LimeAttack: Local Explainable Method for Textual Hard-Label Adversarial Attack

Hai Zhu[1 3*], **Qingyang Zhao**[2], **Weiwei Shang**[1], **Yuren Wu**[3], **Kai Liu**[4]

[1] University of Science and Technology of China
[2] Xidian University
[3]Ping An Technology
[4]Lazada

SA21218029@mail.ustc.edu.cn,21151213588@stu.xidian.edu.cn,wwshang@ustc.edu.cn,leifive@163.com,baiyang.lk@alibaba-inc.com

Abstract

Natural language processing models are vulnerable to adversarial examples. Previous textual adversarial attacks adopt model internal information (gradients or confidence scores) to generate adversarial examples. However, this information is unavailable in the real world. Therefore, we focus on a more realistic and challenging setting, named hard-label attack, in which the attacker can only query the model and obtain a discrete prediction label. Existing hard-label attack algorithms tend to initialize adversarial examples by random substitution and then utilize complex heuristic algorithms to optimize the adversarial perturbation. These methods require a lot of model queries and the attack success rate is restricted by adversary initialization. In this paper, we propose a novel hard-label attack algorithm named LimeAttack, which leverages a local explainable method to approximate word importance ranking, and then adopts beam search to find the optimal solution. Extensive experiments show that LimeAttack achieves the better attacking performance compared with existing hard-label attack under the same query budget. In addition, we evaluate the effectiveness of LimeAttack on large language models and some defense methods, and results indicate that adversarial examples remain a significant threat to large language models. The adversarial examples crafted by LimeAttack are highly transferable and effectively improve model robustness in adversarial training.

Introduction

Deep Neural Networks (DNNs) are widely applied in the natural language processing field and have achieved great success (Kim 2014; Devlin et al. 2019; Minaee et al. 2021; Hochreiter and Schmidhuber 1997). However, DNNs are vulnerable to adversarial examples, which are correctly classified samples altered by some slight perturbations (Jin et al. 2020; Papernot et al. 2017; Kurakin, Goodfellow, and Bengio 2016). These adversarial perturbations are imperceptible to humans but can mislead the model. Adversarial examples seriously threaten the robustness and reliability of DNNs, especially in some security-critical applications (*e.g.*, autonomous driving and toxic text detection (Yang et al. 2021; Kurakin, Goodfellow, and Bengio 2018)). Therefore, adversarial examples have attracted enormous attention on adversarial attacks and defenses in computer vision, natural language processing and speech (Szegedy et al. 2013; Carlini and Wagner 2018; Yu et al. 2022). It is more challenging to craft textual adversarial examples due to the discrete nature of language along with the presence of lexical, semantic, and fluency constraints.

According to different scenarios, textual adversarial attacks can be briefly divided into white-box attacks, score-based attacks and hard-label attacks. In a white-box setting, the attacker utilizes the model's parameters and gradients to generate adversarial examples (Goodman, Zhonghou et al. 2020; Jiang et al. 2020). Score-based attacks only adopt class probabilities or confidence scores to craft adversarial examples (Jin et al. 2020; Li et al. 2020; Ma, Shi, and Guan 2020; Zhu, Zhao, and Wu 2023). However, these attack methods perform poorly in reality due to DNNs being deployed through application programming interfaces (APIs), and the attacker having no access to the model's parameters, gradients or probability distributions of all labels (Ye et al. 2022b). In contrast, under a hard-label scenario, the model's internal structures, gradients, training data and even confidence scores are unavailable. The attacker can only query the black-box victim model and get a discrete prediction label, which is more challenging and realistic. Additionally, most realistic models (*e.g.*, HuggingFace API, OpenAI API) usually have a limit on the number of calls. In reality, the adversarial examples attack setting is hard-label with tiny model queries.

Some hard-label attack algorithms have been proposed (Yu et al. 2022; Ye et al. 2022b; Maheshwary, Maheshwary, and Pudi 2021; Ye et al. 2022a). They follow two-stages strategies: i) generate low-quality adversarial examples by randomly replacing several original words with synonyms, and then ii) adopt complex heuristic algorithms (*e.g.*, genetic algorithm) to optimize the adversary perturbation. Therefore, these attack methods usually require a lot of queries and the attack success rate and quality of adversarial examples are limited by adversary initialization. On the contrary, score-based attacks calculate the word importance based on the change in confidence scores after deleting one word. Word importance ranking improves attack efficiency by preferring to attack words that have a significant impact on the model's predictions (Jin et al. 2020). However, score-based attacks cannot calculate the word importance in a hard-label setting because deleting one token hardly changes the discrete prediction label. Therefore, we want to investigate such a problem:

*Corresponding author.

how to calculate word importance ranking in a hard-label setting to improve attack efficiency?

