 7
    Craft x_{t+1} with m_{t+1}, s_{t+1}, v_{t+1} and x_{t-1},
    Calculate the acceptance probability, α(x_t|x_{t-1}) with Eq. 10
    Sample ϵ from Uniform distribution, Uniform(0,1)
    if ϵ < α(x_t|x_{t-1}) then
        x_{t+1} = x_{t+1}
        Adv_set = [Adv_set, x_{t+1}]
    else
        x_{t+1} = x_t
        Adv_set = [Adv_set, x_{t+1}]
    return Adv_set
Choose the candidate which successfully fools the classifier with lease
  semantic sacrifice as an adversarial example x.
return Adversarial candidate x
```

3.3 Modification reduction with metropolis–hasting algorithm

Besides the success of tampering with the classifier and semantic preservation, the modification rate is also an important factor in evaluating the imperceptibility of adversarial examples. Generally, methods in the literature can generate effective adversarial examples; however, it was hard to guarantee the modification rate is optimally the lowest. To address this, we introduce the Metropolis–Hasting Modification Reduction (MMR), leveraging the Metropolis–Hasting (MH) algorithm to optimize the modification rate by exploring efficient yet minimal substitution combinations for a given adversarial candidate. MMR involves two steps, each employing the MH algorithm: 1) stochastically restoring some attacked words to create a less modified candidate and 2) updating all substitutions without altering the NPW, m. These steps are detailed in Sects. 3.3.1 and 3.3.2 respectively.

3.3.1 Restoring attacked words with MMR

The first step of MMR is probabilistically restoring some attacked words with MH algorithm to test the necessity of the current substitutions. Given an adversarial candidate $\mathbf{x}_t = (m_t, \mathbf{v}_t, \mathbf{s}_t)$ from iteration t in RJA, we aim to generate an adversarial candidate $\mathbf{x}_t^r$ which is constructed by restoring some attacked words in $\mathbf{x}_t$. To sample the restored substitutions, we propose the probability mass function of selecting substitutions $s^r \in \{s_i, w_i\}$ in iteration t as follows:

$$p(s^r|\mathbf{x}_t) = \begin{cases} \dfrac{\exp(I(w_i))}{1+\exp(I(w_i))} & \text{if } s^r = s_i \text{ (continue to attack)}, \\ \dfrac{1}{1+\exp(I(w_i))} & \text{if } s^r = w_i \text{ (attack cancelled)}, \end{cases} \tag{11}$$

$$p_{restore}(\mathbf{x}_t^r|\mathbf{x}_t) = \prod_{s^r \in \mathbf{s}_t} p(s^r|\mathbf{x}_t) \tag{12}$$

where $s^r = s_i$ denotes to continue the attack and $s^r = w_i$ denotes restoring the substitution to the original word w_i, respectively. The $\mathbf{x}_t^r$ is the proposed adversarial candidate with selected substitutions restored from $\mathbf{x}$. With such a probability mass function, the s^r can be sampled by the same strategy of sampling as in Eq. (5). To further investigate the quality of such a candidate, we apply the target distribution, $\pi(\mathbf{x})$, in Eq. (9) to construct the following acceptance probability:

$$\alpha_{restore}(\mathbf{x}_t^r|\mathbf{x}_t) = \min\left(\frac{\pi(\mathbf{x}_t^r)p_{restore}(\mathbf{x}_t|\mathbf{x}_t^r)}{\pi(\mathbf{x}_t)p_{restore}(\mathbf{x}_t^r|\mathbf{x}_t)}, 1\right) \tag{13}$$

to decide whether the proposed adversarial candidate $\mathbf{x}_t^r$ should be accepted as the true candidate.

3.3.2 Updating the combination of substitutions with MMR

Having restored selected substitutions to obtain the adversarial candidate $\mathbf{x}_t^r$ at the t-th iteration, we proceed to the second step: MMR updating. This step is designed to refine attack performance by altering substitution combinations without affecting the NPW, m_t. We apply a methodology similar to the one in Eq. (7) for sampling substitution combinations. In essence, the MMR updating utilizes the candidate proposing function (Eq. 7) to explore alternative substitutions for each attacked word, aiming for enhanced attack efficacy. The formulation for this update, leading to the next adversarial candidate $\mathbf{x}_t^u$, is governed by the subsequent acceptance probability:

$$\alpha_{update}(\mathbf{x}_t^u|\mathbf{x}_t^r) = \min\left(\frac{\pi(\mathbf{x}_t^u)p_{update}(\mathbf{x}_t^r|\mathbf{x}_t^u)}{\pi(\mathbf{x}_t^r)p_{update}(\mathbf{x}_t^u|\mathbf{x}_t^r)}, 1\right), \tag{14}$$

$$p_{update}(\mathbf{x}_t^u|\mathbf{x}_t^r) = \prod_{s_i \in \mathbf{s}_t^r} p(s_i|w_i, m_t^r, \mathbf{x}_t^r), \tag{15}$$

where $p(s_i|w_i, m_t^r, \mathbf{x}_t^r)$ is identical to that in Eq. (7).

By iteratively running T times MH algorithms for substitution restoring and updating with acceptance probabilities in Eqs. (13) and (14), respectively, we can construct the adversarial set $\mathbb{X}' = \{\mathbf{x}_t^u\}_{t=1}^T$ and select the candidate with the highest semantic similarity among the successful candidates that fools the classifier as the final adversarial example $\mathbf{x}^*$. This proposed MMR algorithm will not only be applied to our RJA algorithm but also can help other attack methodologies reduce their modifications. The whole process of MMR is illustrated in Algorithm 2 and block ③–④ in Fig. 2.

Algorithm 2 Metropolis–Hasting Modification Reduction (MMR)

Input: Adversarial candidate $\mathbf{x} = (m, \mathbf{v}, \mathbf{s})$
Output: The final adversarial example $\mathbf{x}^*$
$Adv\_set = [\ \ ]$
for *t in range(T)* **do**
  Fetch $\mathbf{x}_t$ from RJA in iteration t
  Sample $\mathbf{x}_t^r$ to reduce the modifications with Eq. 11
  Calculate the acceptance probability, $\alpha(\mathbf{x}_t|\mathbf{x})$ with Eq. 13
  Sample ϵ from Uniform distribution, Uniform(0,1)
  if $\epsilon < \alpha(\mathbf{x}_t^r|\mathbf{x})$ **then**
    $\mathbf{x}_t^r = \mathbf{x}_t^r$
  else
    $\mathbf{x}_t^r = \mathbf{x}$
  Sample $\mathbf{x}_t^u$ to update the substitutions in $\mathbf{x}_t^r$ with Eq. 15
  Calculate the acceptance probability, $\alpha(\mathbf{x}_t^u|\mathbf{x}_t^r)$ with Eq. 14
  if $u < \alpha(\mathbf{x}_t^u|\mathbf{x}_t^r)$ **then**
    $\mathbf{x}_t^u = \mathbf{x}_t^u$
    Take $\mathbf{x}_t^u$ as RJA's input for next iteration
  else
    $\mathbf{x}_t^u = \mathbf{x}_t^r$
    Take $\mathbf{x}_t^u$ as RJA's input for next iteration
  $Adv\_set = [Adv\_set,\ \mathbf{x}_t^u]$
  return *Adv_set*
Choose the candidate with the least modification from *Adv_set* as the final adversarial example $\mathbf{x}^*$.
