# Bigram and Unigram Based Text Attack via Adaptive Monotonic Heuristic Search

**Xinghao Yang[1]*, Weifeng Liu[2], James Bailey[3], Dacheng Tao[4], Wei Liu[1]**

[1]School of Computer Science, University of Technology Sydney, Australia, [2]School of Information and Control Engineering, China University of Petroleum (East China), China, [3]School of Computing and Information Systems, The University of Melbourne, Australia, [4]School of Computer Science, Faculty of Engineering, The University of Sydney, Australia
xinghao.yang@student.uts.edu.au, liuwf@upc.edu.cn, baileyj@unimelb.edu.au, dacheng.tao@sydney.edu.au, wei.liu@uts.edu.au

**Abstract**

Deep neural networks (DNNs) are known to be vulnerable to adversarial images, while their robustness in text classification are rarely studied. Several lines of text attack methods have been proposed in the literature, such as character-level, word-level, and sentence-level attacks. However, it is still a challenge to minimize the number of word distortions necessary to induce misclassification, while simultaneously ensuring the lexical correctness, syntactic correctness, and semantic similarity. In this paper, we propose the Bigram and Unigram based Monotonic Heuristic Search (BU-MHS) method to examine the vulnerability of deep models. Our method has three major merits. Firstly, we propose to attack text documents not only at the unigram word level but also at the bigram level to avoid producing meaningless outputs. Secondly, we propose a hybrid method to replace the input words with both their synonyms and sememe candidates, which greatly enriches potential substitutions compared to only using synonyms. Lastly, we design a search algorithm, i.e., Monotonic Heuristic Search (MHS), to determine the priority of word replacements, aiming to reduce the modification cost in an adversarial attack. We evaluate the effectiveness of BU-MHS on IMDB, AG's News, and Yahoo! Answers text datasets by attacking four popular DNNs models. Results show that our BU-MHS achieves the highest attack success rate by changing the smallest number of words compared with baselines.

## 1 Introduction

Deep neural networks (DNNs) have exhibited brittleness towards adversarial attacks in the image domain, where an adversarial image is intentionally modified with a only small number of pixel perturbations (Szegedy et al. 2014; Goodfellow, Shlens, and Szegedy 2015). This phenomenon raises great interest in the Computer Vision community, while the vulnerability of DNNs in Natural Language Processing (NLP) field is generally underestimated, especially for those security-sensitive NLP tasks, such as spam filtering, webpage phishing, and sentiment analysis (Atallah et al. 2001).

Compared to image attacks, there are non-trivial difficulties in crafting text adversarial samples. Firstly, the text adversarial samples should be lexical correct, syntactic correct, and semantic similar. This will ensure the adversarial modifications are imperceptible to human readers. Secondly, the words in text sequences are *discrete* tokens instead of continuous pixel values as in images. Therefore, it is infeasible to directly compute the model gradient with respect to every word. Thirdly, making small perturbations on many pixels may still yield a meaningful image from human perception perspectives. However, any small changes, even a single word, to text document can make a sentence meaningless.

| Original Input | Unigram Attack | Bigram Attack |
|---|---|---|
| New York | Fresh York | Empire State |
| Machine Learning | Device Learning | Data Mining |
| Primary School | Major School | Elementary School |

Table 1: Difference between unigram and bigram attacks.

Several text attack methods have been proposed, such as character-level attack, sentence-level attack, and word-level attack (Wang et al. 2019). Character-level attack (e.g., noise $\rightarrow$ nosie) leads to lexical errors, and sentence-level attack (i.e., inserting a whole sentence into the original text) often causes significant semantic changes. To avoid these problems, many recent works focused on word-level attacks that replace the original word with another carefully selected one (Zhang et al. 2019). However, existing methods mostly generate substitution candidates for every individual word (i.e., a unigram), which can easily break commonly used phrases, leading to meaningless outputs (e.g., high school $\rightarrow$ tall school). In addition, when sorting word replacement orders, most algorithms calculate the word important score (WIS) and attack them via a descending order of the WIS. There are different definitions of WIS, such as probability weighted word saliency (PWWS) (Ren et al. 2019) and the changes of DNNs' predictions before and after deleting a word (Jin et al. 2020), etc. One major drawback of using such a static attack order is word substitution inflexibility, e.g., sequentially selecting the top-3 WIS words $\{top1, top2, top3\}$ may not fool a classifier but sometimes the combination $\{top1, top3\}$ can make it.

In this work, we propose a new word-level attack method named Bigram and Unigram based Monotonic Heuristic Search (BU-MHS) which effectively addresses all the draw-

*Corresponding Author (xinghao.yang@student.uts.edu.au).

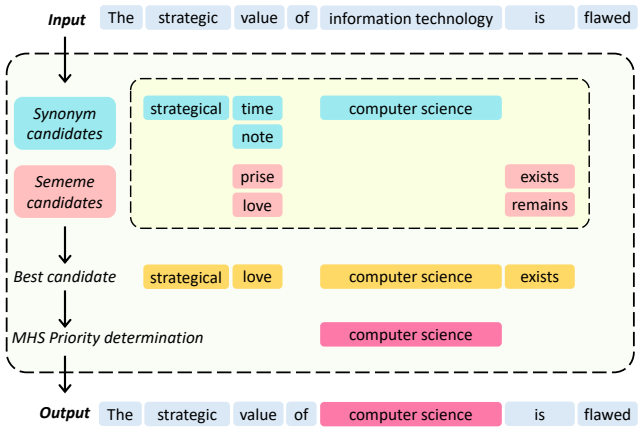


Figure 1: The workflow of BU-MHS. A CNN model is misled from "Sci/Tech" to "Business" by replacing one bigram.

backs above. Unlike traditional unigram word attack, we consider both unigram and bigram substitutions. In our approach, we generate more natural candidates by replacing a bigram with its synonyms (e.g., high school $\rightarrow$ secondary school). Table 1 lists several examples that illustrate the superiority of bigram attacks in comparison with unigram attacks. Additionally, we propose to replace input words by considering both their synonyms and sememe-consistent words. By incorporating these complementary candidates, we have better choices to craft high-quality adversarial texts.