Actually, word importance ranking can reveal the decision boundary to determine the better attack path, but existing hard-label algorithms ignore this useful information because it is hard to obtain. Inspired by local explainable methods (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Shrikumar et al. 2016) for DNNs, which are often used to explain the outputs of black-box models, aim to estimate the token sensitivity on the benign sample. Previous study (Chai et al. 2023) has tried to simply replace deletion-based method with local explainable method to calculate word importance in score-based attack. However, we have verified through experiments that local explainable method does not have a significant advantage over deletion-based method in a score-based scenario. Because the probability distribution of the model's output is available, the influence of each word on the output can be well reflected by deletion-based method. Therefore, compared with score-based attacks, we think local explainable method can play a greater advantage in hard-label attacks where deletion-based method is useless. We adopt the most fundamental and straightforward local explainable method, namely LIME. LIME is easy to understand and more in line with the deletion-based method proposed in score-based attacks, since our goal is to bridge the gap between score-base attacks and hard-label attacks by introducing interpretability method. In fact, local explainable methods are model-agnostic and suitable for conducting word importance estimation for hard-label attacks. However, there are the following difficulties in applying LIME to hard-label attacks: 1) How to allocate LIME and search queries under tiny query budget to achieve optimal results. 2) How to establish a mapping relationship between LIME and word importance in adversarial samples without model's logits output. 3) How to sample reasonably during perturbation execution to achieve optimal results. In subsequent sessions we will explain in detail how to solve these difficulties.

In this work, we propose a novel hard-label attack algorithm named LimeAttack. The application of LIME in hard-label attacks was inspired by the score-based attacks' deletion method. We verify the effectiveness of inside-to-outside attack path in hard-label attacks, then many excellent score-based attacks may provide hard-label attacks more insight. To evaluate the attack performance and efficiency, we compare LimeAttack with other hard-label attacks and take several score-based attacks as references for two NLP tasks on seven common datasets. We also evaluate LimeAttack on the currently state-of-the-art large language models (*e.g.*, ChatGPT). Experiments show that LimeAttack achieves the highest attack success rate compared to other baselines under the tiny query budget. Our contributions are summarized as follows:

- We summarize the shortcomings of the existing hard-label attacks and apply LIME to connect score-base attacks and hard-label attacks and verify the effectiveness of inside-to-outside attack path in hard-label attacks.
- Extensive experiments show that LimeAttack achieves higher attack success rate than existing hard-label attack algorithms under tiny query budget. Meanwhile, adversarial examples crafted by LimeAttack are high quality and difficult for humans to distinguish. [1]
- In addition, we also conduct attacks and evaluations on the currently state-of-the-art large language models. Results indicate that adversarial examples remain a significant threat to large language models. We also have added attack performance on defense methods and convergence results of attack success rate and perturbation rate.

Related Work

Hard-Label Adversarial Attacks

In a hard-label setting, the attacker can only query the victim model and get a discrete prediction label. Therefore, hard-label setting is more practical and challenging. Existing hard-label attacks contain two-stages strategies, *i.e.*, adversary initialization and perturbation optimization. HLBB (Maheshwary, Maheshwary, and Pudi 2021) initializes an adversarial example and adopts a genetic algorithm to optimize the perturbation. TextHoaxer (Ye et al. 2022b) and LeapAttack (Ye et al. 2022a) utilizes semantic similarity and perturbation rate as optimization objective to search for a better perturbation matrix in the continuous word embedding space. TextHacker (Yu et al. 2022) adopts a hybrid local search algorithm and a word importance table learned from attack history to guide the local search. These attack methods often require a lot of queries to reduce the perturbation rate, and the attack success rate and quality of adversary are limited by initialization. Therefore, in this work, we attempt to craft an adversarial example directly from the benign sample. This approach can generate high-quality adversarial examples with fewer queries.

Local Explainable Methods

To improve DNN interpretability and aid decision-making, various methods for explaining DNNs have been proposed and broadly categorized as global or local explainable methods. Global explainable methods focus on the model itself by using the overall knowledge about the model's architecture and parameters. On the contrary, local methods fit a simple and interpretable model (*e.g.*, decision tree) to a single input to measure the contribution of each token. In detail, local explainable methods (Lundberg and Lee 2017; Shrikumar et al. 2016; Štrumbelj and Kononenko 2014) associate all input tokens by defining a linear interpretability model and assumes that the contribution of each token in the input is additive. This is also called the additive feature attribution method. In this paper, local interpretable model-agnostic explanation (LIME) (Ribeiro, Singh, and Guestrin 2016) is applied to calculate word importance, which is a fundamental and representative local explainable method. The intuition of LIME is to generate many neighborhood samples by deleting some original words in the benign example. These samples are then used to train a linear model where the number of features equals to the number of words in the benign sample. The parameters of this linear model are approximated to the importance of each word. As LIME is model-agnostic, it is suitable for hard-label attacks.

[1] Code is available in https://github.com/zhuhai-ustc/limeattack

Limitation of Existing Hard-Label Attack

In order to intuitively compare the difference between LimeAttack and existing hard-label attack algorithms, we create attack search path visualizations in Figure 3. LimeAttack's search paths are represented by green lines, and they move from **inside to outside**. LimeAttack utilizes a local explainable method to learn word importance ranking and generates adversarial examples iteratively from benign samples. This helps LimeAttack to find the nearest decision boundary direction, and costs fewer model queries to attack keywords preferentially. In contrast, previous hard-label attack algorithms' search paths are represented by blue lines, and they move from **outside to inside**. These algorithms typically begin with a randomly initialized adversarial example and optimize perturbation by maximizing semantic similarity between the initialized example and the benign sample, which requires a lot of model queries to achieve a low perturbation rate. Furthermore, their attack success rate and adversary quality are also limited by the adversary initialization.