return The final adversarial example $\mathbf{x}^*$

4 Experiments and analysis

In this section, we comprehensively evaluation the performance of our method against the current state of the art. Besides the main results (Sect. 4.4) of attacking performance and imperceptibility, we also conduct experiments on ablation studies (Sect. 4.5), efficiency analysis (Sect. 4.6), transferability (Sect. 4.7), target attacks (Sect. 4.8), performance front of defense mechanism (Sect. 4.9), adversarial retraining (Sect. 4.10), part-of-speech (POS) preference (Sect. 4.11) and scales of models for robustness(Sect. 4.12)

We evaluate the effectiveness our methods on three widely-used and publicly available benchmark datasets: AG's News (Zhang et al., 2015), Emotion (Saravia et al., 2018), SST2 (Socher et al., 2013) and IMDB(Maas et al., 2011). Specifically, AG's News is a news

Table 2 Datasets and accuracy of victim models before attacks

Dataset	Size	Avg.Length	Class	Task	Model	Accuracy (%)
AG's News	12,700	37.84	4	News topics	BERT-C	94
					TextCNN	90
Emotion	20,000	19.14	6	Sentiment analysis	BERT-C	97
					TextCNN	93
SST2	9613	19.31	2	Sentiment analysis	BERT-C	91
					TextCNN	83
IMDB	50,000	279.48	2	Movie review	BERT-C	93
					TextCNN	88

classification dataset with 127,600 samples belonging to 4 topic classes, *World, Sports, Business, Sci/Tech*. Emotion (Saravia et al., 2018) is a dataset with 20,000 samples and 6 classes, *sadness, joy, love, anger, fear, surprise*. SST2 (Socher et al., 2013) is a binary class (*positive and negative*) topic dataset with 9613 samples. The IMDB dataset (Maas et al., 2011), comprising movie reviews from the Internet Movie Database, is predominantly utilized for binary sentiment classification, categorizing reviews into 'positive' or 'negative' sentiments. The details of these datasets can be found in Table 2.

To ensure reproducibility, we provide the code and data used in our experiments in a GitHub repository.[1]

4.1 Victim models

We apply our attack algorithm to two types of popular and well-performed victim models. The details of the models can be found below.

4.1.1 BERT-based classifiers

To do convincing experiments, we choose three well-performed and popular BERT-based models, which we call BERT-C models (where the letter "C" represents "classifier"), pre-trained by Huggingface.[2] Due to the different sizes of the datasets, the structures of BERT-based classifiers are adjusted accordingly. The BERT classifier for AG's News is structured by the *Distil-RoBERTa-base* (Sanh et al., 2019) connected with two fully connected layers, and it is trained for 10 epochs with a learning rate of 0.0001. For the Emotion dataset, its BERT-C adopts another version of BERT, *Distil-BERT-base-uncased* (Sanh et al., 2019), and the training hyper-parameters remain the same as BERT-C for AG's News. Since the SST2 dataset is relatively small compared with the other two models, the corresponding BERT classifier utilizes a small-size version of BERT, *BERT-base-uncased* (Devlin et al., 2019). As for the IMDB, we employ the *Distil-BERT-base-uncased* for classification tasks.

[1] https://github.com/MingzeLucasNi/RJA-MMR.
[2] https://huggingface.co/.

The test accuracy of these BERT-based classifiers before they are under attacks are listed in Table 2 and these models are publicly accessible[3456].

4.1.2 TextCNN-based models

The other type of victim model is TextCNN (Kim, 2014), structured with a 100-dimension embedding layer followed by a 128-units long short-term memory layer. This classifier is trained 10 epochs by ADAM optimizer with parameters: learning rate $lr = 0.005$, the two coefficients used for computing running averages of gradient and its square are set to be 0.9 and 0.999 ($\beta_1 = 0.9$, $\beta_2 = 0.999$), the denominator to improve numerical stability $\sigma = 10^{-5}$. The accuracy of these TextCNN-base models is also shown in Table 2.

4.2 Baselines

To evaluate the attacking performance, we use the TextAttack (Morris et al., 2020) framework to deploy the following baselines:

- AGA (Alzantot et al., 2018): it uses the combination of restrictions on word embedding distance and language model prediction scores to reduce search space. As for the searching algorithm, it adopts a genetic algorithm, a popular population-based evolutionary algorithm.
- Faster Alzantot Genetic Algorithm (FAGA) (Jia et al., 2019): it accelerates AGA by bounding the searching domain of genetic optimization.
- BERT-Base Adversarial Examples (BAE) (Garg & Ramakrishnan, 2020): it replaces and inserts tokens in the original text by masking a portion of the text and leveraging the BERT-MLM.
- Metropolis–Hasting Attack (MHA) (Zhang et al., 2019): it performs Metropolis–Hasting sampling, which is designed with the guidance of gradients, to sample the examples from a pre-selector that generates candidates by using MLM.
- BERT-Attack (BA)(Liet al., 2020): it takes advantage of BERT-MLM to generate candidates and attacked words by the static WSR descending order.
- Probability Weighted Word Saliency (PWWS) (Ren et al., 2019): it chooses candidate words from WordNet (Miller et al., 1990) and sorts word attack order by multiplying the word saliency and probability variation.
- TextFooler (TF) (Jin et al., 2020): it ranks the important words with similar strategy with Eq. (4). With the important rank, the attacker prioritizes replacing them with the most semantically similar and grammatically correct words until the prediction is altered.
- Particle Swarm Optimization (PSO) (Zang et al., 2020): it selects word candidates from HowNet and employs the POS to find adversarial text. This method treats every sample as a particle whose location in the search space needs to be optimized.

[3] https://huggingface.co/mrm8488/distilroberta-finetuned-age_news-classification.
[4] https://huggingface.co/bhadresh-savani/distilbert-base-uncased-emotion.
[5] https://huggingface.co/echarlaix/bert-base-uncased-sst2-acc91.1-d37-hybrid.
[6] https://huggingface.co/lvwerra/distilbert-imdb.

4.3 Experimental settings and evaluation metrics

For our RJA and RJA-MMR, we use the Universal Sentence Encoder (USE) (Cer et al., 2018) to measure the sentence semantic similarity for target distribution in Eq. (9). We experiment to find $k = 30$ substitution candidates and to find these candidates' substitutions, we use *RoBERTa-large* (Liu et al., 2019) as the MLM with WordPiece (Wu et al., 2016) tokenizer for contextual infilling and utilize OpenHowNet (Qi et al., 2019) with NLTK (Bird et al., 2009) tokenizer as the synonym thesaurus. For the sampling-based algorithms, MHA and the proposed methods (RJA, RJA-MMA), we set the maximum number of iterations T to 1000.

We argue that the quality of adversarial examples is appraised with regard to three key facets: attacking performance, imperceptibility, and fluency. To measure these facets, we use the following five metrics to measure the performance of adversarial attacks:

- Successful attack rate (SAR) is defined as the percentage of attacks where the adversarial examples make the victim models predict a wrong label.
- Modification Rate (Mod) is the percentage of modified tokens. Each replacement, insertion or removal action accounts for one modified token.
- Grammar Error (GErr) is measured by the absolute rate of increased grammatic errors in the successful adversarial examples, compared to the original text, where we use LanguageTool (Naber et al., 2003) to obtain the number of grammatical errors.