More importantly, we propose an effective candidature search method, Monotonic Heuristic Search (MHS), to determine word priorities. The MHS inherits the best-performed candidate combinations from the previous generation and determines the next replacement word with a heuristic search. For instance, if changing the $\{top1\}$ word cannot mislead a classifier, the static methods used in the literature will select the combination $\{top1, top2\}$ in the second iteration, but our adaptive MHS will check more combinations, e.g., $\{top1, top2\}$, $\{top1, top3\}$, etc. Compared with static strategy, the MHS allows us to fool DNNs models with much fewer modifications, which is significant in reducing semantic changes and grammatical mistakes. Figure 1 illustrates an example of our algorithm. Our main contributions in this work are summarized as below:

- We propose to attack text documents not only at the unigram word level but also at the bigram level to generate natural adversarial samples and avoid semantic errors.
- We propose a hybrid approach to generate word substitutions from both synonym candidates and sememe candidates. Such a complementary combination enables us to craft more meaningful adversarial examples.
- We design MHS to effectively prioritize word replacements, which minimizes the number of word modifications and reduces semantic and syntactic mistakes.

## 2 Related Work

Text attack methods can be categorized into character-level, sentence-level, and word-level attacks (Wang, Jin, and He 2019). **Character-level attack** generates adversarial texts by deleting, inserting, or swapping characters (Belinkov and Bisk 2018; Ebrahimi et al. 2018). However, these character-level modifications lead to misspelled words, which can be easily detected by spelling check machines. **Sentence-level attack** concatenates an adversarial sentence before or after the original texts to confuse deep architecture models (Jia and Liang 2017; Wallace et al. 2019a), but they usually lead to dramatic semantic changes and generate human incomprehensible sentences (Gan and Ng 2019; Wallace et al. 2019b). **Word-level attack** replaces original input words with carefully picked words. The core problems are (1) how to select proper candidate words and (2) how to determine the word substitution order. Incipiently, Papernot et al. (2016) projected words into a 128-dimension embedding space and leveraged the Jacobian matrix to evaluates input-output interaction. However, a small perturbation in the embedding space may lead to totally irrelevant words since there is no hard guarantee that words close in the embedding space are semantically similar. Therefore, subsequent studies focused on synonym substitution strategy that search synonyms from the GloVe embedding space, existing thesaurus (e.g., WordNet and HowNet), or BERT Masked Language Model (MLM).

By using GloVe, Alzantot et al. (2018) designed a population-based genetic algorithm (GA) to imitate the natural selection. However, the GloVe embedding usually fails to distinguish antonyms from synonyms. For example, the nearest neighbors for $expensive$ in GloVe space are $\{pricey, cheaper, costly\}$, where $cheaper$ is its antonyms. Therefore, Glove-based algorithms have to use a counter-fitting method to post-process adversary's vectors to ensure the semantic constraint (Mrkšić et al. 2016). Compared with GloVe, utilizing well-organized linguistic thesaurus, e.g., synonym-based WordNet (Miller 1998) and sememe-based HowNet (Dong and Dong 2006), is a simple and easy implementation. Ren et al. (2019) sought synonyms using the WordNet synsets and ranked word replacement order via probability weighted word saliency (PWWS). Zang et al. (2020) manifested that the sememe-based HowNet can provide more substitute words than WordNet and proposed the Particle Swarm Optimization (PSO) to determine which group of words should be attacked. In addition, some recent studies utilized BERT MLM to generate contextual perturbations, such as BERT-Attack (Li et al. 2020) and BERT-based Adversarial Examples (BAE) (Garg and Ramakrishnan 2020). The pre-trained BERT MLM can ensure the predicted token fit in the sentence well, but unable to preserve the semantic similarity. For example, in the sentence "the food was [MASK]", predicting the [MASK] as $good$ or $bad$ are equally fluent but resulting in opposite sentiment label. Notably, all these work focused on unigram attacks.

## 3 Algorithm

This section details the proposed BU-MHS method. Formally, let $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \cdots, \mathbf{X}_N\}$ denote the input space containing $N$ sentences, and $\mathcal{Y} = \{\mathbf{Y}_1, \mathbf{Y}_2, \cdots, \mathbf{Y}_K\}$ represent the output space of $K$ labels. The DNNs classifier $F$ learns a mapping from text space to labels $F : \mathcal{X} \rightarrow \mathcal{Y}$.

### 3.1 Black-box Text Attack

We design our method in black-box settings where no network architectures, intermediate parameters or gradient information are available. The only capability of the black-box adversary is to query the output labels (confidence scores) of the threat model, acting as a standard user.

