# Unmasking Vulnerabilities: Adversarial Attacks via Word-Level Manipulation on NLP Models

**Mingze Ni**

Doctor of Philosophy

University of Technology Sydney
School of Computer Science
Australia

26th August 2024

# Certificate of Original Authorship

I, Mingze Ni, declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science at the University of Technology Sydney. This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis. This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

**Mingze Ni**

Date:26th August 2024

# Abstract

Natural language processing (NLP) models have advanced significantly and are widely used in applications like sentiment analysis, translation, and chatbots. However, they are vulnerable to adversarial attacks, threatening their reliability and real-world adoption. This thesis examines the vulnerabilities of sequence-to-sequence and classification models and introduces techniques for creating effective, imperceptible adversarial examples. The Hybrid Attentive Attack (HAA) method crafts subtle adversarial examples in Neural Machine Translation by focusing on semantically relevant words. The Fraud's Bargain Attack (FBA) uses randomization to improve adversarial example selection for classifiers via the Word Manipulation Process (WMP) and the Metropolis-Hasting sampler. Two algorithms, Reversible Jump Attack (RJA) and Metropolis-Hasting Modification Reduction (MMR), enhance search space and balance changes with attack success. This thesis demonstrates the proposed methods' effectiveness through extensive experiments.

# Acknowledgement

First and foremost, I wish to express my sincere appreciation of my supervisor Associate Professor Wei Liu for taking me as his student, and for his crucial direction, advice, and support throughout the preparation of this thesis. It has been an amazing journey to explore the theories and applications of adversarial attacks for NLP. Without his essential instruction and help, this thesis would not have been accomplished. Finally, a special mention goes out to my parents for supporting me financially and helping me focus on my study. I also commend my friends for encouraging and inspiring me when I encountered difficulties and felt lost in my project, and to other people for their wise advice. I am especially grateful for their support and endorsement throughout this year. I dedicate this thesis to the people that I mentioned.

# Warnings

This thesis includes examples that may be considered offensive or hate speech for the purpose of research and analysis in the field of hate speech detection. These examples are used solely for legitimate academic purposes and do not reflect the views, beliefs, or endorsement of the author or the institution.

# Publications

## Publications related to this thesis

- **Mingze Ni**, Tianqing Zhu, Shui Yu, and Wei Liu: Attacking Neural Machine Translations via Hybrid Attention Learning. In Machine Learning journal (MLJ), 2022. Codes are available at `https://github.com/MingzeLucasNi/HAA`
- **Mingze Ni**, Zhensu Sun, Wei Liu: Fraud's Bargain Attack: Generating Adversarial Text Samples via Word Manipulation Process. In IEEE Transactions on Knowledge and Data Engineering (TKDE), 2024. Codes are available at `https://github.com/MingzeLucasNi/FBA`
- **Mingze Ni**, Zhensu Sun, Wei Liu: Reversible Jump Attack to Textual Classifiers with Modification Reduction. In Machine Learning journal (MLJ), 2024. Codes are available at `https://github.com/MingzeLucasNi/RJA-MMR`
- **Mingze Ni**, Zhensu Sun and Wei Liu: Fraud's Bargain Attacks to Textual Classifiers via Metropolis-Hasting Sampling. In Proceedings of The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI 2023). Codes are available at `https://github.com/MingzeLucasNi/FBA`

## Publications unrelated to this thesis

- **Mingze Ni**, Wei Liu: SleepNet: A Novel Deep Learning Architecture Interweaving Supervised Learning and Unsupervised "Sleep" Cycles. The Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2024) (Under Review)

- Zhensu Sun, Xiaoning Du, Fu Song, Shangwen Wang, **Mingze Ni**, Li Li: Don't Complete It! Preventing Unhelpful Code Completion for Productive and Sustainable Neural Code Completion. In ACM Transactions on Software Engineering and Methodology (TOSEM).

- Zhensu Sun, Xiaoning Du, Fu Song, Shangwen Wang, **Mingze Ni**, Li Li: Don't Complete It! Preventing Unhelpful Code Completion for Productive and Sustainable Neural Code Completion Systems (Poster). In International Conference on Software Engineering (ICSE 2023).

- Zhensu Sun, Xiaoning Du, Fu Song, **Mingze Ni**, Li Li: Coprotector: Protect open-source code against unauthorized training usage with data poisoning. In International World Wide Web Conference (WWW2022)

# CONTENTS

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Deep Neural Networks (DNNs) have demonstrated remarkable success across various applications, such as image classification within the Computer Vision (CV) domain [24, 29, 74], and text recognition in the field of Natural Language Processing (NLP) [9, 26]. However, recent research has exposed a significant vulnerability of DNNs to adversarial attacks, where the input data is deliberately altered to deceive the model [19, 46, 67, 77]. Adversarial attacks in Computer Vision (CV) can have serious consequences, such as causing self-driving cars to misidentify objects or bypass facial recognition systems. Similarly, researchers have shown that NLP models are also highly vulnerable to adversarial attacks [32, 67, 87, 90]. Adversarial attacks in NLP can also cause serious consequences, such as spreading fake news, manipulating online reviews, or causing chatbots to produce harmful responses. Although several attack strategies have gained great success in terms of tampering with the models, the study of adversarial attacks in NLP is still in its early stages, and further research is needed to develop effective defense mechanisms to mitigate their impact.

## 1.1 Background

Adversarial attacks have been a crucial area of research in NLP, as they can help improve the security and reliability of text-based systems. These attacks have significant implications for security and privacy, as attackers can use adversarial examples to evade text-based security systems, impersonate other users, or manipulate the outcome of natural language-based decision-making systems. The inception of text-based attacks can be traced back to 2016, when Papernot et al. [62] conducted a study on the resilience of Recurrent Neural Networks (RNNs) when processing sequential data. In their research, Papernot et al. [62] conclusively demonstrated that RNNs could be completely deceived by making minor alterations to an average of nine words in a 71-word movie review, thus compromising the accuracy of sentiment analysis tasks. In this section, we'll first categorize adversarial attacks in NLP into character-level, word-level, sentence-level, and multi-level, highlighting the effectiveness of word-level attacks. We will discuss why word-level attacks are preferred, considering their balance between detectability and impact. Then, we'll delve into the specifics of how these attacks operate on both sequence-to-sequence and classification models, illustrating their methodologies and effects on these types of NLP models.

### 1.1.1 Attacking Units of NLP Models

Character-level attacks involve the manipulation of individual characters and punctuation marks. These attacks can include character deletion, character insertion, and character substitution. One of the most common techniques used in character-level attacks is the edit distance method, which calculates the number of character-level modifications required to transform the original input into an adversarial example.

Word-level attacks are more sophisticated and involve the manipulation of words using techniques such as insertion, removal, substitution, or switching. These techniques are often used to modify critical or influential words in the text, making them particularly effective in attacking NLP models. For example, attackers may try to insert or substitute negative words in a positive review to manipulate the sentiment analysis system.

Sentence-level attacks are less common, but they involve the insertion, removal, or paraphrasing of entire sentences in the text. These attacks can be challenging to implement, as they require maintaining the coherence and flow of the text while modifying its meaning. One of the most common techniques used in sentence-level attacks is the translation-based method, which involves translating the original text into another language and then back into the original language with modified sentences [16]. Furthermore, multi-level attacks combine techniques from the previous levels of attacks to craft more effective adversarial examples. For example, attackers may use a combination of character-level and word-level attacks to generate more potent adversarial examples that can evade detection by defense mechanisms.

Word-level adversarial attacks are often regarded as optimal in NLP due to their unique combination of imperceptibility, effectiveness, and linguistic naturalness. These attacks provide a sweet spot between detectability and impact; they are subtle enough to avoid immediate detection yet effective enough to alter a model's output significantly. Unlike character-level attacks that might introduce conspicuous spelling mistakes or sentence-level attacks that could drastically change the text's overall structure, word-level modifications are more discreet but still impactful [51]. Moreover, the key strength of word-level attacks lies in their ability to maintain the natural flow of language. By substituting words with synonyms or similar-meaning terms, the original sentence structure and coherence remain largely intact. This subtlety makes it challenging for both humans and automated detection systems to identify the alterations, as they do not notably deviate from standard language usage. Collectively, these factors contribute to the widespread view of word-level attacks as the most efficient and practical form of adversarial attack in NLP, offering an effective blend of stealth, impact, and applicability.

### 1.1.2 Word-level Attacks to Sequence-to-sequence and Classification Models

Word-level adversarial attacks in NLP target sequence-to-sequence models by subtly altering specific words in the input text, which can dramatically change the output while keeping the overall structure and meaning intact [78]. These attacks are effective because sequence-to-sequence models, used in tasks like machine translation and text summarization, heavily rely on the context provided by each word. Techniques such as synonym replacement, homophone substitution, and word insertion or deletion are commonly used to manipulate the model's output. Executing these attacks requires a deep understanding of the model and language, aiming to alter the output significantly while maintaining a semblance of the original text [93]. This area has been the focus of various studies, underscoring the need for more resilient models against such subtle yet impactful word-level adversarial tactics.

Following the discussion on adversarial attacks targeting sequence-to-sequence models, it's crucial to examine how similar strategies impact classification models in NLP. These models, designed for tasks like sentiment analysis, spam detection, and topic categorization, are susceptible to word-level adversarial attacks that can skew their classification outputs. In these attacks, subtle manipulations in the input text, such as replacing key words with synonyms or semantically similar terms, can lead to misclassification. For instance, altering a few words in a product review could change a sentiment analysis model's output from positive to negative. The challenge in executing these attacks lies in modifying the text enough to affect the classification result while keeping the changes inconspicuous [43, 90]. Such attacks highlight the vulnerabilities in classification models and emphasize the importance of enhancing their robustness to maintain accuracy and reliability in real-world applications.

In this thesis, we introduce a novel algorithm, Hybrid Attentive Attack (HAA), designed to target neural machine translation (NMT) systems, a widely-used variant of sequence-to-sequence models. Additionally, we present two sophisticated algorithms aimed at compromising textual classifiers: the Fraud's Bargain Attack (FBA) and the

Reversible Jump Attack with Modification Reduction (RJA-MMR). These algorithms are meticulously crafted to navigate and manipulate the complex landscapes of NMT and textual classification models, showcasing innovative approaches in the field of adversarial machine learning.

## 1.2 Research Objectives

i Perform textual attack to generate semantic preserving text adversarial examples to Neural Machine Translation models, a type of sequence-to-sequence model.

ii Design a performance-boosting textual attack to the NLP classifier by utilizing the sampling methods to pose thrilling attacks via removing, inserting and substituting words.

iii Conduct studies of generating successful adversarial attacks to textual classifiers with minor modification rates based on the random sampling method.

## 1.3 Summary of Research Findings

We summarize the methodology and research findings of this thesis as below:

1. In this thesis, we first propose HAA, which selects influential words by both translation-specific and language-centered attentions and substitutes them with semantics-preserved word perturbations via pre-trained models.
2. We then propose Fraud's Bargain Attack (FBA) utilizing a stochastic process called the Word Manipulation Process (WMP), which considers word substitution, insertion, and removal strategies to generate potential adversarial candidates. The FBA employs the Metropolis-Hasting algorithm to select the best candidates based on a customized acceptance probability, which minimizes semantic deviation from the original sentences.

3. We then introduce Reversible Jump Attack (RJA) with Metropolis-Hasting Modification Reduction (MMR) to enhance the robustness of classifiers. RJA causes NLP classifiers to be at risk by using the Reversible Jump technique to randomly select the number of altered words, the words to be altered, and their replacements for each input. MMR, on the other hand, is a customized algorithm that aims to improve the imperceptibility of the attack, particularly by reducing the modification rate.

## 1.4 Thesis Organization

The remainder of this thesis is organized as follows:

- *Chapter 2:* This chapter presents a survey of adversarial machine learning on the textual attacks, separately.
- *Chapter 3:* This chapter presents the Hybrid Attentive Attacks to Neural Machine Translations and the experimental evaluation results.
- *Chapter 4:* This chapter presents the Fraud's Bargain Attacks to texutal classifiers and reports the experimental validation results.
- *Chapter 5:* This chapter presents Reversible Jump Attack with modifications reduction to generate context-aware text examples and report the experimental results.
- *Chapter 6:* This chapter concludes this thesis and highlights several future research directions.

# Chapter 2

# Literature Review of Adversarial Textual Attack

Deep Neural Networks (DNNs) have seen a surge in popularity due to significant advancements in Artificial Intelligence (AI) and the advent of high-performance computing platforms. Yet, despite their prowess, these models remain susceptible to attacks using adversarial samples. These are maliciously crafted data points which, though only slightly altered from the original input, can deceive the model into making incorrect predictions. This concept is central to adversarial learning, a technique that deliberately supplies misleading input to challenge and probe models. With the increasing emphasis on creating adversarial images and mounting concerns about model security, adversarial learning in the domain of Natural Language Processing (NLP) has garnered substantial attention. In this literature review, our focus will be on exploring research about textual adversarial attacks.

In this chapter, we review the representative character-level (Section 2.2), word-level (Section 2.3), sentence-level (Section 2.4) attacks and the challenges (Section 2.5).

## 2.1 Overview of Adversarial Textual Attack

Despite their advanced capabilities, modern NLP models remain vulnerable to adversarial examples—subtly modified inputs that, while imperceptible to humans, can deceive the algorithm into erroneous behavior [93]. Such adversarial instances have been documented across various domains, including textual classification[1, 77, 90], speech recognition[3], and neural machine translation [6, 13, 78], among others. Beyond the evident security concerns they pose, these adversarial examples underscore crucial gaps in our comprehension of contemporary machine-learning methodologies.

Unlike computer vision models, NLP models inherently operate on discrete, semantically-rich, and readily perceptible input. Given this distinction, our exploration will focus on textual attacks based on the granularity of input units rather than specific attacking methods. We will delve into attacks at the character, word, and sentence levels, as well as multi-level attacks. However, a detailed examination of multi-level attacks will be limited, as they essentially combine the strategies of the first three levels. The subsequent bullet points will highlight the primary aspects of each level of attack:

- Character-level: At this level, individual characters, including punctuation, are manipulated—whether removed, inserted, or replaced—to craft adversarial examples that a language model can still process. Techniques often draw upon gradient-based methods inspired by computer vision attack strategies.
- Word-level: This level encompasses four primary text manipulation methods: word insertion, removal, switching, and substitution. Typically, the focus is on pinpointing and manipulating words that have a significant influence on model interpretation. Recognizing these pivotal words and determining how to alter them is critical for the success of these attacks.
- Sentence-level: Attacks at the sentence level commonly involve inserting, removing, or paraphrasing entire sentences. There's an emerging trend of leveraging text-generation techniques to craft these adversarial examples.

- Multi-level: As the name suggests, multi-level attacks integrate strategies from the character, word, and sentence levels to form a comprehensive attack approach.

In addition to unit-level attacks, textual attacks can be broadly classified into two categories based on the degree of access and understanding the attack possesses regarding a language model's structure and parameters: white box and black box [93].

- White-Box Attacks: For these attacks to be carried out, the adversary must have full, unrestricted access to the classifier model [20]. This encompasses not just the model's overarching architecture but also its intricate details like weights, biases, and gradient information. With such a profound understanding of the model's workings, the attacker can exploit specific vulnerabilities, making these types of attacks particularly potent and potentially more damaging. Since they can pinpoint and leverage weak spots directly, white-box attacks often lead to higher success rates.
- Black-Box Attacks: Reflecting a majority of real-world attack scenarios, black-box attacks operate under the assumption that the attacker does not have detailed insights into the model's architecture, weights, or training process [20]. Their knowledge is generally confined to the model's inputs and outputs. Despite this informational limitation, attackers can still be astutely strategic. By iteratively querying the model and observing its responses, they can glean insights into its behavior, thereby crafting adversarial inputs that might cause the model to err or malfunction.

Both white-box and black-box attacks emphasize the urgent need for durable defense mechanisms in the realm of machine learning. White-box attacks, armed with an intimate understanding of a model, target specific vulnerabilities, offering an in-depth perspective on potential weak points. Conversely, black-box attacks highlight the overarching threats from adversaries operating with only peripheral, constrained knowledge of a model. Despite this, in real-world settings, it is the black-box attacks that often pose a greater concern, given that attackers usually don't have access to proprietary or safeguarded machine learning architectures. Yet, the role of white-box attacks remains pivotal. They not only offer critical insights into potential model vulnerabilities but also set the groundwork that enhances the efficacy of black-box strategies. This foundational understanding allows adversaries to mount more impactful attacks, even with scant model details [13]. Throughout this review, our exploration of attack strategies will span various unit levels, intertwining discussions on both white-box and black-box approaches.

## 2.2 Character Level Attack

As highlighted in the preceding bullet points, character-level attacks encompass strategies that involve the insertion, modification, swapping, or removal of characters, numbers, or special symbols. In 2016, Papernot et al. [62] pioneered white-box attack strategies at the word level, garnering significant interest from researchers in the language modeling domain. Table 2.1 offers a comprehensive overview of the character-level attacks, which will be the focus of our discussion in this section.

Table 2.1: Character level attack with applications

| Year | Reference | Attack Type | Application |
|---|---|---|---|
| 2017 | Hosseini et al. [27] | White & Black box | Classification |
| 2017 | Belinkov and Bisk [2] | Black box | Machine Translation |
| 2018 | Gao et al. [16] | Black box | Classification |
| 2019 | Gil et al. [18] | Black box | Classification |

The *Perspective* API, jointly developed by Google and Jigsaw, was designed to identify toxicity in comments. Nevertheless, in 2017, Hosseini [27] exposed vulnerabilities in Google's *Perspective* API, illustrating that its detection mechanism could be circumvented through basic input alterations. Rather than employing a systematic approach, Hosseini primarily altered toxic words by introducing subtle changes such as inserting a dot (.) or a space between characters or swapping adjacent characters. These tweaks significantly reduced the toxicity score assigned by the API. Some of these adversarial examples are illustrated in Figure 2.2. At a high level, their techniques included controlled adversarial methods designed to mask specific words during translation and targeted methods that either introduced or accentuated particular words. The strategies employed gradient-based optimization and manipulated the text through four primary operations: insertion, character swapping, character replacement, and deletion. For black-box attacks, characters were randomly selected for the prescribed modifications.

| **Original Phrase (Toxicity Score)** | **Original Phrase (Toxicity Score)** |
|---|---|
| Climate change is happening and it's not changing in our favor. If you think differently you're an idiot. (84%) | Climate change is happening and it's not changing in our favor. If you think differently you're an **idiiot**. (20%) |
| They're stupid, it's getting warmer, we should enjoy it while it lasts (86%) | They're **st.upid**, it's getting warmer, we should enjoy it while it lasts (2%) |
| They are liberal idiots who are uneducated (90%) | They are liberal **i.diots** who are **un.educated** (15%) |
| idiots. backward thinking people. nationalists. not accepting facts. susceptible to lies. (80%) | **idiiots**. backward thinking people. **nationaalists**. not accepting facts. susceptible to **1.lies**. (17%) |
| They are stupid and ignorant with no class (91%) | They are **st.upid** and **ig.norant** with no class (11%) |
| It's stupid and wrong (89%) | It's **stuipd** and wrong (17%) |
| If they voted for Hilary they are idiots (90%) | If they voted for Hilary they are **id.iots** (12%) |
| Anyone who voted for Trump is a moron (80%) | Anyone who voted for Trump is a **mo.ron** (13%) |
| Screw you trump supporters (79%) | **S c r e w** you trump supporters (17%) |

Table 2.2: Illustration of the Attack Targeting the *Perspective* Toxic Detection System. The phrases listed in the initial column of the table have been selected from examples available on the *Perspective* website.

Belinkov [2] delved into the domain of character-based neural machine translation. Their approach was distinct in that they did not leverage gradients. Instead, their focus was on bolstering model robustness, employing two primary strategies: the use of structure-invariant word representations and training on noisy texts. Their investigations revealed that models anchored in character convolutional neural networks could adeptly learn representations that were resilient to an array of noise types. Their methodology for generating adversarial examples tapped natural and synthetic noise sources. For natural noise, they curated and extracted errors from various datasets, subsequently replacing the accurate words with these erroneous variants. Regarding synthetic noise, they applied methods such as character swapping, internal word character randomization (excluding

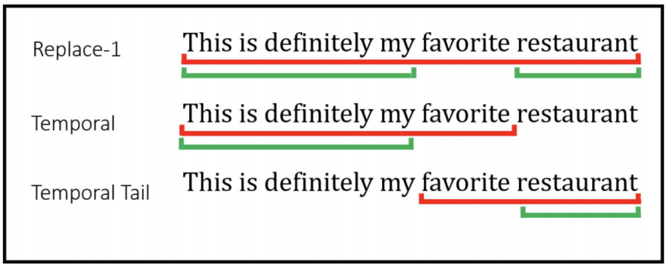


Figure 2.1: The diagram illustrates how the token “favorite” is scored in the input sequence “ This is definitely my favorite restaurant.” Various scoring methods, including Replace1, Temporal Head, and Temporal Tail, involve subtracting the prediction score of the red section from the prediction score of the green section for this token.[16]

the initial and final characters), complete character randomization, and character substitution based on keyboard proximity. To provide specifics, the authors introduced strategies such as swapping adjacent letters within words, a common mistake in rapid typing, rearranging internal characters of words while preserving the first and last letters, fully randomizing word characters, and substituting characters with their immediate keyboard neighbors. These techniques, when applied to words of different lengths, can create anomalies, thereby challenging the standard patterns a system is trained to identify.

Gao [16] introduced the *DeepWordBug* algorithm, specifically designed to induce minute perturbations in text inputs, effectively causing deep-learning classifiers to misclassify them within a black-box environment. Central to their approach was identifying pivotal words ripe for modifications that would lead to the targeted misclassification. To this end, they devised four scoring functions, each meticulously designed to discern these words without necessitating insight into the model’s parameters or architecture. The scoring functions were titled Replace-1 Score (R1S), Temporal Head Score (THS), Temporal Tail Score (TTS), and Combined Score (CS). The initial trio of functions, R1S, THS, and TTS, is elucidated in Figure 2.1. The final function, CS, uniquely amalgamates both THS and TTS. Drawing on insights from these bidirectional measures, it provides a more comprehensive assessment of a word’s significance through its encompassing context. For their empirical studies, the authors leaned on the CS function, given its demonstrably superior efficacy. Once pivotal words were discerned, a series of token alterations—Swap, Substitution, Deletion, and Insertion—were employed to manipulate the identified critical tokens.

Gil et al. [18] adeptly adapted a white-box attack technique for black-box applications. By creating adversarial examples utilizing a white-box method, they subsequently trained a neural network model to emulate this process. Precisely, the generation of adversarial examples via the *HotFlip* method was assimilated into a neural network. This newly evolved model was christened as *DISTFLIP*. A notable merit of their methodology was its independence from the optimization process, thereby accelerating the generation of adversarial examples. Representative adversarial samples from their work can be viewed in Table 2.3.

Table 2.3: Examples of sentences attacked by *DISTFLIP-10* and The Google Perspective toxicity score before and after the attack.

| **Original** | **Adversarial** | **Google score** |
|---|---|---|
| decline=I don't fucking think so | decline=I don't **v**ucking think so | 0.79→0.07 |
| its an AP article you asshole, how is it not neutral??? | its an AP article you ass**n**ole, how is it not neutral??? | 0.96→0.48 |
| I think the 1 million sales is total bullshit though. | I think the 1 million sales is total bullsh**k**t though. | 0.79→0.07 |
| do not know juden! silence is golden juden, now shut up. | do not know juden! silence is golden juden, now **x**h**j**t up. | 0.69→0.37 |
| This article sucks . . WOO WoO WO000000 | This article **x**u**x**ks . . WOO woo w000000o | 0.93→0.22 |
| Also bring back the brendle article you piece of shit. | Also bring back the brendle article you p**k**ece of **x**h**k**t. | 0.98→0.32 |
| to be driven away and die | to be driven away and d**k**e | 0.82→0.32 |

## 2.3 Word Level Attack

Unlike character-level attacks, word-level assaults can be subtler, often eluding human detection even upon meticulous examination of the entire context. This subtlety can make them potentially more damaging than their character-level counterparts. In this section, we delve into various word-level attack strategies, the nuances of which are summarized in Table 2.4

Table 2.4: Word level attack with applications.

| Year | Reference | Attack Type | Application |
|---|---|---|---|
| 2016 | Papernot et al. [62] | White box | Classification |
| 2017 | Samanta and Mehta [71] | White box | Classification |
| 2017 | Liang et al. [46] | White&Black box | Classification |
| 2018 | Kuleshov et al. [40] | White box | Classification |
| 2018 | Li et al. [44] | White box | Classification |
| 2018 | Alzantot et al. [1] | Black box | Classification |
| 2019 | Jia et al. [32] | Black box | Classification |
| 2020 | Li et al. [45] | Black box | Classification |
| 2019 | Ren et al. [67] | Black box | Classification |
| 2020 | Jin et al. [33] | Black box | Classification |
| 2020 | Garg and Ramakrishnan [17] | Black box | Classification |
| 2020 | Zang et al. [90] | Black box | Classification |
| 2020 | Zhang et al. [92] | Black&White box | Classification |
| 2020 | Cheng et al. [6] | Black box | Machine translation & Summarization |
| 2020 | Tan et al. [78] | Black box | Machine translation & Summarization |
| 2021 | Li et al. [43] | Black box | Classification |
| 2021 | Yoo and Qi [89] | Black box | Classification |

The paper by Papernot et al. [62] is credited as the first to create adversarial examples for texts. They employed a computational graph unfolding method to calculate the forward derivative and, with its assistance, the Jacobian can be calculated for further applications. This approach is used to generate adversarial examples via the Fast Gradient Sign Method (FGSM). In their process, the words selected for substitution are chosen randomly, resulting in sentences that don't retain their original meaning or grammatical correctness. Two types of attacks were proposed: adversarial samples and adversarial sequences. For adversarial samples, they utilized an equation inspired by computer vision attacks. In this equation, the adversarial example is crafted from a legitimate sample, with a permutation added to it. The goal is to find the smallest perturbation that causes a change in the model's output. For adversarial sequences, however, the output of a Recurrent Neural Network (RNN) is a sequence, not a category, making the previous equation unattainable. They proposed a modified equation where the adversarial example is obtained by adding the smallest perturbation that keeps the difference between the model's output on the adversarial sequence and the adversarial target within an acceptable error limit. Finally, they apply the FGSM to approximate the adversarial sequence equation. They do this by linearising the model's cost function around its input and choosing a perturbation using the gradient of the cost function with respect to the input itself. The size of the perturbation is controlled by a hyper-parameter.

Samanta and Mehta [71] introduced a model designed to alter a text classification label with the smallest number of changes. This model operates by either adding a new word, deleting an existing one, or replacing one. Initially, the model identifies the words that significantly contribute to the classifier's performance. A word is considered highly contributive if its removal significantly impacts the class probability. To replace words, a pool of potential substitutes is created, comprising synonyms, meaningful words created from typos, and genre-specific words. The model then manipulates these words using three specific strategies. The first strategy involves removing words. Specifically, the model checks if the word under consideration is an adverb and if it has a high contribution score. If both conditions are met, the word is removed from the sample pool, creating a modified sample. If the first strategy isn't applicable, the second strategy comes into play, which involves adding words. In this case, a word is selected from the candidate pool using the Fast Gradient Sign Method (FGSM) [19], following a particular condition. If neither of the first two strategies is applicable, the model resorts to the third strategy: word replacement. In this case, a word is replaced with a genre-specific keyword from the candidate pool, but only if the parts of speech of both the original and replacement words match. If they don't match, the model chooses the next best word from the candidate pool for replacement. This strategy ensures that the modified sample isn't easily detected as an adversarial sample and that the sentence's grammar remains largely intact.

Liang et al. [46] presented a method for creating adversarial text samples that can deceive deep neural network (DNN) based text classifiers, indicating their vulnerability to such attacks. Specifically, the authors proposed both a white-box and a black-box attack strategy based on insertion, deletion, and modification of text. They employed a natural language watermarking technique to generate adversarial samples. In the white-box attack, they used the concepts of Hot Training Phrase (HTP) and Hot Sample Phrase (HSP), obtained through backpropagation, to determine what and where to insert, delete, and modify. For black-box attacks, they used a fuzzing technique to implement a test and obtain HTPs and HSPs. The effectiveness of this method was confirmed through tests on two representative text classification DNNs.