- Perplexity (PPL) denotes a metric used to evaluate the fluency of adversarial examples (Kann et al., 2018; Zang et al., 2020). The perplexity is calculated using small-sized GPT-2 with a 50k-sized vocabulary (Radford et al., 2019).
- Textual similarity (Sim) is measured by the cosine similarity between the sentence embeddings of the input and that of the adversarial sample. We encoded the two sentences with the universal sentence encoder (USE) (Cer et al., 2018).

SAR evaluates attack performance, while Mod and Sim measure imperceptibility. GErr and PPL assess language fluency.

4.4 Experimental results and analysis

The main experimental results of the attacking performance (SAR), the imperceptibility performance (Sim, Mod) and the fluency of adversarial examples (PPL, GErr) are listed in Table 3 and 4. Moreover, we demonstrate adversarial examples crafted by various methods shown in Table 5. We manifest the three contributions mentioned in the Introduction section by answering three research questions:

4.4.1 Does our method make more thrilling attacks compared with baselines?

We compare the attacking performance of the proposed method RJA-MMR and baselines in Table 3. This table demonstrates that RJA-MMR consistently outperforms other competing methods across different data domains, regardless of the structure of classifiers. Further, even RJA, by itself, without using MMR, can craft more menacing adversarial examples than most baselines. We attribute such an outstanding attacking performance to

Table 3 Results on SAR, Mod, and Sim metrics among the baselines and proposed methods on different datasets

Task	Method	AG news			Emotion			SST2			IMDB		
		SAR↑	Mod↓	Sim↑	SAR↑	Mod↓	Sim↑	SAR↑	Mod↓	Sim↑	SAR↑	Mod↓	Sim↑
BERT-C	BAE	41.0	11.3	72	68.7	7.7	88	55.1	11.3	75	66	6.9	*89*
	AGA	15.3	10.9	71	48.1	8.1	89	41.7	12.2	70	71	6.6	90
	FAGA	27.9	14.6	75	78.2	9.8	89	77.1	17.2	69	81	7.3	*89*
	MHA	41.2	14.2	62	77.1	10.2	87	84.5	17.2	72	79.9	6.9	81
	BA	47.7	17.4	73	85.9	9.2	82	77.2	13.5	74	78.1	5.6	87
	PWWS	76.8	17.9	73	91.8	10.2	83	93.9	17.2	84	*99.0*	7.8	80
	TF	89.3	21.6	77	90.1	10.2	82	90.4	15.3	80	96.3	5.6	81
	PSO	93.4	21.6	69	94.3	12.0	87	96.6	17.2	81	99.1	6.3	80
	RJA	95.1	11.1	77	97.1	8.5	88	96.4	15.3	78	92.1	*5.9*	83
	RJA-MMR	**96.7**	**10.3**	**79**	**97.3**	**7.1**	**90**	**98.7**	**11.2**	**86**	**100.0**	**4.3**	**92**
Text-CNN	BAE	39.1	10.3	72.6	85.3	9.8	73	80.1	10.4	70	71	7.9	80
	AGA	33.3	11.3	75.4	81.1	7.7	85	77.4	10.8	75	80	8.8	83
	FAGA	56.1	11.5	80.1	90.1	8.3	80	92.1	15.3	69	83	7.6	87
	MHA	70.0	16.4	71.1	95.1	14.1	56	85.5	17.2	74	91	10.1	81
	BA	70.2	15.4	81.1	97.1	9.3	83	83.4	13.4	70	89	8.7	80
	PWWS	85.3	16.5	81.1	98.2	11.3	79	98.1	13.4	81	99	10.1	81
	TF	77.3	19.4	74.9	91.5	10.9	83	91.0	17.2	75	99	*6.9*	90
	PSO	76.2	15.5	77.3	99.0	9.4	83	92.2	17.2	81	99	9.3	88
	RJA	88.3	11.4	79.1	94.9	9.7	83	94.0	17.2	77	95	6.6	90
	RJA-MMR	**93.8**	**9.9**	**82.2**	**99.3**	**7.4**	**89**	**99.3**	**10.3**	**82**	**100.0**	**3.3**	**91**

The best performance is in bold

Table 4 Results on PPL and GErr metrics among the baselines and proposed methods on different datasets

Task	Method	AG News		Emotion		SST2		IMDB	
		PPL↓	GErr↓	PPL↓	GErr↓	PPL↓	GErr↓	PPL↓	GErr↓
BERT-C	BAE	142	0.19	233	0.10	173	0.15	181	0.21
	AGA	132	0.21	291	0.14	192	0.17	211	0.23
	FAGA	165	0.19	259	0.13	182	0.24	155	0.23
	MHA	223	0.28	311	0.18	210	0.31	160	*0.20*
	BA	281	0.19	301	0.17	200	0.25	100	*0.20*
	PWWS	318	0.22	333	0.16	214	0.21	*89*	0.22
	TF	312	0.28	341	0.19	191	0.17	111	0.22
	PSO	292	0.31	362	0.20	197	0.27	91	0.22
	RJA	155	0.21	281	0.12	201	0.18	77	**0.19**
	RJA-MMR	**141**	**0.18**	**221**	**0.10**	**169**	**0.13**	**61**	**0.19**
Text-CNN	BAE	132	0.19	201	0.10	143	0.13	131	0.23
	AGA	132	0.13	213	0.10	182	0.13	158	0.24
	FAGA	145	0.15	241	0.13	163	0.17	198	0.26
	MHA	211	0.29	248	0.16	164	0.22	156	0.22
	BA	241	0.15	221	0.13	210	0.21	123	0.23
	PWWS	277	0.20	299	0.19	224	0.23	79	0.22
	TF	199	0.21	314	0.21	194	0.21	166	0.22
	PSO	142	0.14	301	0.18	145	0.14	164	0.28
	RJA	154	0.17	294	0.15	164	0.18	*89*	0.23
	RJA-MMR	**127**	**0.12**	**190**	**0.08**	**142**	**0.11**	**65**	**0.19**

The best performance is in bold

Table 5 Adversarial examples of the Emotion dataset for victim classifier BERT-C

Methods	Adversarial example	Success	Confidence (%)
BAE	Made a **wonderful** *nasty* new friend	**Successful**	4.3
AGA	Made a **wonderful** *beautiful* new friend	Failed	94
FAGA	**Make** *Introduced* a **wonderful** *beautiful* new friend	Failed	95
MHA	Made a **wonderful** *newly* friend	Failed	70
BA	Made a **wonderful** *good* **new** *brand* friend	Failed	95
PWWS	**Made** *Seduce* a wonderful **new** *raw admirer*	Failed	99
TF	Made a **wonderful** *strange* new friend	**Successful**	5.0
PSO	**Made** *Doomed* a wonderful new friend	**Successful**	0.92
RJA-MMR	Made a **wonderful** *lovely* new friend	**Successful**	**0.80**

Bold texts are original words, while italic ones are substitutions. Besides the examples, the attack performance is measured by attacking success and confidence in making correct predictions. The lower confidence indicates better performance and the successful attacks and lowest confidence are bold

the two prevailing aspects of RJA. Firstly, RJA optimizes the performance by stochastically searching the domain. Most of the baselines perform a deterministic searching algorithm which could get stuck in the local optima. Differently, such a stochastic mechanism helps skip the local optima and further maximize the attacking performance.