Given a well-trained DNNs classifier $F$, it aims to produce the correct label $\mathbf{Y}_{true} \in \mathcal{Y}$ for any input $\mathbf{X} \in \mathcal{X}$, i.e., $F(\mathbf{X}) = \mathbf{Y}_{true}$, by maximizing the posterior probability:

$$\arg\max_{\mathbf{Y}_i \in \mathcal{Y}} P(\mathbf{Y}_i|\mathbf{X}) = \mathbf{Y}_{true} \tag{1}$$

A rational text attack pursues a human-imperceptible perturbation $\Delta\mathbf{X}$ that can fool the classifier $F$ when it is added to the original $\mathbf{X}$. The altered input $\mathbf{X}^* = \mathbf{X} + \Delta\mathbf{X}$ is defined as the text adversarial example. Generally, a successful adversarial example can mislead a well-trained classifier into either an arbitrary label other than the true label

$$\arg\max_{\mathbf{Y}_i \in \mathcal{Y}} P(\mathbf{Y}_i|\mathbf{X}^*) \neq \mathbf{Y}_{true} \tag{2}$$

or a pre-specified label $\mathbf{Y}^*$

$$\arg\max_{\mathbf{Y}_i \in \mathcal{Y}} P(\mathbf{Y}_i|\mathbf{X}^*) = \mathbf{Y}^* \tag{3}$$

where $\mathbf{Y}^* \neq \mathbf{Y}_{true}$. The attack strategy defined in Eq. (2) and Eq. (3) are known as untargeted attack and targeted attack, respectively. A valid text perturbation needs to satisfy lexical, grammatical, and semantic constraints. As our attack method makes no character modifications, the lexical constraint is naturally retained. Additionally, we propose a bigram substitution strategy to avoid meaningless outputs, and introduce an adaptive search algorithm MHS to minimize the number of word perturbations while preserving the semantic similarity and syntactic coherence.

### 3.2 Replacement Candidate Selection

HowNet annotates words by their sememes, where the sememe is a minimum unit of semantic meaning in linguistics. For example, the word "apple" has multiple sememes, e.g., "fruit", "computer", etc. Words sharing the same sememe tag can be interchangeable in crafting adversarial examples.

**Candidate set creation.** Suppose the input sentence contains $n$ words, i.e., $\mathbf{X} = \{w_1, w_2, \cdots, w_n\}$. For each word $w_i$, we first connect it to its next word $w_{i+1}$ and check if the bigram $(w_i, w_{i+1})$ has synonyms in synonym space $\mathbb{W}$. If yes, we collect all the synonyms to create the bigram candidates set $\mathbb{B}_i$ and skip searching candidates for $w_i$ and $w_{i+1}$ separately[1]. Otherwise, we gather all the candidate words for $w_i$ from the synonym space $\mathbb{W}$ and the sememe space $\mathbb{H}$ and denote them as a subset $\mathbb{S}_i \subset \mathbb{W} \cup \mathbb{H}$. It is worth mentioning that we pose a candidate filter here to make sure all the candidate words in $\mathbb{S}_i$ have the same part-of-speech (POS) tags with $w_i$. Replacing words with the same POS tags (e.g., nouns) can help avoid imposing grammatical errors.

If $w_i$ is a named entity (NE), we enlarge the $\mathbb{S}_i$ by absorbing more same-type NE words. The NE refers to a predefined real-world object that can be symbolized by a proper noun, such as person names, organizations, and locations (Nouvel, Ehrmann, and Rosset 2016). The candidate NE (denoted as $NE_{\text{CAND}}$) must has the same NE type with the original word. It is selected as the most frequently appeared word from the complementary NE set $\mathbb{W} - \mathbb{W}_{\mathbf{Y}_{true}}$, where $\mathbb{W}_{\mathbf{Y}_{true}}$ contains all the NEs of the $\mathbf{Y}_{true}$ class. Then we update the synonym set as $\mathbb{S}_i \leftarrow \mathbb{S}_i \cup NE_{\text{CAND}}$.

Considering polysemy, a word may have more than one sememes defined in HowNet. To guarantee valid substitutions, we take only words that have at least one common sememes with the original word $w_i$ into its candidate set $\mathbb{S}_i$.

**Best candidate selection.** Given the candidate set $\mathbb{S}_i$ (or $\mathbb{B}_i$)[2], every $w_i' \in \mathbb{S}_i$ is a potential candidate for the replacement of word $w_i$. We define the candidate importance score $I_{w_i'}$ for each substitution candidate $w_i'$ as the reduction of prediction probability:

$$I_{w_i'} = P(\mathbf{Y}_{true}|\mathbf{X}) - P(\mathbf{Y}_{true}|\mathbf{X}_i'), \forall w_i' \in \mathbb{S}_i \tag{4}$$

where

$$\mathbf{X} = w_1, w_2, \cdots, w_i, \cdots, w_n \tag{5}$$

$$\mathbf{X}_i' = w_1, w_2, \cdots, w_i', \cdots, w_n \tag{6}$$

Then we pick the word $w_i'$ that achieves the highest $I_{w_i'}$ to be the best substitution word $w_i^*$. Formally, the synonym candidate selection function is given as below

$$w_i^* = R(w_i, \mathbb{S}_i) = \arg\max_{w_i' \in \mathbb{S}_i} I_{w_i'} \tag{7}$$

Repeating this procedure on every word one by one solves the first key issue of our method, as is summarized in Algorithm 1 from line 1 to line 11.

### 3.3 Adaptive Priority Determination

The word priority determination is designed to sort the word replacement order. Given the best substitution word $w_i^*$ for the original $w_i$, we obtain $n$ adversarial examples $\{\mathbf{X}_1^*, \cdots, \mathbf{X}_n^*\}$ with each being modified on one word, i.e., $\mathbf{X}_i^* = \{w_1, \cdots, w_i^*, \cdots, w_n\}$. The change of true label probability between $\mathbf{X}$ and $\mathbf{X}_i^*$ denotes the attack effect that can be achieved by modifying $w_i$ to $w_i^*$:

$$\Delta P_i^* = P(\mathbf{Y}_{true}|\mathbf{X}) - P(\mathbf{Y}_{true}|\mathbf{X}_i^*) \tag{8}$$

A straightforward way of determining the word replacement priority is to sort the words by their $\Delta P_i^*$ descent order and select the top-k ones. However, we empirically find that replacing words in such a static order incrementally always leads to local optima and word over substitution. This means simply selecting top-k words using $\Delta P_i^*$ does not necessarily provide the best word combination in misleading DNNs.