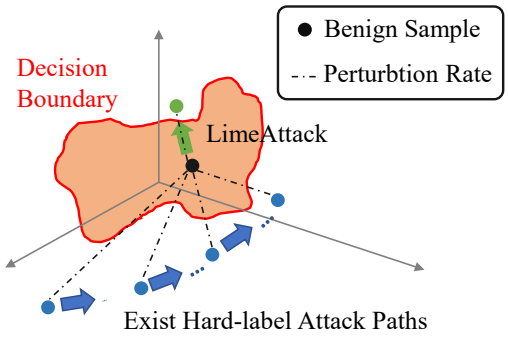

Figure 1: Search paths of existing hard-label attacks and LimeAttack.

Methodology

Problem Formulation

Given a sentence of n words $\boldsymbol{X} = [x_1, x_2, \cdots, x_n]$ and its ground truth label Y, an adversarial example $\boldsymbol{X'} = [x'_1, x'_2, \cdots, x'_n]$ is crafted by replacing one or more original words with synonyms to mislead the victim model $\mathcal{F}$. *i.e.*,

$$\mathcal{F}(\boldsymbol{X'}) \neq \mathcal{F}(\boldsymbol{X}), \quad \text{s.t.} \quad D(\boldsymbol{X}, \boldsymbol{X'}) < \epsilon \tag{1}$$

$D(\cdot, \cdot)$ is an edit distance that measures the modifications between a benign sample $\boldsymbol{X} = [x_1, x_2, \cdots, x_n]$ and an adversarial example $\boldsymbol{X'} = [x'_1, x'_2, \cdots, x'_n]$:

$$D(\boldsymbol{X}, \boldsymbol{X'}) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}(x_i, x'_i) \tag{2}$$

$\mathbb{E}(\cdot, \cdot)$ is a binary variable that equals to 0 if $x_i = x'_i$ and 1 otherwise. A high-quality adversarial example should be similar to the benign sample, and human readers should hardly be able to distinguish the difference. The LimeAttack belongs to the hard-label attack, it has nothing to do with the model's parameters, gradients or confidence scores. The attacker can only query the victim model to obtain a predicted label $\hat{\boldsymbol{Y}} = \mathcal{F}(\hat{\boldsymbol{X}})$.

The Proposed LimeAttack Algorithm

The overall flow chart is shown in Figure 2. LimeAttack follows two steps, *i.e.*, word importance ranking and perturbation execution.

Word Importance Ranking. Given a sentence of n words $\boldsymbol{X}$, we assume that the contribution of all words is additive, and their sum is positively related to the model's prediction. As shown in the Figure 2, we generate some neighborhood samples $\mathcal{X} = [\boldsymbol{X}'_1, \boldsymbol{X}'_2, \cdots, \boldsymbol{X}'_n]$ from a benign example $\boldsymbol{X}$ by randomly replacing some words with '[MASK]'. Usually, sentences with more words often requires more neighbor samples to approximate the word importance. Therefore, we keep the number of neighborhood samples consistent with the number of tokens. We then feed $\mathcal{X}$ to the victim model $\mathcal{F}$ to obtain discrete prediction labels $\hat{\mathcal{Y}} = [\hat{\boldsymbol{Y}}'_1, \hat{\boldsymbol{Y}}'_2, \cdots, \hat{\boldsymbol{Y}}'_n]$. Subsequently, we will fit a linear interpretability model to classify these neighborhood samples:

$$g(\boldsymbol{X}, \boldsymbol{\theta}) = \theta_0 + \sum_{i=1}^{n} \theta_i \mathbb{I}(x_i, \boldsymbol{X}) \tag{3}$$

where $\boldsymbol{\theta}$ is the parameter of the linear model, $\mathbb{I}(\cdot, \cdot)$ is a binary variable that equals to 1 if word x_i in $\boldsymbol{X}$ and 0 otherwise. Therefore, the parameter $\theta_i, i \in [1, n]$ reflects the change without word x_i and is approximated to the word importance. In addition, we have verified through experiments that the linear model (such as LIME) has the same effect as some advanced interpretation methods (such as SHAP) or non-linear models (such as decision tree) under tiny query budgets. SHAP or non-linear models also have a higher computational complexity. The advantages of some advanced interpretation methods or non-linear models will only be reflected when there are a large number of neighborhood samples and queries.

In detail, we transform each neighborhood sample $\boldsymbol{X}'_i$ into the binary vector $\boldsymbol{V}'_i$. If the origin word is removed in $\boldsymbol{X}'_i$, its corresponding vector dimension in $\boldsymbol{V}'_i$ is 0 otherwise 1. Therefore, $\boldsymbol{V}'_i$ has the same length as $\boldsymbol{X}'_i$, which is the length of the benign example. A benign example $\boldsymbol{X}$ is also transformed to $\boldsymbol{V}$. Sometimes neighborhood samples may not necessarily be linearly separable, LIME adopts gaussian kernel to weight the loss for each sample to gather points closest to the original sample, which helps with linear fitting. We give weights $\pi(\boldsymbol{V}'_i, \boldsymbol{V})$ to each neighborhood sample according to their distance from the benign sample (Ribeiro, Singh, and Guestrin 2016).