Secondly, some of the baselines strictly attack the victim words in the order of word saliency rank (WSR), where the domain of the hierarchical search is limited to combinations of the neighboring victim words from the WSR, which would miss the potential optimal victim words combination. Unlike these methods, the RJA would enlarge the searching domain by testing more combinations of substitutions that do not follow the WSR order. Thus, the proposed method RJA achieves the best-attacking performance, with the highest successful attack rate (SAR).

4.4.2 Is RJA-MMR superior to the baselines in terms of imperceptibility?

We evaluate the imperceptibility of different attack strategies in terms of semantic similarities (USE) and modification rate (Mod) between the original input text and its derived adversarial examples, shown in Table 3. It can be seen that the proposed RJA-MMR attains the best performance among the baselines. The outstanding performance of the proposed method is attributed to the mechanisms of RJA and MMR. For semantic preservation, we statistically design the target distribution (Eq. 9) with a strong regularization of the semantic similarity in each iteration. Moreover, the HowNet is a knowledge-graph-based thesaurus that provides part-of-speech (POS) aware substitutions. Compared with the candidates supplied by baselines, the synonyms from HowNet can be more semantically similar to the original words. As for the modification rate, the proposed MMR is mainly designed for restoring the attacked words from successful adversarial examples so that the proposed RJA-MMR perturbs fewer words without sacrificing the attacking performance. Thus we can conclude that the proposed RJA-MMR provides the best performance for imperceptibility among baselines.

4.4.3 Is the quality of adversarial examples generated by the proposed methods better than that crafted by the baselines?

We insist the qualified adversarial examples should be parsing-fluent and grammarly correct. From the Table 4, we can find the RJA-MMR provides the lowest perplexity (PPL), which means the examples generated by RJA-MMR are more likely to appear in the corpus of evaluation. As our corpus is long enough and the evaluation model is broadly used, it indicates these examples are more likely to appear in natural language space, thus eventually leading to better fluency. For the grammar errors, the proposed method RJA-MMR is substantially better than the other baselines, which indicates a better quality of the adversarial examples. We attribute such performance to our method of finding word substitution, constructing the candidates set by intersecting the candidates from HowNet and MLM.

4.5 Ablation study

To rigorously validate the efficacy of the proposed RJA-MMR method, this section conducts a detailed ablation study, dissecting each component to assess its individual impact and overall contribution to the method's performance.

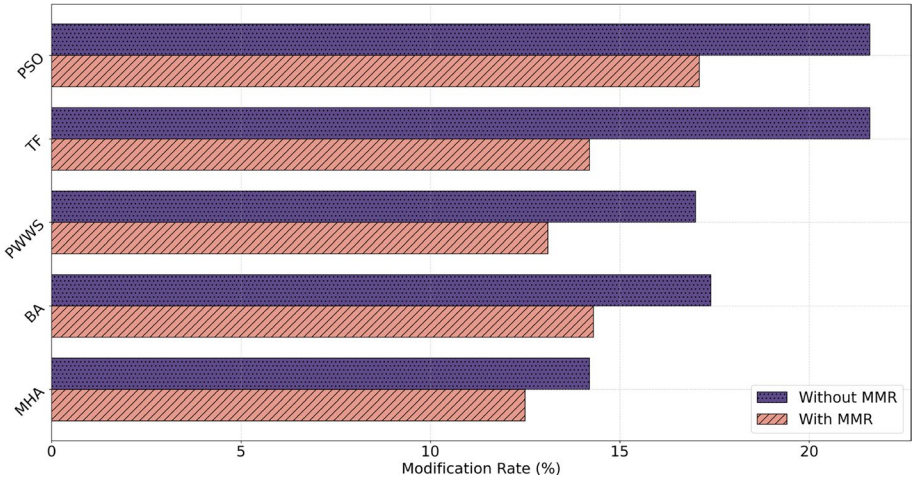

Fig. 3 Comparisons on modification rates among attacking strategies (PSO, TF, PWWS, BA, MHA) with MMR and without MMR to attack the BERT-C on AG News dataset

4.5.1 Effectiveness of RJA

We compare the attacking performance of our Reversible Jump Attack methods (RJA, RJA-MMR) and baselines in Table 3, reflected by SAR. The RJA helps attackers achieve the best attacking performance, with the largest metric SAR across the different downstream tasks. Apart from RJA-MMR, its ablation RJA also surpasses the strong baselines in most cases. Therefore, RJA is effective in terms of attacking performance.

4.5.2 Effectiveness of MMR

MMR is a stochastic mechanism to reduce the modifications of adversarial examples with attacking performance preserved. Besides RJA-MMR, we also apply MMR to different attacking algorithms, including PSO, TF, PWWS, BA and MHA, aiming to demonstrate the advantages of MMR in general.

From Table 3, we can find RJA-MMR has superior performance to RJA with lower modification rates. Moreover, the other baseline analysis results are shown in Fig. 3. It shows that the attacking algorithms with MMR consistently have a lower modification rate than those without MMR. This means that attacking strategies can generally benefit from MMR by making fewer modifications.

4.5.3 Performance versus the number of iterations

The performance of the proposed methods is influenced by the number of iterations, denoted as T. To delve deeper into this relationship, we conducted an extensive ablation study examining the correlation between performance and T. Insights drawn from Fig. 4

Table 6 Performance metrics for RJA-MMR against the TextCNN model on the AG News dataset using varied word candidate selection methods

Methods	SAR	USE	Mod	PPL	GErr
HowNet	85.1	72	13.1	159	0.15
BERT-base	83.1	70	12.1	144	0.15
RoBERTa-large	90.1	73	11.3	156	**0.12**
HowNet+BERT-base	90.8	80	10.9	145	0.14
HowNet+RoBERTa-large	**93.8**	**82**	**9.9**	**127**	**0.12**

The best performances for each metric are highlighted in bold

reveal a positive trend where performance amplifies in tandem with the number of iterations. Notably, performance begins to plateau, indicating convergence, at $T = 100$.

4.5.4 Effectiveness of the word candidates

In our ablation study, detailed in Table 6, we explored the effectiveness of various word candidate selection methods on the performance of RJA-MMR against the TextCNN model, utilizing the AG News dataset. Our evaluation included three strategies: using HowNet, MLMs with BERT-base (Devlin et al., 2019), RoBERTa-large (Liu et al., 2019), and a synergistic approach combining HowNet and MLMs. Individually, HowNet and the MLM approaches showed notable performance, with RoBERTa-large slightly outperforming BERT-base. However, the combination of HowNet and MLMs produced superior results, surpassing the individual methods in all evaluated metrics, highlighting the significant advantage of integrating HowNet with MLMs to enhance the effectiveness of adversarial attacks.

Furthermore, our analysis of combination strategies for generating word candidates revealed that the more sophisticated MLM, RoBERTa-large, yielded a more effective attack performance than its less advanced counterpart, BERT-base. This finding suggests a positive correlation between advancements in MLM technology and enhancements in attack efficacy. We attribute this trend to the ability of more advanced MLMs to generate more relevant and suitable word candidates for use in attack methodologies, thereby increasing the precision and effectiveness of adversarial strategies.