In this paper, we propose the Monotonic Heuristic Search (MHS) method, which adaptively determines the word substitution priority. We first create the initial generation $\mathbb{G}^0$ as an empty set (line 12 of Algorithm 1). Then we set the maximum number of words that can be modified, i.e.,

[1] This means bigram substitution takes precedence.

[2] In the rest of the paper, we slightly abuse the notation by using $\mathbb{S}_i$ to denote the substitution candidate for $w_i$. If $w_i$ belongs to a bigram $(w_i, w_{i+1})$, then $\mathbb{S}_i$ is equivalent to $\mathbb{B}_i$.

**Algorithm 1:** The proposed BU-MHS Attack Algorithm

**Input:** Sample text with $n$ words $\mathbf{X} = (w_1, \cdots, w_n)$
**Input:** Maximum word replacement bond $M$
**Input:** Classifier $F$
**Output:** Adversarial example $\mathbf{X}_{adv}$
/* Select candidates for input words */
1 **for** $i = 1$ **to** $n$ **do**
2  Connect $w_i$ with its next word as $(w_i, w_{i+1})$;
3  Collect bigram candidate set $\mathbb{B}_i$ for $(w_i, w_{i+1})$ from synonym space $\mathbb{W}$;
4  **if** $\mathbb{B}_i \neq \varnothing$ **then**
5  Find the best bigram candidate from $\mathbb{B}_i$;
6  $i \mathrel{+}= 1$ ; ▷ skip attacking $w_{i+1}$
7  **else**
8  Get a synonym-sememe candidate set $\mathbb{S}_i$ for $w_i$ from $\mathbb{W} \cup \mathbb{H}$;
9  **if** $w_i$ *is a NE* **then**
10  $\mathbb{S}_i \leftarrow \mathbb{S}_i \cup NE_{\text{CAND}}$;
11  Find the best unigram candidate from $\mathbb{S}_i$;
12 Create the initial generation with empty $\mathbb{G}^0 = \varnothing$;
13 Set the upper bound $M = \min(M, n)$;
/* The MHS search starts */
14 **for** $m = 1$ **to** $M$ **do**
15  $\mathbb{G}^m = \mathcal{F}\left(\mathbb{G}^{m-1}, \{w_1^*, \cdots, w_n^*\}\right)$;
16  **for** $Candidate \subset \mathbb{G}^m$ **do**
17  Replace $Candidate$ words in $\mathbf{X}$ to craft $\mathbf{X}_{adv}$;
18  **if** $F(\mathbf{X}) \neq F(\mathbf{X}_{adv})$ **then**
19  break ; ▷ successful attack
20  **else**
21  $\Delta P_{adv} = P(\mathbf{Y}_{true}|\mathbf{X}) - P(\mathbf{Y}_{true}|\mathbf{X}_{adv})$;
22 **return** $\mathbf{X}_{adv}$

$M = \min(M, n)$. This threshold forces us to stop the loop if the input example does not admit an adversarial alteration after $M$ times of substitution. The MHS procedure is listed between lines 14-21 of Algorithm 1. For each generation, we first create the population set for the current generation $\mathbb{G}^m$ using the $\mathcal{F}$ function defined in Algorithm 2. Specifically, the $\mathcal{F}$ directly returns all the best substitution synonyms $\{w_1^*, \cdots, w_n^*\}$ as the first generation. Then we iteratively query the classifier $F$ whether its prediction is changed by replacing the first generation candidates. If a population member $\mathbf{X}_{adv}$ achieves a successful attack, the optimization completes and returns the $\mathbf{X}_{adv}$. Otherwise, we calculate the probability shift of $\Delta P_{adv}$ in line 21. If we can not find a successful attack from the first generation, then we need to run the second iteration with the help of $\Delta P_{adv}$.

In the next generation, we recall the $\mathcal{F}$ to construct $\mathbb{G}^m$ with three steps, as listed in lines 5-8 of Algorithm 2. Firstly, we search the most effective element from the previous generation $\mathbb{G}^{m-1}$ that attains the maximal $\Delta P_{adv}$. We denote this best element as $\mathbb{G}_{best}^{m-1}$. Then we wipe out all the candidate words belonging to $\mathbb{G}_{best}^{m-1}$ from the full candidates set $\{w_1^*, \cdots, w_n^*\}$, resulting in $s$ $(1 \leq s \leq n)$ elements left. Finally, we combine the $\mathbb{G}_{best}^{m-1}$ with every remaining candidate $w_i^*$ and assign it to the current population member $\mathbb{G}^m(i)$. The greedy search between lines 16-21 of Algorithm 1 is the same as the first generation but replaces one more word/bigram in every next generation to craft $\mathbf{X}_{adv}$. This procedure does not stop until it successfully finds the adversarial example or reaches the upper threshold of $M$. The heuristic optimization method enables us to preserve the best population member from the previous generation and adaptively determine which word should be altered in the current generation. Based on the MHS, we achieve a higher successful attack rate by replacing a fewer number of words compared with static baselines. This solves the second issue.