Kuleshov et al. [40] defined adversarial examples as inputs that are crafted to retain the same meaning as the original text but can fool text classification algorithms. They introduce the notion of "altered adversarial examples," which involve adding imperceptible perturbations to the original text while maintaining semantic and syntactic similarity. To construct adversarial examples, the authors propose a greedy optimization strategy. The algorithm iteratively replaces words in the input text with their synonyms, aiming to maximize the adversarial objective while satisfying constraints on semantic and syntactic similarity. The optimization process considers valid one-word changes at each step and selects the replacement that improves the objective the most. The algorithm terminates either when the objective surpasses a threshold or when a specified fraction of words has been replaced. The authors utilize thought vectors to capture the semantic similarity between sentences. Thought vectors are mappings of sentences to a vector space where similar sentences are close to each other. The optimization process includes a constraint on the semantic similarity between the original and altered examples using thought vectors. Additionally, the authors introduce a syntactic constraint based on a language model. The language model is trained on the same dataset as the text classifier and measures the probability of a sentence being grammatically correct. The optimization process ensures that the language model probabilities of the original and altered examples are similar, preserving syntactic validity. Overall, the methodology involves a greedy optimization strategy that iteratively replaces words with synonyms while maintaining semantic and syntactic similarity. The approach leverages thought vectors and language models to capture semantic and syntactic constraints, respectively. The experimental results validate the effectiveness of the proposed methodology in constructing adversarial examples in natural language classification.

In 2018, Li et al. [44] demonstrated the vulnerability of Deep Learning-based Text Understanding (DLTU) systems to adversarial text attacks. These attacks involve carefully crafted texts manipulating DLTU systems to produce incorrect results. They introduced a framework called "TextBugger" for generating adversarial texts. The framework includes both white-box and black-box attack techniques, with white-box attacks being more impactful than black-box attacks. For white-box attacks, Li et al. followed a two-step process for generating adversarial examples. They first defined a score function to identify keywords and then manipulated those words accordingly. Drawing inspiration from JSMA (Jacobian-based Saliency Map Approach), which is used in adversarial attacks in computer vision, they formulated their score function using the Jacobian matrix. The score function, denoted as $J_F(x)$, measures the sensitivity of the model's output to changes in the input. To generate adversarial examples, they focused on five types of edits: insertion, deletion, swapping, substitution with visually similar words, and substitution with words having similar meanings. They created all five types of edits and selected the one that yielded the most significant reduction in accuracy. They proposed a three-step approach for black-box attacks within their framework, where gradient information is inaccessible. They first determined the most important sentence and then identified its optimal word. Since the performance of the black-box attack is not satisfying, we consider the contributions to be mainly for white-box attacking. Overall, [44] developed a comprehensive framework, TextBugger, which showcased the susceptibility of DLTU systems to adversarial text attacks. They provided strategies for generating adversarial examples through white-box and black-box attack techniques, highlighting the effectiveness of white-box attacks in particular.

Most researchers' objective was to minimize the number of modified words while maintaining semantic similarity and syntactic coherence between the original and adversarial examples. However, when dealing with black box attacks, where access to the structures and parameters of the DNN models is not available, and gradients cannot be utilized, they needed an alternative approach. Instead of relying on gradient-based optimization, they developed an attack algorithm that leverages population-based gradient-free optimization using genetic algorithms. Alzanot [1] proposed black-box attack techniques based on genetic algorithms. The authors aimed to generate adversarial examples that preserve both semantic and syntactic similarity. They started by randomly selecting a word from a given sentence and replacing it with a suitable replacement word that fits the sentence context. They utilized the GloVe embedding space to calculate the closest neighboring words. To ensure contextual coherence, they used the Google Words language model [5] to remove any words that didn't align with the sentence context. Then, they selected a word that maximized the prediction and replaced the original word with the selected one.

Based on the genetic attack [1], Jia et al. [32] proposed a faster version of the Genetic Attack by applying the Interval Bound Propagation (IBP) training. Specifically, IBP training is a certifiably robust training method that minimizes the upper bound on the worst-case loss induced by any combination of word substitutions. It computes a tractable upper bound on the loss of the worst-case perturbation to ensure robustness against all possible word substitutions within a defined perturbation set. The effectiveness of IBP training is measured by its certified accuracy, which provides a certificate of robustness. IBP-trained models have shown significantly higher robustness. Besides, the key differences between these methods lie in their objectives and outcomes. The genetic attack serves as a tool to test model vulnerabilities by actively seeking weaknesses, whereas IBP training is a proactive approach that focuses on building inherent robustness into the model during the training phase. These methods highlight the contrasting approaches in adversarial machine learning: attacking or testing models versus defending or fortifying them.

In 2020, Linyang Li and colleagues [45] introduced the BERT-Attack, leveraging the capabilities of pre-trained masked language models like BERT. This approach repurposes BERT to challenge its fine-tuned derivatives and other sophisticated neural models in downstream tasks. Demonstrating superior effectiveness, their technique adeptly deceives target models into erroneous predictions, surpassing existing attack methods in terms of success rate and perturbation percentage. Notably, the adversarial samples produced are both linguistically coherent and semantically intact. They have similar strategies to the previous method, finding vulnerable words and substituting them. Since the pre-trained model became a hot topic in natural language processing, Li fine-tuned BERT, a famous pre-trained masked language model, to substitute these vulnerable words.

Ren et al. [67] proposed a rank-based attacking method that uses saliency scores to identify and prioritize words in a text for synonym-swap transformations. This method involves using WordNet [53] to find synonyms or similar entities for each word, then selecting the substitute word that most significantly alters the text's classification probability. By replacing the original words with these substitutes, a new version of the text is generated. The effectiveness of each substitution is evaluated based on the extent to which it changes the classification probability, thereby identifying the most impactful words for adversarial attacks on classification models.

Jin et al. [33] build upon the concept introduced by PWWS, but they adopt a different approach for crafting ranks. In their method, the importance score for a word $w_i$ is calculated based on the change in prediction probability before and after the word is deleted. This approach provides a nuanced understanding of each word's impact on the overall classification. Similarly, Garg and Ramakrishnan [17] follow the same foundational idea but introduces another variation. Their method involves replacing and inserting tokens in the original text. This is achieved by masking a portion of the text and then using the BERT Masked Language Model (MLM) to compute the difference in logits. Both approaches offer alternative methods to assess the influence of specific textual modifications on classification outcomes.

The strategy of finding appropriate word substitutions while preserving semantic meaning is commonly used in word-level attacks. One straightforward approach is to search the embedding space, where candidates with minimal distances can be considered suitable replacements for the original word. However, this method requires computationally expensive metric calculations, which is not ideal. To address this, Zang et al. [90] proposed a sememe-based word substitution technique using Particle Swarm Optimization (PSO) in their work. Sememes are the smallest semantic units in human language. The authors argued that existing methods based on word embeddings and language models can provide numerous replacements but they may not always be semantically correct or contextually relevant. In comparison, their sememe-based approach outperformed the previously mentioned methods, as they demonstrated through their comparisons.

Unlike previous approaches, Zhang et al. [92] employed the Metropolis-Hastings Algorithm (MHA), a Markov Chain Monte Carlo method, to generate adversarial examples in natural language processing. This method overcomes the limitations of traditional optimization techniques, which often struggle with the discrete and complex nature of sentence space, leading to non-fluent adversarial examples. MHA operates in both white-box and black-box settings, with the white-box version utilizing gradients to propose new examples while the black-box version selects target words randomly. This results in superior performance of the white-box attacks compared to the black-box ones. The effectiveness of MHA is demonstrated through experiments on the IMDB and SNLI datasets, where it not only produces more fluent adversarial examples but also improves the robustness and performance of NLP models via adversarial training. MHA marks a significant advancement in the field by creating adversarial examples that are both effective in misleading models and linguistically plausible.

Cheng et al. [6] tackled the intricate task of generating adversarial examples for sequence-to-sequence (seq2seq) models, characterized by discrete textual inputs and outputs with extensive variability. Addressing the challenges of discrete inputs, they introduce a technique that merges projected gradients, group lasso, and gradient regularization. For handling the expansive and diverse output space, they innovatively craft loss functions specifically designed for distinct non-overlapping attacks and attacks targeting specific keywords. This approach is applied to machine translation and text summarization tasks, demonstrating its effectiveness: by altering less than three words, the seq2seq model can be made to produce desired outputs with high success rates. This research provides a significant contribution to the field, particularly in understanding and manipulating seq2seq models, a domain less evaluated compared to CNN-based classifiers. Similarly, Tan et al. [78] introduced a method called Morph to attack sequence-to-sequence models. Morph employs a greedy search strategy, focusing on words classified as nouns, verbs, or adjectives. The goal is to identify and substitute these words with synonyms in a manner that maximally decreases the BLEU score, a metric commonly used for evaluating machine translation quality. By strategically modifying these key words in the source text, Morph aims to significantly impact the translation output. This method allows for precise manipulation of the seq2seq model, targeting its translation capabilities by altering specific parts of the input text.

Dianqi Li et al. [43] proposed CLARE (ContextuaLized AdversaRial Example), which is a text generation model for creating adversarial examples in natural language processing (NLP) tasks. It employs a mask-then-infill procedure to perturb input text while maintaining its similarity to the original text. CLARE employs three contextualized perturbation actions: Replace, Insert, and Merge. In the Replace action, a token at a specific position is substituted with an alternative token. The original token is replaced with a mask, and a candidate token is chosen from a set based on the probabilities assigned by a masked language model. The selected token minimizes the probability of the gold label predicted by the victim model. The Insert action adds extra information to the input by inserting a mask after a token. The masked language model is used to select a candidate token that fits the unmasked context. The Merge action masks a bigram (two consecutive tokens) by replacing it with a mask. The alternative token is selected using the same approach as in the Replace and Insert actions. CLARE applies these perturbation actions iteratively, selecting the highest-scoring action at each step based on its potential to confuse the victim model. By minimizing modifications to the original input, CLARE generates adversarial examples that preserve textual similarity, fluency, and grammaticality. From their experiments and human evaluation, CLARE outperforms baseline methods and strikes a better balance between successful attacks and preserving input-output similarity. Overall, CLARE employs contextualized perturbations, masked language modeling, and scoring mechanisms to generate adversarial examples that maintain the characteristics of the original text across various NLP tasks.

The A2T [89] method enhanced the speed and efficiency of generating adversarial examples in natural language processing. It achieves this through two key innovations: (1) a gradient-based word importance ranking, which calculates the importance of each word in the input text using the gradient of the loss, thus requiring only one forward and backward pass, significantly reducing the number of necessary forward passes for each word, and (2) the use of DistilBERT [72], a semantic textual similarity model, instead of larger encoders like universal sentence encoder (USE) [88]. DistilBERT demands significantly less GPU memory and computational resources. A2T also employs precomputed top-k nearest neighbors in a counter-fitted word embedding for word substitution, ensuring better synonyms and faster attacks. Additionally, an alternative variation named A2T-MLM uses a BERT-based masked language model for word substitution, focusing on preserving grammatical and contextual coherence. According to their experimental results, these improvements result in a faster and more efficient method than previous approaches.

## 2.4 Sentence Level Attack

The thesis presents a comprehensive overview of sentence-level attacks in natural language processing, focusing on various manipulative strategies such as swapping, inserting, deleting, and generating parts of sentences. These methods are detailed in a structured format in Table 2.5, clearly categorizing and comparing different techniques used in sentence-level adversarial attacks. This table is a valuable resource for understanding the diverse approaches employed in altering sentence structures to test and enhance the robustness of NLP models.

Table 2.5: Sentence level attack with application

| Year | Reference | Attack Type | Application |
|---|---|---|---|
| 2017 | Jia and Liang [31] | White box& black box | Question Answering |
| 2017 | Zhao et al. [95] | Black box | Natural Language Inference |
| 2018 | Wang and Bansal [84] | White box | Classification |
| 2019 | Cheng et al. [7] | White box | Machine Translation |

The research by Jia and Liang [31] introduced a novel approach to evaluating natural language processing (NLP) systems, particularly in the context of the SQuAD reading comprehension task. This method, known as adversarial evaluation, diverges from traditional AI evaluation techniques that typically measure average error across a standard test set. Adversarial evaluation specifically focuses on challenging systems with inputs deliberately designed to test their deeper understanding of language. The study implements two distinct types of adversaries for this purpose: ADDSENT and ADDANY. ADDSENT creates adversarial examples by appending sentences to the text that are grammatically coherent and contextually related to the question, yet they do not contradict the correct answer. On the other hand, ADDANY introduces an even higher level of complexity by adding random sequences of English words. These additions aim to confuse the models further. Experimental results demonstrate that both ADDSENT and ADDANY effectively disrupt the typical evaluation mechanisms in question-answering tasks, proving to be potent tools in assessing the true capabilities of NLP systems while still aligning with the correct answers.

The study by Zhao et al. [95] introduces an innovative framework that leverages Generative Adversarial Networks (GANs) to create adversarial examples in natural language processing. This framework is applied to the Stanford Natural Language Inference (SNLI) dataset, with the goal of generating adversarial examples that are grammatically coherent, semantically close to the original input, and capable of revealing the local behavior of models by exploring the semantic space of continuous data representations. To assess the effectiveness of their approach, the researchers implemented these adversarial examples in various applications, including image classification, machine translation, and textual entailment. Their findings demonstrate that the model is adept at producing adversaries that not only maintain logical coherence and common-sense reasoning but also expose vulnerabilities in models like Google Translate during machine translation tasks. This approach is notably distinct from other methods available at the time, primarily because it focuses on generating textual content instead of the usual manipulations of the given input.

In their work, Wang and Bansal [84] made significant improvements to the *ADDSENT* model, originally based on *ADDSENT* by Jia and Liang [31]. They introduced two modifications, collectively referred to as *ADDSENTDIVERSE*. While the *ADDSENT* model generated fake answers mimicking question syntax but without semantic relevance, *ADDSENTDIVERSE* aimed to produce adversarial examples with increased variability by randomizing distractor placements. Additionally, they integrated semantic relationship features from *WordNet* to address antonym-style semantic perturbations present in *ADDSENT*. Their results showed that the *ADDSENTDIVERSE* model outperformed the *ADDSENT* trained model, achieving an average improvement of 24.22% in F1 score.

In the realm of neural machine translation, Cheng et al. [7] introduced *AdvGen*, a gradient-based white-box attack technique. In their research, the authors adopted a greedy approach steered by training loss to efficiently identify optimal solutions for generating adversarial examples. A key aspect of their methodology was the integration of a language model. This inclusion was strategic, as language models are computationally inexpensive and aid in maintaining a degree of semantic coherence in the adversarial outputs. This approach not only facilitates the generation of challenging adversarial attacks but also significantly contributes to the development of adversarial training methods. By utilizing adversarial examples in training, the researchers aimed to enhance the robustness and resilience of models, ensuring better performance against potential attacks. Their work emphasizes the dual utility of adversarial examples, both as tools for testing model vulnerabilities and as means to fortify models against such vulnerabilities.

## 2.5 Discussion and Challenges

In this section, we will discuss the three levels of attacks in natural language processing (NLP), provide a summary of each, highlight some interesting findings, and discuss the challenges that researchers face in this area.

The three levels of attacks that exist in natural language processing (NLP) are character level, word level, and sentence level. In character-level attacks, the adversarial examples are generated by modifying the characters within keywords. Although this attack method can be useful in some scenarios, it is easily detectable by humans due to the altered characters, which limits its effectiveness. In addition, word-level attacks are known to be more detrimental to the models, and the existing methods typically involve identifying significant or vulnerable words and manipulating them through substituting, swapping, deleting, or inserting. Substituting important words with other phrases can maintain the original semantic meaning and avoid detection by humans. As a result, word-level attacks are more powerful and harmful than character-level attacks. When it comes to sentence-level attacks, two trends are currently prevalent. The first trend involves inserting or deleting sentences, which is similar to word-level attacks. The second trend is text generation, where the attacker uses text generation techniques to produce text that generates an adversarial example. However, due to the lack of maturity of text generation techniques, the text produced is often unclear and long-winded, reducing the effectiveness of this approach.

According to Wu's study [85], word-level attacks achieved the best scores; thus, we will focus on the challenges in word-level attacks. Researchers face several challenges in this area. One major challenge is the lack of attention given to adversarial learning in NLP. Most research in adversarial learning has focused on computer vision, but there are significant differences between the two domains. For instance, while image data is continuous, text data is discrete. Therefore, attacks developed for the image domain cannot be utilized in the text domain.

Another significant challenge in word-level attacks in NLP is crafting attacks that are both imperceptible and maintain fluency. The inherent discreteness of language makes it easy for humans to spot alterations in text. This sensitivity to modifications poses a hurdle in developing attacks that remain unnoticed by the human eye. Moreover, preserving the fluency of the generated adversarial examples is crucial. While much of the existing literature focuses on synonym substitution based on thesauri, this approach often neglects the overall textual meaning, leading to a loss of coherence in the modified text. The failure to consider the global context and semantics of the text can result in adversarial examples that are syntactically correct but lack natural flow and coherence, undermining their effectiveness and detectability. Therefore, developing methods that subtly alter text while retaining its original meaning and fluency remains a key challenge in the field.

# Chapter 3

# Attacking Neural Machine Translation via Hybrid Attentive Learning

Deep-learning based natural language processing (NLP) models are proven vulnerable to adversarial attacks. However, there is currently insufficient research that studies attacks to neural machine translations (NMTs) and examines the robustness of deep-learning based NMTs. In this chapter, we aim to fill this critical research gap. When generating word-level adversarial examples in NLP attacks, there is a conventional trade-off in existing methods between the attacking performance and the number of perturbations. Although some literature has studied such a trade-off and successfully generated adversarial examples with a reasonable amount of perturbations, it is still challenging to generate highly successful translation attacks while concealing the changes to the texts. To this end, we propose a novel Hybrid Attentive Attack (HAA) method to locate language-specific and sequence-focused words and make semantic-aware substitutions to attack NMTs. We evaluate the effectiveness of our attack strategy by attacking three high-performing translation models. The experimental results show that our method achieves the highest attacking performance compared with other existing attacking strategies.

This chapter methodically unfolds the intricacies of the Hybrid Attentive Attack (HAA) method. This part introduces the attention mechanism and pre-trained models, which we'll explore more in Section 3.1. The core methodology of HAA is then thoroughly explicated in Section 3.2. Following this, the chapter evaluates the performance of HAA through empirical analysis in Section 3.3 and investigates its transferability and attack preferences in Sections 3.4 and 3.5, respectively. The chapter culminates in Section 3.6, where it presents potential directions for future research. Specifically, the main contributions of this chapter are as follows:

- We propose a novel Hybrid Attentive Attack (HAA) method which identifies the most influential words in an input sequence based on language-specific and sequence-centered attentions.
- We introduce a semantic-aware word substitution strategy for the proposed HAA method to strike a balance between attack effectiveness and imperceptibility.
- We conduct extensive experiments on real-world datasets with three state-of-the-art victim NMTs. Experimental results demonstrate that our proposed method achieves the best performance with a small number of perturbed words.

## 3.1 Preliminary

In this section, we will introduce some preliminary knowledge about the attention mechanisms and BERT variants.

### 3.1.1 Attention in NLP

Originally inspired by human cognitive processes, the concept of attention was subsequently adapted for machine translation to facilitate automatic token alignment [28]. The attention mechanism, a technique for encoding sequence data by assigning importance scores to each element, has seen extensive application and notable enhancements in diverse natural language processing tasks, such as sentiment analysis, text summarization, question answering, and dependency parsing. This section will delve into relevant studies on the application of the attention mechanism in NLP.

The traditional machine translation models [35] are constructed by an encoder-decoder architecture, both of which are recurrent neural networks. An input sequence of source tokens is first fed into the encoder, with which the tokens will be transferred to the hidden representations, and then the decoder will utilize these hidden representations from the encoders as the initial input and output a sequence of dependent tokens. Such an encoder-decoder framework had achieved highest performance compared to purely statistical machine translation models. This design faces two significant challenges. Initially, RNNs struggle with retaining older data, often discarding it after it has passed through several time steps. Additionally, the lack of specific word alignment in the decoding phase results in a diffused focus across the entire sequence. To overcome these obstacles, Bahdanau [12] pioneered the use of attention in encoder-decoder NMT frameworks, which rapidly became a critical component in sequence-to-sequence models within the NMT field. Bahdanau provided such an attention mechanism to model word alignments between input and output sequence, which is an essential aspect of structured output tasks such as translation or text summarization. Based on Bahdanau's attention, Luong proposed two attention models, namely local and global, in context of machine translation tasks [48]. The global attention model is similar to Bahdanau's attention while the local attention is computed with hidden states from the output of the encoder. Luong's attention achieved a better performance than Bahdanau's attention and provided a way of transparentizing the NMTs.

Recurrent architectures rely on sequential processing of input at the encoding step that results in computational inefficiency, as the processing cannot be parallelized [83]. To address this, Vasiwani proposed Transformer architecture that eliminates sequential processing and recurrent connections. Specifically, transformer-based architectures, which are primarily used in modelling language understanding tasks, avoid recurrent structure in neural networks and instead trust entirely on self-attention mechanisms to draw global dependencies between inputs and outputs. To be more specific, the transformer views the encoded representation of the input as a set of key-value pairs,$(K, V)$, whose dimension equals input sequence length. For the decoder, the previous output is compressed into a query $Q$ and the next output is produced by mapping this query and the set of keys and values. Referring to Bahdanau's and Luong's attention, the transformer adopts the scaled dot-product attention: the output is a weighted sum of the values, where the weight assigned to each value is determined by the dot-product of the query with all the keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{n}}\right)\mathbf{V}.$$

Transformer architecture achieved significant parallel processing, shorter training time, and higher accuracy for machine translation without any recurrent components. Besides, self-attention can provide correlations among the contextual words for NLP models, which we will utilize in our proposed algorithm.

### 3.1.2 BERT and Its Variations

Born from the Transformer architecture [83], BERT, which stands for Bidirectional Encoder Representation Transformer [9], undergoes training through two unsupervised tasks: masked language modeling and next sentence prediction. BERT models are extensively pre-trained on vast amounts of unannotated text, enabling fine-tuning for specific tasks and datasets through transfer learning. Thanks to its superior model structure and extensive training data, BERT has consistently achieved state-of-the-art results in numerous NLP tasks [9]. Beyond its accomplishments in language comprehension, BERT has also emerged as a groundbreaking framework for a wide range of natural language processing tasks, including sentiment analysis, sentence prediction, summarization, question answering, natural language inference, and various others.

Over time, numerous new models have drawn inspiration from the BERT architecture but have been tailored to different languages or fine-tuned on domain-specific datasets. One prominent variant of BERT is RoBERTa [47], known as the Robustly Optimized BERT Pretraining Approach, designed to enhance the training process. RoBERTa was developed by extending the training duration of the BERT model, utilizing larger datasets containing longer sequences, and employing larger mini-batches. Through these adjustments, RoBERTa achieved significantly improved results while also incorporating certain modifications to BERT's hyperparameters. Furthermore, RoBERTa does not employ next sentence prediction (NSP) and employs dynamic word masking as part of its approach.

Another notable iteration of the BERT model, referred to as ALBERT [42], aimed to improve upon the training process and outcomes achieved by the BERT architecture. ALBERT introduced techniques such as parameter sharing and factorization to reduce the total number of parameters. The BERT model encompasses millions of parameters, with BERT-Base consisting of approximately 110 million parameters, which not only makes training challenging but also places a heavy computational burden. In response to these challenges, ALBERT was introduced with a reduced parameter count, providing a solution to the issue of excessive parameters associated with BERT.

## 3.2 Methodology

In this section, we first introduce and formulate the attention mechanism in NMT. Then, we elaborate on the proposed two-step attentive adversarial attack on NMTs, which features an attentive word location and a semantic-aware word substitution. Specifically, we first calculate the Hybrid Attention weights consisting of the language-specific translation attention and sequence-centered self-attention to locate the sensitive words. Then, we target to find replacement words using costume-designed selection steps to ensure parsing correctness and semantic preservations.

### 3.2.1 Attentions in NMT

Bahdanau [12] proposed the attention mechanism to help the word alignments, especially for long sentences. We argue that such an attention mechanism reflects the contributions of each input word to the translated results. Therefore, a small perturbation to the most contributing word will have a heavy influence on the translation. The attention model utilizes an encoder-decoder framework for each step $j$; during decoding, they compute an attention score $\alpha_{ji}$ for hidden representation $\boldsymbol{h}$ in $i$ of each input token to obtain and the formulation is below:

$$e_{ji} = a\left(\boldsymbol{s}_{i-1}, \boldsymbol{h}_j\right) \tag{3.1}$$

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T} \exp\left(e_{ik}\right)} \tag{3.2}$$

$$c_j = \sum_{j=1}^{T} \alpha_{ji} \boldsymbol{h}_i \tag{3.3}$$

, where $e_{ji}$ is output of an alignment model $a$, usually a forward neural network, and $\boldsymbol{s_i}$ is the decoder RNN hidden state for time $i$. Using $e_{ji}$, one can score how well the inputs around position $j$ and the output at position $i$ match. $c_{ji}$ is the encoded sentence representation with respect to the current element $\boldsymbol{h}_j$ to measure its similarity with output sequence ($\mathrm{y_1}$,$\mathrm{y_2}\ldots\mathrm{y_t}$), where $\mathrm{y_1}$ is the $t$-th output tokens. The diagram for this attention model is demonstrated in Fig 3.1.

Self-Attention [83] can be applied to many other kinds of NLP tasks besides machine translation. Different from a translation task, the goal is to learn the dependencies between the words in a given sentence and use that information to capture the internal structure of the sentence. In self-attention, there are 3 important variables, Q,K and V, which are vectors used to get better encoding for both our source and target words. All of these three variables are hidden representations from the linear layer. Furthermore, the attention weights of self-attention are also calculated differently from Bahdanau's attention; the formulation is below:

$$\mathrm{Self} = \mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^{\top}}{\sqrt{(d_k)}}\right)\mathbf{V}. \tag{3.4}$$

where $d_k$ is the number of dimensions for key vector $K$. We argue to attack NMTs using self-attention too, as an disturbance to the dependency of source language can also deprave the translation quality.

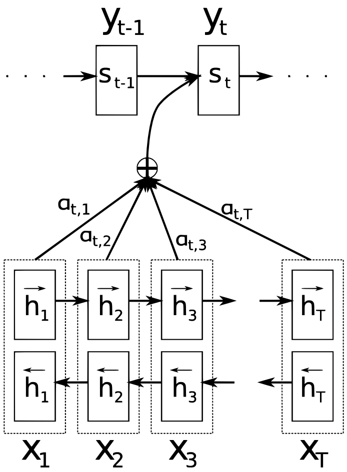


Figure 3.1: Illustration of an attention-based NMT model [12] with RNN based encoder-decoder structures, generating the t-th target token $y_t$ given a input sentence ($x_1$, $x_2$, . . . , $x_T$).

.

### 3.2.2 Problem Formulation

Denoting the source sequence as $S$, the translated target sequence as $Y$, a NMT model can be defined as $f(S) : S \rightarrow Y$. We denote $S = [w_1, \ldots, w_n]$ and $Y = [h_1, \ldots, h_k]$, where $w$ and $h$ denote the words in the source and target sequence, while $n$ and $k$ are the number of words in each respective sequence. To ensure the attack's applicability, we assume a black-box setting where the attacker can only query the NMT model for translated results of a given input, and does not have access to the model parameters, gradients or training data. For an input pair $(S, Y)$, we want to generate an adversarial example $S_{adv}$ such that $f(S_{adv})$ has an obvious semantic difference from $Y$. Additionally, we want $S_{adv}$ to be grammatically correct and semantically similar to $S$.

### 3.2.3 Attentive Word Location

Attention weights in NMT models can be seen as the strength of semantic association between the source and target tokens, by adopting such a mechanism, the performance NMTs are boosted [12]. Hence, we argue that NMTs can be crashed if the attention mechanism is tampered, and the best way of tampering attention is to adopt attention mechanism itself. In this subsection, we introduce the proposed attentive word location scheme and demonstrate different attentive NMT attack implementations based on language-specific and sequence-centered attentions.

## Translation Attentive Attack

Since translation is a cross-language task defined by the source and target languages, it is intuitive to pose language-specific attacks to challenge NMTs' robustness. To this end, we propose a Translation Attentive Attack (TAA) mechanism that focuses on influential words in the translation towards a certain target language. Concretely, we obtain such an attention $\mathcal{A}$ that measures word-wise importance in a specific translation task based on a contextual NMT model [12].

To calculate $\mathcal{A}$, we feed the NMT model with the source sequence to get the translated result $\widehat{Y} = [\widehat{h}_1, \ldots, \widehat{h}_{k'}]$, where $k'$ is the number of words in the attacked target sentence. We then extract a correlation matrix $\mathcal{A}$ from the softmax layer in the model's decoder, thereby formulating the process as $\mathcal{T}(S) : S \rightarrow \mathcal{A}$. The elements in the correlation matrix $\mathcal{A}$ describe the probability distributions of translated words in the target language conditioned on the source sequence $S$, which can be written as:

$$a_{ij} = P(\widehat{h}_j | [w_1, \ldots, w_i, \ldots]) = \frac{\exp\left(e_{ij}\right)}{\sum_{i=1}^{n} \exp\left(e_{ij}\right)}, \tag{3.5}$$

where $P$ denotes probability, and $e_{ij}$ denotes the feature in the model depicting the matching degree between the predicted word $\widehat{h}_j$ in the target language and the input word $w_i$ in $S$. The conditional probabilities reveal the correlation between the input sequence and the predicted sequence in the target language. Given its softmax-normalized distribution, we have $\sum_{i=1}^{n} a_{ij} = 1, \forall j$, therefore it is intuitive to measure $w_i$'s contextual contribution to a translated word $\widehat{h}_j$ using $a_{ij}$ straightforwardly. Further, to find the most influential input words in the translation process, for the whole predicted sequence, we define the language-specific word-wise attention by summing the matrix elements by index $j$, as $\mathbb{A} = [A'_1, \ldots, A'_i, \ldots, A'_n]$, where $A'_i = \sum_{j=1}^{k'} a_{ij}$.

