LBA: Textual Hard-Label Adversarial Attack Under Low Query Budgets

Shixin Guo[1] , **Ming Zhong**[1] , **Xuhong Zhang**[2] , **Dandan Zhao**[1] , **Zhe Wang**[1] , **Bo Zhang**[3] , **Shouling Ji**[2] , **Hao Peng**[1*]
[1]Zhejiang Normal University
[2]Zhejiang University
[3]China Electric Power Research Institute
{spacebound, zhongming, ddzhao, wangzhe97, hpeng}@zjnu.edu.cn, {zhangxuhong, sji}@zju.edu.cn, bobozhangiit@gmail.com

Abstract

Generating high-quality adversarial texts with low query budgets remains a challenging problem in the hard-label scenario. Most existing approaches rely on greedy algorithms, where one position in the text is selected for substitution, followed by the substitutions of other positions. This local search approach may fail to discover high-quality adversarial examples and often leads to excessive query costs. Ideally, an optimal adversarial sample would consider all possible position combinations in the text, but exhaustive search is computationally impractical. To address this challenge, we propose a sampling-based method called LBA, which constructs an approximate distribution of high-quality adversarial examples by integrating both prior and posterior knowledge, and utilizes this distribution for sampling. As sampling progresses, posterior knowledge updates the approximate distribution, which in turn guides more effective sampling. Extensive experiments on six language models, ranging from small-scale to large-scale architectures across four datasets, demonstrate that LBA significantly outperforms state-of-the-art baselines on all evaluation metrics. Additionally, LLM-based assessment indicates that LBA generates more semantically preserved and comprehensible adversarial texts.

1 Introduction

Deep neural networks (DNNs) have achieved remarkable success and are widely applied to natural language processing (NLP) tasks[Brown *et al.*, 2020; Hu *et al.*, 2022; Zhang *et al.*, 2023]. However, as DNNs become increasingly popular and widely applied, potential security problems have also raised significant concerns. It has been observed that DNNs are easily deceived into producing wrong predictions by well-designed malicious examples, known as adversarial attacks[Goodfellow *et al.*, 2015; Szegedy *et al.*, 2014]. To reveal the vulnerabilities and improve the robustness of neural language models, researchers have focused on generating plausible adversarial examples. In recent years, there has been a surge in textual adversarial attacks[Alzantot *et al.*, 2018; Linyang *et al.*, 2020; Zang *et al.*, 2020], further intensifying the pursuit of crafting high-quality adversarial texts.

Based on the outputs the adversary obtained from victim models, existing works on black-box textual adversarial attacks can be categorized into soft-label attacks and hard-label attacks. Soft-label attacks[Li *et al.*, 2020; Jin *et al.*, 2020; Zang *et al.*, 2020] typically calculate the importance of words using the confidence scores output by target models, and then modify words based on the importance ranking until the predicted label is reversed. Since most real-world APIs do not provide confidence scores, hard-label attacks that only require the top-1 label are more practical and challenging. The prevalent hard-label attacks usually adopt one of the following algorithms: (1) heuristic algorithms[Maheshwary *et al.*, 2021a], (2) gradient estimation algorithms[Ye *et al.*, 2022; Ye *et al.*, 2023], or (3) direction estimation algorithms[Liu *et al.*, 2024a] to optimize an initialized adversarial example. In real-world applications, query budgets tend to be limited by monetary costs, and a massive number of similar inputs also increases the risk of exposing the adversary. Therefore, generating high-quality adversarial texts under low query budgets has become a critical concern for hard-label attacks. Although existing methods strive to improve query efficiency, they still struggle with the trade-off between query efficiency and quality. To address this issue, we reveal the limitations of existing hard-label attacks and propose a novel approach.

Existing methods essentially follow a greedy search paradigm, leading to query inefficiency. Their exploration strategy iteratively refines adversarial examples, where each iteration improves upon the result of the previous one to improve quality. To accomplish this, greedy search selects a word position for substitution and locks in the change, then proceeds to the next candidate position without revisiting previous decisions. However, generating high-quality adversarial examples depends on selecting critical positions and appropriate word substitutions for these positions, which involves exploring substitutions for all combinations of perturbable positions. During greedy search, the search space of perturbable position combinations gradually shrinks, blocking the discovery of certain feasible solutions. Ideally, an optimal adversarial sample would consider all possible position

*Corresponding author.

combinations in the text. However, this would introduce a greater challenge of combinatorial explosion.

To address this challenge, we propose a novel method named LBA, which performs sampling from an approximate distribution of high-quality adversarial examples. Sampling from a distribution that statistically abstracts the population and theoretically covers all the combinations offers an effective solution for mitigating the explosion of the search space, compared to directly selecting from the population itself. However, it is not trivial to accurately characterize the distribution of high-quality adversarial examples. To overcome this challenge, we draw inspiration from Metropolis-Hastings (MH) sampling[Chib and Greenberg, 1995] to construct an approximating distribution (i.e., the target function) and sample from it, aiming to obtain samples conforming to the desired distribution. For effective approximation, we leverage both prior and posterior knowledge to shape the target function. The prior is derived from predefined quality constraints (e.g., perturbation rate and semantic similarity). During each attack, the posterior is learned from samples accumulated over previous sampling iterations and is used to iteratively update LBA's understanding of aggressiveness, uncovering which positional changes most strongly induce label inversion. Specifically, we quantify aggressiveness by the importance of each position, measured as its empirical label-flip frequency from adversarial examples discovered in previous iterations. By leveraging this target function, LBA progressively refines its approximation of the desired distribution through iterative sampling, and is self-correcting even when initial priors are too coarse to accurately characterize the distribution of high-quality adversarial examples.