4.6 Platform and efficiency analysis

In this section, we aim to evaluate the efficiency from both empirical and theoretical perspectives. To perform the empirical complexity (EV) evaluation, we carry out all experiments on RHEL 7.9 with the following specification: Intel(R) Xeon(R) Gold 6238R 2.2GHz 28 cores (26 cores enabled) 38.5MB L3 Cache (Max Turbo Freq. 4.0GHz, Min 3.0GHz) CPU, NVIDIA Quadro RTX 5000 (3072 Cores, 384 Tensor Cores, 16GB Memory) (GPU), and 88GB RAM. Table 7 lists the time consumed for attacking BERT and TextCNN classifiers on the Emotion dataset. The metric of time efficiency is second per example, which means a lower metric indicates better efficiency. Results from Table 7 show that our RJA and RJA-MMR run longer than some static counterparts (PWWS, BAE, TF) but are more efficient than the others, such as PSO, FAGA, MHA and BA. Nonetheless,

Table 7 Assessment of attack algorithms' efficiency on the Emotion dataset, utilizing empirical complexity (EC) in seconds per example for practical evaluation and total variance (TV) distance for theoretical convergence speed analysis

Methods	Metric	BAE	FAGA	MHA	BA	PWWS	TF	PSO	RJA	RJA-MMR
BERT-C	EC	*21.7*	162.4	414.0	707.9	**0.7**	40.5	73.8	66.9	56.2
	TV	–	*1.22*	1.14	–	–	–	1.3	0.99	**0.89**
TextCNN	EC	17.4	84.5	191.3	488.1	**0.4**	*28.1*	55.1	51.9	54.1
	TV	–	1.31	1.40	–	–	–	*1.29*	1.11	**1.01**

Lower EC values denote higher efficiency. The top three methods are highlighted in bold, italic, and underlined

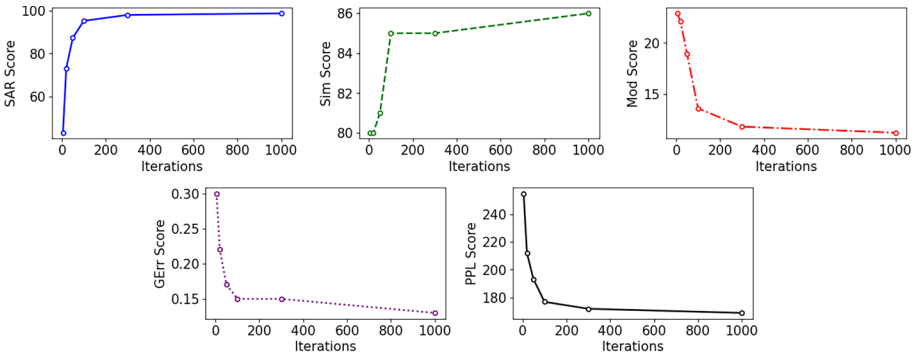

Fig. 4 The progression of SAR, SIM, Mod, GErr, and PPL metrics for SST2 BERT over increased iterations (T). Performance trends and convergence points are visually represented

the results of our methods running longer than some baseline methods indicate the genuine time needed to look for the more optimal adversarial examples.

To theoretically gauge convergence speed, researchers employ the probabilistic concept of Mixing Time (MT), which denotes the duration for a Markov chain to approach its steady-state distribution closely (Kroese et al., 2011). Given that MT is constrained by the total variation distance (TV) between the proposed and target distributions, TV is frequently used as a metric to quantify both the mixing time and speed of convergence (Metropolis et al., 1953a; Green, 1995b). Analysis of Table 7 reveals that the proposed RJA-MMR method registers the lowest Total Variance (TV) distance, indicating superior theoretical performance in terms of convergence speed compared to other methods.

4.7 Transferability

The transferability of adversarial examples refers to its ability to degrade the performance of other models to a certain extent when the examples are generated on a specific classifier (Goodfellow et al., 2015). To evaluate the transferability, we investigate further by exchanging the adversarial examples generated on BERT-C and TextCNN and the results are shown in Fig. 5.

When the adversarial examples generated by our methods are transferred to attack BERT-C and TexCNN, we can find that the attacking performance of RJA-MMR still

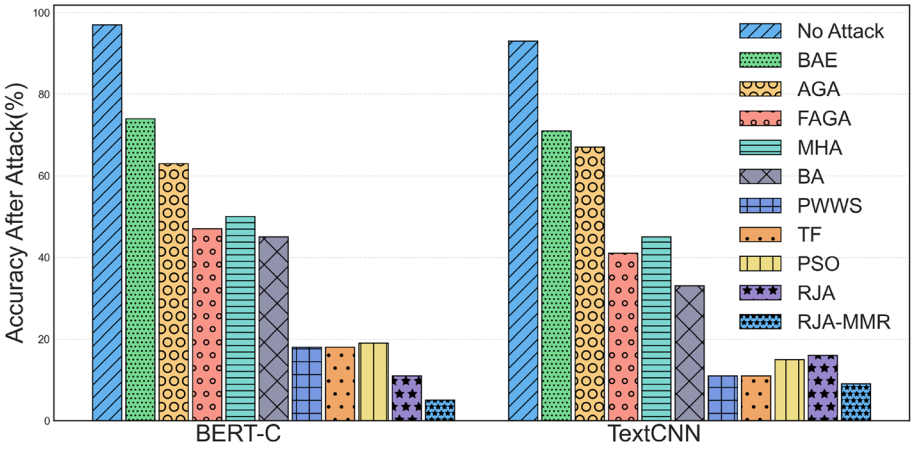

Fig. 5 Performance of transfer attacks to victim models (BERT-C and TextCNN) on Emotion. A lower accuracy of the victim models indicates a higher transfer ability (i.e., the lower, the better)

Table 8 Targeted attack and imperceptibility-preserving performance on the Emotion dataset

Classifers	Attack methods	Metrics				
		SAR↑	Mod↓	PPL↓	GErr↓	Sim↑
BERT-C	PWWS	21.2	14.1	377	0.19	60
	RJA-MMR	**28.0**	**9.2**	**299**	**0.13**	**71**
TextCNN	PWWS	32.6	11.1	345	0.22	63
	RJA-MMR	**57.1**	**10.3**	**256**	**0.17**	**65**

The victim models are BERT-C and TextCNN classifiers, and the baseline is PWWS. The statistics for better performance are vertically highlighted in bold

achieves more than 80% successful rate, which is the best among baselines as illustrated in the Fig. 5. Apart from RJA-MMR, its ablated components RJA also surpass the most baselines. This suggests that the transferring attacking performance of the proposed methods consistently outperforms the baselines.

4.8 Targeted attacks

A targeted attack is to attack the data sample with class y in a way that the sample will be misclassified as a specified target class y' but not other classes by the victim classifier. RJA and MMR can be easily adapted to targeted attack by modifying $1 - F_y(\mathbf{x})$ to $F_{y'}(\mathbf{x})$ in Eq. (9). The targeted attack experiments are conducted on the Emotion dataset. The results demonstrate that the proposed RJA-MMR achieves better performance than PWWS, in terms of attacking performance (SAR), imperceptibility performance (Mod, Sim) and sentence fluency (GErr, PPL) (Table 8).