We can sort the words of the source sequence according to such an attention weight, $\mathbb{A}$, for the first step, and select the top language-specific influential words as the victim words for substitution in the second step, which will be introduced in Section 3.2.4.

### Self-attentive Attack

Beside the language-specific attack that focuses on the translation task between two languages above, the inherent semantics of the input sequence can also be tampered. Thus we propose a sequence-centered Self-Attentive Attack (SAA) which exploits attention from the input sequence itself. We utilize the transformer model [83], $\mathcal{V}(S) : S \to \mathcal{B}$, to extract the self-attention matrix $\mathcal{B}$, whose elements $b_{ij}$ indicate the word-wise weights given positional encodings. Particularly, since such weights are obtained via softmax activation, they are also naturally normalized ($\sum_{i=1}^{n} b_{ij} = 1, \forall j$), and thus they are suitable to quantitatively measure the dependencies among words across the entire input sequence. Therefore, similar to the first step in TAA, we define the sequence-centered self-attention weight as $\mathbb{B} = [B'_1 \dots B'_i \dots B'_n]$, where $B'_i = \sum_{j=1}^{n} b_{ij}$.

Different from the language-specific attention in TAA that emphasizes on contextual alignment between source and target sequences, the sequence-centered attention in SAA can explore long-range dependencies within the input sequence itself, better indicating the word-wise influence on overall language understandings of the sequences.

### Hybrid Attentive Attack

As analyzed above, the translation-attentive attack and self-attentive attack focus on different aspects of NMTs, *i.e.*, the cross-language context alignment and the overall semantic understanding of the source sequence, respectively. We argue that both the two aspects are crucial for NMTs, and an ideal attack for NMTs should combine their advantages. Thus we propose a Hybrid Attentive Attack (HAA) scheme which comprehensively

**Algorithm 1** Hybrid Attentive Attack (HAA)

**Input:** Source and Target sentence pair $S, Y$, number of perturbed words $N$, number of adversarial candidates

**Model:**$\mathcal{T}$: Translation attentive model for TAA. $\mathcal{V}$: Self-attentive transformer for SAA. $\mathcal{M}$: MLM model for word substitution.

**Output:**Adversarial Examples $S_{adv}$

Tokenize $S$
$\mathcal{A} \leftarrow \mathcal{T}(S)$ ▷ elements in $\mathcal{A}$ are represented by $a_{ij}$
$\mathbb{A} \leftarrow \sum_{j=1}^{n} a_{ij}$
$\mathcal{B} \leftarrow \mathcal{V}(S)$ ▷ elements in $\mathcal{B}$ are represented by $b_{ij}$
$\mathbb{B} \leftarrow \sum_{j=1}^{n} b_{ij}$
$\mathbb{H} \leftarrow (1-\lambda)\mathbb{A} + \lambda\mathbb{B}$
$S_{mask} \leftarrow$ mask the top $N$ tokens by $\mathbb{H}$ scores
$\mathbb{S}_{can} = [\quad]$ ▷ create an empty set
**for** i in range($N$) **do**
    $S'_{can} \leftarrow$ the $i$th highest replacement from $\mathcal{M}(S_{mask})$
    **if** $S'_{can} \neq S$ **then**
        $\mathbb{S}_{can}$.append($S'_{can}$)
    **end if**
**end for**
$S_{adv} \leftarrow$ the element of $\mathbb{S}_{can}$ that has the highest semantic similarity with $S$
$S_{adv} \leftarrow$ Detokenize $S_{adv}$
**return** $S_{adv}$

considers the word influence by combining the attention weight from TAA and SAA:

$$\mathbb{H} = (1-\lambda)\mathbb{A} + \lambda\mathbb{B}, \tag{3.6}$$

where $\mathbb{H} = [H'_1 \dots H'_i \dots H'_n]$ and $H'_i$ is the final influence weight for word $w_i$ in the input sentence. The optimal parameter $\lambda$ can be found by a greedy search based on the attack performance measured by BLEU on translated results. The overall workflow of the HAA model is demonstrated in Algorithm 1 with an example shown in Figure 3.2.

### 3.2.4 Semantic-aware Word Substitution

In the above subsection, we locate the most influential words in the input sequence to be attacked. An ideal attack should guarantee sufficient concealment besides having attack effectiveness, enabling the adversarial example to avoid being noticed by the NMT

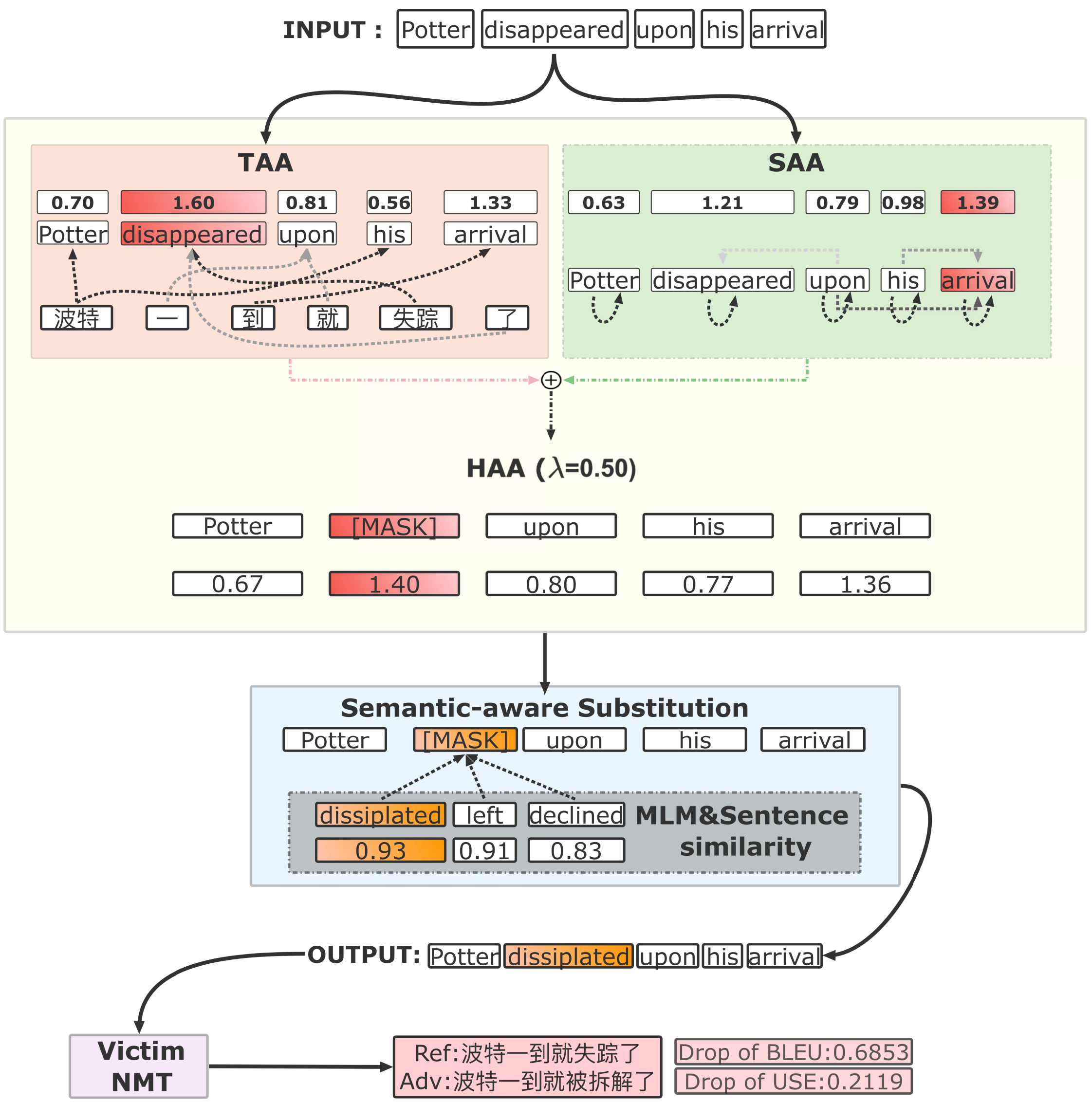


Figure 3.2: An illustrated example of our HAA model. In this example, HAA generates an adversarial example with one word perturbed to attack an English-Chinese translation. The arrows inside the TAA box, and those in the SAA box, respectively represent the utilisation of translation and self-attention weights. The numbers inside the semantic-aware substitution box represents the sentence-level semantic similarity. The TAA, SAA, HAA and Semantic-aware Substitution workflows are reflected in Lines 2-3, Lines 4-5, Lines 6, and Lines 7-15 in Algorithm 1, respectively.

model. Therefore, we further argue a qualified adversarial example $S_{adv}$ should preserve semantics and be grammatically correct, constraining reasonable deviations from the original input sequence.

We propose to design such a semantic-aware word substitution approach based on the semantic feature similarity between the tampered sequence and the original one. We mask a victim word one at a time by a descending order of the attention score to get $S_{mask}$, and utilise an MLM model $\mathcal{M}(S_{mask}) : S_{mask} \rightarrow S'_{can}$, where $S'_{can}$ is a mask-filled sentence. At each iteration, we utilize $\mathcal{M}$ to generate $n^*$ best adversarial example candidates, $\mathbb{S}_{can} = [S'_{(can,1)}, \ldots, S'_{(can,p)}, \ldots, S'_{(can,n^*)}]$, according to corresponding logits from $\mathcal{M}$, and we use a pre-trained semantic retrieval model, universal sentence encoder (USE) [88], to calculate the cosine feature distance between the candidate $S'_{(can,p)}$ and the original sequence $S$. Then we select $S_{adv}$ with highest similarity to the original one as the adversarial example. By such a semantic-aware word substitution, we can complete the NMT adversarial attack process and strike a balance between influencing the translation result and concealing the perturbations with similar semantics.

## 3.3 Experiments

We empirically evaluated and assessed our proposed attacking strategies (TAA, SAA and HAA) on a task of translating English to Chinese to three well-performed world-leading NMTs: Google Cloud Translation, Baidu Cloud Translation and Helsinki NMT [81]. To deeply explore the attacking performance, we not only attack the victim model but also make transfer attacks which utilize the adversarial examples generated on one victim NMT to attack other NMTs.

### 3.3.1 Datasets

To get sufficient training data, we utilized 4 datasets as our training set for training the language-specific NMT and sequence-centered transformer models utilized for the TAA, the SAA, and the MLM for semantic-aware word substitution. Three of the training sets are Commentary [80], Infopankki [81] and the Openoffice [81], are publicly available, while the other, YYeTs subs[1], is scripted by us from YYeTs website (provided in the codes of corresponding paper [58]), which provides human translated movie and drama subtitles. The details of the train set can be found in Table 3.1.

[1] `https://m.yysub.net/`

Table 3.1: introduces details about datasets used in the experiments.

| Dateset | YYeTs Subs | Comme -ntary | Infop-ankki | Openo-ffice | WMT20 T1 | WMT20 T2 | ALT.P (test) |
|---|---|---|---|---|---|---|---|
| Size | 500k | 69k | 30k | 69k | 6.0k | 6.0k | 1.0k |
| Avg.len | 7.83 | 46.14 | 9.92 | 6.16 | 14.10 | 16.51 | 16.54 |
| Min.len | 1 | 1 | 1 | 1 | 3 | 2 | 2 |
| Max.len | 67 | 229 | 144 | 221 | 130 | 199 | 204 |
| Content | Subs | News | Science | Education | Wiki | Wiki | News |
| Purpose | Training Set | | | | Testing Set | | |

To get reliable experimental results, we test attacking strategies on 3 other public datasets, WMT20 T1, WMT20 T2 [81] and ALT-P(test) [69]. WMT is the main event for machine translation and machine translation research, which provides reliable multilingual datasets from Wikipedia. To diverse the sources of test set, we also include ALT-P dataset on news. The details of the test set can be found in Table 3.1.

### 3.3.2 Victim Models

We test the proposed attacking strategies on three well-performed NMTs: Google Cloud Translation[2] (Google.T), Baidu Cloud Translation[3] (Baidu.T), and Helsinki NMT (Hel.T) [81]. The first two NMTs are cloud translation platforms, which are used for commercial purposes while the other NMT, Helsinki NMT is based on MarianNMT[34] from Microsoft for academic purpose.

### 3.3.3 Baselines

We compare our proposed strategies with 5 word-level attack strategies below:

- RAND: randomly selects victim words in the target sentences and utilize the proposed semantic-aware substitution strategy to construct the adversarial examples.

[2] https://cloud.google.com/translate
[3] https://api.fanyi.baidu.com/

- Morpheus-Attack (Morph) [78], greedily searches for words, from *noun*, *verb*, or *adjective* tags, maximally decreasing BLEU on source language side, and substitute them with synonyms.
- BERT-ATTACK (BERT.A) [45]: utilizes BERT to locate the victim words by ranking the differences between the logits of original words and BERT-predicted words, and then make substitutions with BERT.
- Seq2sick [6]: crafts the adversarial example by depraving the targeted logits of victim NMT with regularization on preserving semantic similarity.
- PSO [90]: selects word candidates from HowNet and employs the PSO to find adversarial text for classifier. We adjust the metric from classification logits to BLEU.

### 3.3.4 Evaluation Metrics

We use metrics based on BLEU and USE [88] to evaluate attacking performance on the target language side and the semantic preservation on the source language side. BLEU evaluates the sentence pairs in term of word alignment while USE is a multilingual pre-trained language model to evaluate the semantic similarity.

Since changes of the original input will always lead to changes of the translated output, we examine how much more changes an attacked output has compared to those of the unattacked translation. So instead of directly using BLEU and USE on translated outputs, we define BLEU drop ratio (BDR) and USE drop ratio (UDR) to evaluate attacks:

$$\text{BDR} = \frac{\text{BLEU}(Y, f(S)) - \text{BLEU}(Y, f(S_{adv}))}{\text{BLEU}(Y, f(S))} \tag{3.7}$$

$$\text{UDR} = \frac{\text{USE}(Y, f(S)) - \text{USE}(Y, f(S_{adv}))}{\text{USE}(Y, f(S))} \tag{3.8}$$

where $S$ and $Y$ denote input sentence and translation reference, and $f(\cdot)$ is the victim

NMT model.

In addition, we also evaluate how much word perturbations are made on the original inputs by using BLEU and USE on the attacked source language. To distinguish from the metrics used on the target language side, we use S-BLEU and S-USE for denoting changes made on the source language.

### 3.3.5 Experimental Settings

This section presents the models used for HAA and the results of greedy searching for $\lambda$. We examine how the number of perturbed words affects attack performance, testing 1 to 5 word perturbations per sentence in our comparisons.

#### 3.3.5.1 Model Structures

In this subsection, we introduce the structure of the language-specific NMT for TAA, transformer for SAA, and the MLM for semantic-aware word substitution. All of these 3 models are trained and fine-tuned on the same train datasets mentioned in Table 3.1.

- **TAA**: The architecture of TAA consists of a 2-layer stacked LSTM, plus a Luong's translation attention layer to process of the output of LSTM [48]. To be more specific, the encoder takes a list of subtoken IDs to an embedding vector for each subtoken via an embedding layer. Further, we processes the embeddings into a new sequence with a LSTM. After encoding, the features of input sentences will be passed into a decoder, and the decoder's job is to generate predictions for the next output token. The decoder receives the complete encoder output and uses a LSTM to keep track of what it has generated so far. To get translation attention, the decoder will utilize its LSTM output as the query to the attention over the encoder's output, producing the context vector. After the LSMT in decoder, we adopt the Luong's translation attention to combine the LSTM output and the context vector generate the translation attention matrix. For the last step, decoders generates logit predictions for the next tokens based on the attention matrix. For the hyper-parameter, we set 1024 hidden units, 256 embedding dimmensions, 64 batch size, with Adam optimizer.

- **SAA**: SAA is designed to get sequence-centered attention weights on the source language, therefore it will be trained with only the data in source language. Since the data is unlabeled and sequential, we utilize BERT-base-uncased [9], one of the best unsupervised language models, as the transformer to extract the sequence-centered attention weights. The hyper-parameters of this model are public available. To adjust the model to our dataset, it will be fine-tuned on our datset with Adam optimizer with learning rate 0.001 and batch size 128.

- **MLM for semantic-aware substitution**: MLMs mask the words in the train set and are given a task to fill these masks, therefore utilize these models can help to find parsing substitutions for the proposed methods. We utilize a public pre-trained model, RoBERTa-large [47], as our candidate to generate parsing and semantic-preserving adversarial examples.

#### 3.3.5.2 Optimization of $\lambda$

In the experiments, our proposed method, HAA, utilizes a greedy search for the best hyper-parameter $\lambda$ to combine language-specific and sequence-centered attention. The objective used for searching is BLEU and the search is within the validation set which contains 1000 samples separated from the training set. Greed search is used for the optimal hyper-parameter $\lambda$ within $[0, 0.01, \ldots, 1]$ with a step size of 0.01 for each victim model and the searched results for the three victim NMTs (Google, Baidu and Helsinki translations) are shown in Figure 3.3.

From search results in Fig 3.3, we can find that the optimal $\lambda$ values for the three victim models are $\lambda_{\text{Google}} = 0.68$, $\lambda_{\text{Baidu}} = 0.47$ and $\lambda_{\text{Hel.T}} = 0.41$. Therefore, we can find the $\lambda$ can be different for different victim NMTs in our experimental settings. Since $\lambda$ is utilized to control the weight of SAA and TAA, it can show the preference between SAA

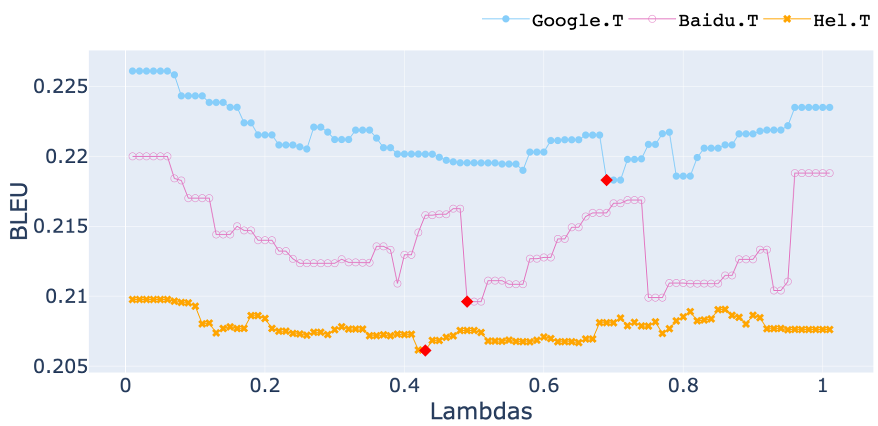


Figure 3.3: The process of searching for the best $\lambda$ for Google, Baidu and Helsinki NMT. The discovered optimal $\lambda$ values are highlighted in red.

and TAA. From the results, we find that for different victim NMTs, the proposed HAA will have different preferences: TAA is preferred for Google translation while SAA is preferred for Baidu Translation and Helsinki Translation. Besides, as $\lambda$ is searched based on the performance of NMTs, there is no doubt that $\lambda$ can be different due to the different NMTs' performance on datasets so that this preference can be different in datasets.

### 3.3.6 Main Results and Analysis

We show the results for greedy searching process in Fig 3.3. The main results of attacking performance and semantic preserving performance on different test data sets are shown in Tables 3.2, 3.3, 3.4, and Figures 3.4, 3.5, and 3.6. In addition to the statistics of the results, an example of learned attentions for the proposed methods is shown in Table 3.5 and an adversarial example is also shown in Table 3.6 to show the differences of attacks. We validate the advantages of our proposed methods (i.e., TAA, SAA and HAA) from the following three aspects:

#### 3.3.6.1 Does HAA have superior attack performance compared to baselines?

We assess the effectiveness of attentive methods (TAA, SAA, HAA) versus non-attentive baselines across datasets, shown in Figures 3.4, 3.5, and Table 3.4, by compar-

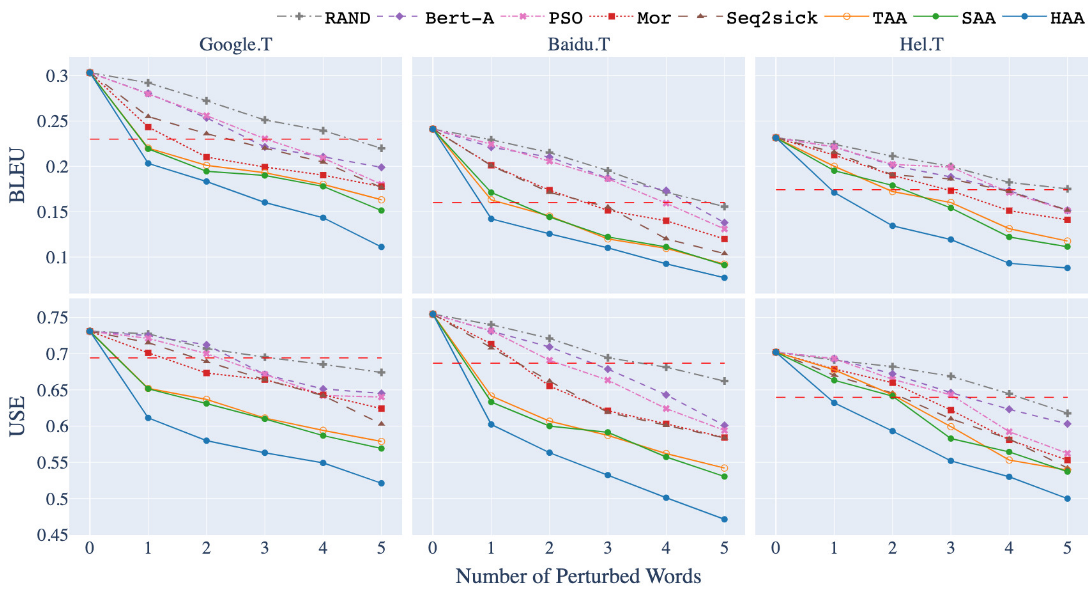


Figure 3.4: Attacking performance (BLEU, USE) on the WMT20 T1 dataset towards different numbers of perturbed words ranging from 1 to 5 for three victims, NMT, Goolge.T, Baidu.T and Helsinki.T.

ing the decreases in BLEU and USE scores for original and attacked translations. It can thus be concluded that the proposed method HAA achieves the best attacking performance, with the largest metric score drops for both word alignment (BLEU) and semantic understanding (USE). Particularly, HAA consistently outperforms other competing methods across different data domains, regardless of the number of perturbed words. Apart from HAA itself, its different attentive components TAA and SAA, also surpass the non-attentive baselines in most cases.

#### 3.3.6.2 Balance between attack performance and the number of perturbed words.

Concerning the trade-off between effectiveness and imperceptibility, we evaluate the attack's imperceptibility from both appearance and semantic modification perspectives, the first of which is the number of words perturbed. As shown in Fig 3.4, Fig 3.5 and Fig 3.6, comparing the numbers of words needed to achieve identical drops of metric scores (marked by the horizontal red dashed lines), we can find that HAA perturbs the fewest words, for it theoretical focuses on the most influential words with both language-specific and sequence-centered attentions. Thus we can conclude that the proposed HAA

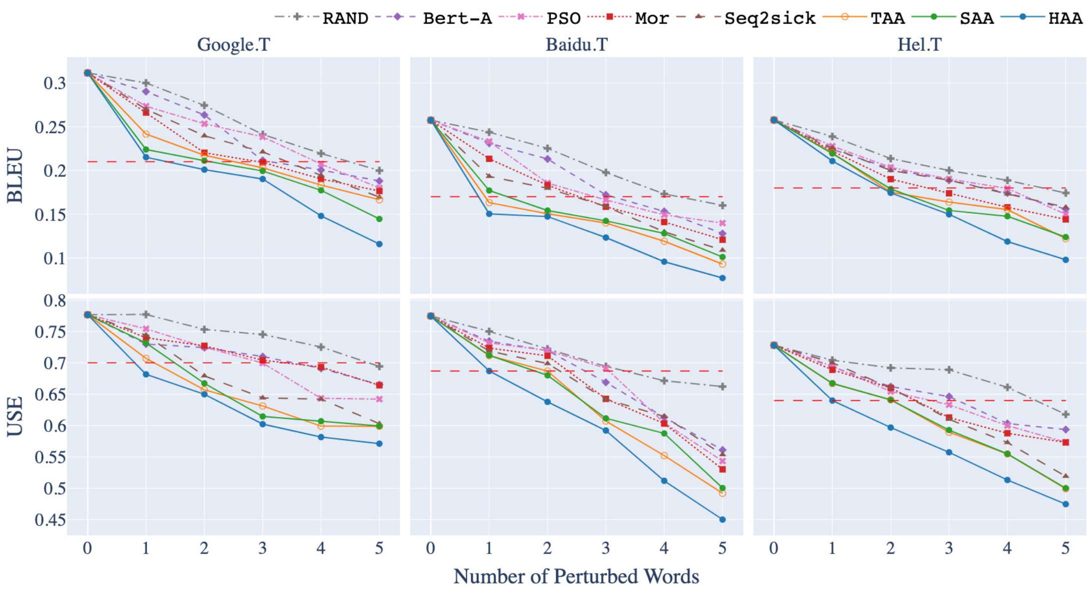


Figure 3.5: Attacking performance (BLEU, USE) to the WMT20 T2 dataset towards different numbers of perturbed words ranging from 1 to 5 for three victims, NMT, Goolge.T, Baidu.T and Helsinki.T.

more successfully balances attacking performance and the appearance modifications to the sequence.

#### 3.3.6.3 How well does HAA reserve the semantic meaning of the original input sentences?

To further investigate the attack's imperceptibility, we evaluate the semantic similarities between the original input sentence and its derived adversarial sample (i.e., S-BLEU and S-USE) shown in Table 3.2, Table 3.3 and Table 3.4 on different datasets. All of the table demonstrates the attacking methods based our semantic-aware substitution, SAA TAA HAA and RAND, are the best methods in most cases in terms of semantic preserving. In some cases, our methods are not the best, but they are still comparable to the best method PSO by a close margin in semantic preservation. However, PSO's preservation comes at the price of much inferior performance, as is shown by its BDR and UDR. Thus we can conclude that proposed HAA provides the one of the best balances between attack performance and semantics preservation.

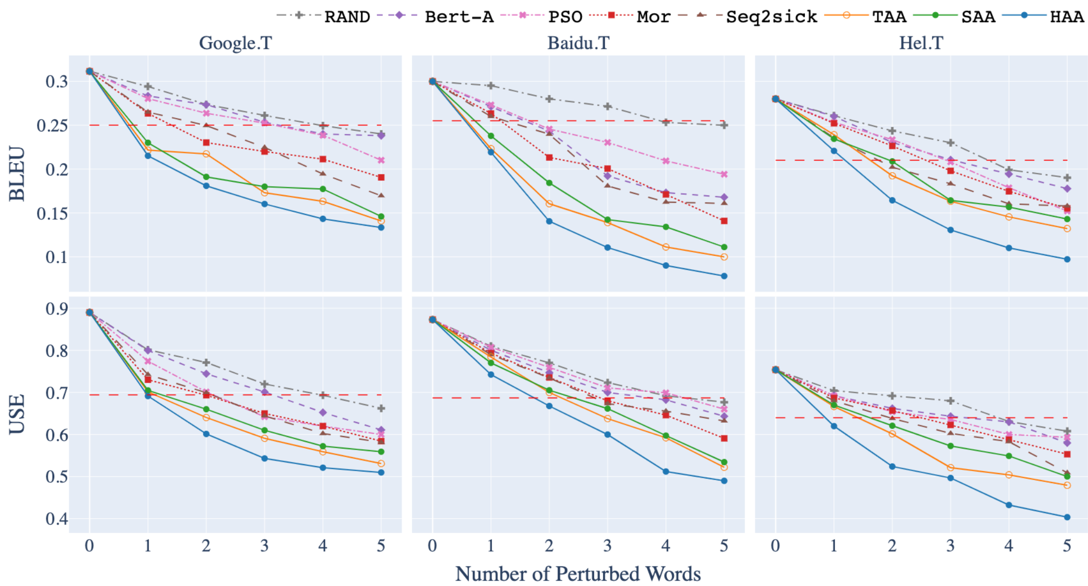


Figure 3.6: Attacking performance (BLEU, USE) to the ALT.P dataset towards different numbers of perturbed words ranging from 1 to 5 for three victims, NMT, Goolge.T, Baidu.T and Helsinki.T. The horizontal red dashed lines indicate the numbers of words needed to achieve identical drops of metric scores.