The sampling process is an iterative cycle of generating a candidate sample and evaluating it for acceptance or rejection. At each iteration, we first generate a candidate by replacing a randomly chosen word in the sample from the previous iteration with a synonym, following the principle of with-replacement sampling. Subsequently, we incorporate the target function and a transition proposal to probabilistically determine whether to accept and then query the candidate. As iterations progress, query-derived posterior knowledge refines the estimated importance of positions in the target function, leading to more efficient sampling. In turn, improved sampling yields more informative observations, creating a positive feedback loop. To sum up, our contributions are as follows:

- We reveal the limitations of existing hard-label textual adversarial attacks. Furthermore, to the best of our knowledge, we are the first to formulate query-efficient adversarial text generation as a distribution-based sampling problem.
- We propose LBA[1] (**L**ow-query **B**udget hard-label **A**ttack), a novel method for generating high-quality adversarial texts in the hard-label scenario.
- We conduct experiments on six language models, ranging from small-scale to large-scale architectures, including BERT, LLaMA 2, and GPT-4o, across four datasets. The results demonstrate that LBA significantly outperforms existing hard-label baselines.

[1]The code and supplementary material are available at https://github.com/SpeisBound/LBA

2 Related Work

2.1 Soft-Label Adversarial Attack

Soft-label adversarial attacks rely on the confidence scores of target models to craft adversarial examples. Typically, these methods [Li *et al.*, 2020; Jin *et al.*, 2020; Li *et al.*, 2021; Maheshwary *et al.*, 2021b; Garg and Ramakrishnan, 2020; Gao *et al.*, 2024] calculate the importance of words based on confidence scores and then perturb those words according to their importance ranking until adversarial examples are generated. For instance, Textfooler[Jin *et al.*, 2020] uses importance ranking to sequentially perform synonym substitutions. Additionally, Optimization-based methods[Alzantot *et al.*, 2018; Zang *et al.*, 2020] utilize combinatorial optimization techniques to craft adversarial texts. For instance, PSO[Zang *et al.*, 2020] treats adversarial text generation as a combinatorial optimization problem and adopts a particle swarm optimization algorithm to address the challenge.

2.2 Hard-Label Adversarial Attack

Existing hard-label attacks typically adopt heuristic algorithms [Maheshwary *et al.*, 2021a; Peng *et al.*, 2023], gradient estimation-based search [Ye *et al.*, 2022; Ye *et al.*, 2023; Liu *et al.*, 2024b], or direction estimation-guided search [Liu *et al.*, 2024a] to approach the decision boundary of the target model. HLA[Maheshwary *et al.*, 2021a] explores hard-label adversarial attacks in NLP by employing the genetic algorithm to optimize adversarial texts. However, it is easy to fall into the local optimum and requires numerous queries to maintain its population. TextHoaxer [Ye *et al.*, 2022] first formulates the hard-label adversarial attack as a gradient-based optimization problem in the word embedding space. It introduces a perturbation matrix to estimate the gradient at each position, thereby determining the fixed substitute word for each position. To mitigate inaccurate gradient estimation, LeapAttack [Ye *et al.*, 2023] implements an interchange between discrete substitutions and continuous vectors, thereby enhancing the effectiveness of gradient utilization in discrete text. To alleviate the issue of gradient estimation consuming numerous queries, HQA-Attack [Liu *et al.*, 2024a] evaluates position importance and sequentially predicts the perturbation direction for each position, representing each direction with a transition synonym to avoid querying the entire synonym set. However, the reliance on the greedy search paradigm limits the potential of existing methods to further enhance query efficiency. In contrast to the aforementioned textual hard-label methods, LBA adopts sampling-based strategy as its search paradigm to efficiently explore the search space.

3 Proposed Attack

3.1 Problem Formulation

In this paper, we focus on adversarial text generation under the black-box hard-label setting, where the adversary can

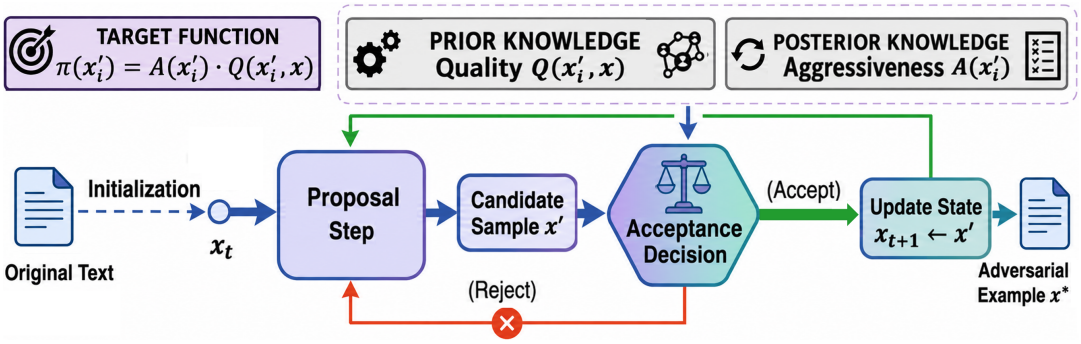


Figure 1: Pipeline of the sampling-based method LBA, which leverages both prior and posterior knowledge to shape the target function.

only access the top-1 predicted label from the target language model. The target model, denoted by a function f, accepts an input text $x = [w_1, w_2, ..., w_n]$ with n words and outputs a predicted label $f(x) = y$, which matches the corresponding ground-truth label. The goal of the attack is to craft a high-quality adversarial text x' by synonym substitution in x under a query budget of η. Specifically, x' is designed to mislead the victim model f into outputting an incorrect prediction result, i.e., $f(x) \neq f(x')$. The term "high-quality" denotes adversarial texts characterized by high similarity and fluency, which correspond to low perturbation rates, high semantic similarity, and fluent sentences. These attributes are crucial to aligning with the definition of adversarial texts, which requires them to be imperceptible to humans. Formally, we define the optimization problem as:

$$\arg\max_{x'} \; Q(x', x)$$
$$\text{s.t.} \quad \begin{cases} Q(x', x) \propto F(x') + S(x', x) \\ f(x) \neq f(x') \\ Qrs(x') \leq \eta \end{cases} \tag{1}$$

where $Q(x', x)$ denotes the quality of x', which is positively correlated with the fluency $F(x')$ and similarity $S(x', x)$ of the adversarial example. $Qrs(x')$ represents the number of queries required to craft x', and η is the given query budget.