$$\pi(\boldsymbol{V}'_i, \boldsymbol{V}) = \exp\left(-d(\boldsymbol{V}'_i, \boldsymbol{V})^2 / \sigma^2\right) \tag{4}$$

where $d(\cdot, \cdot)$ is a distance function. We adopt the cosine similarity as the distance metric.

$$d(\boldsymbol{V}'_i, \boldsymbol{V}) = \frac{\boldsymbol{V}'_i \cdot \boldsymbol{V}}{\sqrt{|\boldsymbol{V}'_i||\boldsymbol{V}|}} \tag{5}$$

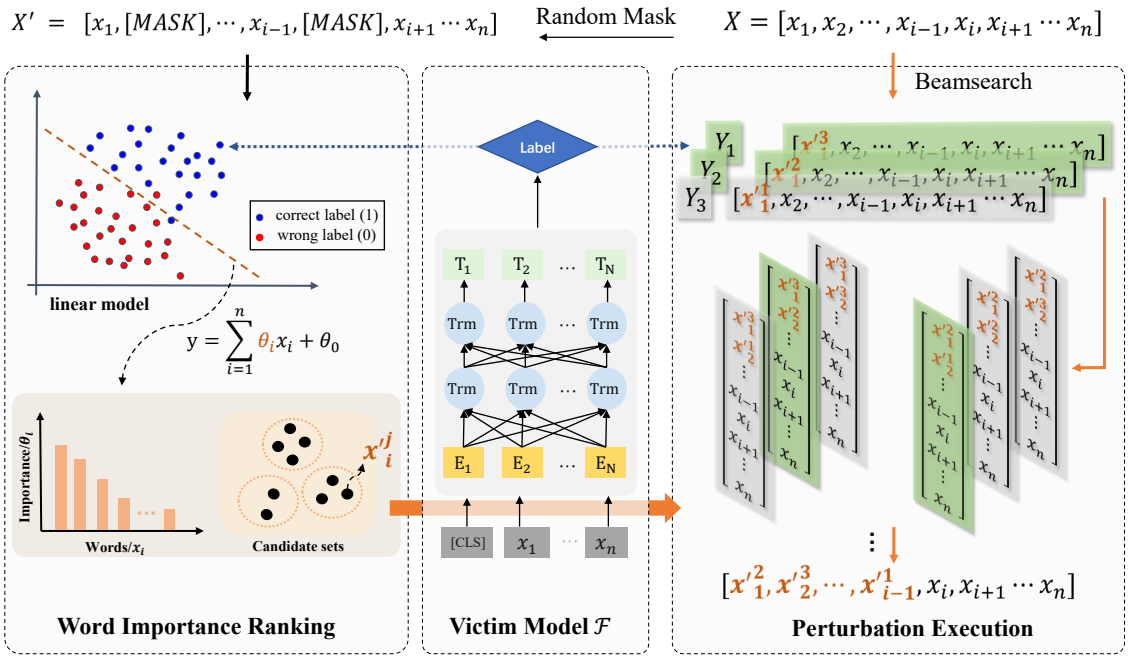

Figure 2: Overview of LimeAttack. It consists of two modules, *i.e.*, word importance ranking and perturbation execution. We first generate some neighborhood examples by masking some words in the benign sample, and then adopt linear model to approximate the importance of each word x_i. Then, we select candidate sets in the counter-fitted embedding space for each word. Finally, we adopt beam search (beam size $b = 2$ in the figure) to generate adversarial examples iteratively.

Finally, we calculate the optimal parameters $\boldsymbol{\theta}^*$:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{n} \pi(\boldsymbol{V}_i', \boldsymbol{V})\{\hat{\boldsymbol{Y}}_i' - g(\boldsymbol{X}_i')\}^2 + \Omega(\boldsymbol{\theta}) \quad (6)$$

where $\Omega(\boldsymbol{\theta})$ is the non-zero of parameters, which is a measure of the complexity of the linear model. After optimizing $\boldsymbol{\theta}$, the importance of each word x_i is equal to θ_i. LIME can be seen as an approximation of the model's decision boundary in the original sample. The parameters can be interpreted as the margin, the larger the margin, the larger the importance of this word in approximating the decision boundary. We will filter out stop words using NLTK[2] firstly and calculate the importance of each word. To ensure that LimeAttack has generated high-quality adversarial examples rather than just negative examples. We only adopt synonym replacement strategy and construct the synonym candidate set $\mathcal{C}(x_i)$ for each word x_i by selecting the top k nearest synonyms in the counter-fitted embedding space (Mrkšić et al. 2016).