Table 3.2: Comparisons of performance on WMT20 T1 dataset in terms of semantic preservation (S-BLEU, S-USE) and attacking performance (BDR, UDR) averaged across different number of perturbed words for victim models. For each column, the highest, the second and third highest score are highlighted in bold, underlined and italic, respectively.

| Attack method 1 | Google.T | | | | Baidu.T | | | | Hel.T | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE |
| RAND | 15.94% | 4.53% | 0.3637 | 0.8124 | 19.81% | 7.26% | 0.3600 | 0.8243 | 14.22% | 5.87% | *0.3634* | 0.8225 |
| BERT.A | 22.20% | 9.55% | 0.3622 | 0.7431 | 22.90% | 10.85% | 0.3611 | 0.7723 | 19.20% | 7.77% | 0.3602 | 0.7712 |
| PSO | 23.81% | 7.65% | **0.3659** | 0.8312 | 24.80% | 12.42% | 0.3671 | **0.8325** | 18.37% | 10.11% | 0.3605 | 0.8321 |
| Morph | 32.69% | 7.65 % | 0.3601 | *0.8319* | 34.78% | 15.78% | 0.3618 | 0.8303 | 25.04% | 11.84% | 0.3637 | 0.8301 |
| Seq2sick | 27.97% | 9.34% | 0.3552 | 0.7508 | 37.69% | 15.88% | 0.3621 | 0.7590 | 20.86% | 13.11% | 0.3631 | 0.7545 |
| SAA | 38.48% | 16.59% | 0.3631 | 0.8312 | *46.94%* | 28.04% | *0.3646* | 0.8305 | 34.19% | 14.86% | *0.3634* | **0.8342** |
| TAA | *36.87%* | *15.92%* | 0.3639 | 0.8377 | 47.75% | *22.09%* | 0.3641 | *0.8310* | *32.51%* | *14.19%* | **0.3647** | *0.8333* |
| HAA | **47.17%** | **22.72%** | 0.3639 | **0.8395** | **54.59%** | **25.67%** | **0.3794** | 0.8319 | **47.66%** | **20.03%** | 0.3632 | 0.8334 |

Table 3.3: Comparisons of performance on WMT20 T2 dataset in terms of semantic preservation (S-BLEU, S-USE) and attacking performance (BDR, UDR) averaged across different number of perturbed words for victim models. For each column, the highest, the second and third highest score are highlighted in bold, underlined and italic, respectively.

| Attack method | Google.T | | | | Baidu.T | | | | Hel.T | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE |
| RAND | 20.71% | 4.86% | 0.3609 | *0.8288* | 22.31% | 9.63% | 0.3619 | 0.8288 | 21.18% | 7.59% | 0.3622 | 0.8288 |
| BERT.A | 25.98% | 10.79% | 0.3601 | 0.7660 | 32.07% | 15.04% | 0.3621 | 0.7677 | 26.18% | 13.32% | 0.3634 | 0.7632 |
| PSO | 25.92% | 9.35% | **0.3666** | **0.8400** | 32.60% | 14.87% | 0.3671 | **0.8388** | 26.63% | 12.13% | **0.3655** | **0.8445** |
| Morph | 32.77% | 9.14% | 0.3611 | 0.8339 | 38.77% | 17.11% | 0.3617 | *0.8321* | 31.05% | 14.22% | 0.3647 | 0.8308 |
| Seq2sick | 29.69% | 14.70% | 0.3599 | 0.7538 | 40.13% | 16.66% | 0.3633 | 0.7598 | 26.58% | 15.88% | *0.3643* | 0.7548 |
| SAA | 38.58% | *17.10%* | *0.3643* | 0.8278 | *45.38%* | *20.19%* | *0.3649* | 0.8319 | 36.02% | 18.79% | 0.3639 | 0.8349 |
| TAA | *35.03%* | 17.78% | 0.3645 | 0.8265 | 48.27% | 21.26% | 0.3647 | 0.8311 | *35.05%* | *18.92%* | 0.3641 | 0.8322 |
| HAA | **44.12%** | **20.53%** | 0.3633 | 0.8275 | **53.85%** | **25.67%** | **0.3677** | 0.8332 | **41.64%** | **23.58%** | 0.3642 | *0.8328* |

Table 3.4: Comparisons of performance on ALT.P dataset in terms of semantic preservation (S-BLEU, S-USE) and attacking performance (BDR, UDR) averaged across different number of perturbed words for victim models. For each column, the highest, the second and third highest score are highlighted in bold, underlined and italic, respectively.

| Attack method | Google.T | | | | Baidu.T | | | | Hel.T | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE | Avg. BDR | Avg. UDR | Avg. S-BLEU | Avg. S-USE |
| RAND | 15.34% | 18.04% | 0.3312 | **0.7932** | 10.00% | 15.88% | 0.3410 | **0.7499** | 19.71% | 12.05% | 0.3297 | <u>0.7361</u> |
| BERT.A | 17.28% | 21.17% | 0.3301 | 0.7362 | 30.015% | 18.16% | 0.3329 | <u>0.7477</u> | 23.34% | 14.90% | <u>0.3333</u> | *0.7332* |
| PSO | 20.05% | 25.00% | 0.3262 | <u>0.7410</u> | 23.15% | 16.69% | 0.3252 | 0.7388 | 26.71% | 15.74% | 0.3201 | 0.7155 |
| Morph | 28.36% | 26.35% | 0.3301 | 0.7321 | 34.04% | 21.07% | *0.3371* | 0.7332 | 28.10% | 17.55% | *0.3331* | 0.7201 |
| Seq2sick | 20.05% | 25.01% | 0.3211 | 0.7191 | 33.07% | 20.16% | 0.3231 | 0.7214 | 32.89% | 20.07% | 0.3243 | 0.7011 |
| SAA | *40.63%* | *30.20%* | **0.3643** | 0.7319 | *46.01%* | <u>25.13%</u> | 0.3321 | 0.7329 | *35.18%* | *22.72%* | **0.3400** | 0.7321 |
| TAA | <u>41.17%</u> | <u>32.09%</u> | *0.3334* | 0.7293 | <u>51.06%</u> | <u>25.13%</u> | <u>0.3347</u> | 0.7391 | <u>37.66%</u> | <u>26.43%</u> | 0.3234 | 0.7310 |
| HAA | **46.51%** | **35.59%** | <u>0.3353</u> | *0.7377* | **57.42%** | **31.01%** | **0.3417** | *0.7410* | **48.34%** | **34.30%** | 0.3299 | **0.7388** |

Table 3.5: Examples for attentions learned by proposed methods (TAA, SAA and HAA). The examples are red, blue and green for TAA, SAA, and HAA, respectively. The opacity of each word depends on its corresponding attention weight which is placed in the brackets after each token.

| | |
|---|---|
| TAA | However, [0.68] after [1.58] a [3.26] few [4.16] victories, [6.68] the [6.40] campaign [9.37] falters. [7.74] |
| SAA | However, [0.80] after [0.79] a [0.68] few [0.98] victories, [0.75] the [0.72] campaign [0.79] falters. [0.94] |
| HAA | However, [0.75] after [1.10] a [1.72] few [2.25] victories, [3.12] the [2.99] campaign [4.22] falters. [3.66] |

To further validate the effectiveness of our word replacement strategy, we conduct an extensive experiment on our semantic-preserving performance by a task of substituting the same victim words located by our hybrid attention. We select 3 common substituting baselines:

- Default masked-word filling (HA.Def): utilize MLM to fill the mask without a consideration to the semantic preservation
- Synonyms (HA.Syn): replace the victim words with synonym from the WordNet [53]
- Word embedding distance ranking (HA.Rank): search the word embedding space in GloVe [63] to set the word, with smallest distance ($l_2$) to victim word, as the replacement.

The results from Table 3.7 show that HAA (semantic-aware substitution) achieves the best semantic-preserving performance on attacking the same position. Clearly, HAA can provide more parsing-correct and semantic-preserved adversarial examples than other methods.

Table 3.6: Adversarial examples (adv.) crafted by proposed methods and baselines, and their corresponding translated results (Tran.). The semantic preserving (S-BLEU, S-USE) and attacking performance (BDR, UDR) metrics are provided in the brackets after the adversarial and translated sentence, respectively. The translation attacked by HAA made a completely wrong causality of between the "war" and the "mustache" by stating "The war of beard sparked off this atrocity" in the translation.

| | |
|---|---|
| BERT.A | Adv. This atrocity sparked off the war of the mustache, a millennia ~~spanning~~ long that saw the empires of the Elves and Dwarves crumble into ruins. (S-BLEU: 0.9440, S-USE: 0.9865)<br>Tran.这场暴行引发了胡子战争, ~~在~~ 这场长达数千年的冲突中,精灵帝国和矮人帝国陷入废墟。(BDR: 3.31%, UDR: -3%) |
| Morph | Adv. This atrocity sparked off the war of the mustache, a millennia spanning that saw the empires of the Elves and Dwarves ~~crumble~~ disintegrate into ruins. (S-BLEU: 0.9120, S-USE: 0.9193)<br>Tran.这~~场~~一暴行引发了胡子战争,在长达数千年的~~冲突~~战争中,精灵帝国和矮人帝国~~陷入废墟~~ 解体。 (BDR: -7.76%, UDR: 5.19%) |
| PSO | Adv. This atrocity sparked off the war of the mustache, ~~a~~ one millennia spanning that saw the empires of the Elves and Dwarves crumble into ruins. (S-BLEU: 0.9669, S-USE: 0.9958)<br>Tran.这场暴行引发了胡子战争, ~~在~~长达数千年~~的冲突中~~ 之久,精灵~~帝国~~和矮人帝国~~陷入废墟~~ 崩溃。 (BDR: 23.37%, UDR: 1.90%) |
| Seq2sick | Adv. This atrocity sparked off the war of the mustache, a millennia spanning that saw the ~~empires~~ king of the Elves and Dwarves crumble into ruins. (S-BLEU: 0.9460, S-USE: 0.9663)<br>Tran.这场暴行引发了胡子战争, ~~在~~长达数千年~~的冲突中~~之久,精灵~~帝国~~和矮人~~帝国~~国王们 ~~陷入废墟~~ 都陷入了困境。(BDR: 21.33%, UDR: 5.789%) |
| SAA | Adv. This atrocity sparked off the war of the mustache, a millennia spanning that saw the empires of the Elves and ~~Dwarves~~ Dwarfs crumble into ruins. (S-BLEU: 0.9739, S-USE: 0.9832)<br>Tran.这场暴行引发了胡子战争,长达数千年的~~冲突~~中,精灵帝国和矮人帝国~~陷入废墟~~ 奔溃。 (BDR: 21.33%, UDR: 5.78%) |
| TAA | Adv. This atrocity sparked off the war of the ~~mustache~~ beard, a millennia spanning conflict that saw the empires of the Elves and Dwarves crumble into ruins. (S-BLEU: 0.9329, S-USE: 0.9420)<br>Tran.这场暴行~~引发了胡子战争~~ 是胡子战争引发的, ~~在长达数千年的冲突中~~ 几千年来, 精灵~~帝国~~和矮人帝国~~陷入废墟~~都陷入困境。(BDR: 37.43%, UDR: 13.86%) |
| HAA | Adv. This atrocity sparked off the war of the ~~mustache~~ beard, a millennia spanning conflict that saw the empires of the Elves and Dwarves crumble into ruins. (S-BLEU: 0.9329, S-USE: 0.9420)<br>Tran.这场暴行~~引发了胡子战争~~ 是胡子战争引发的, ~~在长达数千年的冲突中~~ 几千年来, 精灵~~帝国~~和矮人帝国~~陷入废墟~~都陷入困境。(BDR: 37.43%, UDR: 13.86%) |

Table 3.7: Comparisons among different word substituting methods.

| Metrics | HA.Def | HA.Syn | HA.Rank | HAA |
|---|---|---|---|---|
| Avg.S-BLEU | 0.3634 | 0.3521 | 0.3631 | **0.3642** |
| Avg.S-USE | 0.7639 | 0.6733 | 0.7591 | **0.8328** |

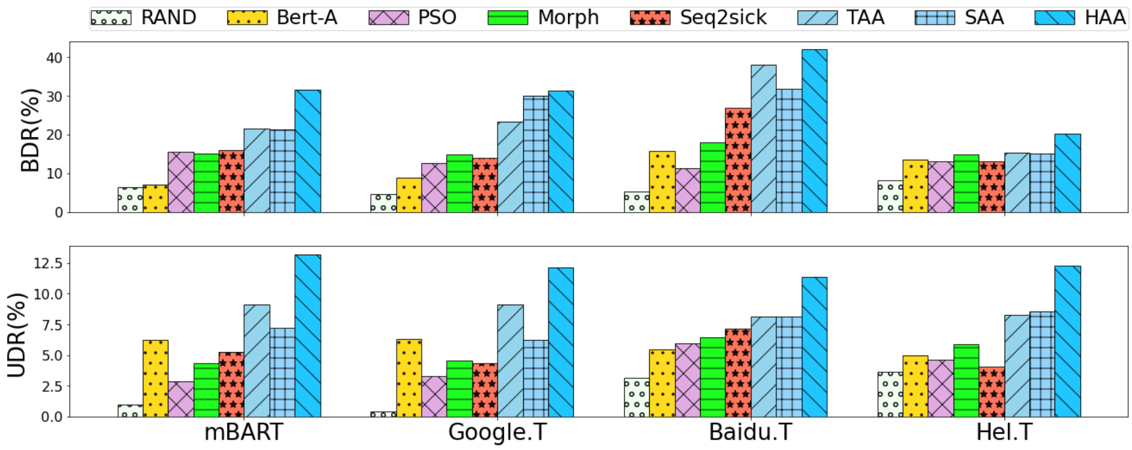


Figure 3.7: Attacking performance (BDR, UDR) of transferred attacks from mBART to Google, Baidu and Helsinki NMT models.

## 3.4 Transferability

The transferability of adversarial examples is defined as whether the adversarial examples targeting at a specific model $f$ can also mislead another model $f^{'}$. To evaluate transferability, we apply one-word-perturbation adversarial examples generated by different methods on mBART-large-cc25 [79], a sequence-to-sequence transformer from Facebook, to attack Google, Baidu and Helsinki translation models. Figure 3.7 shows the results on the original mBART NMT and other transferred models. It can be concluded from this figure that our attentive methods (TAA, SAA, and HAA) achieve the best attack performance on the three transferred NMT models, demonstrating the effectiveness of our methods in terms of attack transferability.

Table 3.8: Distributions of POS tags for different attack strategies. The percentages are calculated row-wise. For each row, the most, second and third highest percentage is highlighted in bond, underlined italic, respectively.

| Models | Noun | Verb | Adj. | Adv. | Others |
|---|---|---|---|---|---|
| PSO | **40.89%** | 9.12% | 15.77% | 17.89% | *16.33%* |
| Seq2sick | **40.11%** | 14.90% | 19.30% | 8.77% | *16.92%* |
| BERT.A | **78.42%** | 3.90% | 9.92% | 2.51% | *5.25%* |
| Morph | 35.51% | 9.77% | **44.19%** | *10.53%* | 0.0% |
| TAA | **48.37%** | 24.33% | 6.15% | 0.04% | *17.15%* |
| SAA | **44.71%** | 27.10% | *17.56%* | 6.57% | 4.06% |
| HAA | **51.14%** | 23.86% | *17.41%* | 2.79% | 4.80% |

## 3.5 Attacking Preference

As the superiority of proposed method in terms of attacking performance, we collect some statistics to research the attacking preference, described by speech (POS) tags, for different attacking strategies. In this subsection, we analyze statistics on POS as shown in Table 3.8, and aim to analyse the more vulnerable POS tags by a comparison between the proposed methods and baselines.

Words that are assigned to the same part of speech (POS) tags generally present similar syntactic importance, we investigate attacking strategies' preference on POS tags for further lingual analysis. We apply Stanford PSO tagger [82] 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 3.8 demonstrate that generally all the attacking methods tend to focus on *noun*, which we can suppose is the most sensitive POS category for translation. However, the proposed attacking strategies (TAA, SAA and HAA) tends to take a larger proportion of *Verbs* than any other methods, thus we may conclude that *Verb* might be the second adversarially vulnerable POS tag.

## 3.6 Summary and Discussion

This chapter highlights the susceptibility of Neural Machine Translation (NMT) models to adversarial attacks, which not only disrupt the translation of specific words but also their contextual environment. A method named HAA is introduced, which strategically selects and substitutes influential words, thereby affecting the translation of other words in the sequence. This method has proven to provide an optimal balance between the number of altered words and the effectiveness of the attack. In addition, this chapter suggests that adversarial examples are features, not bugs, and proposes adversarial retraining as a potential defense strategy. This involves integrating adversarial examples into the training set to enhance the model's robustness against such attacks.

# Chapter 4

# Fraud's Bargain Attack to Textual Classifiers via Metropolis-Hasting Sampling

It has been proven that adversarial examples expose vulnerabilities of natural language processing (NLP) models [32, 67, 87, 90]. The performance of existing techniques for generating adversarial examples is limited due to searching for a sub-optimal adversarial example, which would often cause attack failures. In this chapter, we introduce Fraud's Bargain Attack (FBA), a novel approach that leverages the Word Manipulation Process (WMP) to expand the search space and generate high-quality adversarial examples with increased probability. FBA employs a conditional Metropolis-Hastings sampler to select adversarial examples from the WMP, enhancing its effectiveness.

Compared with most literature attacks, FBA has two outstanding advantages: **a large searching space** and **an adaptive setting of NPW**. Instead of perturbing the input texts only with word substitutions, FBA can generate adversarial examples by manipulating words in the input text through insertion, substitution, and deletion. Besides, different from WIR, WMP selects the attacked position stochastically by a customized word distribution, with which it is possible for every word in the context to be chosen according to its importance. More word manipulation operations and attacked positions provide a significantly larger searching domain, which is theoretically expected to generate more effective adversarial examples than literature. In addition, we regulate the MH sampler with the imperceptibility of the attacks and minimal deviations from the original semantics. In this way, the NPW, usually preset in previous studies, can be adaptive to different input texts, an approach that can achieve many successful attacks. Our main contributions to this work are as follows:

- We design a stochastic process, the Word Manipulation Process (WMP), which creates a large search space for adversarial candidates by taking word insertion, removal, and substitution into account of the stochastic processes.
- We propose a highly effective adversarial attack method, Fraud's Bargain Attack (FBA), which applies the Metropolis-Hasting (MH) algorithm to construct an acceptance probability and use it to adaptively select high-quality adversarial candidates generated by WMP. The use of the acceptance probability helps our attack method jump out of the local optima and generate solutions closer to the global optima.
- We evaluate our attack method on real-world public datasets. Our results show that methods achieved the best performance in terms of both attack success and semantics preservation.

This chapter is structured as follows. Firstly, we introduce the formulation and properties of MCMC in Section 4.1. Then we detail our proposed method in Section 4.2 and 4.3. We evaluate the performance of the proposed method through empirical analysis in Section 4.4. We conclude the chapter with suggestions for future work in Section 4.5.

## 4.1 Preliminaries

Markov chain Monte Carlo methods create samples from a continuous random variable, with probability density proportional to a known function. To generate a sample that reflects the target distribution, a Markov chain is constructed with the target distribution as its equilibrium. Recording states from this chain yields a sample of the target distribution. As more samples are produced, the sample distribution increasingly resembles the true target distribution. Metropolis Hasting (MH) sampler [50], from which MCMC methods originate, applies the following setting: suppose that we wish to generate samples from an arbitrary multidimensional probability density function (PDF):

$$f(\mathbf{x}) = \frac{p(\mathbf{x})}{\mathcal{Z}}, \tag{4.1}$$

where $p(\mathbf{x})$ represents a defined positive function, and $\mathcal{Z}$ is a normalizing constant, which may or may not be known. The term $q(\mathbf{y}|\mathbf{x})$ denotes a proposal or instrumental density. This Markov transition density describes the transition process from state $\mathbf{x}$ to state $\mathbf{y}$. The MH algorithm is based on the following "trial-and-error" strategy by defining an acceptance probability $\alpha(\mathbf{y}|\mathbf{x})$ as follows:

$$\alpha(\mathbf{y}|\mathbf{x}) = \min\left\{\frac{f(\mathbf{y})q(\mathbf{x} \mid \mathbf{y})}{f(\mathbf{x})q(\mathbf{y} \mid \mathbf{x})}, 1\right\} \tag{4.2}$$

which decides whether the new state $y$ is accepted or rejected. To be more specific, we sample a random variable $u$ from a Uniform distribution ranging from 0 to 1, $u \sim Unif(0, 1)$, and if $u < \alpha(\mathbf{x}, \mathbf{y})$ then $\mathbf{y}$ is accepted as the new state. Otherwise, the chain remains at $\mathbf{x}$. The fact that the equilibrium distribution of the MH Markov Chain is equal to the target distribution is guaranteed by the local/detailed balance equation.

A good property of the MH sampler is that in order to evaluate the accept rate $\alpha(\mathbf{x}, \mathbf{y})$, it is only necessary to know the positive function $q(\mathbf{y} \mid \mathbf{x})$, which is also known as kernel density. In addition, the efficiency of the MH sampler depends on the choice of the transition density function $q(\mathbf{y} \mid \mathbf{x})$. Ideally, $q(\mathbf{y} \mid \mathbf{x})$ should be "close" to $f(\mathbf{y})$, with respect to $\mathbf{x}$.

## 4.2 Word Manipulation Process

In this section, we detail the Word Manipulation Process (WMP) and shall explain our strategy for selecting the generated candidates in the next section. Let $D = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ denote a dataset with $m$ data samples, where $x$ and $y$ are the input text and its corresponding class. The victim classifier $F$ learns from text space to class space through a categorical distribution, $F(\cdot) : \mathcal{X} \rightarrow (0, 1)^K$, where $\mathcal{X}$ represents text space and $K$ is the number of classes. Given the input text $x = [w_1, \ldots, w_i, \ldots, w_n]$ with $n$ words, we denote an adversarial candidate of $x$ as $x'$, and denote the final chosen adversarial example as $x^*$.

From the current text state $x$, WMP takes three steps to implement perturbations. The first step is to sample an action $e$ from the set {insert(0), substitute(1), remove(2)}. Then we determine the position $l$ in the sentence to conduct the chosen manipulation $e$ by drawing $l$ from a customized categorical distribution. For the third step, if word insertion or substitution is chosen, WMP will provide an insertion or substitution candidate. The candidate is selected from a combination of synonyms of the original words and a pre-trained masked language model (MLM), such as BERT[9], XML [8] and MPNet [76]. The algorithm of WMP is demonstrated in Algorithm 2. Details of the three steps of WMP are elaborated as follows:

### 4.2.1 Action

we draw $e \in \{0, 1, 2\}$ from a categorical distribution:

$$p(e|x) = \begin{cases} \mathbf{P}_{ins} & e = 0, \\ \mathbf{P}_{sub} & e = 1, \\ \mathbf{P}_{rem} & e = 2 \end{cases} \tag{4.3}$$

where $\mathbf{P}_{ins} + \mathbf{P}_{sub} + \mathbf{P}_{rem} = 1$, and $\mathbf{P}_{ins}$, $\mathbf{P}_{sub}$, and $\mathbf{P}_{rem}$ represent probability of insertion (0), substitution (1) and removing (2), respectively. The probabilities of these types of attacks can be set by the attacker's preference.

### 4.2.2 Position

Given a certain action $e$, we need to select one target word at location $l$ in the sentence to implement the attack. Considering the effectiveness of the selection, higher probabilities should be assigned to the words with more influence. To solve this, we use the changes of victim classifiers' logits, $I = [I_{w_1}, \ldots, I_{w_i}, \ldots, I_{w_n}]$, before and after deleting a word. Such a drop of logits for $i$th word, $I_{w_i}$, is mathematically formulated as:

$$\begin{aligned} I_{w_i} &= F_{y,logit}(x) - F_{y,logit}(x_{w_i}), \\ x_{w_i} &= [w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n] \end{aligned} \tag{4.4}$$

where $F_{y,logit}(\cdot)$ is the classifier returning the logit of the correct class, and $x_{w_i}$ is the text with $w_i$ removed. Differently with word importance rank (WIR), we utilize drops of logits $I$ to craft categorical distribution on position $l$, $p(l|e, x)$, by putting $I$ to a softmax function as following:

$$p(l|e, x) = \text{softmax}(I) \tag{4.5}$$

This way, locations of words (tokens) are assigned with probabilities according to the words' influence on the classifier.

### 4.2.3 Word Candidates

Different actions require different searching strategies for word candidates. To find the word candidates for substitution and insertion, we utilize an MLM and synonyms of the original words (calculated by nearest neighbors using L2-norm of word embeddings), for parsing-fluency and semantic preservation, respectively. As for word removals, we design a hesitation mechanism to maintain a probabilistic balance with word insertion. The details are demonstrated in the following paragraphs.

#### 4.2.3.1 Candidates for Substitution Attacks

We mask the word on the selected position to construct a masked sentence $x^*_{sub} = [w_1, \ldots, [MASK], \ldots, w_n]$ and feed this masked sentence $x^*_{sub}$ into a MLM $\mathcal{M}(\cdot) : \mathcal{X} \rightarrow (0, 1)^d$ to obtain a distribution about word candidates $o$ across all the words in the dictionary size $d$. The distribution is below:

$$p_{\mathcal{M}}(o|l, e, x) = \mathcal{M}(x^*_{sub}) \tag{4.6}$$

The MLM relies on softmax to output a distribution on the dictionary but most words from the dictionary can be grammarly improper and the probability of selecting one of these words can be high. To this end, we tend to create another distribution with respect to the $k$ top word candidates from the MLM. By mixing such a distribution with the MLM distribution, the probability of selecting grammarly improper words can be decreased. To construct such a $k$ top words distribution, we choose the $k$ top words $G^{sub}_{\mathcal{M}} = \{w^{sub}_{(\mathcal{M},1)}, \ldots, w^{sub}_{(\mathcal{M},k)}\}$ from $\mathcal{M}$ and treat every word from this set equally important:

$$p^{top}_{\mathcal{M}}(o|l, e, x) = \mathbb{1}(o \in G^{sub}_{\mathcal{M}})\frac{1}{k}, \tag{4.7}$$

where $\mathbb{1}(\cdot)$ is an indicator function. In such a setting, the top candidates from MLM are attached with more importance.

Although the MLM can find parsing-fluent word candidates, these candidates cannot ideally preserve semantics. Therefore, we perform synonym extraction by gathering a word candidate set for top $k$ replacements of the selected word. Specifically, we use L-2 norm as the metric to perform kNN inside word embedding space from BERT, and construct such a synonym candidates set $G_{nn} = \{w_{(\mathcal{M},1)}, \ldots, w_{(\mathcal{M},k)}\}$, with top-k nearest neighbors from the embedded spaces as synonyms for the word on selected position $l$. In this way, we construct the synonym words distribution as follows:

$$p_{nn}(o|l, e, x) = \mathbb{1}(o \in G_{nn})\frac{1}{k} \tag{4.8}$$

As we tend to generate parsing-fluent and semantic-preserving adversarial candidates, we combine the distributions in Eq. 4.6, Eq. 4.7 and Eq. 4.8 to construct a mixture distribution as the final distribution to draw the substitution:

$$\begin{aligned} p_{sub}(o|e, l, x) =& a_1 p_{\mathcal{M}}(o|l, e, x) + a_2 p^{top}_{\mathcal{M}}(o|l, e, x) \\ &+ a_3 p_{nn}(o|l, e, x), \end{aligned} \tag{4.9}$$

where $a_1 + a_2 + a_3 = 1$, $a_1, a_2, a_3 \in (0, 1)$, while $a_1$, $a_2$ and $a_3$ are hyper-parameters for weighing the corresponding distribution.

#### 4.2.3.2 Candidates for Insertion Attacks

Searching word candidates for insertion attacks follows a similar logic as substitutions but without a synonym search. We construct the mask sentence:

$$x^*_{ins} = [w_1, \ldots, w_{l-1}, [MASK], w_l, \ldots, w_n]$$

by inserting a masked token on the left side of selected position, then apply the MLM $\mathcal{M}$ to the mask sentence for extracting the output of the softmax layer, $\mathcal{M}(x^*_{ins})$. Following the same logic as Eq. 4.7, we select the top $k$ word candidates $G^{ins}_{\mathcal{M}} = \{w_{(\mathcal{M},1)}, \ldots, w_{(\mathcal{M},k)}\}$. Similarly with Eq. 4.9, insertion word candidate distribution can be constructed as follows:

$$p_{ins}(o|l, e, x) = b_1 \mathcal{M}(x_{ins}) + b_2 \mathbb{1}(o \in G^{ins}_{\mathcal{M}})\frac{1}{k} \tag{4.10}$$

where $b_1 + b_2 = 1$, $b_1, b_2 \in (0, 1)$, $b_1$ and $b_2$ are hyper-parameters for weighing the corresponding distribution.