3.2 Framework of LBA

The proposed approach is summarized in Algorithm 1, which first utilizes random initialization to generate an initial adversarial example x_0' and then optimizes it using the sampling-based strategy.

Initialization

To establish a starting point for sampling, we use random initialization, an approach commonly used in existing hard-label adversarial attacks[Maheshwary *et al.*, 2021a; Ye *et al.*, 2023; Ye *et al.*, 2022; Liu *et al.*, 2024a], to create an initial adversarial example. This process begins by randomly selecting a word w_j from the input text x and substituting it with a randomly chosen synonym from its synonym set $Syn(w_j)$. This substitution is repeated for other words in x until an example $x_0' = [w_1', w_2', ..., w_n']$ is generated, such that $f(x) \neq f(x_0')$. Before substituting a word, synonyms with mismatched part-of-speech (POS) tags are filtered out to improve grammatical consistency. To further improve the semantic similarity of the initial adversarial text, we follow previous attacks to substitute some original words back into x_0'. Under limited query budgets, fast random initialization is a practical choice. Moreover, compared to greedy methods that progressively shrink perturbable positions during iteration, ourmethod is not more sensitive to initialization.

Sampling-Based Strategy

As shown in Figure 1, we employ a sampling-based strategy inspired by MH sampling to explore the search space from the initial adversarial example. MH sampling enables drawing samples from an intractable distribution by constructing a tractable approximation. It provides theoretical convergence guarantees under ergodicity, a condition our adaptive target function naturally satisfies as statistics accumulate. The sampling is an iterative process consisting of the following steps:

- Propose a candidate sample by substituting the word at a randomly selected perturbable position in the current sample (initially set to x_0') with a synonym.
- Calculate the acceptance rate of the candidate sample based on the target function and transition proposal.
- Accept or reject the candidate: if rejected, retain the current sample; if accepted, query the victim model to validate aggressiveness and update the current sample accordingly.

In the following, we provide a detailed introduction to the three steps of sampling:

(1) Proposing a Candidate Sample. This step aims to generate a candidate sample from the search space composed of all perturbable positions. In existing methods, the search space of positions gradually shrinks, which can block the discovery of certain feasible solutions. To avoid this, we generate candidate samples using with-replacement modification over the perturbable positions. Let l_p denote the set of perturbable positions, i.e., the indices where the words differ between the initial adversarial example and its original counterpart. We generate the candidate sample x_i' by substituting the j-th word w_j in the current example $x^\star$ with a synonym s_j^k that is randomly picked from $Syn(w_j) = \{s_j^1, s_j^2, ..., s_j^r\}$. The index j is randomly picked from the set l_p. There are two reasons for selecting indices from l_p. As shown in previous work, if the perturbable positions are expanded to include all words, the search space becomes excessively large, making

Algorithm 1: LBA

Input : Original text $x = [w_1, w_2, ..., w_n]$, ground-truth label y, target model f.
Output: Adversarial example x'.

Generate an initial adversarial example $x'_0 = [w'_1, w'_2, ..., w'_n]$ by randomly substitution
Current example $x^\star \leftarrow x'_0$, optimal adversarial example $x' \leftarrow x'_0$, sampling counter $i \leftarrow 0$
while *Query budget is not exhausted* **do**
  $i \leftarrow i + 1$, select an index j form perturbable position indices l_p
  Propose a candidate sample x'_i by substituting the j-th word of $x^\star$ with a synonym s_j^k
  $\alpha(x'_i, x^\star) \leftarrow$ Calculate the acceptance rate of x'_i via Eq. (2)
  if *Uniform*$(0, \beta) < \alpha(x'_i, x^\star)$ **then**
    $y_i \leftarrow$ obtain the prediction result of x'_i
    if $y_i \neq y$ **then**
      Further improve the quality of $x^\star$
    Updating current example $x^\star$ and optimal adversarial example x'
Return x'

it difficult to achieve effective coverage. Furthermore, these positions, compared to unchanged words, have already been validated for effectiveness during the initialization process.

(2) Calculating the Acceptance Rate. To select samples that better align with the desired distribution and control the stochasticity of sampling transitions, we use the acceptance rate to determine whether a proposed sample should be accepted. The acceptance rate is calculated by comparing the relative probabilities of the candidate and the current sample with respect to the target function and transition proposal. Specifically, the target function is designed to approximate the distribution of high-quality adversarial examples. The accuracy of the distribution approximation directly influences the effectiveness of the obtained samples and plays a guiding role in the overall sampling process. The transition proposal is asymmetric, measured by the change in semantic similarity between samples, guiding the search toward semantically coherent adversarial examples. The acceptance rate of a proposed candidate x'_i, is defined as:

$$\alpha(x'_i, x^\star) = \min\left(1, \frac{\pi(x'_i) g(x^\star \mid x'_i)}{\pi(x^\star) g(x'_i \mid x^\star)}\right) \tag{2}$$

where $\pi(x'_i)$ denotes the target function, and $g(x^\star \mid x'_i)$ represents the transition proposal. A detailed explanation of these components is provided below.

The target function $\pi(x'_i)$ is designed for approximating the distribution of high-quality adversarial examples. Methodologically, a task-driven approach that maximizes the use of knowledge yields a more accurate approximation. Accordingly, the target function incorporates two key attributes, namely aggressiveness and quality, to capture the essential characteristics of high-quality adversarial examples. Aggressiveness reflects the attack capability of inducing label inversion, informed by posterior knowledge learned from samples accumulated over previous sampling iteration. Quality measures the similarity between candidate examples and original inputs, as well as fluency, derived from predefined quality constraints. Specifically, $\pi(x'_i)$ is defined as:

$$\pi(x'_i) = A(x'_i) \cdot Q(x'_i, x) \tag{3}$$

where $A(x'_i)$ and $Q(x'_i, x)$ quantify the aggressiveness and quality, respectively.