Table 9 A comparative analysis of attack performance (SAR) against BERT-C when subjected to two defense mechanisms, FGWS and RanMASK, across IMDB and SST2 datasets

Datasets	Defense	BAE	FAGA	MHA	PWWS	PSO	RJA-MMR
IMDB	FGWS	37.7	18.0	34.9	66.1	80.0	**88.1**
	RanMASK	39.1	19.2	40.1	55.3	81.0	**83.1**
SST2	FGWS	38.1	40.1	61.0	63.7	79.9	**81.7**
	RanMASK	41.1	16.4	39.6	71.3	77.1	**86.7**

Performance metrics are highlighted in bold to emphasize superior results

4.9 Attacking models with defense mechanism

Defending against textual adversarial attacks is paramount in ensuring the integrity and security of machine learning models used in natural language processing applications. Effective defense mechanisms encompass two multi-faceted approaches that include: 1) robust model training, utilizing adversarial training techniques to increase models' resilience against malicious inputs. 2) malicious input detection, aiming to identify and mitigate adversarial examples without actively altering the machine learning model's structure or training process.

To ensure a thorough evaluation of our proposed attack methods, we've integrated two distinct defense mechanisms into our assessment. For passive defense, we adopted the Frequency-Guided Word Substitutions (FGWS) (Mozes et al., 2021) approach, which excels at identifying adversarial examples. Conversely, for active defense, we incorporated Random Masking Training (RanMASK) (Zeng et al., 2023), a technique that bolsters model resilience via specialized training routines. We perform the adversarial attack to the BERT-C on the two datasets IMDB and SST2, and the results are presented in Table 9. The results show that our method outperforms the baselines.

4.10 Adversarial retraining

This section explores RJA-MMR's potential in improving downstream models' accuracy and robustness. Following (Li et al., 2021), we use RJA-MMR to generate adversarial examples from AG's News training instances and include them as additional training data. We inject different proportions of adversarial examples into the training data for the settings of a BERT-based MLP classifier and a TextCNN classifier without any pre-trained embedding. We provide adversarial retraining analysis by answering the following two questions:

4.10.1 Can adversarial retraining help achieve better test accuracy?

As shown in Fig. 6, when the training data is accessible, adversarial training gradually increases the test accuracy while the proportions of adversarial data are smaller than roughly 30%. Based on our results, we can see that a certain amount of adversarial data can help improve the models' accuracy, but too much such data will degrade the performance. This means that the right amount of adversarial data will need to be determined empirically, which matches the conclusions made from previous research (Jia et al., 2019; Yang et al., 2021).

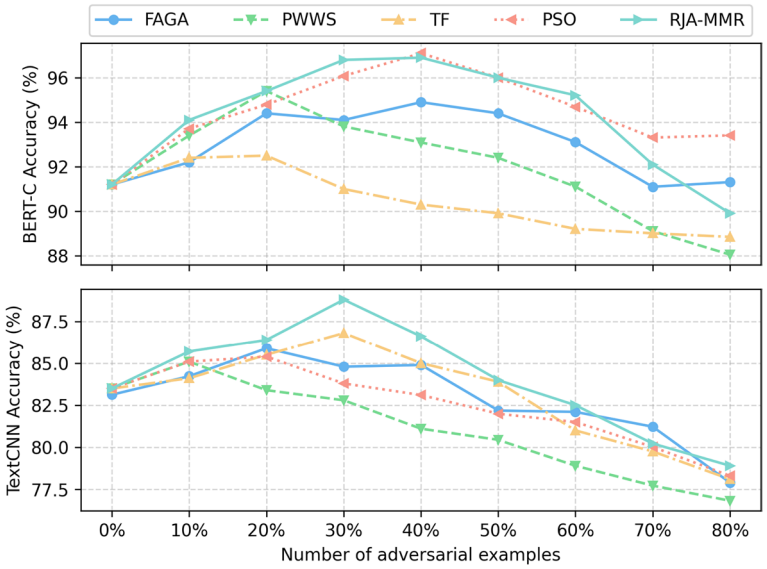

Fig. 6 Results of adversarially trained BERT and TextCNN by inserting the different numbers of adversarial examples to the training set. The accuracy is based on the performance of the SST2 test set

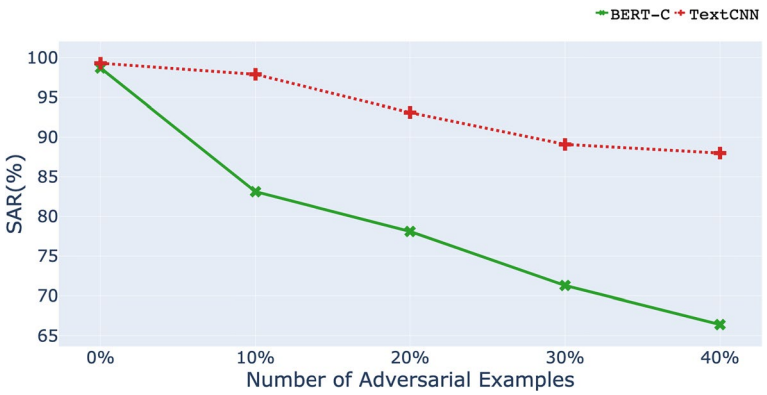

Fig. 7 The success attack rate (SAR) of adversarially retrained models with different numbers of adversarial examples. A lower SAR indicates a victim classifier is more robust to adversarial attacks

Does adversarial retraining help the models defend against adversarial attacks? To evaluate this, we use RJA-MMR to attack the classifiers trained with different proportions (0%, 10%, 20%, 30%, 40%) of adversarial examples. A higher success rate (SAR) indicates a victim classifier is more vulnerable to adversarial attacks. As shown in Fig. 7, adversarial training helps to decrease the attack success rate by more than 10% for the BERT classifier (BERT-C) and 5% for TextCNN. These results suggest that the proposed RJA-MMR can be used to improve downstream models' robustness by joining its generated adversarial examples to the training set.

Table 10 POS preference with respect to choices of victim words among attacking methods

Methods	Noun (%)	Verb (%)	Adj. (%)	Adv. (%)	Others (%)
BAE	*30*	14	13	**41**	2
AGA	**44**	*21*	11	5	19
FAGA	**34**	11	*22*	14	19
MHA	**54**	9	*21*	4	12
BA	**68**	9	4	9	*10*
PWWS	**54**	9	*18*	3	16
TF	*31*	10	**39**	10	10
PSO	**48**	9	15	**19**	9
RJA	*28*	12	19	11	**30**
RJA-MMR	*22*	17	13	17	**31**

The tags with the horizontally highest and second highest proportion are bold and italic, respectively

4.11 Parts of speech preference

Regarding the superiority of the proposed method in attacking performance, we investigate its attacking preference, described by parts of speech (POS), for further linguistic analysis. In this subsection, we break down the attacked words in AG's News dataset by part-of-speech tags with Stanford PSO tagger (Toutanova et al., 2003), and the collected statistics are shown in Table 10. By analyzing the results, we expect to find the more vulnerable POS by comparing the proposed methods and baselines.

We apply PSO tagger to annotate them with POS tags, including *noun*, *verb*, *adjective (Adj.)*, *adverb (Adv.)* and *others* (i.e., pronoun preposition, conjunction, etc.). Statistical results in Table 10 demonstrate that all the attacking methods heavily focus on the *noun*. Presumably, in the topic classification task, the prediction heavily depends on *noun*. However, the proposed attacking strategies (RJA and RJA-MMR) tend to take a more significant proportion of *others* than any other methods; thus we might conclude that *Others* (pronoun, preposition and conjunction) might be the second adversarially vulnerable. Since these tags (pronouns, prepositions and conjunction) do not carry much semantics, we think these tags will not linguistically and semantically affect prediction but possibly impact the sequential dependencies, which could contaminate the contextual understanding of the classifiers and then subsequently cause wrong predictions.