#### 4.2.3.3 Candidates for Removal Attacks

Since word candidates for insertion and substitution can be drawn from a significantly large dictionary, these two actions will provide a large variance of adversarial candidates. Differently, removal does not require a selection of word candidates but directly removes the word on the selected position, which will lead to a low variety of adversarial candidates crafted by removal. The consequence of such a low variety is that the probability of crafting the same adversarial candidate is much higher than inserting and substituting. To balance such a probability, we design a Bernoulli distribution to determine word removals. Specifically, we craft the removal word candidates set $G^{rem} = \{0, 1\}$, where 0 and 1 represent remaining and removing the selected word, respectively. The distribution is as follows:

$$p_{rem}(o|e,l,x) = \begin{cases} 1 - \frac{1}{k} & o = 0, \\ \frac{1}{k} & o = 1 \end{cases} \tag{4.11}$$

With the above distribution, the $o = 1$ (i.e., to remove the word) is selected to replace the original word with probability $\frac{1}{k}$. The rationale of using $\frac{1}{k}$ is to decrease the probability of repeatedly proposing the same perturbed sentence with action removal such that it is approximate to the probability of word replacement and insertion as in Eq. 4.6 and Eq. 4.10: while each replacement word has a probability of $\frac{1}{k}$ for being chosen, the removal, $o = 1$, has the same probability of being selected in removal attacks.

#### 4.2.3.4 Integration of the three WMP steps

With the three WMP steps, we summarize the probability density function for word candidates:

$$p(o|e,l,x) = \begin{cases} p_{ins}(o|e,l,x) & e = 0, \\ p_{sub}(o|e,l,x) & e = 1, \\ p_{rem}(o|e,l,x) & e = 2 \end{cases} \tag{4.12}$$

By iteratively running WMP $T$ times with an initial start at original input text ($x'_0 = x$), we can get a sequence of adversarial candidates $x'_T = [x'_1, \dots, x'_t, \dots, x'_T]$. By applying the Bayes rule, we can derive the WMP's distribution from the iteration $t$ to $t+1$ as the following equation:

$$\begin{aligned} \mathrm{WMP}(x'_{t+1}|x'_t) =& p(e, l, o|x'_t) \\ =& p(e|x'_t)p(l|e, x'_t)p(o|e, l, x'_t) \end{aligned} \tag{4.13}$$

### 4.2.4 The Theoretical Merit of WMP

WMP is expected to own two major merits: enlarging the searching domain and the ability to correct possible wrong manipulation. The merits can guarantee by the aperiodicity in the following theorem.

**Theorem 1** *Word Manipulation Process (WMP) is aperiodic.*

*Proof:* Suppose we have two arbitrary text samples $x_i, x_j \in \mathcal{X}$ from text space $\mathcal{X}$. $x_i = [w^i_1, \dots, w^i_{n_i}]$ $x_j = [w^j_1, \dots, w^{n_j}_1]$ have $n_i$ and $n_j$ words, respectively. To prove the process is aperiodic by definition, we need to show:

$$\exists N \leqslant \infty,\ \mathbb{P}(x^{(N)} = x_j|x_i) > 0, \tag{4.14}$$

which means that there always exist $\exists N \leqslant \infty$ that can make the probability of transfer $x_i$ to $x_j$ after $N$ times larger than zero.

Because text dataset is discrete and WMP is time-discrete, WMP is a Markovian process. Therefore, we apply the Chapman–Kolmogorov equation, to derive the following equation:

$$\begin{aligned} \mathbb{P}\left(x^{(N)} = x_j|x_i\right) = &\sum_{x^{(t)} \in \mathcal{X}} \mathrm{WMP}\left(x^{(1)}|x_i\right) \mathrm{WMP}\left(x^{(2)}|x^{(1)}\right) \\ &\cdots \mathrm{WMP}\left(x^{(N-1)}|x^{N-2}\right) \mathrm{WMP}\left(x_j|x^{(N-1)}\right), \end{aligned} \tag{4.15}$$

where WMP($\cdot$) denotes the Word Manipulation process (WMP). We try to prove aperiodicity with a special case, let $N = n_i + n_j$ $\mathcal{A}$: first inserting all the words from $x_j$ to the $x_i$, then remove all words from $x_i$. This process can be illustrated as follows:

$$\begin{aligned}\mathcal{A}: \quad x_i & \xrightarrow[insert]{n_j\text{times}} x^{n_j} = [w_1^i, \ldots, w_{n_i}^i, w_1^j, \ldots, w_1^{n_j}] \\ & \xrightarrow[remove]{n_j\text{times}} x_j = [w_1^j, \ldots, w_1^{n_j}]\end{aligned}$$

As $\mathcal{A}$ is the special case of the $N$ times iterations, we have:

$$\mathbb{P}(\mathcal{A}) \leqslant \mathbb{P}\left(x^{(N)} = x_j | x_i\right). \tag{4.16}$$

Moreover, the WMP inserts one word on any position based on softmax, which outputs non-zero probabilities, therefore we can derive:

$$0 < \mathbb{P}(\mathcal{A}) \leqslant \mathbb{P}\left(x^{(N)} = x_j | x_i\right). \tag{4.17}$$

Therefore, we find that for arbitrary $x_i, x_j \in \mathcal{X}$, there always exist $N = n_i + n_j \leqslant \infty$ that can make the probability of transfer $x_i$ to $x_j$ after $N$ time larger than zero, i.e., $\mathbb{P}(x^{(N)} = x_j | x_i) > 0$. According to the definition, we successfully prove the Theorem 1. $\square$

Aperiodicity implies that, given a large enough number of iterations $T$, an arbitrary $x$ can be perturbed to any text $x'$ in text space, i.e., WMP$(x|x') \neq 0$. The first merit of aperiodicity, WMP theoretically guarantees that the searching domain is enlarged to generate the most effective adversarial candidates. Since WMP samples the adversarial candidates with randomness, there is a tiny possibility of crafting a bad adversarial candidate. With aperiodicity, WMP is eligible for correcting this bad manipulation by reversing the bad sample $x'_{t+1}$ to the previous state $x'_t$.

## 4.3 Adversarial Candidate Selection by Metropolis-Hasting Sampling

After WMP generates adversarial candidates, a naive method is to greedily test all the adversarial canididates and choose the best-performed candidates as adversarial examples. However, such a brute-force approach not only is time consuming but also can end up with over-modified adversarial examples. To this end, we propose Fraud's Bargain Attack (FBA), which utilizes the Metropolis-Hasting (MH) algorithm to enhance the WMP via selecting adversarial candidates evaluated by a customized adversarial distribution.

### 4.3.1 Adversarial Distribution

We argue that adversarial examples should work with imperceptible manipulations to input text. Therefore, given text $x$, we construct the adversarial target distribution $\pi(x') : \mathcal{X} \to (0, 1)$ to measure the classifier's depravation once under attack with a heavy penalty on change of semantics. Concretely, we measure the classifier's depravation by defining a measure of distance to perfection, $R$, based on the confidence of make wrong predictions $1 - F_y(x')$, where $F_y : \mathcal{X} \to [0, 1]$ is the confidence of predicting correct class. The higher the value of the distance to perfection $R$, the more successful the attack. Meantime, we add a regularizer on semantic similarity, $\mathrm{Sem}(\cdot) =$. Thus the mathematical formulation is as follows:

$$\pi(x'|x, \lambda) = \frac{R + \lambda \mathrm{Sem}(x', x)}{C} \tag{4.18}$$

$$R = \begin{cases} 1 - F_y(x') & F_y(x') > \frac{1}{K}, \\ 1 - \dfrac{1}{K} & F_y(x') \leqslant \frac{1}{K} \end{cases} \tag{4.19}$$

$$C = \sum_{x' \in \mathcal{X}} R + \lambda \mathrm{Sem}(x', x), \tag{4.20}$$

**Algorithm 2** Word Manipulation Process

**Input:** Number of iterations: $T$, Input text: $x$
**Output:** A set of adversarial candidates: $[x'_1, \ldots, x'_T]$

candidates=[ ]
$x'_0 = x$
**for** $t$ in $T$ **do**
    Given $x'_{t-1}$, sample $e$ with Eq. 4.4
    Given $x'_{t-1}$, $e$, sample $l$ with Eq. 4.5
    Given $x'_{t-1}$, $e$, $l$, sample $o$ with Eq. 4.12
    Craft $x'_t$ by taking action $e, l, o$ to $x'_{t-1}$
    candidate.append($x'_t$)
**end for**
**return** candidates

where $K$ is the number of classes. In Eq. 4.18, we construct adversarial distribution by utilizing a hyper-parameter $\lambda$ to combine the attack performance $R$ and semantic similarity $Sem(\cdot)$. In Eq. 4.20, $C$ represents the constant normalizing $\pi(x')$ to ensure the distribution condition, $\sum_{x' \in \mathcal{X}} \pi(x'|x, \lambda) = 1$. To keep more semantics, we let Sem$(x', x)$ denote the semantic similarity between adversarial example $x'$ and original text $x$. In general, the $Sem(\cdot)$ is implemented with the cosine similarity between sentence encodings from a pre-trained sentence encoder, such as USE [4].

In spite of the use of the semantic regularizer, we argue that a high $R$ might still cause a thrilling semantic loss because the value of $\pi(x'|x)$ might go up with large increases of $R$ and small drops of semantic similarity $Sem(x', x)$. Thus, for a further improvement on semantic preservation, we let the $R$ be associated with a cut-off value at $\frac{1}{K}$ when the class is successfully misclassified (i.e., when $F_y(x') \leqslant \frac{1}{K}$). Note that when $F_y(x') \leqslant \frac{1}{K}$, the classifier will misclassify $x'$ to one of the other $K - 1$ classess other than $y$. By having the mechanism of setting $R$ to $\frac{1}{K}$ whenever misclassification is achieved, all successful adversarial examples will have the same $R$ value. This way, their optimization with $\pi$ will then focus on maximizing their semantic similarity $Sem(x', x)$ with the original texts.

### 4.3.2 Fraud's Bargain Attack via Metropolis Hasting Sampler

Metropolis Hasting [50] simulates a target distribution by using a proposing function to offer a trial state which is then accepted or rejected according to a customized acceptance probability. Specifically, given a target distribution $Q(\cdot)$, the MH sampler utilizes a proposing function $q(s_{t+1}|s_t)$ (transition density from $s_t$ to $s_{t+1}$) to construct a Markov Chain, whose equilibrium distribution is our target distribution. By this probabilistic mechanism, the proposing function would propose a trial state $s_{t+1}$ given the current state $s_t$ and the acceptance probability $\alpha(s_{t+1}|s_t)$, as shown in Eq. 4.21:

$$\alpha(s_i, s_{i+1}) = \min\left(1, \frac{Q\left(s_{i+1}\right)}{Q(s_i)} \frac{q\left(s_i \mid s_{i+1}\right)}{q\left(s_{i+1} \mid s_i\right)}\right) \tag{4.21}$$

Based on such a setting, we construct FBA by considering the adversarial distribution (Eq. 4.18) and the WMP as the MH's target distribution and proposing function, respectively. In each iteration of FBA, we use WMP to propose a trial state $x_{t+1}$ and calculate the acceptance probability $\alpha(x_{t+1}|x_t)$. FBA's acceptance probability in Eq. 4.21 can then be mathematically formulated by using WMP as follows:

$$\alpha(x_{t+1}|x_t) = \min\left(1, \frac{\pi\left(x_{t+1}\right)}{\pi(x_t)} \frac{\mathrm{WMP}\left(x_t \mid x_{t+1}\right)}{\mathrm{WMP}\left(x_{t+1} \mid x_t,\right)}\right) \tag{4.22}$$

where $\mathrm{WMP}\left(x_t \mid x_{t+1}\right)$ can be guaranteed non-zero by Theorem 1, and calculated by reversing the WMP process: removing the inserted word, inserting the removed word and recovering substituted word. After calculating $\alpha(x_{t+1}|x_t)$, we sample $u$ from a uniform distribution, $u \sim Unif(0, 1)$, if $u < \alpha(x_{t+1}|x_t)$ we will accept $x_{t+1}$ as the new state, otherwise the state will remain as $x_t$. By running $T$ iterations, FBA generates a set of adversarial candidates, and we will choose the one with lowest modification among the successful adversarial candidates that flips the predicted class. The whole process of FBA is illustrated in the Algorithm 3.

**Algorithm 3** Fraud's Bargain Attack (FBA)

**Input:** Input text: $x$, Number of Sample: $T$
**Output:** An adversarial example

$Adv\_set = [\quad]$
$x_1 = x$
**for** t in range($T$) **do**
  Sample $x_t$ from WMP given $x_{t-1}$ with Eq. 4.13
  Sample $u$ from Uniform distribution, Uniform(0,1)
  Calculate the acceptance probability, $prob = \alpha(x_{t+1}, x_t)$ with Eq. 4.22
  **if** $u < prob$ **then**
    $x_t = x_{t+1}$
    $Adv\_set$.append($x_{t+1}$)
  **else**
    $x_t = x_t$
    $Adv\_set$.append($x_t$)
  **end if**
  **return** $Adv\_set$
**end for**
Choose the candidate with the least modification as adversarial example $x^*$.
**return** adversarial example $x^*$

## 4.4 Experiments and Analysis

We evaluate the effectiveness of methods on widely-used and publicly available datasets with well-performed victim classifiers. We provide codes and data with the published paper [61] to ensure reproducibility.

### 4.4.1 Main Experimental Settings

In this subsection, the basical experimental settings such as the datasets, victim models, baselines and the evaluation metrics used for the performance evaluation will be introduced.

#### 4.4.1.1 Datasets and Victim Models

In this section, we detail the three benchmark datasets and the two well-performed textual classifiers. We conduct experiments on four publicly accessible benchmark data-

Table 4.1: Datasets and accuracy of victim models before attacks.

| Dataset | Size | Task | Model | Accuracy |
|---|---|---|---|---|
| AG's News | 127000 | News topics | BERT-C<br>TextCNN | 94%<br>90% |
| Emotion | 20000 | Sentiment analysis | BERT-C<br>TextCNN | 97%<br>93% |
| SST2 | 9613 | Sentiment analysis | BERT-C<br>TextCNN | 91%<br>83% |
| IMDB | 50000 | Movie review | BERT-C<br>TextCNN | 93%<br>88% |

sets. AG's News [94] is a news classification dataset with 127,600 samples belonging to 4 topic classes. Emotion [73] is a dataset with 20,000 samples and 6 classes. SST2 [75] is a binary class topic dataset with 9,613 samples. IMDB [75] is a binary class topic dataset with 50,000 labeled samples. Details of these datasets can be found in Table 4.1.

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

#### BERT-based Classifiers

To do convincing experiments, we choose three well-performed and popular BERT-based models, which we call BERT-C models, pre-trained by Huggingface[1]. 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* [72] 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* [72], 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* [9]. The test accuracies of these BERT-based classi-

[1] `https://huggingface.co/`

fiers before they are under attack are listed in Table 4.1 which are publicly accessible [2] [3] [4].

#### TextCNN-based models

The other type of victim model is TextCNN [38], structured with a 100-dimension embedding layer followed by a 128-unit 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 4.1. The hyper-parameters for training this models, is based on the literatures [55, 86]. Additionally, the victim models are employed to evaluate the attack performance, testing whether the proposed methods can compromise robustness. Consequently, these models do not need to achieve state-of-the-art performance.

[2] https://huggingface.co/mrm8488/distilroberta-finetuned-age_news-classification

[3] https://huggingface.co/bhadresh-savani/distilbert-base-uncased-emotion

[4] https://huggingface.co/echarlaix/bert-base-uncased-sst2-acc91.1-d37-hybrid

Table 4.2: Adversarial examples of Emotion dataset for victim classifier BERT-C. Original words are highlighted in blue, while substitutions are indicated in red. The attack performance measured by true class scores is placed inside the brackets. The lower true class score indicates better performance. The successful attacks and lowest true class scores are bold.

| Attacks | Adversarial examples |
|---|---|
| A2T (Unsuccessful attack. True class score = 41.31%) | i spent wandering around still kinda dazed and not ~~feeling~~ sense particularly ~~sociable~~ social but because id been in hiding for a couple for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with folks |
| BAE (Unsuccessful attack. True class score = 33.25%) | i spent wandering around still kinda dazed and not ~~feeling~~ being particularly sociable but because id been in hiding for a couple for days and it was getting to be a ~~little~~ bit unhealthy i made myself go down to the cross and hang out with folks |
| FAGA (**Successful attack**. True class score = 13.32%) | i spent wandering around still kinda dazed and not feeling particularly ~~sociable~~ sympathetic but because id been in hiding for a ~~couple~~ few for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with folks |
| BERT.A (Unsuccessful attack. True class score = 77.04%) | i spent wandering around still kinda dazed and not ~~feeling~~ being particularly ~~sociable~~ happy but because id been in hiding for a couple for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with folks |
| CLARE (**Successful attack**. True class score = 10.54%) | i spent wandering around still kinda dazed and not feeling particularly ~~sociable~~ lonely but because id been in hiding for a couple for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with folks. |
| PWWS (**Successful attack**. True class score = 21.11%) | i spent wandering around still kinda dazed and not ~~feeling~~ palpate particularly sociable but because id been in hiding for a couple for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with folks |
| PSO (**Successful attack**. True class score = 6.90%) | i spent wandering around still kinda dazed and not ~~feeling~~ considering particularly sociable but because id been in hiding for a couple for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with ~~folks~~ dudes |
| FBA (**Successful attack**. True class score = **0.63%**) | i spent wandering around still kinda dazed and not ~~feeling~~ sensing particularly sociable but because id been in hiding for a couple for days and it was getting to be a little unhealthy i made myself go down to the cross and hang out with folks |

Table 4.3: Adversarial examples of AG's News dataset for victim classifier TextCNN. Original words are highlighted in blue, while substitutions are indicated in red. The attack performance measured by true class scores is place inside the brackets. The lower true class score indicates better performance. The successful attacks and lowest true class score are bold.

| Attacks | Adversarial examples |
|---|---|
| A2T (Unsuccessful attack. True class score = 45.59%) | Card fraud unit nets 36,000 cards In its first two years, the UK's dedicated card fraud unit, has recovered 36,000 stolen ~~cards~~ items and 171 arrests - and ~~estimates~~ says it saved 65m. |
| BAE (Unsuccessful attack. True class score = 73.25%) | Card fraud unit nets ~~36,000~~ stolen cards In its first two years, the UK's dedicated card fraud unit, has recovered 36,000 stolen cards and 171 ~~arrests~~ accounts - and estimates it saved 65m. |
| FAGA (Unsuccessful attack. True class score = 37.32%) | Card fraud unit ~~nets~~ recovered 36,000 cards In its first two years, the UK's dedicated card fraud unit, has recovered 36,000 ~~stolen~~ credit cards and 171 arrests - and estimates it saved 65m. |
| BERT.A (Unsuccessful attack. True class score = 30.43%) | Card fraud unit nets 36,000 cards In its first two years, the ~~UK's~~ new ~~dedicated~~ largest card fraud unit, has recovered 36,000 stolen cards and 171 arrests - and ~~estimates~~ claims it saved 65m. |
| CLARE (**Successful attack**. True class score = 9.05%) | Card fraud unit nets 36,000 cards In its first two years, the UK's dedicated card ~~fraud~~ collection ~~unit~~ center, has recovered 36,000 stolen cards and 171 arrests - and estimates it saved 65m. |
| PWWS (**Successful attack**. True class score = 18.31%) | ~~Card~~ The fraud unit nets 36,000 cards In its first two years, the UK's dedicated card fraud unit, has ~~recovered~~ reported 36,000 stolen cards and 171 arrests - and estimates it saved 65m. |
| PSO (Unsuccessful attack. True class score = 44.01%) | Card fraud unit nets 36,000 cards In its first two ~~years~~ ages, the UK's dedicated card fraud unit, has recovered 36,000 stolen cards and ~~171~~ many arrests - and estimates it saved 65m. |
| FBA (**Successful attack**. True class score = **1.92%**) | Card fraud unit nets 36,000 cards In its first two years, the UK's dedicated card fraud unit, has recovered 36,000 stolen cards and emancipation 171 arrests - and estimates it saved 65m. |

Table 4.4: Adversarial examples of SST2 dataset for victim classifier TextCNN. Original words are highlighted in blue, while substitutions are indicated in red. The attack performance measured by true class scores is placed inside the brackets. The lower true class score indicates better performance. The successful attacks and lowest true class scores are bold.

| Attacks | Adversarial examples |
|---|---|
| A2T (**Successful attack**. True class score = 21.31%) | an often-deadly ~~boring~~ short, strange reading of a classic whose witty dialogue is ~~treated~~ laced with a baffling casual approach |
| BAE (Unsuccessful attack. True class score = 32.10%) | an often-deadly boring , strange reading of a ~~classic~~ novel whose witty ~~dialogue~~ commentary is treated with a baffling casual approach |
| FAGA (Unsuccessful attack. True class score = 51.12%) | an ~~often-deadly~~ extremely boring , strange reading of a ~~classic~~ characters whose witty dialogue is treated with a baffling casual approach |
| BERT.A (Unsuccessful attack. True class score = 63.41%) | an often-deadly boring , ~~strange~~ critical reading of a classic whose ~~witty~~ spoken dialogue is treated with a baffling casual approach |
| CLARE (Unsuccessful attack. True class score = 10.54%) | an often-deadly boring , strange reading of a classic whose ~~witty~~ entire dialogue is treated with a baffling ~~casual approach~~ casualness |
| PWWS (Unsuccessful attack. True class score = 51.11%) | an often-deadly boring, ~~strange~~ casual reading of a classic whose witty dialogue is treated with a ~~baffling~~ more casual approach |
| PSO (**Successful attack**. True class score = 7.90%) | an often-~~deadly~~ somewhat boring, ~~strange~~ humorous reading of a classic whose witty dialogue is treated with a baffling casual approach |
| FBA (**Successful attack**. True class score = **5.11%**) | an ~~often-deadly~~ often-harmful boring, strange reading of a classic whose witty dialogue is treated with a baffling casual approach |

#### 4.4.1.2 Baselines

To evaluate the attacking performance, we use the Textattack [55] framework to deploy the following baselines:

- Faster Alzantot Genetic Algorithm (FAGA) [32] accelerate Alzantot Genetic Algorithm [1], by bounding the searching domain of genetic optimization.
- BAE [17] replaces and inserts tokens in the original text by masking a portion of the text and leveraging the BERT-MLM.
- BERT-Attack [45] takes advantage of BERT MLM to generate candidates and attack words by the static WIR descending order.
- A2T [89] uses a gradient-based word importance ranking method to iteratively replace each word with synonyms generated from a counter-fitted word embedding.
- CLARE (Li, 2021) [43] implements a series of context-sensitive perturbation steps on the input. This process resembles a localized mask-then-infill approach, where a specific portion of the input is masked and subsequently completed using a pre-trained Masked Language Model (MLM).
- In PWWS (Ren, 2019) [67], the selection of potential words is derived from WordNet [54]. The approach prioritizes words for alteration by calculating a product of their significance in the text and the degree of change they cause in the output probability.
- Particle Swarm Optimization (PSO) by Zang et al. (2020)[90] involves sourcing word alternatives from HowNet [10] and utilizing PSO for generating adversarial text. In this framework, each sample is viewed as a particle whose position requires optimization within the search space.

#### 4.4.1.3 Evaluation Metrics and Experimental Setting

We use the following metrics to measure the performance of adversarial attacks.

- Successful attack rate (SAR): the percentage of adversarial examples that can successfully attack the victim model.

- Textual similarity: the cosine similarity between the input and its adversary. We calculate this using the universal sentence encoder USE [4].
- Modification Rate (Mod) is the percentage of modified tokens. Each replacement, insertion or removal action accounts for one modified token.
- Recall-Oriented Understudy for Gisting Evaluation (ROUGE): the overlap of n-grams between the candidate sentence and reference sentence. Since the modification rate cannot measure the similarity of word alignment such as word ordering and sentence length, we adopt ROUGE to measure the alignment similarity between adversarial examples and original sentences.
- Grammar Error Rate (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 [57] to obtain the number of grammatical errors.
- Perplexity (PPL) denotes a metric used to evaluate the fluency of adversarial examples and is broadly applied in the literature [43, 90]. The perplexity is calculated using small-sized GPT-2 with a 50k-sized vocabulary [66].

Table 4.5: The successful attack rate (SAR) of attack algorithms. The higher values of SAR indicate better performance. For each row, the highest SAR is highlighted in bold, the second highest SAR is highlighted by underline, and the third highest SAR is denoted with italic.

| Dataset | Model | Attack Methods | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | WMP | A2T | BAE | FAGA | BERT.A | CLARE | PWWS | PSO | FBA |
| AG's News | BERT-C | 21.39% | 15.25% | 19.50% | 25.17% | 47.71% | *71.32%* | <u>76.75%</u> | 67.76% | **81.90%** |
| | TextCNN | 25.19% | 33.31% | 39.13% | 56.10% | 70.21% | *79.31%* | <u>85.31%</u> | 76.24% | **93.12%** |
| Emotion | BERT-C | 44.91% | 48.06% | 62.68% | 78.21% | 85.90% | <u>99.06%</u> | 91.75% | *94.76%* | **99.15%** |
| | TextCNN | 83.11% | 81.11% | 85.34% | 90.10% | 95.11% | 97.11% | <u>98.20%</u> | <u>99.00%</u> | **100%** |
| SST2 | BERT-C | 52.43% | 41.74% | 55.14% | 77.07% | 77.21% | *94.79%* | 93.9% | <u>96.62%</u> | **99.31%** |
| | TextCNN | 71.69% | 77.41% | 80.15% | 92.14% | 83.23% | 85.55% | <u>98.13%</u> | *92.20%* | **100%** |
| IMDB | BERT-C | 79.1% | 76.1% | 80.3% | 90.4% | 89.1% | *95.1%* | 94.1% | **100.0%** | **100.0%** |
| | TextCNN | 77.0% | 81.0% | 89.3% | 92.5% | 91.1% | 98.4% | *99.6%* | **100.0%** | **100.0%** |

Table 4.6: The imperceptibility performance (ROUGE, USE, Mod) of attack algorithms. The higher values of ROUGE and USE indicate better performance while the lower Mod indicates better performance. For each row, the best performance is highlighted in bold, the second best is highlighted in underline, and the third best is denoted with italic.

| Dataset | Model | Metrics | Attack Methods | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | WMP | A2T | BAE | FAGA | BERT.A | CLARE | PWWS | PSO | FBA |
| AG's News | BERT-C | ROUGE ↑ | 0.5149 | 0.8311 | 0.8145 | 0.7721 | 0.8003 | 0.6673 | *0.8356* | **0.8521** | <u>0.8453</u> |
| | | USE ↑ | 0.7083 | 0.7143 | 0.7234 | *0.7534* | 0.7334 | 0.6603 | 0.7332 | <u>0.7732</u> | **0.8001** |
| | | Mod ↓ | 17.2% | *13.3%* | <u>11.3%</u> | 14.6% | 17.4% | 14.2% | 17.9% | 21.6% | **11.0%** |
| | TextCNN | ROUGE ↑ | 0.5149 | 0.8291 | 0.8355 | 0.7663 | <u>0.8643</u> | 0.6969 | 0.8366 | *0.8401* | **0.8711** |
| | | USE ↑ | 0.7083 | 0.7542 | 0.7257 | 0.8014 | *0.8103* | 0.7121 | <u>0.8111</u> | *0.8103* | **0.8224** |
| | | Mod ↓ | 19.2% | 11.9% | **10.3%** | *11.5%* | 15.4% | 14.1% | 16.5% | 15.5% | **10.3%** |
| Emotion | BERT-C | ROUGE ↑ | 0.5021 | 0.6111 | 0.5991 | 0.5911 | <u>0.6401</u> | 0.4441 | 0.6141 | *0.6256* | **0.6410** |
| | | USE ↑ | 0.8803 | 0.8951 | 0.8831 | 0.8940 | 0.8714 | 0.8201 | *0.9012* | <u>0.9210</u> | **0.9246** |
| | | Mod ↓ | 9.9% | *9.1%* | <u>7.7%</u> | 9.8% | 9.2% | 11.2% | 10.2% | <u>12.0%</u> | **7.3%** |
| | TextCNN | ROUGE ↑ | 0.6210 | 0.6891 | 0.6413 | *0.6932* | 0.6201 | 0.5623 | <u>0.6994</u> | 0.6613 | **0.7304** |
| | | USE ↑ | 0.7982 | *0.8530* | 0.7341 | 0.80412 | 0.8315 | 0.5631 | <u>0.8920</u> | 0.8342 | **0.9046** |
| | | Mod ↓ | 13.0% | 10.3% | 9.8% | <u>8.3%</u> | *9.3%* | 15.3% | 19.0% | 18.0% | **8.0%** |
| SST2 | BERT-C | ROUGE ↑ | 0.6621 | 0.6642 | <u>0.7521</u> | *0.7201* | 0.6352 | 0.5501 | 0.6341 | 0.7041 | **0.7540** |
| | | USE ↑ | 0.7938 | 0.8421 | 0.7304 | 0.7310 | 0.8564 | 0.7434 | 0.8422 | 0.8104 | **0.8820** |
| | | Mod ↓ | 18.1% | *13.4%* | <u>11.3%</u> | 17.2% | 13.5% | 22.1% | 17.2% | 17.2% | **10.1%** |
| | TextCNN | ROUGE ↑ | 0.6393 | 0.6499 | <u>0.7139</u> | 0.6821 | 0.6532 | 0.5701 | 0.6393 | *0.7001* | **0.7340** |
| | | USE ↑ | 0.6901 | 0.7521 | 0.7024 | 0.6931 | 0.7410 | 0.7009 | <u>0.8210</u> | *0.8162* | **0.8709** |
| | | Mod ↓ | 15.9% | 16.1% | <u>10.4%</u> | 15.3% | *13.4%* | 17.1% | *13.4%* | 17.2% | **10.0%** |
| IMDB | BERT-C | ROUGE ↑ | 0.7934 | 0.8123 | 0.8153 | *0.8234* | 0.8112 | 0.8233 | <u>0.8531</u> | 0.8021 | **0.8621** |
| | | USE ↑ | 0.8118 | 0.8561 | 0.8321 | 0.8246 | *0.8691* | 0.8369 | <u>0.9080</u> | 0.8600 | **0.9102** |
| | | Mod ↓ | 8.0% | 6.7% | 7.7% | 6.7% | **5.9%** | 8.3% | *6.2%* | 9.2% | **5.9%** |
| | TextCNN | ROUGE ↑ | 0.8104 | 0.8321 | 0.8611 | *0.8814* | 0.8412 | 0.8600 | <u>0.8821</u> | 0.8123 | **0.9121** |
| | | USE ↑ | 0.8211 | 0.8905 | 0.8613 | 0.7714 | *0.8911* | 0.7979 | <u>0.9000</u> | 0.8600 | **0.9092** |
| | | Mod ↓ | 7.1% | 7.1% | 8.7% | <u>4.7%</u> | 5.5% | 6.3% | *5.2%* | 7.6% | **3.9%** |