To achieve the estimation of aggressiveness, we transform the statistics from query feedback during the sampling history into position importance scores. Previous studies[Jin *et al.*, 2020] have shown that only some key words act as influential signals for the final prediction. Without confidence scores, posterior knowledge from previous queries can identify positions more strongly correlated with aggressiveness. Therefore, we employ the empirical label-flip frequency to measure the importance of word position. Specifically, the importance of a position is determined by the frequency at which its perturbation induces label inversion, with this frequency calculated based on all discovered adversarial examples. These frequencies are normalized into final importance scores. The aggressiveness component is defined as follows:

$$\begin{aligned} A(x'_i) &= \mathrm{softmax}(\mathbf{f})_j = \frac{\exp(f(j))}{\sum_{t=1}^{n} \exp(f(t))} \\ \mathbf{f} &= [f(1), f(2), ..., f(j), f(n)] \end{aligned} \tag{4}$$

where $f(j)$ represents the number of times that position j, the index selected for substitution in the previous step, has appeared among the indices that induce label inversion.

The other component of the target function is the quality $Q(x', x)$. According to the definition in Equation 1, "high-quality" denotes adversarial texts characterized by high similarity and fluency. We evaluate similarity using semantic similarity and perturbation rate, and assess fluency using the increase in grammatical errors and perplexity. Since the priorities among these features differ, we integrate these four metrics via weighted summation, and it is defined as:

$$\begin{aligned} Q(x'_i, x) =& \lambda_1 \mathbb{S}(x'_i, x) + \lambda_2 \mathbb{C}(x'_i, x'_0) \\ &+ \lambda_3 \mathbb{G}(x'_i) + \lambda_4 \mathbb{P}(x'_i, x) \end{aligned} \tag{5}$$

where λ_1, λ_2, λ_3 and λ_4 are weight parameters. $\mathbb{S}(x'_i, x)$ measures the semantic similarity between x'_i and x as follows:

$$\mathbb{S}(x'_i, x) = \cos\left(Enc(x'_i), Enc(x)\right) \tag{6}$$

where $Enc(\cdot)$ is the universal sentence encoder[Cer *et al.*, 2018], a machine learning model that converts sentences into high-dimensional vectors. The term $\mathbb{C}(x'_i, x'_0)$ quantifies the reduction in perturbation rate, with larger values indicating a smaller perturbation rate in x'_i compared to the initial adversarial example x'_0. $\mathbb{G}(x'_i)$ evaluates grammatical error changes, calculated with the help of LanguageTool, where $\mathbb{G}(x'_i)$ is set to 0 if x'_i contains more grammatical errors than $x^\star$, and 1 otherwise. The perplexity term $\mathbb{P}(x'_i, x)$ calculates the perplexity of x'_i and is defined as:

$$\mathbb{P}(x'_i, x) = \min\left(2, \frac{ppl(x'_i) - ppl(x)}{ppl(x'_0) - ppl(x)}\right) \tag{7}$$

where $ppl(x)$ is calculated with the help of GPT-2.

The transition proposal $g(x^\star \mid x_i')$, representing the transition probability between the current sample and the candidate sample, is defined as:

$$g(x_i \mid x^\star) \propto 1 - \mathbb{S}(x^\star, x), \quad g(x^\star \mid x_i') \propto 1 - \mathbb{S}(x_i', x) \quad (8)$$

The reason for choosing semantic similarity as the primary element of the transition proposal is that it plays a crucial role in guiding the search toward the decision boundary while avoiding abrupt jumps. To some extent, previous works[Maheshwary *et al.*, 2021a; Liu *et al.*, 2024a] view the search process as iteratively optimizing the semantic similarity between the original example and the adversarial example.

(3) Updating the Adversarial Example. This step determines whether to accept a candidate sample. Given the acceptance rate $\alpha(x_i', x^\star)$ for x_i,A value is sampled from a uniform distribution over $(0, \beta)$. If $\alpha(x_i', x^\star)$ exceeds this sampled value, the candidate sample is accepted. This randomness allows LBA to occasionally accept lower-probability candidates, enabling proper exploration of the target distribution. Upon acceptance, $x^\star$ is updated to x_i', and the target model is queried to obtain label y_i. If y_i differs from the ground truth label, x_i is identified as an adversarial example. Finally, original words are substituted back where possible while preserving the adversarial condition.

4 Experiments

4.1 Datasets and Query Budgets

We randomly select 500 samples from the test sets of four commonly used datasets, including two short-text datasets and two long-text datasets: (1) **MR**[Pang and Lee, 2005], a movie review dataset; (2) **AG**[Zhang *et al.*, 2015], a 4-class news classification dataset; (3) **YELP**[Zhang *et al.*, 2015], a binary sentiment classification dataset; and (4) **IMDB**[Maas *et al.*, 2011], another binary classification dataset. We allocate three query budgets, for each dataset based on the average number of queries required by a strong soft-label attack, Textfooler[Jin *et al.*, 2020]. In all experiments, all queries, including queries used to initialize adversarial examples in hard-label attacks, are counted in the given budgets.