4.12 Robustness versus the scale of pre-trained models

Examining Tables 3 and 4, a question arises: Does increasing the scale of a model enhance its robustness? To explore this, we conducted a study applying our proposed attack methods to victim models of varying sizes on the Emotion dataset.

To provide a more nuanced analysis, we recognize that limiting our comparison to the two initial versions of BERT-base and large as introduced by (Devlin et al., 2019)-does not sufficiently support robust experimental outcomes. Hence, we have incorporated several

Table 11 Robustness of BERT Models of Different Sizes on the Emotion Dataset

Versions	BERT tiny	BERT mini	BERT small	BERT medium
Size	L = 2, H = 128	L = 2, H = 128	L = 4, H = 512	L=4, H=512
SAR	99.9	9.2	98.4	97.5
Mod	5.7	6.2	6.8	7.0
Sim	93	92	91	91

These models are trained with the same datasets and hyper-parameter but with different numbers of transformer layers (**L**) and hidden embedding sizes (**H**)

widely recognized versions published subsequent to the original BERT paper. Specifically, we analyzed four versions of BERT as documented in Turc et al. (2019): BERT Tiny,[7] BERT Mini,[8] BERT Small,[9] and BERT Medium.[10] Notably, the most downloaded version among these has reached up to 6,559,486 monthly downloads on Huggingface alone. Our findings, detailed in Table 11, demonstrate a positive correlation between model size and experimental robustness, confirming the value of incorporating a diverse range of model sizes into our analysis.

5 Conclusion and future work

In recent years, the safety and fairness of NLP models have greatly been threatened by adversarial attacks. Many researchers have raised concerns about the robustness of the NLP classifiers because of their broad downstream tasks, such as fake news detection, sentiment analysis, and email spam detection. To improve classifiers' robustness, we have presented RJA-MMR which consists of two algorithms, Reversible Jump Attack (RJA) and Metropolish-Hasting Modification Reduction (MMR). RJA poses threatening attacks to NLP classifiers by applying the Reversible Jump algorithm to adaptively sample the number of perturbed words, victim words and their substitutions for individual textual input. While MMR is a customized algorithm to help improve the imperceptibility, especially to lower the modification rate, by utilizing the Metropolis–Hasting algorithm to restore the attacked words without affecting attacking performance. Experiments demonstrate that RJA-MMR delivers the best attack success, imperceptibility and sentence fluency among strong baselines.

Although the adversarial examples can threaten the NLP models, these examples are not bugs but features (Ilyas et al., 2019). To protect the models from the attacks, we conduct extensive experiments with a defense strategy, adversarial retraining, which is done by joining the adversarial examples in the training set and then retraining the models with the newly constructed training set. Unsurprisingly, in our experiments, the robustness of the classifiers has been greatly improved, while the accuracy of these models on clean data drops when an excessive amount of adversarial examples are injected.

[7] https://huggingface.co/prajjwal1/bert-tiny.
[8] https://huggingface.co/prajjwal1/bert-mini.
[9] https://huggingface.co/prajjwal1/bert-small.
[10] https://huggingface.co/prajjwal1/bert-medium.

Since the adversarial attack is one of the most effective methods to test the robustness of a model, the proposed attacks raise some concerns about deep neural networks (DNNs) and large pre-trained models. As DNNs and pre-trained language models achieved great success, most existing well-performed NLP classifiers are based on these techniques. Such popularity of these techniques could put textual classifiers at high risk because attackers can make effective attacks by utilizing DNNs and large pre-trained models. Thus a safer way of applying these techniques is a promising future research direction. At the same time, we also plan to pertinently study and design defense strategies to further improve the robustness of NLP classifiers under future adversarial attacks.

Author contributions Mingze Ni contributed to conceptualization, theoretical analysis, experiments and draft preparation; Zhensu Sun contributed to experiments, draft preparation, writing review; Wei Liu contributed to conceptualization, theoretical analysis, draft writing and editing.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. Not applicable.

Availability of data and materials All of the datasets are available on Huggingface (https://huggingface.co/datasets) and on our GitHub site (https://github.com/MingzeLucasNi/RJA-MMR.git).

Code availability All codes from our experiments are available at https://github.com/MingzeLucasNi/RJA-MMR.git.

Declarations

Conflict of interest Not applicable.

Ethical approval Not applicable.

Consent to participate The authors give their consent to participate.

Consent for publication The authors give their consent to the publication of all information in this paper.

References

Alzantot, M., Sharma, Y., Elgohary, A., Ho, B. J., Srivastava, M., & Chang, K. W. (2018). Generating natural language adversarial examples. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 2890–2896).

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly Media, Inc.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Strope, B. Kurzweil, R. (2018). Universal sentence encoder for English. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations* (pp. 169–174).

Cheng, M., Yi, J., Chen, P. Y., Zhang, H., & Hsieh, C. J. (2020). Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 34, No. 04, pp. 3601-3608).

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Dong, Z., & Dong, Q. (2003). Hownet-a hybrid language and knowledge resource. In *International conference on natural language processing and knowledge engineering, 2003. Proceedings. 2003* (pp. 820–824). IEEE.

Dong, Z., Dong, Q., & Hao, C. (2010). Hownet and its computation of meaning. In *Coling 2010: Demonstrations*, (pp. 53–56).

Ebrahimi, J., Rao, A., Lowd, D., & Dou, D. (2018). Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 2: Short Papers)* (pp. 31–36).

Fan, X., Li, B., & Sisson, S. (2018). Rectangular bounding process. *Advances in Neural Information Processing Systems, 31*.

Fan, Y., & Sisson, S. A. (2011). Reversible jump MCMC. In *Handbook of Markov Chain Monte Carlo*, (pp. 67–92).

Gan, W. C., & Ng, H. T .(2019). Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual meeting of the association for computational linguistics*, (pp. 6065–6075).

Garg, S., & Ramakrishnan, G. (2020). Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, (pp. 6174–6181).

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). *Explaining and harnessing adversarial examples*. In *CoRR*arXiv: abs/1412.6572

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika, 82*(4), 711–732.

Haase, J. F., Dellantonio, L., Celi, A., Paulson, D., Kan, A., Jansen, K., & Muschik, C. A. (2021). A resource efficient approach for quantum and classical simulations of gauge theories in particle physics. *Quantum, 5*, 393.

Harrison, B., Purdy, C., & Riedl, M. O. (2017). Toward automated story generation with Markov chain Monte Carlo methods and deep neural networks. In *Thirteenth artificial intelligence and interactive digital entertainment conference*.

Herrmann, H. (1986). Fast algorithm for the simulation of Ising models. *Journal of Statistical Physics, 45*(1), 145–151.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., & Madry, A. (2019) Adversarial examples are not bugs, they are features. *Advances in Neural Information Processing Systems*, *32*.

Iyyer, M., Wieting, J., Gimpel, K., & Zettlemoyer, L. (2018b). Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: Human language technologies, Volume 1 (Long Papers)* (pp. 1875–1885). Association for Computational Linguistics, New Orleans, Louisiana.