Table 4.7: The quality of generated adversarial examples measured by fluency (PPL) and grammar error rate (GErr) of attack algorithms. The lower values of PPL and GErr indicate better performance. For each row, the best quality is highlighted in bold, the second best quality is highlighted by underline, and the third best quality is denoted with italic.

| Dataset | Model | Metrics | Attack Methods | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | WMP | A2T | BAE | FAGA | BERT.A | CLARE | PWWS | PSO | FBA |
| AG's News | BERT-C | PPL ↓ | 331 | 291 | 142 | *165* | 281 | 233 | 321 | 292 | **133** |
| | | GErr ↓ | 0.22 | 0.19 | 0.19 | 0.19 | 0.19 | 0.21 | 0.22 | 0.31 | **0.18** |
| | TextCNN | PPL ↓ | 281 | 199 | 142 | 145 | 241 | 213 | 277 | *143* | **133** |
| | | GErr ↓ | 0.23 | 0.17 | 0.19 | *0.15* | *0.15* | 0.19 | 0.20 | 0.14 | **0.13** |
| Emotion | BERT-C | PPL ↓ | 281 | *251* | 233 | 259 | 301 | 298 | 333 | 336 | **229** |
| | | GErr ↓ | 0.19 | 0.19 | **0.10** | *0.13* | 0.17 | 0.14 | 0.16 | 0.20 | **0.10** |
| | TextCNN | PPL ↓ | 288 | 268 | 201 | 241 | *221* | 256 | 299 | 301 | **182** |
| | | GErr ↓ | 0.16 | *0.11* | **0.10** | 0.13 | 0.13 | 0.16 | 0.19 | 0.18 | **0.10** |
| SST2 | BERT-C | PPL ↓ | 213 | 211 | *173* | 182 | 200 | 155 | 214 | 197 | **142** |
| | | GErr ↓ | 0.20 | *0.18* | 0.15 | 0.24 | 0.25 | 0.19 | 0.21 | 0.27 | **0.14** |
| | TextCNN | PPL ↓ | 198 | 164 | 142 | 163 | 210 | 174 | 224 | *145* | **140** |
| | | GErr ↓ | 0.19 | 0.21 | *0.14* | 0.17 | 0.21 | 0.17 | 0.23 | **0.13** | **0.13** |
| IMDB | BERT-C | PPL ↓ | 91 | 62 | *70* | 90 | 83 | 101 | 88 | 90 | **60** |
| | | GErr ↓ | 0.28 | 0.22 | *0.19* | 0.21 | *0.19* | **0.18** | *0.19* | 0.22 | **0.18** |
| | TextCNN | PPL ↓ | 103 | 69 | 79 | 99 | 80 | 89 | 69 | 99 | **66** |
| | | GErr ↓ | 0.33 | 0.29 | 0.22 | 0.26 | 0.23 | **0.20** | **0.20** | 0.28 | **0.20** |

For settings of FBA, we set WMP action proposing probability in Eq. 4.4 as $\mathbf{P}_{ins} = 0.2$, $\mathbf{P}_{sub} = 0.6$, and $\mathbf{P}_{rem} = 0.2$. While setting up the distribution for selecting the substitution and insertion words, we believe MLM and Synonyms are equally important. Therefore we set the weights of these two methods equal by setting $a_1 = 0.1, a_2 = 0.4, a_3 = 0.5$ and $b_1 = 0.5, b_2 = 0.5$ from Eq. 4.9 and Eq. 4.10, respectively.

### 4.4.2 Experimental Results and Analysis

The experimental results of attacking performance (SAR) and the imperceptibility performance (ROUGE, USE) and sentence quality (GErr, PPL) are listed in Table 4.5 and Table 4.6 and Table 4.7, respectively. To give an intuitive of the generated examples, we also show two generated adversarial examples in Tables 4.2, 4.3 and 4.4. We manifest the three contributions mentioned in the beginning of the chapter by asking four research questions:

**(a) Does our FBA method make more thrilling attacks to baselines?**

We compare the attacking performance of the proposed FBA method and baselines in Table 4.5. To be more specific, Table 4.5 demonstrates that FBA consistently outperforms other competing methods across different data domains, regardless of the structure of classifiers. It can thus be concluded that the proposed method FBA achieves the best attacking performance, with the largest successful attack rate (SAR). We attribute such an outstanding attacking performance to the two prevailing aspects of FBA. Firstly, the proposed FBA could enlarge the searching domain by removing, substituting and inserting words compared with the strategies with only substitution such that FBA provides more possible attacking combinations. Secondly, FBA optimizes the performance by stochastically searching the domain. Most of the baselines perform a deterministic searching algorithm with Word Importance Rank (WIR) could get stuck in the local optima. Differently, such a stochastic mechanism helps skip the local optima and further maximize the attacking performance.

**(b) Is FBA superior to the baselines in terms of imperceptibility?**

We evaluate the imperceptibility of different attack strategies, in terms of semantic similarities (USE), modification rate (Mod) and word alignment (ROUGE) between the original input text and its derived adversarial examples, shown in Table 4.6. Specifically, Table 4.6 demonstrates the proposed FBA mostly attains better performance than baselines. FBA is comparable to PSO for ROUGE while attacking BERT-C on AG's News, however, FBA maintains a higher semantic similarity (USE) than PSO on that data set. This means PSO's performance comes at the price of much inferior imperceptibility performance. Thus we can conclude that the proposed FBA provides the best performance for imperceptibility among baselines. There are two important reasons for such an outstanding performance for imperceptibility. The first factor is the customized target distribution in Eq. 4.18, which could help to avoid over-modifications. The second reason is that we apply both MLM and kNN to find the best substitution candidates which can provide more semantic similar substitutions.

**(c) Is the quality of adversarial examples generated by the FBA better than that crafted by the baselines?**

High-quality adversarial examples should be parsing-fluent and grammarly correct. From Table 4.7, we can find that FBA provides the lowest perplexity (PPL), which means the examples generated by FBA 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 FBA is substantially better than the other baselines, which indicates better quality of the adversarial examples. We attribute such performance to our method of finding word substitution, constructing the candidates set by applying both MLM and kNN for synonym searching.

Table 4.8: Comparisons between the FBA and its ablation WMP on AG's News dataset. The better performance is highlighted in bold.

| Models | Attack | Metrics | | | | | |
|---|---|---|---|---|---|---|---|
| | | SAR↑ | ROUGE↑ | USE↑ | Mod↓ | PPL↓ | GErr↓ |
| BERT-C | WMP | 21.39% | 0.5149 | 0.7083 | 17.2% | 331 | 0.22 |
| | FBA | **81.90%** | **0.8453** | **0.8001** | **11.0%** | **133** | **0.18** |
| TextCNN | WMP | 25.19% | 0.5149 | 0.7083 | 19.2% | 281 | 0.23 |
| | FBA | **93.12%** | **0.8711** | **0.8224** | **10.3%** | **133** | **0.13** |

**(d) Does the Metropolis-Hasting (MH) algorithm benefit the selection of the best adversarial candidates?**

To test the performance of the Metroplis-Hasting algorithm, we did an ablation study by making a comparison between FBA and WMP whose adversarial candidates are not selected by the Metropolis-Hasting algorithm. Specifically, we perform these two attacks, FBA and WMP, on two classifiers, BERT-C and TextCNN pre-trained on dataset AG's News, and the experimental results are shown in Table 4.8. From Table 4.8, FBA achieved better performance in both attack (SAR), imperceptibility (USE, Mod, ROUGE) and sentence quality (PPL, GErr), thus we can conclude that the Metropolis-Hasting algorithm is effective in selecting the adversarial candidates.

### 4.4.3 Ablation Studies

**(a) Evaluating the Effectiveness of MH**

To test the performance of the Metroplis-Hasting algorithm, we did an ablation study by making a comparison between FBA and WMP whose adversarial candidates are not selected by the Metropolis-Hasting algorithm. Specifically, we perform these two attacks, FBA and WMP, to two classifiers, BERT-C and TextCNN pre-trained on dataset AG's News, and the experimental results are shown in Tables 4.5, 4.6 and 4.7. From these tables , FBA achieved better performance in both attack (SAR), imperceptibility (USE, Mod, ROUGE) and sentence quality (PPL, GErr), thus we can conclude that the Metropolis-

Hasting algorithm is effective in selecting the adversarial candidates.

### (b) Evaluating the Impact of Actions: Removal, Insertion, and Substitution

To evaluate the influence of different actions on attack performance, we set up six distinct pairs with associated probabilities, as outlined in Table 4.9. In order to achieve equilibrium between the actions of removal and insertion, we consistently set their probabilities to be equal, represented as $\mathbf{P}_{ins} = \mathbf{P}_{rem}$.

The analysis of results from Table 4.9 can be categorized into three key facets: attack performance (SAR), imperceptibility (ROUGE, USE, Mod), and the quality of the generated examples (PPL, GErr). For attack performance, the group with $\mathbf{P}_{sub} = 0.4$ exhibited the highest efficacy. Meanwhile, the groups with $\mathbf{P}_{sub} = 0.6$ and $\mathbf{P}_{sub} = 0.8$ trailed closely, differentiated by a slim margin. The initial three groups underperformed, primarily due to constraints that only consider examples with a USE exceeding 0.5 and a modification rate below 0.25. Hence, even though these groups could generate adversarial candidates capable of deceiving the target models, such examples are not regarded as successful adversarial instances. Furthermore, the group focusing solely on substitution also showcased a commendable success rate against overall performance.

Regarding imperceptibility, we observed an initial increase in performance with a rise in the substitution probability $\mathbf{P}_{sub}$, which later began to decline. This trend can be attributed to the notion that a higher likelihood of insertions and removals can impact imperceptibility. Specifically, inserting or removing words may compromise language semantics, alignment, and parsing, as these actions can introduce significant losses or semantic redundancies. Simultaneously, if attackers focus exclusively on substitution, imperceptibility may suffer due to altering a larger set of words to bolster the attack's success. Hence, an optimal substitution probability likely exists that harmoniously balances imperceptibility. As for the quality of adversarial examples, there's a distinct pattern: the greater the substitution probability, the higher the quality. The experimental data suggests that inserting and removing affect sentence quality.

Table 4.9: Performance comparison of FBA on the AG News dataset against the TextCNN victim model using different action probabilities. The top three performances are highlighted in bold, underlined, and italics.

| $\mathbf{P}_{sub}$ | $\mathbf{P}_{ins}$ | SAR | ROUGE | USE | Mod | PPL | GErr |
|---|---|---|---|---|---|---|---|
| 0.0 | 0.5 | 69.12% | 0.6711 | 0.7411 | 16.3% | 231 | 0.21 |
| 0.2 | 0.4 | 78.10% | 0.7011 | 0.7524 | 15.3% | 201 | 0.19 |
| 0.4 | 0.3 | **94.12%** | 0.7911 | 0.7771 | 15.1% | 179 | 0.17 |
| 0.6 | 0.2 | <u>93.12%</u> | **0.8711** | *0.8224* | **10.3%** | *133* | <u>0.13</u> |
| 0.8 | 0.1 | *91.12%* | <u>0.8211</u> | **0.8401** | <u>12.3%</u> | <u>112</u> | <u>0.13</u> |
| 1.0 | 0.0 | 87.01% | *0.8011* | <u>0.8333</u> | *14.3%* | **110** | **0.12** |

Our findings indicate that the performance is suboptimal when focusing solely on substitution or when excluding substitution altogether. This underscores the importance of considering all actions – substitution, removal, and insertion – to bolster the attack's effectiveness. It's imperative to gauge the overall success of adversarial attacks across three dimensions: attack potency, imperceptibility, and sentence quality. Engaging in adversarial attacks often necessitates trade-offs between imperceptibility and sentence quality, as documented in [51, 59]. Given varying attack objectives, attackers can adjust the substitution probability. Based on our experiments, a substitution probability of 0.6 (denoted as $\mathbf{P}_{sub} = 0.6$) is recommended, as it strikes an ideal balance between attack efficacy and imperceptibility without undermining the textual quality.

#### (c) Evaluating the Effectiveness of Word Candidates Selection

Choosing the appropriate word candidates for substitution and insertion actions is crucial, as it directly impacts the success rate of attacks and imperceptibility. To evaluate the effectiveness of our word candidates selection method, we undertook an ablation study. This study compared performances utilizing a thesaurus (specifically, WordNet[54]), Masked Language Model (MLM), and Nearest Neighbors (NN) under L1, L2, and infinite norms. As shown from Table 4.10, our proposed method (MLM+$L_1$, MLM+$L_2$, MLM+$L_{\text{inf}}$) for word candidate search exhibited superior performance than the baselines. We attribute this success to two main factors. Firstly, WMP utilizes NN to identify 'potential' synonyms that, although not always precise, capture the desired meaning. Secondly,

Table 4.10: Performance metrics for FBA against the TextCNN model on the AG News dataset using varied word candidate selection methods. The best three performances for each metric are highlighted in bold, underline, and italics.

| Methods | SAR | ROUGE | USE | Mod | PPL | GErr |
|---|---|---|---|---|---|---|
| WordNet | 55.12% | 0.7103 | 0.7231 | 15.1% | 260 | 0.21 |
| $L_{\text{inf}}$ | 63.06% | 0.7401 | 0.7010 | 14.8% | 281 | 0.20 |
| $L_1$ | 70.19% | 0.7419 | 0.7533 | 14.1% | 209 | 0.18 |
| $L_2$ | 72.88% | 0.7812 | 0.7695 | 14.0% | 201 | 0.19 |
| MLM | 81.12% | 0.7731 | 0.7031 | 15.3% | 178 | *0.17* |
| MLM+$L_{\text{inf}}$ | *88.12%* | *0.8441* | *0.8001* | *13.3%* | *140* | *0.17* |
| MLM+$L_1$ | <u>92.42%</u> | **0.8713** | <u>0.8194</u> | <u>11.1%</u> | <u>136</u> | **0.14** |
| MLM+$L_2$ | **93.12%** | <u>0.8711</u> | **0.8224** | **10.3%** | **133** | <u>0.13</u> |

Table 4.11: The time efficiency of attack algorithms evaluated with BERT-C on the Emotion and IMDB dataset. The metric of efficiency is second per example, which means a lower metric indicates a better efficiency. The horizontally best 3 methods will be bold, underlined and italic.

| Datasets | WMP | A2T | BAE | FAGA | BERT.A | CLARE | PWWS | PSO | FBA |
|---|---|---|---|---|---|---|---|---|---|
| Emotion | 100.7 | 162.4 | <u>21.7</u> | 414.0 | 707.9 | 130.5 | **0.7** | *73.8* | 120.2 |
| IMDB | *155.7* | 431.4 | <u>81.1</u> | 781.0 | 1007.9 | 170.5 | **3.7** | 166.9 | 159.2 |

the MLM is crucial in improving the sentence's parsing structure. Furthermore, our observations indicate that the L2 norm marginally surpasses the L1 norm and significantly outperforms the infinite norm. While NN methods are not limited to any particular norm, our experimental results demonstrate the commendable efficacy of both L1 and L2 norms.

### 4.4.4 Derivative Attacks and Retraining

In this subsection, we will explore various derivative attacks leveraging the proposed methods. These include transfer attacks, where the attack model is transferred to a different target model; target attacks, which aim to cause misclassifications for specific targeted samples; and attacks targeting defense mechanisms employed to safeguard machine learning models. We will delve into the intricacies of each attack type and evaluate their effectiveness against state-of-the-art models and defenses.

Table 4.12: Targeted attack results on Emotion dataset. The better-performed attack is highlighted in bold.

| Models | Attack | Metrics | | | | | |
|---|---|---|---|---|---|---|---|
| | | SAR↑ | ROUGE↑ | USE↑ | Mod↓ | PPL↓ | GErr↓ |
| BERT-C | PWWS | 21.23% | 0.4541 | 0.6012 | 13.1% | 341 | 0.28 |
| | FBA | **57.21%** | **0.5101** | **0.7732** | **11.3%** | **299** | **0.22** |
| TextCNN | PWWS | 32.61% | 0.5603 | 0.6320 | 21.3% | 411 | 0.29 |
| | FBA | **65.07%** | **0.6198** | **0.6511** | **15.1%** | **223** | **0.28** |

#### 4.4.4.1 Transferability

Transferability of adversarial examples implies whether the adversarial samples generated to mislead a seimportant evaluation metric in adversarial attacks [43, 87, 90]. To evaluate the transferability of the adversarial attacks, we exchange the adversarial examples generated on BERT-C and TextCNN, and let them attack the other side. Fig 4.1 shows the classification accuracy results of transferred adversarial examples. Note that the lower the accuracy, the higher the transferability. From Fig 4.1, it can be seen that our method attains the best transfer attack performance.

#### 4.4.4.2 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. The targeted attack is regarded as a more harmful attack compared with untargeted attacks, since targeted adversarial attacks give the attackers more control over the final predicted label of the perturbed text. FBA can be easily adapted to targeted attack by modifying $1 - F_y(x')$ to $F_{y'}(x')$ in the definition of $R$ in Eq. 4.19. The targeted attack experiments are conducted on the Emotion dataset. The results are shown in Table 4.12 which demonstrates that the proposed FBA achieves better attacking performance (SAR) and imperceptibility performance (ROUGE, USE).

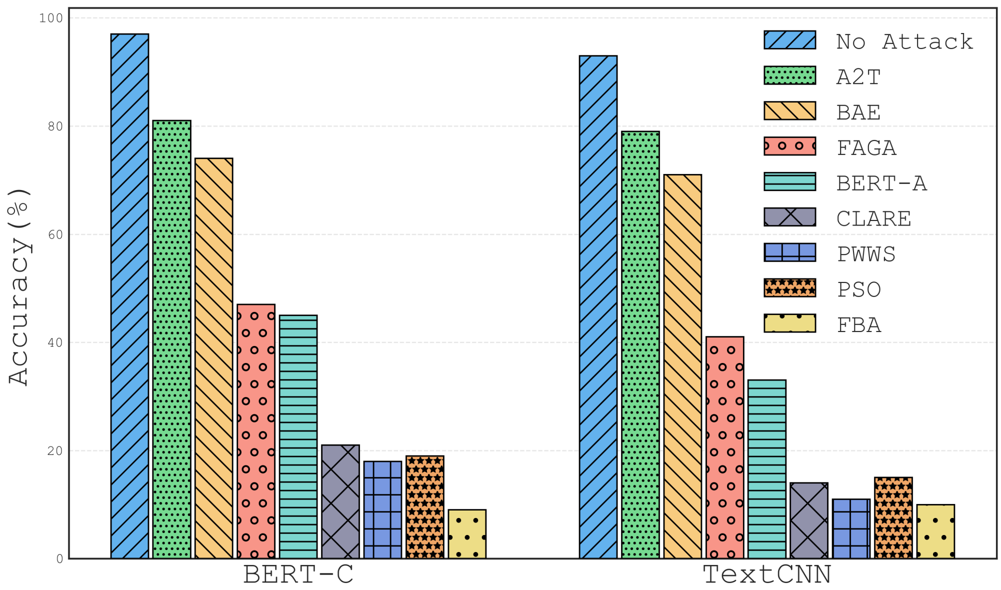


Figure 4.1: Performance of transfer attacks to victim models (BERT-C and TextCNN) on Emotion. The decreased accuracy of the victim models indicates an increased level of transferability, with lower values indicating improved performance in this regard.

#### 4.4.4.3 Adversarial Retraining

Since adversarial examples should be regarded as features rather than bugs [30], adversarial retraining is an effective way of using these features to improve the model's accuracy and robustness. To test the accuracy of adversarially retrained classifiers, we randomly generate 1000, 2000, 3000, 4000, 5000, and 6000 adversarial examples from the training set of SST2 and then append them to the training set for retraining the TextCNN. Figure 4.2 shows the model's accuracy on the clean test set after adversarial training versus appending the different numbers of adversarial examples. From Figure 4.2, we find that the classifier trained with adversarial examples achieves the best accuracy for adding the same number of adversarial examples from FBA. In addition, we also evaluate the robustness of the retrained model by applying FAGA to attack the retrained models. Results in Fig 4.3 show that all retrained victim models can defend against the attacks to a certain degree, and the retrained model with adversarial data from FBA is even more robust than baselines. The reason for improved accuracy and robustness is that adversarial attacks can augment informational features targeting the weak spots of the classifier. At the same time, FBA with the best attacking performance can generate more robust and

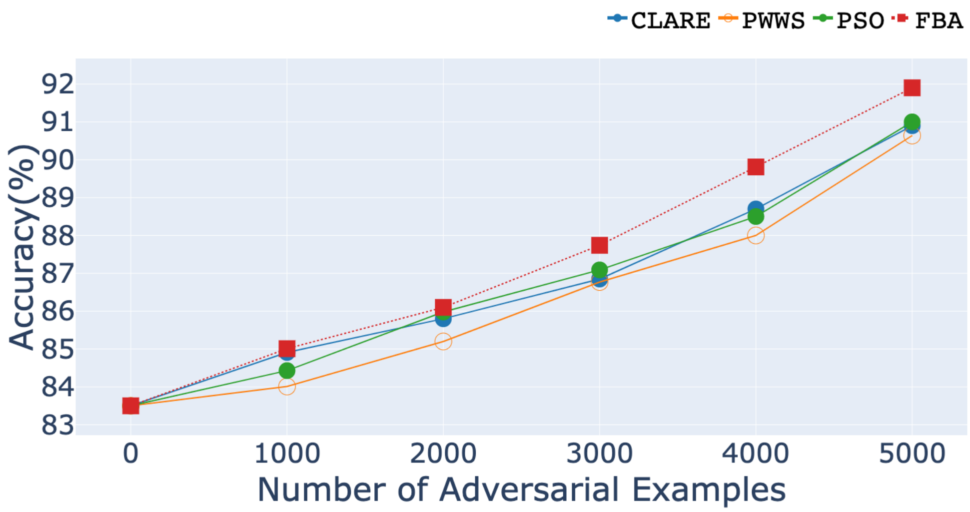


Figure 4.2: Retraining accuracy of TextCNN with different numbers of adversarial examples included in the retraining. The higher the accuracy, the better the performance of the retraining.

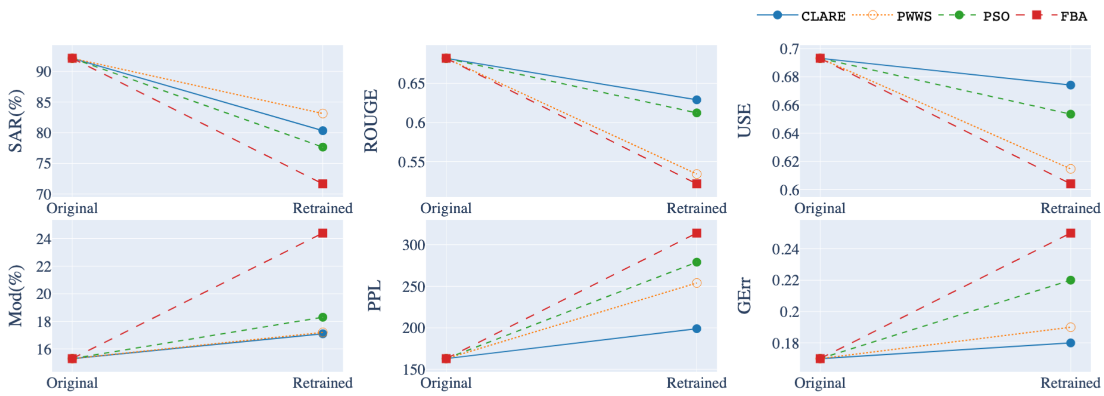


Figure 4.3: We employ FAGA to attack the adversarial retrained TextCNNs which joins adversarial examples from different attacking strategies (CLARE, PWWS, PSO and FBA) to the training set of SST2. The lower metrics (SAR, ROUGE, USE) suggest a better performance in robustness while The higher metrics (Mod, PPL, GErr) suggest a better performance in robustness.

informative features to adapt the original data domain to the true domain. Therefore, FBA achieves the best retrain performance in accuracy and robustness.

### 4.4.5 Complexity and Qualitative Results

Experiments were run on a RHEL 7.9 system with an Intel(R) Xeon(R) Gold 6238R CPU (2.2GHz, 28 cores - 26 enabled, 38.5MB L3 Cache), an NVIDIA Quadro RTX 5000 GPU (3072 Cores, 384 Tensor Cores, 16GB memory), and 88GB RAM.

Table 4.11 presents the time taken to attack BERT and TextCNN classifiers on the Emotion dataset. Time efficiency is measured in seconds per example, where a lower value denotes better efficiency. As observed from Table 4.11, while our WMP and FBA methods take longer than certain static baselines like PWWS and BAE, they outperform others such as CLARE, FAGA, A2T, and BA in terms of efficiency. It is noted that the extended run time of our methods compared to some baseline approaches suggests the additional time invested in seeking more optimal adversarial examples.

## 4.5 Summary and Discussion

In conclusion, the chapter presents a novel FBA algorithm for creating natural language adversarial examples, which not only enables successful attacks on textual classifiers but also enhances the models' accuracy and robustness through adversarial retraining. While adversarial examples are considered features, not bugs, they pose a threat to NLP models. Adversarial retraining, despite its effectiveness, can be costly and potentially degrade model accuracy.

# Chapter 5

# Reversible Jump Attack to Textual Classifiers with Modification Reduction

Crafting optimal adversarial examples requires balancing successful attacks and controlled imperceptibility. Existing optimization algorithms like Genetic Attack (GA)[1] and Particle Swarm Optimization (PSO) [90] face challenges like low efficiency due to large search spaces and compromised semantic integrity from synonym substitutions. Hierarchical search methods based on word saliency rank (WSR) have drawbacks: 1) Difficulty setting the optimal number of perturbed words for large datasets; 2) Reduced search domain by only attacking words ordered by WSR. To address these limitations, we propose two algorithms: Reversible Jump Attack (RJA) using randomization to enlarge the search space, and Metropolis-Hasting Modification Reduction (MMR) to enhance the imperceptibility of generated adversarial examples.

To address identified challenges, we introduce two novel black-box, word-level adversarial algorithms: Reversible Jump Attacks (RJA) and MH Modification Reduction (MMR). RJA employs the Reversible Jump sampler to dynamically adjust the number of perturbed words and select high-quality adversarial candidates based on semantic similarity. MMR aims to reverse RJA's changes by restoring original words and updating substitutions to maintain attack effectiveness. By integrating RJA and MMR, our approach

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Word Saliency | 3.1 | -1.3 | 13.1 | 8.3 | 0.1 | 14.1 | -0.1 | -4.2 | 11.1 | 5.9 | |
| Original input | I | do | not | like | this | movie | because | I | hate | thrillers | Original Label: 0 |
| Genetic Attack | I | do | not | like | this | movie | because | I | hate | **science** | Predicted Label: 1 |
| PWWS Attack | I | do | **really** | like | this | **photo** | because | I | **dislike** | thrillers | Predicted Label: 0 |
| RJA-MMR | I | do | not | like | this | movie | **as** | I | hate | thrillers | Predicted Label: 1 |

Figure 5.1: An example to show attack performance of optimizing attack (Genetic attack), hierarchical 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.

optimizes adversarial attacks by balancing semantic coherence with minimal perturbations. This effectiveness is clearly illustrated in an example Fig 5.1 where RJA-MMR significantly outperforms traditional Genetic and hierarchical attacks.