4.2 Victim Models

Based on the classification in [Zhao *et al.*, 2023], we select six representative language models from three categories as victim models: Neural Language Models (NLMs), Pretrained Language Models (PLMs), and Large Language Models (LLMs). For NLMs and PLMs, we follow the setup used in existing hard-label adversarial attacks and target three models: **WordCNN**[Kim, 2014], **WordLSTM**[Hochreiter and Schmidhuber, 1997], and **BERT**[Devlin *et al.*, 2019]. All the NLMs and PLMs are provided by TextAttack [Morris *et al.*, 2020]. For LLMs, we adopt the following three popular models as victims: **LLaMA2-7b-chat** (LLaMA2, [Touvron *et al.*, 2023]), **Vicuna-7b-v1.5** (Vicuna, [Du *et al.*, 2022]), and **GPT-4o-mini-tts-2025-03-20** (GPT-4o, [OpenAI, 2023]). For the generation parameters of the two open-source LLMs, i.e., LLaMA2 and Vicuna, we set the temperature to 0.1 and the maximum tokens to 15.

4.3 Baselines

We choose two strong soft-label attacks and four state-of-the-art hard-label attacks as our baselines. These baselines include: (1) **Textbugger**[Li *et al.*, 2020], a soft-label attack that leverags word importance ranking to guide perturbations; (2) **Textfooler**[Jin *et al.*, 2020], a soft-label attack baseline; (3) **HLA**[Maheshwary *et al.*, 2021a], a heuristic-based hard-label attack method; (4) **Texthoaxer**[Ye *et al.*, 2022], a gradient-based hard-label attack method; (5) **LeapAttack**[Ye *et al.*, 2023], another gradient-based hard-label attack method; and (6) **HQA-Attack**[Liu *et al.*, 2024a], a semantic similarity-guided hard-label attack method. We grant the soft-label attacks access to the confidence scores output by the target models to successfully execute the attacks.

4.4 Implementation Setting

Following prior work[Ye *et al.*, 2023; Ye *et al.*, 2022; Liu *et al.*, 2024a], all the hard-label attacks utilize the same initialization method originally proposed by HLA [Maheshwary *et al.*, 2021a] for fair comparison. For the quality component weights in our target function, we set $\lambda_1 = 0.25$, $\lambda_2 = 0.50$, $\lambda_3 = 0.15$, and $\lambda_4 = 0.10$. These values were determined through a sensitivity analysis, where we first optimized the allocation between the similarity group ($\lambda_1 + \lambda_2$) and the fluency group ($\lambda_3 + \lambda_4$), then tuned the ratio within each group. Detailed results are provided in Appendix C.

4.5 Evaluation Metrics

We evaluate the quality of adversarial examples from two dimensions: similarity and fluency. Following previous studies[Ye *et al.*, 2022; Ye *et al.*, 2023; Liu *et al.*, 2024a], we use two metrics **perturbation rate** (Pert.) and **semantic similarity** (Sim.), to quantify the similarity. The semantic similarity is measured by the cosine similarity between the original text and its adversary, calculated using the Universal Sentence Encoder[Cer *et al.*, 2018]. To assess the fluency of adversarial texts, we employ two metrics: **perplexity** (PPL) and the **increase in grammatical errors** (Ige.). We utilize GPT-2[Radford *et al.*, 2019] to calculate the perplexity of texts. The increase in grammatical errors refers to the percentage of increased grammar mistakes compared to the original text.

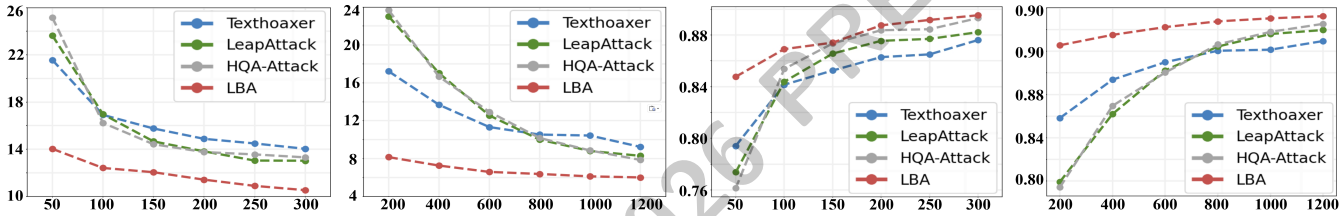
To more objectively evaluate the aggressiveness and quality of adversarial texts, we further utilize GPT-4o as an evaluator and propose two metrics: **consistency rate** (CR) and **fully understood rate** (FUR). The CR metric measures the percentage of adversarial examples classified by GPT-4o as matching their ground truth labels, distinguishing whether aggressiveness stems from altering the original semantics:

$$CR = \frac{\sum_{x' \in X_{adv}} \mathbf{I}(E_{val}(x') = y)}{\mid X_{adv} \mid} \quad (9)$$

where X_{adv} is the set of adversarial texts, y is the ground truth label of the original text, $E_{val}(x')$ denotes GPT-4o's predicted label, and $\mathbf{I}(\cdot)$ returns 1 if the condition holds. The FUR metric is used to assess whether GPT-4o can fully understand the content of adversarial text, highlighting the robustness of the attack. A "fully understood" response indicates