**Algorithm 2:** Function of Generation Creation ($\mathcal{F}$)

**Input:** Last generation combinations $\mathbb{G}^{m-1}$
**Input:** The best substitution words $\{w_1^*, \cdots, w_n^*\}$
**Output:** Current generation combinations $\mathbb{G}^m$
1 Initialize current generation $\mathbb{G}^m = \varnothing$;
2 **if** $\mathbb{G}^{m-1} = \varnothing$ **then**
3  $\mathbb{G}^m = \{w_1^*, \cdots, w_n^*\}$ ; ▷ 1st generation
4 **else**
5  Search for the most effective element $\mathbb{G}_{best}^{m-1}$ from $\mathbb{G}^{m-1}$ that achieves the highest $\Delta P_{adv}$;
6  Remove all words of $\mathbb{G}_{best}^{m-1}$ from $\{w_1^* \cdots w_n^*\}$, resulting in $s$ $(1 \leq s \leq n)$ elements left;
7  **for** $i = 1$ **to** $s$ **do**
8  Create new generation element $\mathbb{G}^m(i)$ by combing $\mathbb{G}_{best}^{m-1}$ with $i^{th}$ remaining element $w_i^*$;
9 **return** $\mathbb{G}^m$

# 4 Experiments

We evaluate the effectiveness of our BU-MHS method on widely used text datasets. We provide code and data with a fully anonymous link[3] to ensure reproducibility.

## 4.1 Datasets and Victim Models

**Datasets** We conduct experiments on three publicly available benchmarks. IMDB (Maas et al. 2011) is a binary sentiment classification dataset containing 50,000 movie reviews. AG's News (Zhang, Zhao, and LeCun 2015) is a news classification dataset with 127600 samples belonging to 4 topic classes. Yahoo! Answers (Zhang, Zhao, and LeCun 2015) is a ten-class topic dataset with 1,400,000 train samples and 60,000 test samples. The average text length (without punctuation) of IMDB (227 words) is much longer than AG's News (38 words) and Yahoo! Answers (32 words).

**Victim models** We apply our attack algorithm on four popular victim models. Word CNN (Kim 2014) is stacked by a word embedding layer with 50 embedding dimensions, a convolutional layer with 250 filters, and each kernel size of 3. Character-based CNN (Char CNN) (Zhang, Zhao, and LeCun 2015) is composed of a 69-dimensional character embedding layer, 6 convolutional layers, and 3 densely-connected layers. Word LSTM passes the input sequence through a 100-dimension embedding layer, concatenating a 128-units long short-term memory layer, and following a

[3] https://github.com/AdvAttack/TextAttack

| Dataset | Model | Test Accuracy |
|---|---|---|
| IMDB | Word CNN | 87.97% |
| | Bi-LSTM | 85.71% |
| AG's News | Word CNN | 90.75% |
| | Char CNN | 89.24% |
| | Word LSTM | 91.62% |
| Yahoo! Answers | Word CNN | 71.21% |
| | Bi-LSTM | 71.60% |

Table 2: Test accuracy of four DNNs models before attacks.

dropout of 0.5. Bidirectional LSTM (Bi-LSTM) consists of a 128-dimension word embedding layer, a bidirectional layer that wraps 64 LSTM units, a dropout of 0.3, and a fully connected layer for classification. Table 2 lists the classification accuracy on the original legitimate test samples.

### 4.2 Baselines

We compare our method with representative black-box word-level attack algorithms as listed below.

- RAND attack randomly selects a synonym from WordNet and ranks the attack order by our MHS algorithm.
- Word saliency attack (WSA) (Li, Monroe, and Jurafsky 2016) gets replacement words from WordNet and rephrases texts in the word saliency (WS) descending order. The word saliency is similar to Eq. (4) but replaces $w_i$ with $unknown$.
- PWWS (Ren et al. 2019) chooses candidate words from WordNet and sorts word attack order by multiplying the word saliency and probability variation.
- PSO (Zang et al. 2020) selects word candidates from HowNet and employs the PSO to find adversarial text. This method treats every sample as a particle where its location in the search space needs to be optimized.
- TextFooler (TEFO) (Jin et al. 2020) obtains synonyms from Glove space and defines the WIS by iteratively deleting input words and calculating the DNNs score changes.
- BERT-ATTACK (BEAT) (Li et al. 2020) takes advantage of BERT MLM to generate candidates and attack words by the static WIS descending order. The WIS is similar to Eq. (4) but changes $w_i$ to [MASK].

### 4.3 Evaluation Metrics and Experiment Settings

**Evaluation Metrics.** We use two metrics to evaluate the text attack performance, i.e., successful attack rate (SAR) and the average number of word substitutions. The SAR is defined as the misclassification rate of the classifier $F$ after it is attacked. To quantify the perturbation cost, we count the number of word substitutions in each sample and compute the average number for each dataset to denote the adversarial attack cost. Intuitively, a rational hacker targets to achieve a high SAR by substituting a small number of words.

**Experimental Settings.** We train all the DNNs models using the ADAM optimizer (Kingma and Ba 2015), where parameters are set as: learning rate $= 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$. We deploy BU-MHS and the first three baselines on Keras. The PSO, TEFO, and BEAT are tested on the TextAttack framework (Morris et al. 2020), where the Char CNN model and the Yahoo! Answers are currently unavailable. We set the upper bound of word replacement number as $M = 20$ and use the recommended parameters for all baselines. Their attack performance is assessed on 1000 test samples of each dataset as the conventional setting (Zang et al. 2020; Jin et al. 2020). For our BU-MHS, the synonym candidates and sememe candidates are picked from WordNet and HowNet, respectively.

### 4.4 Experimental Results and Analysis

The experimental results of SAR and the average word replacement number are listed in Table 3 and Table 4, respectively. We manifest the three contributions mentioned in the Introduction by asking three research questions:

**Q1: Is our adaptive MHS superior to static baselines?** To validate this, we design the U-MHS that searches substitution words from *only* WordNet and attacks text *only* at the unigram word level - the same as WSA and PWWS, but employs our MHS to determine the word substitution priority. Experimental results in Table 3 and Table 4 show that U-MHS achieves higher SAR and changes a much smaller number of words comparing with static counterparts (WSA, and PWWS). Besides, the RAND delivers higher SAR than WSA on IMDB and AG's News. This also illustrates the merit of our adaptive MHS.