Jia, R., & Liang, P. (2017). Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 conference on empirical methods in natural language processing* (pp. 2021–2031). Association for Computational Linguistics, Copenhagen, Denmark.

Jia, R., Raghunathan, A., Göksel, K., & Liang, P. (2019). Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, (pp. 4129–4142).

Jin, D., Jin, Z., Zhou, J. T., & Szolovits, P. (2020). Is Bert really robust? A strong baseline for natural language attack on text classification and entailment. In *AAAI*.

Kang, X., & Ren, F. (2011). Sampling latent emotions and topics in a hierarchical Bayesian network. In *2011 7th international conference on natural language processing and knowledge engineering*, (pp. 37–42).

Kann, K., Rothe, S., & Filippova, K. (2018). Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd conference on computational natural language learning*, (pp. 313–323).

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *EMNLP*.

Kroese, D. P., Taimre, T., & Botev, Z. I. (2011). *Handbook of Monte Carlo Methods*. Wiley.

Kumagai, K., Kobayashi, I., Mochihashi, D., Asoh, H., Nakamura, T., & Nagai, T. (2016). Human-like natural language generation using Monte Carlo tree search. In *CC-NLG*.

Li, D., Zhang, Y., Peng, H., Chen, L., Brockett, C., Sun, M. T., & Dolan, B. (2021). Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: Human language technologies*, (pp. 5053–5069).

Li, L., Ma, R., Guo, Q., Xue, X., & Qiu, X. (2020). Bert-attack: Adversarial attack against Bert using Bert. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, (pp. 6193–6202).

Liang, B., Li, H., Su, M., Bian, P., Li, X., & Shi, W. (2018). Deep text classification can be fooled. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 4208–4215). AAAI Press, IJCAI'18.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V., (2019.) *Roberta: A robustly optimized Bert pretraining approach*. ArXiv arXiv: abs/1907.11692

Maas, A., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies* (pp. 142–150). Association for Computational Linguistics, Portland, Oregon, USA. http://www.aclweb.org/anthology/P11-1015

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. (1953). Equation of state calculations by fast computing machines. *The Journal of Chemical Physics, 21*(6), 1087–1092.

Michel, P., Li, X., & Neubig, G., (2019). *On evaluation of adversarial perturbations for sequence-to-sequence models*. arXiv preprint arXiv:1903.06620

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. In *ICLR*.

Miller, G. A. (1992). WordNet: A lexical database for English. *Communications of the ACM, 38*, 39–41.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography, 3*(4), 235–244.

Morris, J. X., Lifland, E., Yoo, J. Y., Grigsby, J., Jin, D., & Qi, Y. (2020). Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 conference on empirical methods in natural language processing: System demonstrations*, (pp. 119–126).

Mozes, M., Stenetorp, P., Kleinberg, B., & Griffin, L. D. (2021). Frequency-guided word substitutions for detecting textual adversarial examples. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main volume*, (pp. 171–186).

Mrkšić, N., Séaghdha, D. Ó., Thomson, B., Gasić, M., Rojas-Barahona, L., Su, P.H., Vandyke, D., Wen, T.H. & Young, S. (2016). Counter-fitting word vectors to linguistic constraints. In *Proceedings of the 2016 xonference of the North American chapter of the association for computational linguistics: Human language technologies*, (pp. 142–148).

Naber, D. (2003). A rule-based style and grammar checker. *Citeseer*.

Ni, M., Wang, C., Zhu, T., Yu, S., & Liu, W. (2022). Attacking neural machine translations via hybrid attention learning. *Machine Learning, 111*(11), 3977–4002.

Papernot, N., McDaniel, P., Swami, A., & Harang, R. (2016). *Crafting adversarial input sequences for recurrent neural networks*. In *CoRR*arXiv: abs/1604.08275

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical methods in natural language processing (EMNLP)*, (pp. 1532–1543).

Qi, F., Yang, C., Liu, Z., Dong, Q., Sun, M., & Dong, Z. (2019). *Openhownet: An open sememe-based lexical knowledge base*. arXiv preprint arXiv:1901.09957

Qi, F., Chang, L., Sun, M., Ouyang, S., & Liu, Z. (2020). Towards building a multilingual sememe knowledge base: Predicting sememes for babelnet synsets. In *Proceedings of the AAAI conference on artificial intelligence*, (pp. 8624–8631).

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI*.

Ren, S., Deng, Y., He, K., & Che, W. (2019) Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, (pp. 1085–1097).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th annual meeting of the association for computational linguistics (Volume 1: Long Papers)* (pp. 856–865). Association for Computational Linguistics, Melbourne, Australia.

Rincent, R., Kuhn, E., Monod, H., Oury, F. X., Rousset, M., Allard, V., & Le Gouis, J. (2017). Optimization of multi-environment trials for genomic selection based on crop models. *Theoretical and Applied Genetics, 130*, 1735–1752.

Rubinstein, R. (1999). The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability, 1*(2), 127–190.

Samanta, S., & Mehta, S. (2018). Generating adversarial text samples. In *European conference on information retrieval* (pp. 744–749). Springer.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *Distilbert, a distilled version of Bert: Smaller, faster, cheaper and lighter*. arXiv: abs/1910.01108

Saravia, E., Liu, H. C. T., Huang, Y. H., Wu, J., & Chen, Y. S. (2018). CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 3687–3697). Association for Computational Linguistics, Brussels, Belgium.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642). Association for Computational Linguistics, Seattle, Washington, USA.

Tan, S., Joty, S., Kan, M. Y., & Socher, R. (2020). It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 2920–2935). Association for Computational Linguistics.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 conference of the North American Chapter of the association for computational linguistics on human language technology—Volume 1* (pp. 173–180). Association for Computational Linguistics, USA, NAACL '03.

Turc, I., Chang, M. W., Lee, K., & Toutanova, K. (2019). *Well-read students learn better: The impact of student initialization on knowledge distillation*. In *CoRR* arXiv: org/abs/1908.08962

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., & Dean, J. (2016). *Google's neural machine translation system: Bridging the gap between human and machine translation*. arXiv preprint arXiv:1609.08144

Yang, X., Liu, W., Bailey, J., Tao, D., & Liu, W. (2021). Bigram and unigram based text attack via adaptive monotonic heuristic search. In *Proceedings of the AAAI conference on artificial intelligence*, (pp. 706–714).

Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., & Sun, M. (2020). Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, (pp. 6066–6080).

Zeng, J., Xu, J., Zheng, X., & Huang, X. (2023). Certified robustness to text adversarial attacks by randomized [mask]. *Computational Linguistics, 49*(2), 395–427.

Zhang, H., Zhou, H., Miao, N., & Li, L. (2019). Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, Florence, Italy.

Zhang, W. E., Sheng, Q. Z., Alhazmi, A., & Li, C. (2020). Adversarial attacks on deep-learning models in natural language processing: A survey. *ACM Transactions on Intelligent Systems and Technology (TIST), 11*(3), 1–41.

Zhang, X., Zhao, J. J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *NIPS*.

Zou, W., Huang, S., Xie, J., Dai, X., & Chen, J. (2020). A reinforced generation of adversarial examples for neural machine translation. In *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 3486–3497). Association for Computational Linguistics.