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 chapter is structured as follows. We first review adversarial attacks for NLP models and the Markov Chain Monte Carlo methods in NLP in Section 5.1. Then, we detail our proposed method in Section 5.2. We evaluate the performance of the proposed method through empirical analysis in Section 5.3. We conclude the chapter with suggestions for future work in Section 5.4.

## 5.1 Preliminary

Markov chain Monte Carlo (MCMC) [50], a statistically generic method for approximate sampling from an arbitrary distribution, can be applied in a variety of fields, such as optimization [70], machine learning [14], quantum simulation [22] and icing models [25]. The main idea is to generate a Markov chain whose equilibrium distribution is equal to the target distribution [39]. There exist various algorithms for constructing chains, including the Gibbs sampler, Reversible Jump sampler [21], and Metropolis-Hasting (MH) algorithm [50]. 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 [41] 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 [23] 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 [36] 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.

Despite the applications in NLP, the MCMC can be applied to adversarial attacks on

NLP models. [92] 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 cannot access 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.

The Metropolis-Hasting (MH) [50] 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:

$$\alpha(\mathbf{z}'|\mathbf{z}) = \min\left\{\frac{f(\mathbf{z}')q(\mathbf{z} \mid \mathbf{z}')}{f(\mathbf{z})q(\mathbf{z}' \mid \mathbf{z})}, 1\right\} \tag{5.1}$$

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) [21] 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 [15], dimension reduction [68], and cross-dimensional optimization [39]. 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:

$$\alpha(\mathbf{z'}_{(m')}|\mathbf{z}_{(m)}) = \min\left\{\frac{f(\mathbf{z'}_{(m')})q(\mathbf{z}_{(m)} \mid \mathbf{z'}_{(m')})}{f(\mathbf{z}_{(m)})q(\mathbf{z'}_{(m')} \mid \mathbf{z}_{(m)})}, 1\right\} \tag{5.2}$$

$$q\left(\mathbf{z'}_{(m')}|\mathbf{z}_{(m)}\right) = p\left(\mathbf{z'}_{(m')}|m', \mathbf{z}_{(m)}\right) p\left(m'|\mathbf{z}_{(m)}\right), \tag{5.3}$$

where $m$ denotes the dimensions of the vector $\mathbf{z}_{(m)}$, $q\left(\mathbf{z'}_{(m')}|\mathbf{z}_{(m)}\right)$ in Eq. 5.3 illustrates the new transition function and $p\left(m'|\mathbf{z}_{(m)}\right)$ is the dimensional transition item. Comparing MH and RJS 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 [92].

## 5.2 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).

### 5.2.1 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 $Sem(x, \mathbf{x}^*) \leqslant \varepsilon$ for a domain-specific semantic similarity function $Sem(\cdot) : \mathcal{X} \times \mathcal{X} \to (0, 1)$, where the bound $\varepsilon \in \mathbb{R}$ 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 5.1.

Table 5.1: 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 $i$th 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. |
| $F(\cdot) : \mathcal{X} \to \mathcal{Y}$ | The victim classifier. |
| $Sem(\cdot) : \mathcal{X}^2 \to (0, 1)$ | The function measuring the semantic similarity. |
| $p(\mathbf{x}_{t+1}\|\mathbf{x}_t) : \mathcal{X} \to (0, 1)$ | The transition function from state $\mathbf{x}_t$ to $\mathbf{x}_{t+1}$. |
| $\pi(x) : \mathcal{X} \to (0, 1)$ | Target distribution. |
| $\alpha(\mathbf{x}_{t+1}\|\mathbf{x}_t) : \mathcal{X} \to (0, 1)$ | The acceptance probability. |

### 5.2.2 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

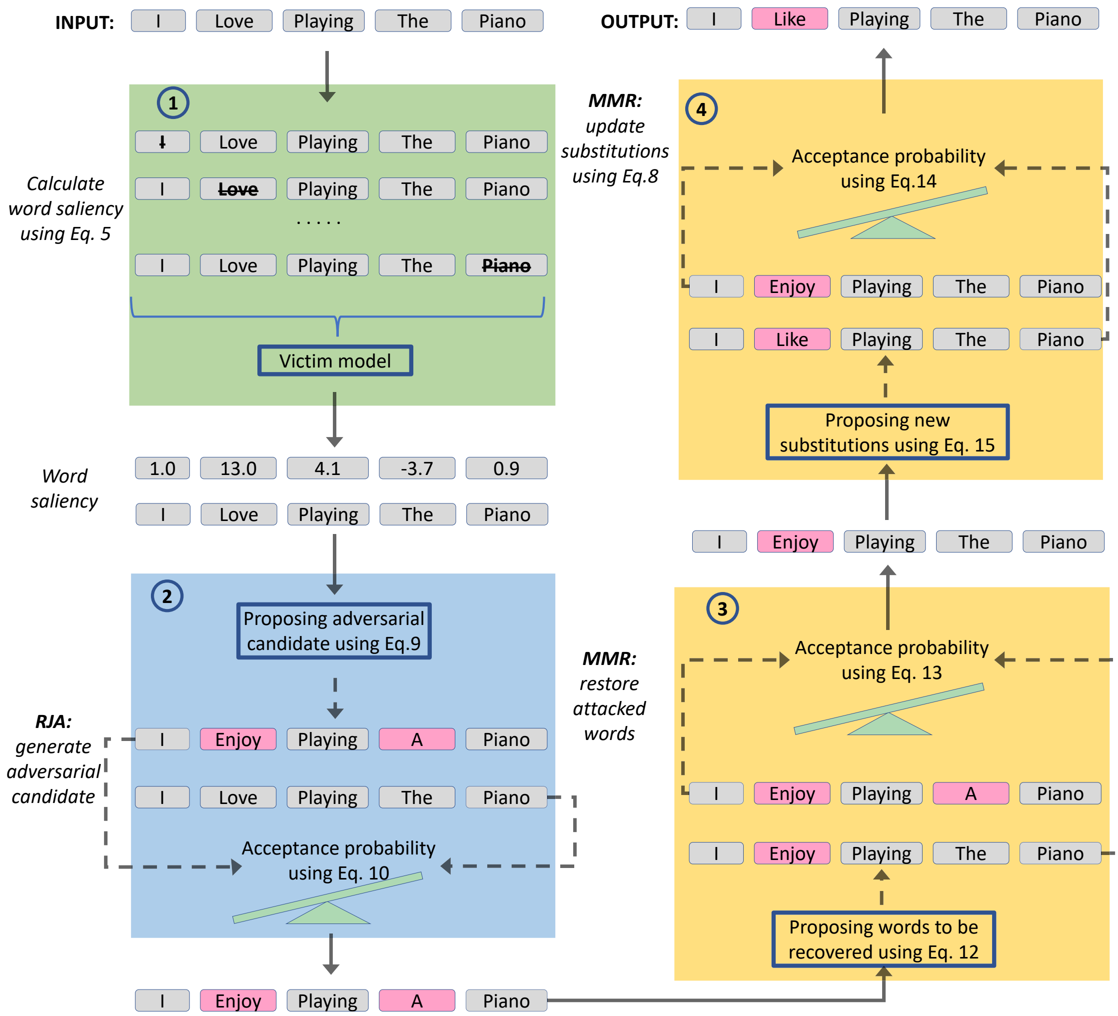


Figure 5.2: The workflow of our RJA-MMR. In this example, HAA generates an adversarial example with one word perturbed to attack a sentimental classifier with two labels (positive and negative). The block ① shows the calculation of word saliency. After obtaining the word saliency, we perform RJA in block ② which reflects the lines 4-15 in Algorithm 4. 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 5, respectively.

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. 5.2). This process iteratively refines the adversarial examples, guided by an **acceptance probability** mechanism.

This section first presents the transition function (Section 5.2.2.1) and then elaborates on the acceptance probability (Section 5.2.2.2), which builds upon the transition function.

#### 5.2.2.1 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 $I(w_i)$:

$$I(w_i) = F_{logit}(x) - F_{logit}(x \backslash w_i), \tag{5.4}$$
$$x \backslash w_i = [w_1, \ldots, w_{i-1}, w_{i+1}, \ldots, w_n],$$

where $F_{logit}(\cdot)$ is the classifier returning the logit of the correct class, and $x \backslash w_i$ 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 5.2.

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:

$$p(m_{t+1}|\mathbf{x}_t) = \begin{cases} \dfrac{\exp(l_1)}{\exp(l_1) + \exp(l_2)} & m_{t+1} = m_t - 1, \\ \dfrac{\exp(l_2)}{\exp(l_1) + \exp(l_2)} & m_{t+1} = m_t + 1, \end{cases} \tag{5.5}$$

$$\text{where} \quad l_1 = \sum_{w_i \in \mathbf{v}_t} I(w_i), \quad l_2 = \sum_{w_i \notin \mathbf{v}_t} I(w_i).$$

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 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 $\eta$ 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:

$$p(v_{tgt}|\mathbf{x}_t, m_{t+1}) = \begin{cases} \frac{1}{m_t} \quad v_{tgt} \in \mathbf{v}_t & \text{if } m_{t+1} = m_t - 1, \\ \frac{1}{n-m_t} \quad v_{tgt} \in \mathbf{x} \backslash \mathbf{v}_t & \text{if } m_{t+1} = m_t + 1. \end{cases} \tag{5.6}$$

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 [11] based thesauri such as OpenHowNet [64] and BabelNet [65] These thesauri are context-aware and at the same time can provide more synonyms than common thesaurus such as WordNet [52]. 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. 5.7.

$$p(s_{tgt}|w_{tgt}, m_{t+1}, \mathbf{x}_t) = \frac{1}{[\mathbb{G}]} \tag{5.7}$$

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. 5.5, 5.6 and 5.7, the final transition function is:

$$p_{_{RJA}}\left(\mathbf{x}_{t+1}|\mathbf{x}_t\right) = p\left(m_{t+1}|\mathbf{x}_t\right) p\left(w_{tgt}|m_{t+1}, \mathbf{x}_t\right) p\left(s_{tgt}|w_{tgt}, m_{t+1}, \mathbf{x}_t\right) \tag{5.8}$$

#### 5.2.2.2 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:

$$\pi(\mathbf{x}) = \frac{\left(1 - F_p(\mathbf{x})\right) Sem\,(x, \mathbf{x})}{C}, \tag{5.9}$$

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 [4]. $C = \sum_{\mathbf{x} \in \mathcal{X}} \left(1 - F_p(\mathbf{x})\right) Sem\,(x, \mathbf{x})$ is a positive normalizing factor to make $\sum_{\mathbf{x} \in \mathcal{X}} \pi(\mathbf{x}) = 1$ and $F_p(\cdot) : \mathcal{X} \to (0, 1)$ denotes the confidence of making right predictions where $\mathcal{X}$ represents text space. From Eq. 5.9, 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. 5.9 and **transition function** in Eq. 5.8, we formulate the acceptance probability for RJA, $\alpha_{_{RJA}}(\mathbf{x}_{t+1}|\mathbf{x}_t)$, as follows:

$$\alpha_{_{RJA}}(\mathbf{x}_{t+1}|\mathbf{x}_t) = \min\left\{\frac{\pi(\mathbf{x}_{t+1}) p_{_{RJA}}(\mathbf{x}_t|\mathbf{x}_{t+1})}{\pi(\mathbf{x}_t) p_{_{RJA}}(\mathbf{x}_{t+1}|\mathbf{x}_t)}, 1\right\} \tag{5.10}$$

After calculating $\alpha(\mathbf{x}_{t+1}|\mathbf{x}_t)$, we sample a random number $\varepsilon$ from a uniform distribution, $\varepsilon \sim Uniform(0, 1)$, if $\varepsilon < \alpha(\mathbf{x}_{t+1}|\mathbf{x}_t)$ 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 4 and block ② in Fig 5.2.

### 5.2.3 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),

**Algorithm 4** Reversible Jump Attack (RJA)

**Input: Input text:** $x$**, Number of iterations:** $T$
**Output: Adversarial candidate x**

$Adv\_set = [\quad]$
$\mathbf{x}_0 = x$
**for** t+1 in range($T$) **do**
  Sample $m_{t+1}$ given $\mathbf{x}_t$ with Eq. 5.5
  Sample $\mathbf{s}_{t+1}$ given $\mathbf{x}_t$ and $m_{t+1}$ with Eq. 5.6
  Sample $\mathbf{v}_{t+1}$ given $\mathbf{v}_t$, $m_{t+1}$ and $\mathbf{s} + \mathbf{1}$ with Eq. 5.7
  Craft $\mathbf{x}_{t+1}$ with $m_{t+1}$, $\mathbf{s}_{t+1}$, $\mathbf{v}_{t+1}$ and $\mathbf{x}_{t-1}$,
  Calculate the acceptance probability, $\alpha(\mathbf{x}_t|\mathbf{x}_{t-1})$ with Eq. 5.10
  Sample $\varepsilon$ from Uniform distribution, Uniform(0,1)
  **if** $\varepsilon < \alpha(\mathbf{x}_t|\mathbf{x}_{t-1})$ **then**
    $\mathbf{x}_{t+1} = \mathbf{x}_{t+1}$
    $Adv\_set = [Adv\_set,\ \mathbf{x}_{t+1}]$
  **else**
    $\mathbf{x}_{t+1} = \mathbf{x}_t$
    $Adv\_set = [Adv\_set,\ \mathbf{x}_{t+1}]$
  **end if**
**end for**
**return** An Adversarial candidate set $Adv\_set$
Choose the candidate which successfully fools the classifier with lease semantic sacrifice as an adversarial example **x**.
**return** Adversarial candidate **x**

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 Sections 5.2.3.1 and 5.2.3.2 respectively.

#### 5.2.3.1 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:

$$p(s^r|\mathbf{x}_t) = \begin{cases} \dfrac{\exp(I(w_i))}{1+\exp(I(w_i))} & \text{if } s^r = s_i \text{ (continue to attack),} \\ \dfrac{1}{1+\exp(I(w_i))} & \text{if } s^r = w_i \text{ (attack cancelled),} \end{cases} \tag{5.11}$$

$$p_{restore}(\mathbf{x}_t^r|\mathbf{x}_t) = \prod_{s^r \in \mathbf{s}_t} p(s^r|\mathbf{x}_t) \tag{5.12}$$

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. 5.5. To further investigate the quality of such a candidate, we apply the target distribution, $\pi(\mathbf{x})$, in Eq. 5.9 to construct the following acceptance probability:

$$\alpha_{restore}(\mathbf{x}_t^r|\mathbf{x}_t) = \min\left(\frac{\pi(\mathbf{x}_t^r)p_{restore}(\mathbf{x}_t|\mathbf{x}_t^r)}{\pi(\mathbf{x}_t)p_{restore}(\mathbf{x}_t^r|\mathbf{x}_t)}, 1\right) \tag{5.13}$$

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

#### 5.2.3.2 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. 5.7 for sampling substitution combinations. In essence, the MMR updating utilizes the candidate proposing function (Eq. 5.7) 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:

$$\alpha_{update}(\mathbf{x}_t^u|\mathbf{x}_t^r) = \min\left(\frac{\pi(\mathbf{x}_t^u)p_{update}(\mathbf{x}_t^r|\mathbf{x}_t^u)}{\pi(\mathbf{x}_t^r)p_{update}(\mathbf{x}_t^u|\mathbf{x}_t^r)}, 1\right), \tag{5.14}$$

$$p_{update}(\mathbf{x}_t^u|\mathbf{x}_t^r) = \prod_{s_i \in \mathbf{s}_t^r} p(s_i|w_i, m_t^r, \mathbf{x}_t^r), \tag{5.15}$$

where $p(s_i|w_i, m_t^r, \mathbf{x}_t^r)$ is identical to that in Eq. 5.7.

By iteratively running $T$ times MH algorithms for substitution restoring and updating with acceptance probabilities in Eq. 5.13 and Eq. 5.14, 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 $\mathbf{x}^*$. The suggested MMR technique has potential applications beyond our RJA method, potentially minimizing alterations required in various other attack strategies as well. The whole process of MMR is illustrated in Algorithm 5 and block ③-④ in Fig 5.2.

## 5.3 Experiments and Analysis

In this section, we comprehensively evaluate the performance of our method against the current state of the art and the basical experimental setting is provided in Sec. 5.3.1. Besides the main results (Sec. 5.3.2) of attacking performance and imperceptibility, we also conduct experiments on ablation studies (Sec. 5.3.3), derivative attacks (Sec. 5.3.4), adversarial retraining (Sec. 5.3.5), efficiency and attacking performance (Sec. 5.3.6).

We evaluate the effectiveness our methods on three widely-used and publicly available benchmark datasets: AG's News [94], Emotion [73], SST2 [75] and IMDB[49]. Specifically, AG's News is a news classification dataset with 127,600 samples belonging to 4 topic classes, *World, Sports, Business, Sci/Tech*. Emotion [73] is a dataset with 20,000 samples and 6 classes, *sadness, joy, love, anger, fear, surprise*. SST2 [75] is a binary class (*positive and negative*) topic dataset with 9,613 samples. The IMDB dataset [49], comprising movie reviews from the Internet Movie Database, is predominantly utilized

**Algorithm 5** 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. 5.11
  Calculate the acceptance probability, $\alpha(\mathbf{x}_t|\mathbf{x})$ with Eq. 5.13
  Sample $\varepsilon$ from Uniform distribution, Uniform(0,1)
  **if** $\varepsilon < \alpha(\mathbf{x}_t^r|\mathbf{x})$ **then**
    $\mathbf{x}_t^r = \mathbf{x}_t^r$
  **else**
    $\mathbf{x}_t^r = \mathbf{x}$
  **end if**
  Sample $\mathbf{x}_t^u$ to update the substitutions in $\mathbf{x}_t^r$ with Eq. 5.15
  Calculate the acceptance probability, $\alpha(\mathbf{x}_t^u|\mathbf{x}_t^r)$ with Eq. 5.14
  **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**
    $\mathbf{x}_t^u = \mathbf{x}_t^r$
  **end if**
  Take $\mathbf{x}_t^u$ as RJA's input for next iteration
  $Adv\_set = [Adv\_set,\ \mathbf{x}_t^u]$
  **return** $Adv\_set$
**end for**
**return** An Adversarial candidate set $Adv\_set$
Choose the candidate with the least modification from $Adv\_set$ as the final adversarial example $\mathbf{x}^*$.
**return** The final adversarial example $\mathbf{x}^*$

for binary sentiment classification, categorizing reviews into 'positive' or 'negative' sentiments. The details of these datasets can be found in Table 5.2. To ensure reproducibility, we provide the code and data used in the published paper [60].

### 5.3.1 Main Experiments Settings

In this subsection, the basic experimental settings such victim models, baselines and metrics will be introduced.

Table 5.2: 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<br>TextCNN | 94%<br>90% |
| Emotion | 20,000 | 19.14 | 6 | Sentiment analysis | BERT-C<br>TextCNN | 97%<br>93% |
| SST2 | 9,613 | 19.31 | 2 | Sentiment analysis | BERT-C<br>TextCNN | 91%<br>83% |
| IMDB | 50,000 | 19.31 | 2 | Movie review | BERT-C<br>TextCNN | 93%<br>88% |

#### 5.3.1.1 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.

**BERT-based Classifiers** 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[1]. 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* [72] 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* [72], 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* [9]. The test accuracy of these BERT-based classifiers before they are under attacks are listed in Table 5.2 and these models are publicly accessible[2] [3] [4] [5].

[1] https://huggingface.co/

[2] https://huggingface.co/mrm8488/distilroberta-finetuned-age_news-classification

[3] https://huggingface.co/bhadresh-savani/distilbert-base-uncased-emotion

[4] https://huggingface.co/echarlaix/bert-base-uncased-sst2-acc91.1-d37-hybrid

[5] https://huggingface.co/lvwerra/distilbert-imdb

**TextCNN-based models** The other type of victim model is TextCNN [38], structured with a 100-dimension embedding layer followed by a 128-unit 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 5.2.

#### 5.3.1.2 Baselines

To evaluate the attacking performance, we use the TextAttack [55] framework to deploy the following baselines:

- AGA [1]: 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) [32]: it accelerates AGA by bounding the searching domain of genetic optimization.
- BERT-Base Adversarial Examples (BAE) [17]: 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) [92]: 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)[45]: it takes advantage of BERT-MLM to generate candidates and attacked words by the static WIR descending order.
- Probability Weighted Word Saliency (PWWS) [67]: it chooses candidate words from WordNet [54] and sorts word attack order by multiplying the word saliency and probability variation.

- TextFooler (TF) [33]: it ranks the important words with similar strategy with Eq. 5.4. 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) by Zang et al. (2020) [90] involves sourcing word alternatives from HowNet [10] and utilizing PSO for generating adversarial text. In this framework, each sample is viewed as a particle whose position requires optimization within the search space.

#### 5.3.1.3 Experimental Settings and Evaluation Metrics

For our RJA and RJA-MMR, we use the Universal Sentence Encoder (USE) [4] to measure the sentence semantic similarity for target distribution in Eq. 5.9. We experiment to find $k = 30$ substitution candidates, and to find these candidates' substitutions, we use *RoBERTa-large* [47] as the MLM for contextual infilling and utilize OpenHowNet [64] 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 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 [57] to obtain the number of grammatical errors.
- Perplexity (PPL) denotes a metric used to evaluate the fluency of adversarial examples [37, 90]. The perplexity is calculated using small-sized GPT-2 with a 50k-sized vocabulary [66].

- 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) [4].

Table 5.3: Results on SAR, Mod, and Sim metrics among the baselines and proposed methods on different datasets. The best performance is in bold.

| 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** |

Table 5.4: Results on PPL and GErr metrics among the baselines and proposed methods on different datasets. The best performance is in bold.

| 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** |

### 5.3.2 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 Tables 5.3 and 5.4. Moreover, we demonstrate adversarial examples crafted by various methods shown in Table 5.5. We manifest the three contributions mentioned in the Introduction section by answering three research questions:

#### 5.3.2.1 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 5.3. 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 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 importance rank (WIR), where the domain of the hierarchical search is limited to combinations of the neighbouring victim words from the WIR, 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 WIR order. Thus, the proposed method RJA achieves the best-attacking performance, with the highest successful attack rate (SAR).

Table 5.5: Adversarial examples of the Emotion dataset for victim classifier BERT-C. Original words are highlighted in blue, while substitutions are indicated in red. Besides the examples, the attack performance is measured by attacking success (Succ.) and confidence (Conf.) in making correct predictions. The lower confidence indicates better performance, and the successful attacks and lowest confidence are bold.

| Methods | Adversarial example | Succ. | Conf. |
|---|---|---|---|
| BAE | made a ~~wonderful~~ nasty new friend | **Yes** | 4.3% |
| AGA | made a ~~wonderful~~ beautiful new friend | No | 94% |
| FAGA | ~~made~~ introduced a ~~wonderful~~ beautiful new friend | No | 95% |
| MHA | made a ~~wonderful~~ ~~new~~ newly friend | No | 70% |
| BA | made a ~~wonderful~~ good ~~new~~ brand friend | No | 95% |
| PWWS | ~~made~~ seduce a wonderful ~~new~~ raw ~~friend~~ admirer | No | 99% |
| TF | made a ~~wonderful~~ strange new friend | **Yes** | 5.0% |
| PSO | ~~made~~ doomed a wonderful new friend | **Yes** | 0.92% |
| RJA-MMR | made a ~~wonderful~~ lovely new friend | **Yes** | **0.80%** |

#### 5.3.2.2 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 5.4. 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. 5.9) with a strong regularization of the semantic similarity in each iteration. Moreover, the HowNet are knowledge-graph-based thesaurus which 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.

#### 5.3.2.3 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 5.4, 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 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.

### 5.3.3 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.

#### 5.3.3.1 Effectiveness of RJA

We compare the attacking performance of our Reversible Jump Attack methods (RJA, RJA-MMR) and baselines in Table 5.3, 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 shows surpasses the strong baselines in most cases. Therefore, RJA is effective in terms of attacking performance.

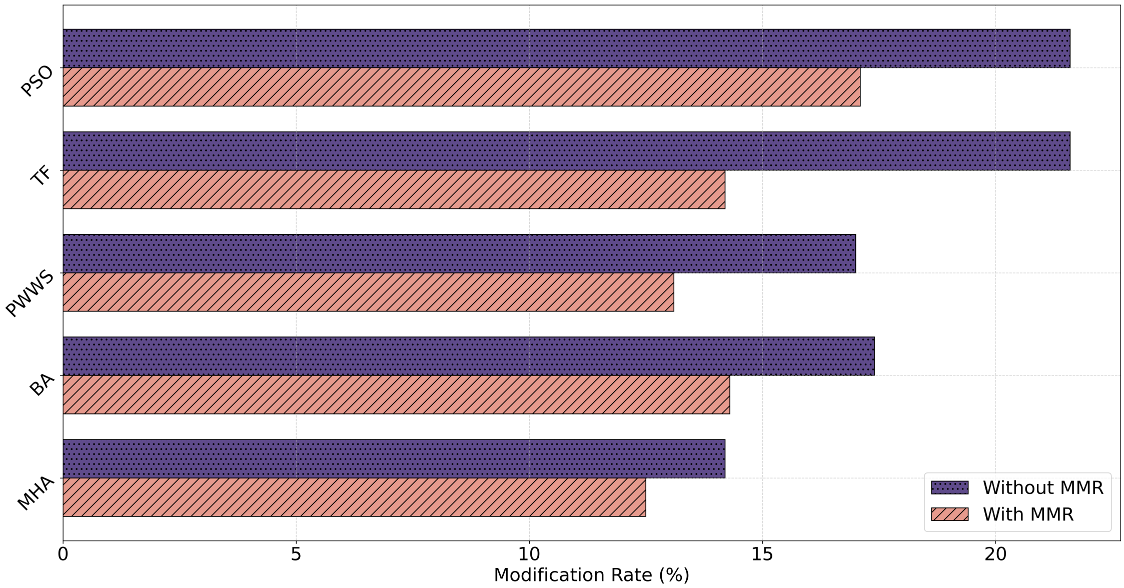


Figure 5.3: Comparisons on modification rates among attacking strategies (PSO, TF, PWWS, BA, MHA) with MMR and without MMR to attack the BERT-C on AG News dataset.

#### 5.3.3.2 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 5.3,we can find RJA-MMR has superior performance to RJA with lower modification rates. Moverover, the other baseline analysis results are shown in Fig 5.3. 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.

#### 5.3.3.3 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 Figure

Table 5.6: Performance metrics for RJA-MMR against the TextCNN model on the AG News dataset using varied word candidate selection methods. The best performances for each metric are highlighted in bold.

| Methods | SAR | USE | Mod | PPL | GErr |
|---|---|---|---|---|---|
| HowNet | 85.1 | 72 | 13.1 | 159 | 0.15 |
| MLM | 90.1 | 73 | 11.3 | 156 | **0.12** |
| HowNet+MLM | **93.8** | **82** | **9.9** | **127** | **0.12** |

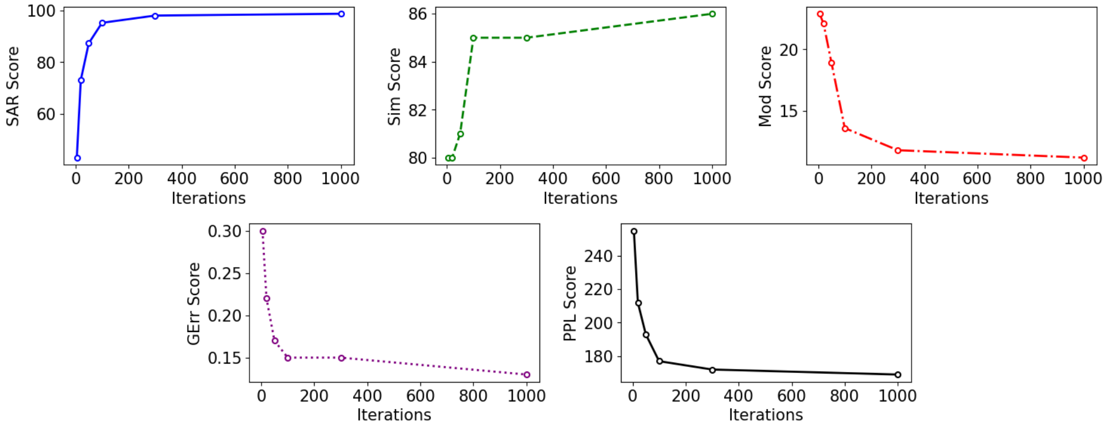


Figure 5.4: shows the progression of SAR, SIM, Mod, GErr, and PPL metrics for SST2 BERT over increased iterations (T). Performance trends and convergence points are visually represented.