**Q2: Is the hybrid of synonym and sememe beneficial?** We present a hybrid version of U-MHS, i.e., HU-MHS, which is all the same with U-MHS but integrates HowNet to search synonym-sememe candidates. Table 3 shows that HU-MHS accomplishes the highest SAR in all cases and outperforms U-MHS by a large margin. Intriguingly, Table 4 exhibits that HU-MHS achieves such high SAR by using the fewest word substitutions. This strongly suggests the profit of incorporating HowNet in the candidate selection step.

**Q3: What's the advantage of combining the bigram attack?** The bigram substitution is vitally significant in improving semantic smoothness and generating meaningful sentences. To show this, we list three adversarial examples from IMDB (Table 6), AG's News (Table 7) and Yahoo! Answers (Table 8). We can see from the adversarial examples that our bigram substitution can greatly reduce the semantic variations. For example, in Table 7, our method replaces one bigram (Olympic Games $\rightarrow$ Olympiad) but causes less semantic variation than HU-MHS changing two unigrams.

Overall, Table 3 and Table 4 elaborate that the HU-MHS, BU-MHS, and U-MHS almost swept the top-3 results on all datasets, indicating the superiority of our method.

**Platform and Efficiency Analysis.** We conduct all experiments on Enterprise Linux Workstation 7.7 with 2.7GHz CPU frequency and 176GB memory. Table 5 lists the time consuming of various methods on AG's News dataset. Table 5 shows that our BU-MHS is more efficient than the dynamic PSO but costs more time than static counterparts.

| Dataset | Model | $M = \frac{n}{2}$ | $M = n$ | | | $M = 20$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BEAT | PSO | TEFO | PWWS | WSA | PWWS | RAND | U-MHS | HU-MHS | BU-MHS |
| IMDB | Word CNN | 91.02% | **100%** | **100%** | *94.60%* | 38.85% | 90.56% | 81.13% | 95.58% | **100%** | **100%** |
| | Bi-LSTM | 90.15% | **100%** | **100%** | 99.76% | 81.70% | 97.09% | 94.79% | *98.90%* | **100%** | **100%** |
| AG's News | Word CNN | *86.91%* | 85.48% | 82.75% | 85.48% | 77.98% | 81.98% | 82.42% | 85.76% | **92.77%** | 91.77% |
| | Word LSTM | 77.52% | 79.87% | *85.17%* | 79.87% | 74.73% | 76.81% | 79.76% | 81.07% | **88.40%** | 86.21% |
| | Char CNN | - | - | - | 75.48% | 68.33% | 75.25% | 75.26% | *80.70%* | **92.17%** | 91.49% |
| Yahoo! Answers | Word CNN | - | - | - | 68.39% | 66.82% | 68.23% | 51.96% | *70.27%* | **90.14%** | 88.58% |
| | Bi-LSTM | - | - | - | 67.76% | 66.20% | 67.29% | 51.64% | *67.92%* | **87.95%** | 85.60% |

Table 3: The successful attack rate (SAR) of various attack algorithms. For each row, the highest SAR is highlighted in bold, the second highest SAR is highlighted in underline, and the third highest SAR is denoted with italic.

| Dataset | Model | $M = \frac{n}{2}$ | $M = n$ | | | $M = 20$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | BEAT | PSO | TEFO | PWWS | WSA | PWWS | RAND | U-MHS | HU-MHS | BU-MHS |
| IMDB | Word CNN | 7.5 | *3.42* | 8.01 | 8.625 | 16.15 | 5.87 | 7.31 | 4.5 | **2.07** | 2.11 |
| | Bi-LSTM | 7.89 | 5.22 | 8.13 | 5.53 | 10.55 | 5.16 | 5.67 | *4.238* | **2.28** | 2.33 |
| AG's News | Word CNN | 5.97 | 5.02 | 7.42 | 6.29 | 9.51 | 6.03 | 5.23 | *4.815* | **4.16** | 4.32 |
| | Word LSTM | 6.24 | 5.82 | 8.49 | 8.05 | 10.12 | 7.72 | 6.19 | *5.97* | **5.74** | 5.99 |
| | Char CNN | - | - | - | 5.59 | 8.23 | 5.44 | 5.06 | *4.389* | **3.41** | 3.52 |
| Yahoo! Answers | Word CNN | - | - | - | 3.34 | 3.95 | 3.15 | 3.65 | *2.78* | **1.85** | 1.89 |
| | Bi-LSTM | - | - | - | 3.47 | 3.83 | 3.2 | 3.74 | *2.98* | **2.28** | 2.4 |

Table 4: The average number of word substitutions of various attack methods. For each row, the smallest word substitution number is highlighted in bold, the second smallest is denoted in underline, and the third smallest is represented with italic.

| BEAT | PSO | TEFO | PWWS | WSA | RAND | BU-MHS |
|---|---|---|---|---|---|---|
| 1.94 | 6.52 | 0.14 | 1.05 | 0.79 | 2.97 | 2.87 |
| 1.05 | 10.4 | 0.17 | 2.62 | 2.7 | 5.6 | 8.45 |

Table 5: The top and bottom row shows the runtime (hours) by attacking Word CNN and Word LSTM, respectively.