Dataset	Method	Tiny Budget					Tight Budget					Moderate Budget				
		ASR.(%)	Pert.(%)	Sim.	Ige.(%)	PPL	ASR.(%)	Pert.(%)	Sim.	Ige.(%)	PPL	ASR.(%)	Pert.(%)	Sim.	Ige.(%)	PPL
MR	*Textbugger*	32.57	10.16	0.9268	6.34	207.03	58.92	12.10	0.9105	6.37	212.83	63.28	13.32	0.9033	6.98	223.52
	Textfooler	13.69	11.39	0.8846	0.82	211.34	51.87	13.90	0.8609	0.88	193.60	73.24	14.89	0.8514	1.08	195.86
	HLA		30.95	0.7201	2.98	427.86		27.17	0.7468	2.23	371.42		24.14	0.7751	1.88	331.31
	Texthoaxer		21.56	0.7941	2.24	304.17		16.91	0.8415	1.62	243.44		15.76	0.8525	1.46	224.70
	LeapAttack	90.66	23.63	0.7737	2.30	325.37	93.57	16.96	0.8438	1.71	234.07	94.61	14.66	0.8656	1.35	217.16
	HQA-Attack		25.15	0.7614	2.31	343.57		16.23	0.8537	1.52	229.11		14.42	0.8735	1.51	208.17
	LBA		**14.03**	**0.8477**	**1.47**	**209.33**		**12.41**	**0.8691**	**1.22**	**207.07**		**12.05**	**0.8741**	**0.88**	**194.47**
AG	*Textbugger*	23.94	7.64	0.9490	2.58	212.96	45.55	13.33	0.9169	4.19	256.12	56.78	16.84	0.9006	5.33	291.55
	Textfooler	16.10	6.27	0.9567	0.16	183.03	36.23	10.21	0.9269	0.74	223.14	55.93	14.38	0.9007	1.15	261.18
	HLA		39.32	0.6691	2.61	678.43		35.12	0.7002	2.40	594.65		28.98	0.7466	2.23	485.43
	Texthoaxer		25.42	0.7878	1.80	475.20		18.06	0.8398	1.07	342.72		17.71	0.8417	0.85	337.16
	LeapAttack	77.97	31.76	0.7288	2.32	567.38	86.02	18.91	0.8346	1.27	346.83	87.71	15.84	0.8617	0.96	312.46
	HQA-Attack		33.65	0.7176	2.45	584.73		19.63	0.8332	1.28	366.63		14.87	0.8648	1.03	292.64
	LBA		**14.30**	**0.8501**	**0.74**	**292.53**		**13.66**	**0.8521**	**0.48**	**285.75**		**12.74**	**0.8653**	**0.35**	**278.27**
YELP	*Textbugger*	38.40	8.82	0.9388	3.96	120.02	63.24	9.23	0.9369	3.83	110.65	76.39	9.43	0.9374	3.89	109.06
	Textfooler	32.24	7.67	0.9352	0.95	101.61	63.04	8.04	0.9374	0.89	96.48	80.08	8.31	0.9376	0.87	97.70
	HLA		34.22	0.6920	2.78	438.88		25.34	0.7772	2.59	328.51		20.16	0.8021	2.36	210.70
	Texthoaxer		17.22	0.8582	2.25	198.83		13.68	0.8937	1.64	173.69		11.33	0.9099	1.40	139.97
	LeapAttack	97.13	23.03	0.7991	2.89	281.54	99.38	17.03	0.8619	2.22	221.51	99.59	12.55	0.9023	1.46	166.95
	HQA-Attack		23.68	0.7941	2.76	283.89		16.66	0.8695	2.02	216.36		12.93	0.9003	1.60	167.51
	LBA		**8.15**	**0.9256**	**1.05**	**105.15**		**7.26**	**0.9352**	**0.79**	**103.03**		**6.59**	**0.9424**	**0.76**	**97.46**
IMDB	*Textbugger*	44.78	4.70	0.9722	2.24	99.92	69.78	6.54	0.9606	2.79	102.24	76.12	7.71	0.9521	3.22	107.08
	Textfooler	40.30	3.06	0.9833	0.34	80.68	66.04	4.45	0.9733	0.49	84.43	75.00	5.38	0.9676	0.64	88.27
	HLA		22.87	0.8249	2.42	268.39		19.87	0.8549	2.12	223.39		18.78	0.8767	2.29	198.19
	Texthoaxer		14.14	0.9028	1.67	175.68		9.99	0.9373	1.16	138.13		8.77	0.9453	1.14	133.21
	LeapAttack	87.31	19.88	0.8436	2.30	230.82	88.81	10.91	0.9233	1.22	157.60	93.28	9.28	0.9349	1.08	140.48
	HQA-Attack		17.85	0.8643	2.07	211.92		13.98	0.8985	1.72	190.54		10.14	0.9288	1.15	152.95
	LBA		**4.21**	**0.9728**	**0.49**	**89.95**		**3.89**	**0.9752**	**0.46**	**88.93**		**3.68**	**0.9767**	**0.47**	**87.50**

Table 1: Comparison of perturbation rate (Pert.), semantic similarity (Sim.), increase in grammatical errors(Ige.) and perplexity (PPL) with given query budgets when attacking BERT. For a fair comparison, methods with low attack success rates (ASR.) - more than 20% below other baselines (denoted by underscored values) - are excluded from the quality comparison.



(a) MR (Pert.) (b) YELP (Pert.) (c) MR (Sim.) (d) YELP (Sim.)

Figure 2: Comparison of Pert. (a,b) and Sim. (c,d) under various query budgets for attacks against BERT. The x-axis is the query number. The y-axis is the perturbation rate (%) in (a) and (b), and the semantic similarity in (c) and (d).

that the text is coherent and fluent, suggesting that perturbations introduced by the attack are more difficult to detect. FUR is defined as follows:

$$FUR = \frac{\sum_{x' \in X_{adv}} \mathbf{I}(E_{val}(x') = 1)}{|X_{adv}|} \tag{10}$$

where $E_{val}(x')$ denotes whether GPT-4o's response is classified as "fully understood" (represented by 1).

4.6 Experimental Results

Comparison of Similarity and Fluency

We performed experiments with the given query budgets against WordCNN, WordLSTM, and BERT. Due to limited space, we present results on BERT in Table 1, and others in Appendix A.1. As shown in Table 1, LBA significantly outperforms baselines under all query budgets, achieving the highest similarity (Pert., Sim.) and fluency (Ige., PPL.) across all victim models and datasets. For example, when attacking BERT under the tiny budget, our method reduces the average Pert., Ige., and PPL by 56.18%, 58.56%, and 47.76%, respectively, while improving the average Sim. by 11.74%, compared to the second-best baseline. Moreover, it can be observed that on datasets with larger combinatorial spaces, namely the long-text datasets YELP and IMDB, the performance gap in the quality metrics between LBA and other baselines is more significant compared to the other two datasets. This trend further highlights the effectiveness of LBA's distribution-based sampling strategy in efficiently considering all possible position combinations. To further analyze the distribution of generated samples in terms of similarity, we also compare the cumulative distributions of perturbation rate and semantic similarity for adversarial examples in Appendix A.2. As the query budget increases from tiny to tight and moderate levels, the results show that LBA maintains its superiority over other hard-label baselines.