Perturbation Execution. Adversarial examples generation is a combinatorial optimization problem. Score-based attack iterates by selecting the token that causes the greatest change in model's logits each time. But there is no such information in the hard-label attack. Therefore, we can only rely on the similarity between the adversarial sample and the original sample for iteration. The problem is that the similarity and attack success rate are not completely linearly correlated. As shown in the Table.7, greedily selecting the adversarial sample with the lowest similarity each time cannot ensure that the final attack success rate is optimal. We hope that each sampling is uniformly distributed to balance attack success rate and semantic similarity. For each origin word x_i, we replace it with $c \in \mathcal{C}(x_i)$ to generate an adversarial example $\boldsymbol{X}' = [x_1, \cdots, x_{i-1}, c, x_{i+1}, \cdots, x_n]$, then we calculate the semantic similarity between the benign sample X and the adversarial example $\boldsymbol{X}'$ by universal sentence encoder (USE)[3]. We first sort candidates by similarity and sample b adversarial examples each time to enter the next iteration. In detail, We have formulated the following sampling rules: (1) Sampling $\lfloor b/3 \rfloor$ adversarial examples with the highest semantic similarity. (2) Sampling $\lfloor b/3 \rfloor$ adversarial examples with the lowest semantic similarity. (3) Sampling $\lfloor b/3 \rfloor$ of the remaining adversarial samples randomly.

Experiments

Tasks, Datasets and Models

We adopt seven common datasets, such as MR (Pang and Lee 2005), SST-2 (Socher et al. 2013), AG (Zhang, Zhao, and LeCun 2015) and Yahoo (Yoo et al. 2020) for text classification. SNLI (Bowman et al. 2015) and MNLI (Williams, Nangia, and Bowman 2018) for textual entailment, where MNLI

[2]https://www.nltk.org/

[3]https://tfhub.dev/google/ universal-sentence-encoder

includes a matched version (MNLIm) and a mismatched version (MNLImm). In addition, we have trained three neural networks as victim models, including CNN (Kim 2014), LSTM (Hochreiter and Schmidhuber 1997) and BERT (Devlin et al. 2019).

Baselines

We have chosen the following existing hard-label attack algorithms as our baselines: HLBB (Maheshwary, Maheshwary, and Pudi 2021), TextHoaxer (Ye et al. 2022b), LeapAttack (Ye et al. 2022a) and TextHacker (Yu et al. 2022) as our baselines. Additionally, we have included some classic score-based attack algorithms, such as TextFooler (TF) (Jin et al. 2020), PWWS (Ma, Shi, and Guan 2020) and BertAttack (Li et al. 2020) for references, which obtain additional confidence scores for attacks and are implemented on the TextAttack framework (Morris et al. 2020).

Automatic Evaluation Metrics

We use four metrics to evaluate the attack performance: attack success rate (ASR), perturbation rate (Pert), semantic similarity (Sim) and query number (Query). Specifically, given a dataset $\mathcal{D} = \{(\boldsymbol{X}_i, \boldsymbol{Y}_i)\}_{i=1}^{N}$ consisting of N samples $\boldsymbol{X}_i$ and corresponding ground truth labels $\boldsymbol{Y}_i$, attack success rate of an adversarial attack method $\mathcal{A}$, which generates adversarial examples $\mathcal{A}(\boldsymbol{X})$ given an input $\boldsymbol{X}$ to attack a victim model $\mathcal{F}$, is defined as (Wang et al. 2021):

$$ASR = \sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \frac{\mathbb{I}[\mathcal{F}(\mathcal{A}(\boldsymbol{X})) \neq \boldsymbol{Y}]}{|\mathcal{D}|} \tag{7}$$

The perturbation rate is the proportion of the number of substitutions to the number of original tokens, which has been defined in Eq 2. The semantic similarity is measured by the Universal Sentence Encoder (USE). Most papers (Maheshwary, Maheshwary, and Pudi 2021; Ye et al. 2022a) have adopted USE. In order to maintain consistency and facilitate comparability, we have also utilized USE. Query number is the number of model queries during the attack. The robustness of a model is inversely proportional to the attack success rate, while the perturbation rate and semantic similarity together reveal the quality of adversarial examples. Query number reveals the attack efficiency.

Implementation Details

We set the kernel width $\sigma = 25$, the number of neighborhood samples equal to the number of the benign sample's tokens, and the beam size $b = 10$. For a fair comparison, all baselines follow the same settings: synonyms are selected from counter-fitted embedding space and the number of each candidate set $k = 50$, the same 1000 texts are sampled for baselines to attack. The results are averaged on five runs with different seeds (1234,2234,3234,4234 and 5234) to eliminate randomness. In order to improve the quality of adversarial examples, the attack succeeds if the perturbation rate of each adversarial example is less than 10%. We set a tiny query budget of 100 for hard-label attack, which corresponds to real-world settings. (*e.g.*, The HuggingFace free Inference API typically limits calls to 200 times per minute.)

Experiments Results

Attack Performance. Table 1 and 2 show that LimeAttack outperforms existing hard-label attacks on text classification and textual entailment tasks, achieving higher attack success rates and lower perturbation rates in datasets such as SST-2, AG, and MNLI. Unlike existing hard-label attacks that require many queries to optimize the perturbation, LimeAttack adopts a local explainable method to calculate word importance ranking and attacks key words first. This approach can generate adversarial examples with a high attack success rate, even under tiny query budgets.

Query Budget. As illustrated in Figure 3, LimeAttack still maintains a stable attack success rate and a smoother attack curve under different query budgets, which means that regardless of high or low query budget, LimeAttack often have a stable and excellent attack performance. Comparing the attack performance in low query and high query budgets can provide a more comprehensive evaluation. However, attack without considering the query budget is more of an ideal situation, it shows the upper limit of an attack algorithm. A large number of queries are expensive, we believe attack performance under low query budget is more practical.