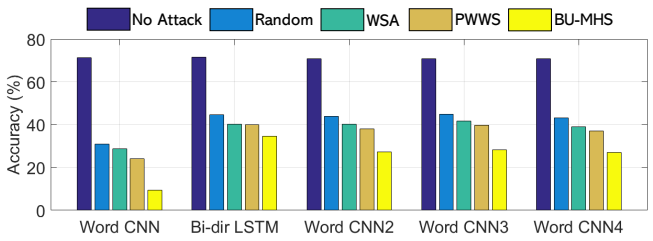


Figure 2: Transfer attack on Yahoo! Answers. Lower accuracy indicates higher transfer ability (the lower the better).

### 4.5 Transferability

The transferability of adversarial examples implies whether the adversarial samples generated to mislead a specific model $F$ can mislead other models $F'$. To evaluate the transferability, we construct three more CNN models named Word CNN2, Word CNN3, and Word CNN4. Different from the previous Word CNN model, Word CNN2 has one more fully connected layer, Word CNN3 replaces the Relu nonlinear function with Tanh, and Word CNN4 adds one convolutional layer. We apply the adversarial examples generated on Word CNN to attack these three new models and the LSTM model. Figure 2 shows the results on the original Word CNN and transferred models. It can be seen from Figure 2 that our method attains the best transfer attack performance, elaborating the strength of our method in transfer attack.

### 4.6 Adversarial Retraining

Adversarial retraining is an effective way to improve the model's robustness by joining the adversarial examples to the training set. In this experiment, we randomly generate and append {500, 1000, 1500, 2000} AG's New adversarial samples to its training set and retrain the Word CNN model. Figure 3 shows the five-run mean accuracy of Word CNN on the clean test set after adversarial training. From Figure 3 we know that our method generates more effective adversarial samples than PWWS in improving the model robustness. We also evaluate if the retrained model is immune to adversarial attacks by attacking the retrained model. Results in Table 9 show that retrained victim models can defend against the attacks to a certain degree. Additionally, our BU-MHS brings higher SAR than PWWS after retraining, indicating the BU-MHS is harder to defend by adversarial retraining.

### 4.7 Targeted Attack Evaluations

Algorithm 1 can be easily adapted to make targeted attack with slight modifications, e.g., change line 18 from

| | |
|---|---|
| PWWS (Successful attack. True label score = 45.74%) | I think that the movie was really ~~*good*~~ **dear**. Subject, acting and ~~*Nusrat*~~ **BAD** ~~*Fateh*~~ **BAD** ALi Khan's music were ~~*marvellous*~~ **tall**. Although the director has succeeded in showing the status of women in rural areas and how they suffer at the hands of male-dominated culture, he has neglected ~~*Phoolan*~~ **BAD**'s character a ~~*bit*~~ **piece** and has focussed more on the violence faced by her. |
| HU-MHS (Successful attack. True label score = 2.69%) | I think that the movie was ~~*really*~~ **presumably** good. Subject, acting and Nusrat Fateh ALi Khan's music were marvellous. Although the director has succeeded in showing the status of women in ~~*rural*~~ **rustic** areas and how they suffer at the hands of male-dominated culture, he has neglected Phoolan's character a bit and has focussed more on the violence faced by her. |
| BU-MHS (Successful attack. True label score = 41.69%) | I think that the movie was really good. Subject, acting and Nusrat Fateh ALi Khan's music were marvellous. Although the director has succeeded in showing the status of women in ~~*rural areas*~~ **country** and how they suffer at the hands of male-dominated culture, he has neglected Phoolan's character a bit and has focussed more on the violence faced by her. |

Table 6: Adversarial examples of IMDB (attack Word CNN). Italic texts are attacked words, while bold ones are substitutions.

| | |
|---|---|
| PWWS (Successful attack. True label score = 37.17%) | Afghan women ~~*make*~~ **arrive** brief Olympic ~~*debut*~~ **introduction**. Afghan women made a short-lived debut in the Olympic Games on Wednesday as 18-year-old judo wildcard Friba Razayee was defeated after 45 seconds of her first ~~*match*~~ **peer** in the under-70kg middleweight. |
| HU-MHS (Successful attack. True label score = 41.13%) | Afghan women make brief Olympic ~~*debut*~~ **introduction**. Afghan women made a short-lived debut in the Olympic Games on Wednesday as 18-year-old judo wildcard Friba Razayee was defeated after 45 seconds of her first ~~*match*~~ **supply** in the under-70kg middleweight. |
| BU-MHS (Successful attack. True label score = 40.02%) | Afghan women make brief Olympic debut. Afghan women made a short-lived debut in the ~~*Olympic Games*~~ **Olympiad** on Wednesday as 18-year-old judo wildcard Friba Razayee was defeated after 45 seconds of her first match in the under-70kg middleweight. |

Table 7: Adversarial examples by attacking Word LSTM model on AG's News dataset.

| | |
|---|---|
| PWWS (Failure. True label score = 75.85%) | How do I ~~*become*~~ **go** a baseball ~~*player*~~ **actor**? |
| HU-MHS (Successful attack. True label score = 31.79%) | How do I become a ~~*baseball*~~ **cubicle** player? |
| BU-MHS (Successful attack. True label score = 4.95%) | How do I become a ~~*baseball player*~~ **ballplayer**? |

Table 8: Adversarial examples by attacking Bi-LSTM model on Yahoo! Answers dataset.

| Methods | Before retraining | | After retraining | |
|---|---|---|---|---|
| | SAR | # words | SAR | # words |
| PWWS | 81.98% | 6.03 | 73.42% | 8.165 |
| BU-MHS | 91.77% | 4.32 | 77.36% | 7.66 |

Table 9: Attack the retrained Word CNN on AG's News. "# words" denotes the average number of word substitutions.