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

#### 5.3.3.4 Effectiveness of the Word Candidates

In our ablation study outlined in Table 5.6, we assessed the impact of different word candidate selection methods on the RJA-MMR's performance against the TextCNN model using the AG News dataset. The evaluation encompassed three distinct strategies: HowNet, MLM, and a synergistic combination of both. Individually, HowNet and MLM demonstrated commendable performances, with MLM slightly edging ahead. However, the confluence of HowNet and MLM delivered unparalleled results, outclassing the individual methods across all metrics. This underscores the enhanced efficacy achieved through the integration of HowNet and MLM in bolstering adversarial attack potency.

Table 5.7: Robustness of BERT Models of Different Sizes on the Emotion Dataset

| Metric | BERT Tiny | BERT Base | BERT Large | BERT Huge |
|---|---|---|---|---|
| SAR | 98.1 | 97.3 | 94 | 90 |
| Sim | 6.0 | 7.1 | 7.7 | 7.9 |
| Mod | 93 | 90 | 88 | 87 |

#### 5.3.3.5 Robustness versus the Scale of Pre-trained Models

Examining Tables 5.3 and 5.4, 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. The findings, presented in Table 5.7, confirm an increase in robustness correlating with model size augmentation.

### 5.3.4 Derivative Attacks

In this subsection, we will explore various derivative attacks leveraging the proposed methods. These include transfer attacks, where the attack model is transferred to a different target model; target attacks, which aim to cause misclassifications for specific targeted samples; and attacks targeting defense mechanisms employed to safeguard machine learning models. We will delve into the intricacies of each attack type and evaluate their effectiveness against state-of-the-art models and defenses.

#### 5.3.4.1 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 [19]. 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 5.5.

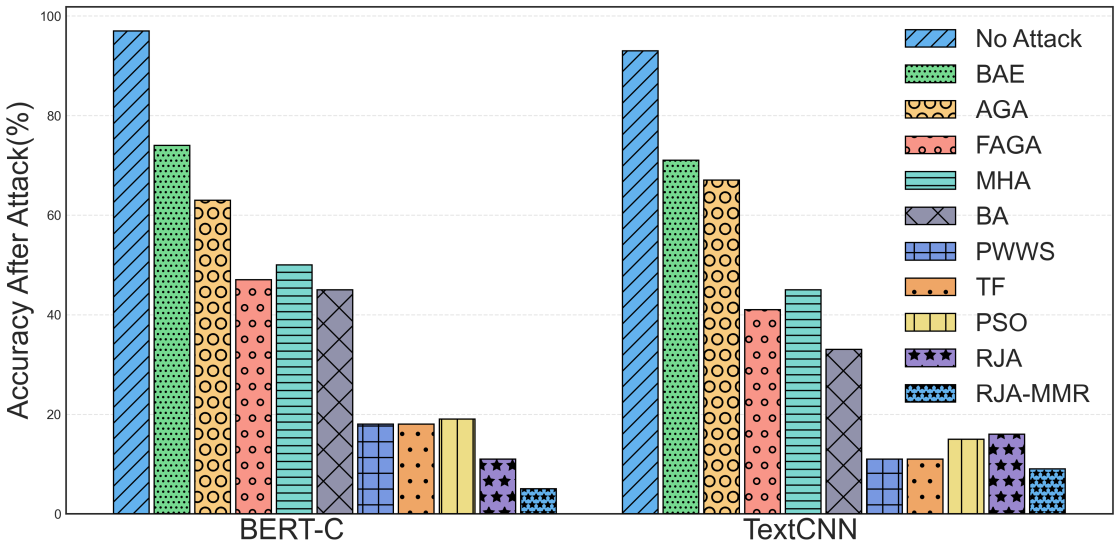


Figure 5.5: Performance of transfer attacks to victim models (BERT-C and TextCNN) on Emotion. A lower accuracy of the victim models indicates a higher transfer ability (i.e., the lower, the better).

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 achieves more than 80% successful rate, which is the best among baselines as illustrated in the Fig 5.5. 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.

#### 5.3.4.2 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. 5.9. The targeted attack experiments are conducted on the Emotion dataset. The results are shown in Table 5.8, which demonstrates that the proposed RJA-MMR achieves better performance than PWWS, in terms of attacking performance (SAR), imperceptibility performance (Mod, Sim) and sentence quality (GErr, PPL).

Table 5.8: Targeted attack and imperceptibility-preserving performance on the Emotion dataset. The victim models are BERT-C and TextCNN classifiers, and the baseline is PWWS. The statistics for better performance are vertically highlighted in bold.

| 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** |

#### 5.3.4.3 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)[56] approach, which excels at identifying adversarial examples. Conversely, for active defense, we incorporated Random Masking Training (RanMASK)[91], 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 5.9. The results show that our method outperforms the baselines.

Table 5.9: A comparative analysis of attack performance (SAR) against BERT-C when subjected to two defense mechanisms, FGWS and RanMASK, across IMDB and SST2 datasets. Performance metrics are highlighted in bold to emphasize superior results.

| 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** |

### 5.3.5 Adversarial Retraining

This section explores RJA-MMR's potential in improving downstream models' accuracy and robustness. Following [43], 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:

#### 5.3.5.1 Can adversarial retrainig help achieve better test accuracy?

As shown in Fig. 5.6, 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 [32, 87].

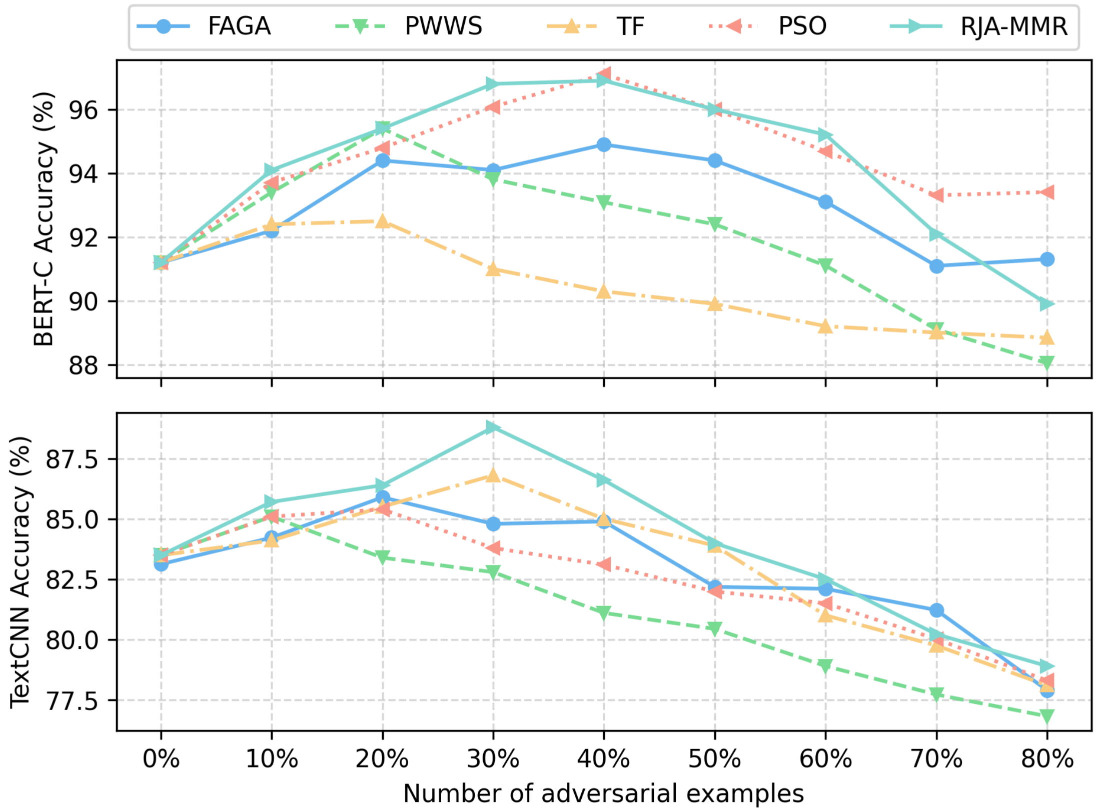


Figure 5.6: Results of adversarially trained BERT and TextCNN by inserting the different numbers of adversarial examples to the training set. The accuracy is based on the performance of the SST2 test set.

#### 5.3.5.2 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 5.7, 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.

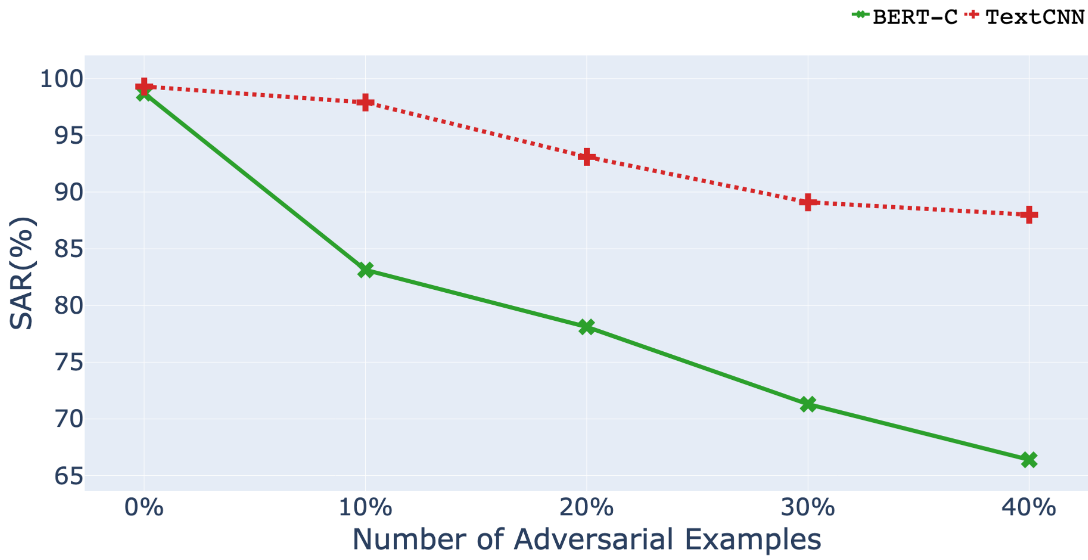


Figure 5.7: The success attack rate (SAR) of adversarially retrained models with different numbers of adversarial examples. A lower SAR indicates a victim classifier is more robust to adversarial attacks.

### 5.3.6 Efficiency and Attacking Preference

This section explores RJA-MMR's efficiency and attacking preference in terms of part-of-speech (POS).

#### 5.3.6.1 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 [82], and the collected statistics are shown in Table 5.10. 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 5.10 demonstrate that all the attacking methods heavily focus on the *noun*. Presumably, in the topic classification task, the prediction heavily depends

Table 5.10: POS preference with respect to choices of victim words among attacking methods. The tags with the horizontally highest and second highest proportion are bold and italic, respectively

| 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%** |

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.

#### 5.3.6.2 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 5.11 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 5.11 show that our RJA and RJA-MMR run longer than some static counterparts (PWWS,

Table 5.11: 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. Lower EC values denote higher efficiency. The top three methods are highlighted in bold, italic, and underlined.

| Methods | Metric | BAE | FAGA | MHA | BA | PWWS | TF | PSO | 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** |

BAE, TF) but are more efficient than the others, such as PSO, FAGA, MHA and BA. Nonetheless, 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 [39]. 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[21, 50]. Analysis of Table 5.11 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.

## 5.4 Summary and Discussion

In conclusion, this chapter of your thesis discusses the vulnerability of NLP classifiers to adversarial attacks. It introduces a novel method, RJA-MMR, which consists of two algorithms: Reversible Jump Attack (RJA) and Metropolis-Hasting Modification Reduction (MMR). RJA poses a threat to NLP classifiers by adaptively sampling the number of perturbed words, victim words, and their substitutions. MMR improves imperceptibility and lowers the modification rate by restoring attacked words without affecting attack performance. The RJA-MMR method has proven to deliver the best attack success, imperceptibility, and sentence quality among strong baselines. Adversarial examples, while posing a threat, are considered features, not bugs. A defense strategy, adversarial retraining, is proposed and tested, involving the integration of adversarial examples into the training set. This strategy has significantly improved the robustness of the classifiers, although the accuracy on clean data decreases when an excessive amount of adversarial examples are injected. This chapter contributes to the ongoing research on enhancing the robustness of NLP models against adversarial attacks.

Different from the HAA proposed in Chapter 3, the FBA in Chapter 4 and RJA-MMR aim to fool textual classifiers rather than NMTs. The key distinction lies in their objectives: degrading translation quality for NMTs (a continuous measure) versus flipping classes for classifiers (a discrete outcome). In Chapter 4, we primarily focused on enhancing attacking performance, with imperceptibility as a secondary concern. To address this, we introduce RJA-MMR in this chapter, comprising RJA and MMR. Both components mitigate excessive modifications without compromising the attacking effectiveness of FBA, as evidenced by the results in Tables 4.6 and 5.3.

# Chapter 6

# Conclusion and Future Work

In this chapter, a brief conclusion is presented in Section 6.1, and then several potential future research directions are proposed in Section 6.2.

## 6.1 Conclusion of This Thesis

In this thesis, we have undertaken a thorough investigation into the vulnerabilities of NMT and textual classifiers under adversarial attacks. Our exploration has yielded significant insights into the frailties of these models. It has provided a valid way to test and enhance the robustness of the current NMTs and textual classifiers. In Chapter 3, we have proposed HAA, which selects influential words by both translation-specific and language-centered attention and substitutes them with semantics-preserved word perturbations. Adversarial examples generated by our proposed method will affect not only the victim's word translation but also other words' translations. Experiments demonstrate that HAA delivers the best balance between the number of perturbed words and attacking performance among the competing methods. Although the generated adversarial examples can threat the NMTs, adversarial examples are not bugs but features [30]. To protect the NMT from the proposed attack, we believe that one possible defence strategy is adversarial retraining, which is usually done by joining the adversarial examples in the training set then retraining the models with the newly constructed training set. Although we did not perform the adversarial retraining in experiments, due to the lack of access to the victim models' structure since the Google and Baidu translations are online service and Helsinki NLP does not specify their model structures, by joining the adversarial features into model training, the model can be theoretically more robust against adversarial attacks. At the outset of this journey, we identified a crucial gap in the understanding of deep-learning-based NMTs' vulnerability to adversarial attacks. This initial finding steered us towards a more nuanced exploration of the weaknesses inherent in these models. Our first major contribution in this area was the development of the Hybrid Attentive Attack (HAA) method. This approach targeted the trade-offs between attacking performance and text perturbations in word-level adversarial examples. By focusing on key language-specific and sequence-focused words, HAA enabled semantic-aware substitutions that proved highly effective in attacking NMTs.

Building on the foundation of HAA, we then proposed the Fraud's Bargain Attack (FBA), utilizing the novel Word Manipulation Process (WMP). This methodology expanded the search space for adversarial examples, facilitating the generation of high-quality adversarial examples with increased success probabilities. This advancement marked a significant step forward in adversarial machine learning within the realm of NLP. In chapter 4, we specify FBA to exploit a stochastic process WMP to generate the adversarial candidates from an enlarged domain and employs the MH sampler to improve the quality of these candidates. With FBA, we not only make successful attacks on textual classifiers but also improve the accuracy and robustness of models by adversarial retraining. To protect the models from the attacks, 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, is proven effective for defending against these attacks in our experiments. However, adversarial retraining can be extremely expensive, and some research indicates that adversarial retraining can degrade the accuracy of the models [43, 67]. To the best of our knowledge, existing defence methods for adversarial examples mainly focus on the image domain, the defence methods studied in the text domain are not effective enough to prevent these maliciously crafted adversarial examples. Therefore, developing effective and robust defence schemes is a promising direction for future work which we plan to pursue.

The pinnacle of our research was the introduction of the Reversible Jump Attack (RJA) and Metropolis-Hasting Modification Reduction (MMR) algorithms. These methodologies revolutionized the generation of adversarial examples, balancing high effectiveness with heightened imperceptibility. RJA's innovative randomization mechanism and MMR's application of the Metropolis-Hasting sampler set new standards in the field by enhancing the subtlety of perturbations while maintaining attack success. Specifically, 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 quality among strong baselines.

In summary, this thesis represents a significant stride towards understanding and improving the security of NLP models via adversarial attacks. The methods developed challenge existing paradigms and pave the way for more resilient and robust AI systems. As AI continues to evolve, it is imperative that research in adversarial machine learning progresses in parallel to ensure secure and beneficial advancements in AI for society.

## 6.2 Future Work

This thesis introduces innovative adversarial attack methods aimed at assessing the performance of Deep Neural Networks (DNNs) and enhancing their robustness through adversarial retraining. Building on these investigations, we propose several promising avenues for future research. Our findings have profound implications for the future of

NLP, particularly concerning the security and robustness of neural network models.

- Advanced Defensive Techniques: Explore the development of novel defensive mechanisms tailored to counter the sophisticated adversarial attacks identified in this research. Investigate the integration of machine learning-based defenses, anomaly detection, and robust training methods to enhance model resilience. Additionally, it examines the potential of ensemble methods and adversarial training to fortify NLP models against a diverse range of adversarial perturbations.
- Transferability to Other NLP Applications: Adapt the Hybrid Attentive Attack (HAA), Fraud's Bargain Attack (FBA), Reversible Jump Attack (RJA), and Metropolis-Hasting Modification Reduction (MMR) methodologies for broader applications within NLP. Explore the transferability of attack methods across various NLP tasks, such as sentiment analysis, text summarization, and named entity recognition. Assess the robustness and effectiveness of the introduced methods in diverse linguistic contexts and languages beyond the scope of machine translation.
- Data Augmentation Strategies: Investigate advanced data augmentation techniques, including synthetic data generation and diversity-enhancing methods, to enrich the training dataset. Explore the impact of augmented datasets on improving model generalization and resilience against adversarial attacks. Consider the combination of data augmentation with privacy-preserving strategies to ensure the responsible use of sensitive information.
- Ethical Considerations in Adversarial Attacks: Address ethical considerations related to the responsible use of adversarial attacks in AI research. Explore guidelines and frameworks for conducting ethical adversarial machine learning research, considering potential societal implications. Investigate transparency and disclosure practices when using adversarial attacks, ensuring clear communication about the intent and impact of the research.

These detailed future directions aim to provide a comprehensive understanding of the proposed research avenues, emphasizing advanced defensive strategies, broader application of attack methodologies, enhanced data augmentation techniques, and ethical considerations in the evolving field of adversarial machine learning for NLP.

# Bibliography


[1] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, 2018.

[2] Y. Belinkov and Y. Bisk. Synthetic and natural noise both break neural machine translation. *International Conference on Learning Representations (ICLR)*, 2017.

[3] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, and W. Zhou. Hidden voice commands. In *25th USENIX Security Symposium USENIX Security 16)*, pages 513–530, 2016.

[4] D. Cer, Y. Yang, S.-y. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, et al. Universal sentence encoder for english. In *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*, pages 169–174, 2018.

[5] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. T. Koehn, and T. Robinson. One billion word benchmark for measuring progress in statistical language modeling. In *Interspeech*, 2013.

[6] M. Cheng, J. Yi, P.-Y. Chen, H. Zhang, and C.-J. Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3601–3608, 2020.

[7] Y. Cheng, L. Jiang, and W. Macherey. Robust neural machine translation with doubly adversarial inputs. In A. Korhonen, D. Traum, and L. Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy, July 2019. Association for Computational Linguistics.

[8] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale. In *ACL*, 2020.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[10] Z. Dong and Q. Dong. Hownet-a hybrid language and knowledge resource. In *International Conference on Natural Language Processing and Knowledge Engineering, 2003. Proceedings. 2003*, pages 820–824. IEEE, 2003.

[11] Z. Dong, Q. Dong, and C. Hao. Hownet and its computation of meaning. In *Coling 2010: Demonstrations*, pages 53–56, 2010.

[12] K. C. Dzmitry Bahdanau and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, abs/1409.0473, 2015.

[13] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. HotFlip: White-box adversarial examples for text classification. In I. Gurevych and Y. Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[14] X. Fan, B. Li, and S. Sisson. Rectangular bounding process. *Advances in Neural Information Processing Systems*, 31, 2018.

[15] Y. Fan and S. A. Sisson. Reversible jump mcmc. *Handbook of Markov Chain Monte Carlo*, pages 67–92, 2011.

[16] 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)*, pages 50–56. IEEE, 2018.

[17] 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)*, pages 6174–6181, 2020.

[18] Y. Gil, Y. Chai, O. A. Gorodissky, and J. Berant. White-to-black: Efficient distillation of black-box adversarial attacks. In *North American Chapter of the Association for Computational Linguistics*, 2019.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2015.

[20] S. Goyal, S. Doddapaneni, M. M. Khapra, and B. Ravindran. A survey of adversarial defenses and robustness in nlp. *ACM Computing Surveys*, 55(14s):1–39, 2023.

[21] P. J. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

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

[23] B. Harrison, C. Purdy, and M. O. Riedl. Toward automated story generation with markov chain monte carlo methods and deep neural networks. In *Thirteenth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2017.

[24] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] H. Herrmann. Fast algorithm for the simulation of ising models. *Journal of statistical physics*, 45(1):145–151, 1986.

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[27] H. Hosseini, S. Kannan, B. Zhang, and R. Poovendran. Deceiving google's perspective api built for detecting toxic comments. *arXiv preprint arXiv:1702.08138*, 2017.

[28] D. Hu. An introductory survey on attention mechanisms in nlp problems. In *Proceedings of SAI Intelligent Systems Conference*, pages 432–448. Springer, 2019.

[29] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[30] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systemas*, 32, 2019.

[31] R. Jia and P. Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031, Copenhagen, Denmark, Sept. 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215.

[32] R. Jia, A. Raghunathan, K. Goksel, and P. Liang. 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)*, pages 4129–4142, 2019.

[33] 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. In *AAAI*, 2020.

[34] M. Junczys-Dowmunt, R. Grundkiewicz, T. Dwojak, H. Hoang, K. Heafield, T. Neckermann, F. Seide, U. Germann, A. Fikri Aji, N. Bogoychev, A. F. T. Martins, and A. Birch. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia, July 2018.

[35] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, Oct. 2013.

[36] X. Kang and F. Ren. Sampling latent emotions and topics in a hierarchical bayesian network. *2011 7th International Conference on Natural Language Processing and Knowledge Engineering*, pages 37–42, 2011.

[37] K. Kann, S. Rothe, and K. Filippova. Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 313–323, 2018.

[38] Y. Kim. Convolutional neural networks for sentence classification. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.

[39] D. P. Kroese, T. Taimre, and Z. I. Botev. *Handbook of Monte Carlo Methods*. John Wiley & Sons, 2011.

[40] V. Kuleshov, S. Thakoor, T. Lau, and S. Ermon. Adversarial examples for natural language classification problems, 2018.

[41] K. Kumagai, I. Kobayashi, D. Mochihashi, H. Asoh, T. Nakamura, and T. Nagai. Human-like natural language generation using monte carlo tree search. In *CC-NLG*, 2016.

[42] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations. *ArXiv*, abs/1909.11942, 2020.

[43] D. Li, Y. Zhang, H. Peng, L. Chen, C. Brockett, M.-T. Sun, and W. B. Dolan. 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*, pages 5053–5069, 2021.

[44] J. Li, S. Ji, T. Du, B. Li, and T. Wang. Textbugger: Generating adversarial text against real-world applications. *Network and Distributed Systems Security (NDSS) Symposium 2019*, 2019.

[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)*, pages 6193–6202, 2020.

[46] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi. Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*, 2017.

[47] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692, 2019.

[48] T. Luong, H. Pham, and C. D. Manning. Effective approaches to attention-based neural machine translation. In *EMNLP*, 2015.

[49] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[50] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[51] P. Michel, X. Li, G. Neubig, and J. Pino. On evaluation of adversarial perturbations for sequence-to-sequence models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, 2019.

[52] G. A. Miller. Wordnet: A lexical database for english. *Commun. ACM*, 38:39–41, 1992.

[53] G. A. Miller. *WordNet: An Electronic Lexical Database*. MIT press, 1998.

[54] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4): 235–244, 1990.

[55] J. X. Morris, E. Lifland, J. Y. Yoo, J. Grigsby, D. Jin, and Y. Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp, 2020.

[56] M. Mozes, P. Stenetorp, B. Kleinberg, and L. Griffin. 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*, pages 171–186, 2021.

[57] D. Naber et al. A rule-based style and grammar checker. *Citeseer*, 2003.

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

[59] M. Ni, Z. Sun, and W. Liu. Fraud's bargain attacks to textual classifiers via metropolis-hasting sampling (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[60] M. Ni, Z. Sun, and W. Liu. Reversible jump attack to textual classifiers with modification reduction. *Machine Learning*, pages 1–31, 2024.

[61] M. Ni, Z. Sun, and W. Liu. Frauds bargain attack: Generating adversarial text samples via word manipulation process. *IEEE Transactions on Knowledge & Data Engineering*, 36(07):3062–3075, jul 2024. ISSN 1558-2191. doi: 10.1109/TKDE.2024.3349708.

[62] 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*, pages 49–54. IEEE, 2016.

[63] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[64] F. Qi, C. Yang, Z. Liu, Q. Dong, M. Sun, and Z. Dong. Openhownet: An open sememe-based lexical knowledge base. *arXiv preprint arXiv:1901.09957*, 2019.

[65] F. Qi, L. Chang, M. Sun, S. Ouyang, and Z. Liu. Towards building a multilingual sememe knowledge base: predicting sememes for babelnet synsets. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8624–8631, 2020.

[66] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. *OpenAI*, 2019.

[67] 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*, pages 1085–1097, 2019.

[68] R. Rincent, E. Kuhn, H. Monod, F.-X. Oury, M. Rousset, V. Allard, and J. Le Gouis. Optimization of multi-environment trials for genomic selection based on crop models. *Theoretical and Applied Genetics*, 130(8):1735–1752, 2017.

[69] H. Riza, M. Purwoadi, Gunarso, T. Uliniansyah, A. A. Ti, S. M. Aljunied, L. C. Mai, V. T. Thang, N. P. Thai, V. Chea, R. Sun, S. Sam, S. Seng, K. M. Soe, K. T. Nwet, M. Utiyama, and C. Ding. Introduction of the asian language treebank. In *2016 Conference of The Oriental Chapter of International Committee for Coordination and Standardization of Speech Databases and Assessment Techniques (O-COCOSDA)*, pages 1–6, 2016. doi: 10.1109/ICSDA.2016.7918974.

[70] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and computing in applied probability*, 1(2):127–190, 1999.

[71] S. Samanta and S. Mehta. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*, 2017.

[72] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[73] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404.

[74] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[75] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics.

[76] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867, 2020.

[77] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[78] S. Tan, S. Joty, M.-Y. Kan, and R. Socher. It's morphin' time! Combating linguistic discrimination with inflectional perturbations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, July 2020. Association for Computational Linguistics.

[79] Y. Tang, C. Tran, X. Li, P.-J. Chen, N. Goyal, V. Chaudhary, J. Gu, and A. Fan. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv e-prints*, pages arXiv–2008, 2020.

[80] J. Tiedemann. Parallel data, tools and interfaces in opus. In *LREC*, 2012.

[81] J. Tiedemann. The Tatoeba Translation Challenge – Realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1174–1182. Association for Computational Linguistics, Nov. 2020.

[82] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer. 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*, NAACL '03, page 173–180, USA, 2003. Association for Computational Linguistics.

[83] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[84] Y. Wang and M. Bansal. Robust machine comprehension models via adversarial training. *arXiv preprint arXiv:1804.06473*, 2018.

[85] W. Wu, D. Arendt, and S. Volkova. Evaluating neural machine comprehension model robustness to noisy inputs and adversarial attacks. *arXiv preprint arXiv:2005.00190*, 2020.

[86] X. Yang, W. Liu, J. Bailey, D. Tao, and W. Liu. Bigram and unigram based text attack via adaptive monotonic heuristic search. In *AAAI*, 2021.

[87] X. Yang, W. Liu, J. Bailey, D. Tao, and W. Liu. Bigram and unigram based text attack via adaptive monotonic heuristic search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 706–714, 2021.

[88] Y. Yang, D. Cer, A. Ahmad, M. Guo, J. Law, N. Constant, G. H. Abrego, S. Yuan, C. Tar, Y.-H. Sung, et al. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*, 2019.

[89] J. Y. Yoo and Y. Qi. Towards improving adversarial training of nlp models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 945–956, 2021.

[90] Y. Zang, F. Qi, C. Yang, Z. Liu, M. Zhang, Q. Liu, and M. Sun. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, 2020.

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

[92] H. Zhang, H. Zhou, N. Miao, and L. Li. Generating fluent adversarial examples for natural languages. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy, July 2019. Association for Computational Linguistics.

[93] 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 (TIST)*, 11(3):1–41, 2020.

[94] X. Zhang, J. J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. In *NIPS*, 2015.

[95] Z. Zhao, D. Dua, and S. Singh. Generating natural adversarial examples. *arXiv preprint arXiv:1710.11342*, 2017.