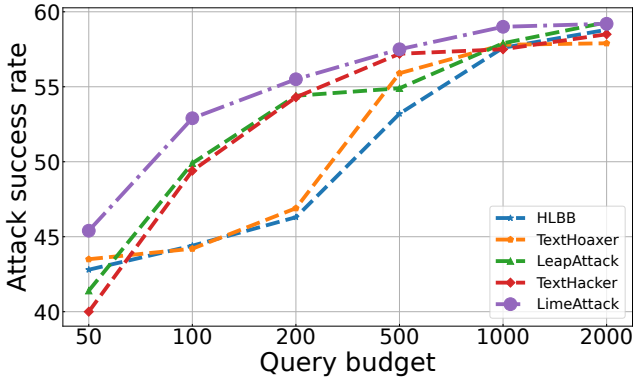

Figure 3: Attack success rate of different attacks under different query budgets on CNN-MR.

Adversary Quality. High-quality adversarial examples should be both fluent and context-aware, while also being similar to benign samples to evade human detection. We utilize Language-Tools[4] and USE to detect grammatical errors and measure semantic similarity. As shown in Table 3, LimeAttack has the lowest perturbation rate and grammatical error, though its semantic similarity is lower than HLBB, TextHoaxer, and LeapAttack. Because these methods take the similarity into account during the attack, thus LimeAttack exhibits lower similarity than other methods. Considering all metrics, LimeAttack is still dominant.

Evaluation on Large Language Models. Large language models (LLMs), also known as foundation models (Bommasani et al. 2021), have achieved impressive performance on various natural language processing tasks. However, their robustness to adversarial examples remains unclear (Wang et al.

[4]https://www.languagetool.org/

Model	Attack	MR		SST-2		AG		Yahoo	
		ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓
CNN	HLBB	44.4	5.4	33.4	**5.6**	17.7	3.3	41.8	3.6
	TextHoaxer	44.2	**5.2**	38.1	**5.6**	15.7	**2.9**	39.9	**3.3**
	LeapAttack	43.1	5.3	40.0	5.7	20.2	3.2	40.4	3.4
	TextHacker	49.4	6.2	38.1	6.3	20.5	6.2	38.1	5.9
	LimeAttack	**49.9**	5.3	**42.8**	**5.6**	**20.9**	**2.9**	**43.7**	3.7
LSTM	HLBB	41.2	**5.2**	33.1	5.7	15.2	3.1	38.4	**3.3**
	TextHoaxer	39.3	5.4	36.4	5.6	14.7	**2.7**	37.1	**3.3**
	LeapAttack	40.0	5.3	39.8	5.6	15.9	3.1	37.6	**3.3**
	TextHacker	45.8	6.1	35.2	6.4	16.5	6.2	36.8	5.9
	LimeAttack	**47.6**	5.4	**40.1**	**5.5**	**17.3**	**2.7**	**40.3**	3.7
BERT	HLBB	26.6	5.6	23.0	5.8	12.7	3.2	36.3	3.6
	TextHoaxer	27.0	5.5	24.9	5.8	9.8	3.0	32.7	**3.3**
	LeapAttack	26.5	**5.4**	26.1	5.8	13.7	**2.9**	34.1	3.4
	TextHacker	26.5	6.5	25.4	6.3	12.9	5.5	31.3	6.3
	LimeAttack	**29.2**	5.9	**27.8**	**5.7**	**14.6**	**2.9**	**37.4**	3.8

Table 1: The attack success rate (ASR.,%↑) and perturbation rate (Pert.,%↓) of different hard-label attack algorithms on three models for text classification under a query budget of 100.

Dataset	HLBB		TextHoaxer		LeapAttack		TextHacker		LimeAttack	
	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓
SNLI	24.9	**8.3**	24.7	**8.3**	28.3	**8.3**	22.8	**8.3**	**29.1**	8.4
MNLIm	41.9	7.8	40.9	**7.7**	49.1	**7.7**	38.2	7.8	**49.7**	**7.7**
MNLImm	47.8	**7.5**	45.6	7.6	56.0	7.6	44.3	7.7	**56.3**	7.6

Table 2: The ASR.,%↑ and Pert.,%↓ of LimeAttack and other baselines on BERT for textual entailment under a query budget of 100.

Attack	ASR.↑	Pert.↓	Sim.↑	Gram.↓
HLBB	23.0	5.8	99.2	1.6
TextHoaxer	24.9	5.8	**99.2**	1.7
LeapAttack	26.1	5.8	99.1	1.5
TextHacker	25.4	6.3	96.0	1.9
LimeAttack	**27.8**	**5.7**	96.4	**1.5**

Table 3: ASR.,%↑, Pert.,%↓, Sim.,%↑ and Gram.,↓ of different hard-label attack algorithms on SST-2 dataset for BERT under query budget of 100.

Model(size)	ASR.↑	Pert.↓	Sim.↑	Acc.↑
BART-L (407M)	42.0	5.15	93.7	87.0
DeBERTa-L (435M)	52.0	5.82	92.9	79.0
T5-L (780M)	28.0	5.59	95.1	**93.0**
GPT3(175B)	**61.0**	**4.82**	95.2	82.0
ChatGPT (175B)	25.0	5.62	**95.3**	92.0

Table 4: The evaluation of LimeAttack on large language models. We attack these large language models on MR dataset under query budget of 100.