$F(\mathbf{X}) \neq F(\mathbf{X}_{adv})$ to $F(\mathbf{X}_{adv}) = \mathbf{Y}_{target}$. The targeted attack experiments are conducted on AG's News dataset, and the four target labels are: 0 (World), 1 (Sports), 2 (Business) and 3 (Sci/Tech). We found similar results on four labels but just report the results when $\mathbf{Y}_{target} = 0$ due to the space limitation. The results shown in Table 10 indicate that our BU-MHS attains a much higher SAR and replaces less words than PWWS for all victim models. This means our method is powerful for both targeted and untargeted attack.

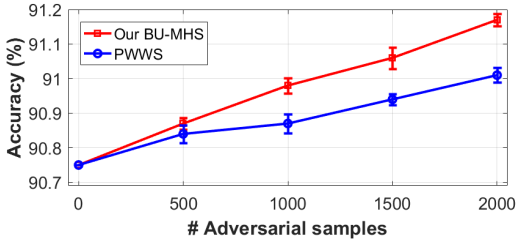


Figure 3: Adversarial retraining results. The higher the accuracy, the more robust of the model after retraining.

| Model | SAR | | # words replace | |
|---|---|---|---|---|
| | PWWS | BU-MHS | PWWS | BU-MHS |
| Word CNN | 87.21% | **94.67%** | 4.78 | **3.69** |
| Char CNN | 32.61% | **65.07%** | 11.56 | **9.08** |
| Word LSTM | 77.92% | **87.87%** | 7.09 | **5.68** |

Table 10: Targeted attack results on AG's News dataset.

## 5 Conclusions

In this paper, we have proposed a novel BU-MHS algorithm for crafting natural language adversarial samples. The BU-MHS exploits unigram and bigram modifications to avoid the semantic errors and employs adaptive MHS to reduce attack cost. The hybrid synonym-sememe approach provides more candidate options. Future research directions include designing defense methods via an $n$-gram strategy.

## Acknowledgments

Weifeng Liu was supported in part by the National Natural Science Foundation of China (Grant No.61671480), in part by the Open Project Program of the National Laboratory of Pattern Recognition (NLPR) (Grant No.202000009), and in part by the Major Scientific and Technological Projects of CNPC (Grant No.ZD2019-183-008). Dacheng Tao was supported by Australian Research Council Projects FL-170100117, DP-180103424, and IH-180100002.

## References

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

Atallah, M. J.; McDonough, C. J.; Raskin, V.; and Nirenburg, S. 2001. Natural language processing for information assurance and security: an overview and implementations. In *Proceedings of the 2000 workshop on New security paradigms*, 51–65.

Belinkov, Y.; and Bisk, Y. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Dong, Z.; and Dong, Q. 2006. *Hownet and the computation of meaning*. World Scientific.

Ebrahimi, J.; Rao, A.; Lowd, D.; and Dou, D. 2018. Hotflip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 31–36.

Gan, W. C.; and Ng, H. T. 2019. Improving the Robustness of Question Answering Systems to Question Paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 6065–6075.

Garg, S.; and Ramakrishnan, G. 2020. BAE: BERT-based Adversarial Examples for Text Classification. *arXiv preprint arXiv:2004.01970* .

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*.

Jia, R.; and Liang, P. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2021–2031.

Jin, D.; Jin, Z.; Tianyi Zhou, J.; and Szolovits, P. 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Kim, Y. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 1746–1751.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Li, J.; Monroe, W.; and Jurafsky, D. 2016. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220* .

Li, L.; Ma, R.; Guo, Q.; Xue, X.; and Qiu, X. 2020. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984* .

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 142–150.

Miller, G. A. 1998. *WordNet: An electronic lexical database*. MIT press.

Morris, J. X.; Lifland, E.; Yoo, J. Y.; and Qi, Y. 2020. TextAttack: A framework for adversarial attacks in natural language processing. *arXiv preprint arXiv:2005.05909* .

Mrkšić, N.; Séaghdha, D. O.; Thomson, B.; Gašić, M.; Rojas-Barahona, L.; Su, P.-H.; Vandyke, D.; Wen, T.-H.; and Young, S. 2016. Counter-fitting word vectors to linguistic constraints. In *The 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 142–148.

Nouvel, D.; Ehrmann, M.; and Rosset, S. 2016. *Named entities for computational linguistics*. Wiley Online Library.

Papernot, N.; McDaniel, P.; Swami, A.; and Harang, R. 2016. Crafting adversarial input sequences for recurrent neural networks. In *IEEE Military Communications Conference*, 49–54.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 1085–1097.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Wallace, E.; Feng, S.; Kandpal, N.; Gardner, M.; and Singh, S. 2019a. Universal trigger sequences for attacking and analyzing NLP. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2153–2162.

Wallace, E.; Rodriguez, P.; Feng, S.; Yamada, I.; and Boyd-Graber, J. 2019b. Trick me if you can: Human-in-the-loop generation of adversarial examples for question answering. *Transactions of the Association for Computational Linguistics* 7: 387–401.

Wang, W.; Wang, L.; Wang, R.; Wang, Z.; and Ye, A. 2019. Towards a Robust Deep Neural Network in Texts: A Survey. *arXiv preprint arXiv:1902.07285* .

Wang, X.; Jin, H.; and He, K. 2019. Natural language adversarial attacks and defenses in word level. *arXiv preprint arXiv:1909.06723* .

Zang, Y.; Qi, F.; Yang, C.; Liu, Z.; Zhang, M.; Liu, Q.; and Sun, M. 2020. Word-level Textual Adversarial Attacking

as Combinatorial Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 6066–6080.

Zhang, W. E.; Sheng, Q. Z.; Alhazmi, A. A. F.; and Li, C. 2019. Generating textual adversarial examples for deep learning models: A survey. *CoRR, abs/1901.06796* .

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.