Comparison of Attack Under More Query Budgets

In real-world scenarios, both limited and ample query budgets may be encountered. Therefore, to more comprehensively evaluate the applicability of LBA, we further compare the Pert. and Sim. of hard-label attacks across various query budgets. Considering the effect of text length on the search space, we choose two datasets with varying lengths: the short-text MR dataset and the long-text YELP dataset.

Dataset	Method	LLaMA2					Vicuna					GPT-4o				
		ASR.(%)	Pert.(%)	Sim.	Ige.(%)	PPL	ASR.(%)	Pert.(%)	Sim.	Ige.(%)	PPL	ASR.(%)	Pert.(%)	Sim.	Ige.(%)	PPL
MR	Texthoaxer	89.58	21.03	0.7905	2.08	308.29	96.61	14.77	0.8502	1.21	228.04	71.68	32.60	0.6619	2.71	440.93
	LeapAttack		25.22	0.7461	2.54	351.06		16.34	0.8388	2.17	242.68		33.17	0.6578	2.83	442.51
	HQA-Attack		25.06	0.7569	2.42	363.66		16.14	0.8537	1.16	231.93		31.77	0.6809	2.69	436.88
	LBA		**14.88**	**0.8322**	**1.41**	**243.41**		**11.37**	**0.8729**	**1.09**	**206.63**		**19.78**	**0.7855**	**2.02**	**308.33**
YELP	Texthoaxer	63.11	27.86	0.7471	3.42	349.59	74.69	22.96	0.8001	2.78	318.34	50.74	33.11	0.6939	3.48	456.67
	LeapAttack		29.25	0.7343	3.65	403.38		23.24	0.7924	2.61	317.54		34.63	0.6859	3.75	464.63
	HQA-Attack		28.13	0.7459	3.54	346.41		23.82	0.7930	2.77	325.70		31.35	0.7134	3.04	351.41
	LBA		**12.94**	**0.8792**	**1.54**	**123.22**		**9.76**	**0.9096**	**1.10**	**96.80**		**22.31**	**0.7728**	**2.58**	**238.21**

Table 2: Comparison of Pert., Sim., Ige. and PPL with tiny query budgets in attacks against target LLMs.

Dataset	Method	Acc(%)	CR (%)	FUR (%)
MR	HLA	91.5	60.61	28.41
	Texthoaxer		66.11	34.45
	LeapAttack		65.45	33.41
	HQA-Attack		62.38	32.79
	LBA		**66.20**	**38.03**
AG	HLA	89.4	79.03	9.68
	Texthoaxer		82.06	13.74
	LeapAttack		75.54	12.50
	HQA-Attack		73.16	13.13
	LBA		**87.43**	**15.51**
YELP	HLA	98.4	92.62	15.57
	Texthoaxer		93.02	25.97
	LeapAttack		91.97	22.41
	HQA-Attack		92.59	24.36
	LBA		**94.67**	**32.41**
IMDB	HLA	93.3	61.02	32.20
	Texthoaxer		88.78	46.93
	LeapAttack		88.89	32.20
	HQA-Attack		89.27	49.77
	LBA		**91.88**	**67.95**

Table 3: Results of utility evaluation using GPT-4o. Acc stands for GPT-4o prediction accuracy of original texts.

The results shown in Figure 2 indicate that LBA's performance continues to improve as the query budget increases and always surpasses that of other methods. This suggests that LBA effectively adapts to both limited and ample query budgets, making it well-suited for real-world applications.

Comparison of Attacking LLMs

To further evaluate the practicality of LBA on LLMs, we conduct experiments on representative LLMs under the tiny query budget. In view of the observed performance on NLMs and PLMs, we exclude HLA and select TextHoaxer, LeapAttack, and HQA-Attack as the baselines. We use 500 texts from two representative datasets, MR (a short-text dataset) and YELP (a long-text dataset), as the target samples. As shown in Table 2, the results show that LBA generates the highest quality adversarial texts in all three LLMs. For example, in attacks on dataset YELP, compared with the second best results, LBA achieves a 12.39% lower perturbation rate, 1.28% lower increase in grammatical errors, and 185.67 lower perplexity, and an 11.74% higher semantic similarity compared to the second-best method. Overall, the results not only confirm the robust practicality of LBA but also highlight the need to enhance the adversarial robustness of LLMs.

4.7 Evaluation by LLM Assistant

To more accurately evaluate the utility of the generated adversarial text, we leverage GPT-4o to automatically assess the adversarial examples across two aspects: aggressiveness and quality. Aggressiveness is measured by CR, which helps distinguish whether the aggressiveness of adversarial examples stems from altering the original meaning, further evaluating the semantic preservation of the samples. Quality is measured by FUR, which reflects whether the fluency of the samples aligns with natural language, further evaluating the comprehensibility of the samples. We use the adversarial texts generated against BERT with the tiny query budget. As shown in Table 3, LBA achieves the highest CR and FUR among the hard-label baselines, indicating that these adversarial texts remain the most semantically preserved and comprehensible. Human evaluation results are shown in Appendix A.3.

Method	Pert.(%)	Sim.	Ige.(%)	PPL	LIHR (%)
Quality	16.57	0.8345	1.51	236.83	21.77
Aggressiveness	19.59	0.8276	1.87	263.44	30.56
LBA	14.03	0.8477	1.47	209.33	28.84

Table 4: Ablation study results of LBA.