2023). To evaluate the effectiveness of LimeAttack on LLMs, we select some popular models such as DeBERTa-L (Kojima et al. 2022), BART-L (Lewis et al. 2019), Flan-T5 (Raffel et al. 2020), GPT-3 (text-davinci-003) and ChatGPT (gpt-3.5-turbo) (Brown et al. 2020). Due to the limited API calls, we sample 100 texts from MR datasets and attacked the zero-shot classification task of these models. As Table 4 shows, LimeAttack successfully attacked most LLMs under tight query budgets. Although these models have high accuracy on zero-shot tasks, their robustness to adversarial examples still needs to be improved. ChatGPT and T5-L are more robust to adversarial examples. The robustness of the victim model is related to origin accuracy. The higher the origin accuracy, the stronger the victim model's ability to defense adversarial examples.

Attack Performance on Defense Methods. To evaluate the effectiveness of LimeAttack on defense methods, we use A2T (Yoo and Qi 2021) and ASCC (Dong et al. 2021) to enhance the defense ability of BERT on SNLI, and conducted attack experiments on this basis. As shown in Table 5, LimeAttack still has a certain attack effect and outcomes

Defense Method	HLBB		TextHoaxer		LeapAttack		TextHacker		LimeAttack	
	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓	ASR.↑	Pert.↓
None	24.9	**8.3**	24.7	**8.3**	28.3	**8.3**	22.8	**8.3**	**29.1**	8.4
A2T	20.6	9.3	21.4	**9.5**	23.5	**9.4**	19.8	**9.1**	**24.5**	**9.4**
ASCC	13.2	**6.5**	13.4	6.5	14.3	**6.4**	12.5	7.2	**15.8**	6.7

Table 5: The evaluation of hard-label attacks on defense methods based on BERT-SNLI under query budget of 100.

other baselines on these defense methods.

Ablation Study

Effect of Word Importance Ranking. To validate the effectiveness of word importance ranking, we removed the word importance ranking strategy and instead randomly selected words to perturb to evaluate its effectiveness. Table 6 shows that without the word importance ranking, the attack success rate decreased by 9% and 6% on the MR and SST-2 datasets, respectively. Furthermore, adversarial examples generated by random selection had higher perturbation rates and required more queries. This indicates the importance of the word importance ranking in guiding LimeAttack to focus on crucial words, leading to a more efficient attack with lower perturbation rates.

Effect of Sampling Rules. To verify the effectiveness of LimeAttack's sampling rules, we will replace this strategy with one of three common sampling rules: (1) selecting b adversarial examples with the highest semantic similarity, (2) selecting b adversarial examples with the lowest semantic similarity, or (3) randomly selecting b adversarial examples. The results in Table 7 show that LimeAttack outperforms other sampling rules with a higher attack success rate and lower perturbation rate. Additionally, it has a comparable (second highest) semantic similarity and number of queries.

	MR		SST-2	
	Random	LIME	Random	LIME
Pert.↓	6.1	**5.6**	6.4	**5.9**
ASR.↑	30.1	**39.3**	32.1	**36.5**
Sim.↑	94.6	**94.8**	94.2	**94.6**
Query.↓	157.2	**153.3**	148.1	**132.5**

Table 6: Comparison between word importance ranking learned by LIME and random selecting for BERT under query budget of 1000.

Human Evaluation

3We selected 200 adversarial examples BERT-MR. Each adversarial example was evaluated by two human judges for semantic similarity, fluency and prediction accuracy. The entire human evaluation is consistent with TextFooler (Jin et al. 2020). In detail, we ask human judges to put a 5-point Likart scale (1-5 corresponds to very not fluent/similar, not fluent/similar, uncertain, fluent/similar, very fluent/similar

Sample Rule	ASR.↑	Pert.↓	Sim.↑	Query.↓
Method 1	35.8	5.76	**95.02**	164.65
Method 2	31.5	6.13	93.79	**87.45**
Method 3	32.1	6.09	94.50	107.05
LimeAttack	**39.3**	**5.65**	94.81	153.03

Table 7: Comparison between different sample rules on MR dataset for BERT under query budget of 1000.

respectively) to evaluate the the similarity and fluency of adversarial examples and benign samples. The results are listed in the Table 8, semantic similarity is 4.5, which means adversarial samples are similar to original sample. The prediction accuracy here is to make humans to predict what the label of this sentence is (such as it is positive or negative for sentiment analysis). 76.7% means majorities of adversarial examples have the same attribute as original samples from humans' perspective but mistake victim model.

	Ori	Adv
Prediction Accuracy	81.2%	76.7%
Fluency	4.4	4.1
Semantic Similarity	4.5	

Table 8: The semantic similarity, fluency and prediction accuracy of original texts and adversarial examples evaluated by human judges for BERT-MR.