4.8 Ablation Study and Case Study

We study LBA's performance with and without the aggressiveness and quality components on MR against BERT under a tight query budget. To further validate the effectiveness of the aggressiveness component, we introduce the **label inversion hit rate** (LIHR), which measures the proportion of accepted samples that successfully induce label flipping. Results in Table 4 show that the aggressiveness component improves LIHR, while the quality component enhances sample quality. Overall, this demonstrates that both the aggressiveness and quality components play crucial roles in approximating the distribution of high-quality adversarial examples. We also list some adversarial texts crafted by LBA, which are shown in Appendix B. These adversarial texts further demonstrate that LBA can generate high-quality adversarial texts with low-query budgets.

5 Conclusion

In this paper, we present LBA, a novel sampling-based hard-label attack that generates high-quality adversarial texts with low query budgets. Our approach overcomes the limitations of existing methods by designing a sampling-based strategy to mitigate the explosion of the search space, thereby improving the query efficiency. Through extensive experiments on six language models and four datasets, we demonstrate that LBA significantly outperforms existing hard-label attacks across all evaluation metrics. Furthermore, our results show that the adversarial texts generated by LBA are not only effective but also comprehensible, as evaluated by GPT-4o.

Acknowledgments

This work was supported by the Key Project of the National Natural Science Foundation of China under grant no. 62536007, the National Natural Science Foundation of China under grant no. 62502454, the Zhejiang Province Science Foundation under grant no. LD24F020002, LQ24F020025, LZYQ25F020003, the Zhejiang Province's 2025 "Leading Goose + X" Science and Technology Plan under grant no. 2025C02034, and the Jinhua Science and Technology Plan under Grant 2023-1-091.

References

[Alzantot *et al.*, 2018] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, B. Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. *EMNLP*, pages 2890–2896, 2018.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Cer *et al.*, 2018] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris 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.

[Chib and Greenberg, 1995] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335, 1995.

[Devlin *et al.*, 2019] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[Du *et al.*, 2022] Zhengxiao Du, Yujie Qian, Xiao Liu, Ming Ding, Jiezhong Qiu, Zhilin Yang, and Jie Tang. Glm: General language model pretraining with autoregressive blank infilling. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 320–335, 2022.

[Gao *et al.*, 2024] Chongyang Gao, Kang Gu, Soroush Vosoughi, and Shagufta Mehnaz. Semantic-preserving adversarial example attack against bert. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*, pages 202–207, 2024.

[Garg and Ramakrishnan, 2020] Siddhant Garg and Goutham 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.

[Goodfellow *et al.*, 2015] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997.

[Hu *et al.*, 2022] Junjie Hu, Hiroaki Hayashi, Kyunghyun Cho, and Graham Neubig. DEEP: DEnoising entity pre-training for neural machine translation. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1753–1766, 2022.

[Jin *et al.*, 2020] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025, 2020.

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *EMNLP*, pages 1746–1751, 2014.

[Li *et al.*, 2020] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS*, 2020.

[Li *et al.*, 2021] Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and William 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.

[Linyang *et al.*, 2020] Li Linyang, Ma Ruotian, Guo Qipeng, Xue Xiangyang, and Qiu Xipeng. Bert attack: Adversarial attack against bert using bert. *EMNLP 2020*, pages 6193–6202, 2020.

[Liu *et al.*, 2024a] Han Liu, Zhi Xu, Xiaotong Zhang, Feng Zhang, Fenglong Ma, Hongyang Chen, Hong Yu, and Xianchao Zhang. Hqa-attack: toward high quality black-box hard-label adversarial attack on text. *Advances in Neural Information Processing Systems*, 36:51347-51358, 2024.

[Liu *et al.*, 2024b] Zhaorong Liu, Xi Xiong, Yuanyuan Li, Yan Yu, Jiazhong Lu, Shuai Zhang, and Fei Xiong. Hygloadattack: Hard-label black-box textual adversarial attacks via hybrid optimization. *Neural Networks*, 178:106461, 2024.

[Maas *et al.*, 2011] L. Andrew Maas, E. Raymond Daly, T. Peter Pham, Dan Huang, Y. Andrew Ng, and Christopher Potts. Learning word vectors for sentiment analysis. *ACL*, pages 142–150, 2011.

[Maheshwary *et al.*, 2021a] Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. Generating natural language attacks in a hard label black box setting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13525–13533, 2021.

[Maheshwary *et al.*, 2021b] Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. A strong baseline for query efficient attacks in a black box setting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8396–8409, 2021.

[Morris *et al.*, 2020] John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, 2020.

[OpenAI, 2023] R OpenAI. Gpt-4 technical report. *arXiv*, pages 2303–08774, 2023.

[Pang and Lee, 2005] Bo Pang and Lillian Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. *meeting of the association for computational linguistics*, pages 115–124, 2005.

[Peng *et al.*, 2023] Hao Peng, Shixin Guo, Dandan Zhao, Xuhong Zhang, Jianmin Han, Shouling Ji, Xing Yang, and Ming Zhong. Textcheater: A query-efficient textual adversarial attack in the hard-label setting. *IEEE Transactions on Dependable and Secure Computing*, (01):1–16, 2023.

[Radford *et al.*, 2019] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[Touvron *et al.*, 2023] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[Ye *et al.*, 2022] Muchao Ye, Chenglin Miao, Ting Wang, and Fenglong Ma. Texthoaxer: budgeted hard-label adversarial attacks on text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 3877–3884, 2022.

[Ye *et al.*, 2023] Muchao Ye, Jinghui Chen, Chenglin Miao, Ting Wang, and Fenglong Ma. Leapattack: Hard-label adversarial attack on text via gradient-based optimization. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2307–2315, 2023.

[Zang *et al.*, 2020] Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. Word-level textual adversarial attacking as combinatorial optimization. pages 6066–6080, 2020.

[Zhang *et al.*, 2015] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA, 2015.

[Zhang *et al.*, 2023] Biao Zhang, Barry Haddow, and Alexandra Birch. Prompting large language model for machine translation: a case study. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, pages 41092–41110, 2023.

[Zhao *et al.*, 2023] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.