Conclusion

In this work, we summarize the previous score-based attacks and hard-label attacks and propose a novel hard-label attack algorithm called LimeAttack. LimeAttack adopts a local explainable method to approximate the word importance ranking, and then utilizes beam search to generate high-quality adversarial examples with tiny query budget. Experiments show that LimeAttack achieves a higher attack success rate than other hard-label attacks. In addition, we have evaluated LimeAttack's attack performance on large language models and some defense methods. The adversarial examples crafted by LimeAttack are high-quality, high transferable and improves victim model's robustness in adversarial training. LimeAttack has verified the effectiveness of inside-to-outside attack path in hard-label. Then many excellent score-based attacks may provide hard-label attacks more insight.

References

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 632–642.

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, 1877–1901.

Carlini, N.; and Wagner, D. 2018. Audio adversarial examples: Targeted attacks on speech-to-text. In *2018 IEEE Security and Privacy Workshops*, 1–7.

Chai, Y.; Liang, R.; Samtani, S.; Zhu, H.; Wang, M.; Liu, Y.; and Jiang, Y. 2023. Additive Feature Attribution Explainable Methods to Craft Adversarial Attacks for Text Classification and Text Regression. *IEEE Transactions on Knowledge and Data Engineering*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4171–4186.

Dong, X.; Luu, A. T.; Ji, R.; and Liu, H. 2021. Towards robustness against natural language word substitutions. *International Conference on Learning Representations*.

Goodman, D.; Zhonghou, L.; et al. 2020. FastWordBug: A fast method to generate adversarial text against NLP applications. *arXiv preprint arXiv:2002.00760*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. In *Neural Computation*, volume 9, 1735–1780.

Jiang, H.; He, P.; Chen, W.; Liu, X.; Gao, J.; and Zhao, T. 2020. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2177–2190.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 2020. 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, 8018–8025.

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

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2018. Adversarial examples in the physical world. In *Artificial Intelligence Safety and Security*, 99–112.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, L.; Ma, R.; Guo, Q.; Xue, X.; and Qiu, X. 2020. BERT-ATTACK: Adversarial Attack Against BERT Using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 6193–6202.

Lundberg, S. M.; and Lee, S.-I. 2017. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*.

Ma, G.; Shi, L.; and Guan, Z. 2020. Adversarial Text Generation via Probability Determined Word Saliency. In *International Conference on Machine Learning for Cyber Security*, 562–571.

Maheshwary, R.; Maheshwary, S.; and Pudi, V. 2021. Generating natural language attacks in a hard label black box setting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 13525–13533.

Minaee, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlu, M.; and Gao, J. 2021. Deep learning–based text classification: a comprehensive review. In *ACM Computing Surveys*, volume 54, 1–40.

Morris, J.; Lifland, E.; Yoo, J. Y.; Grigsby, J.; Jin, D.; and Qi, Y. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, 119–126.

Mrkšić, N.; Ó Séaghdha, D.; Thomson, B.; Gašić, M.; Rojas-Barahona, L. M.; Su, P.-H.; Vandyke, D.; Wen, T.-H.; and Young, S. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 142–148.

Pang, B.; and Lee, L. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 115–124.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. In *The Journal of Machine Learning Research*, volume 21, 5485–5551.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international*

conference on knowledge discovery and data mining, 1135–1144.

Shrikumar, A.; Greenside, P.; Shcherbina, A.; and Kundaje, A. 2016. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*.

Socher, R.; Perelygin, A.; Wu, J.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 1631–1642.

Štrumbelj, E.; and Kononenko, I. 2014. Explaining prediction models and individual predictions with feature contributions. In *Knowledge and information systems*, volume 41, 647–665.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Wang, B.; Xu, C.; Wang, S.; Gan, Z.; Cheng, Y.; Gao, J.; Awadallah, A. H.; and Li, B. 2021. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*.

Wang, J.; Hu, X.; Hou, W.; Chen, H.; Zheng, R.; Wang, Y.; Yang, L.; Huang, H.; Ye, W.; Geng, X.; et al. 2023. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. In *arXiv preprint arXiv:2302.12095*.

Williams, A.; Nangia, N.; and Bowman, S. 2018. A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1112–1122.

Yang, X.; Liu, W.; Tao, D.; and Liu, W. 2021. BESA: BERT-based Simulated Annealing for Adversarial Text Attacks. In *International Joint Conference on Artificial Intelligence*, 3293–3299.

Ye, M.; Chen, J.; Miao, C.; Wang, T.; and Ma, F. 2022a. 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*, 2307–2315.

Ye, M.; Miao, C.; Wang, T.; and Ma, F. 2022b. TextHoaxer: Budgeted Hard-Label Adversarial Attacks on Text. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 3877–3884.

Yoo, J. Y.; Morris, J.; Lifland, E.; and Qi, Y. 2020. Searching for a Search Method: Benchmarking Search Algorithms for Generating NLP Adversarial Examples. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, 323–332.

Yoo, J. Y.; and Qi, Y. 2021. Towards Improving Adversarial Training of NLP Models. *Findings of the Association for Computational Linguistics: EMNLP*.

Yu, Z.; Wang, X.; Che, W.; and He, K. 2022. Learning-based Hybrid Local Search for the Hard-label Textual Attack. *arXiv preprint arXiv:2201.08193*.

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

Zhu, H.; Zhao, Q.; and Wu, Y. 2023. BeamAttack: Generating High-quality Textual Adversarial Examples through Beam Search and Mixed Semantic Spaces. *arXiv preprint arXiv:2303.07199*.