Evaluating Adversarial Defense in the Era of Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) have demonstrated superior performance in many natural language processing tasks. Existing works have shown that LLMs are not robust to adversarial attacks, questioning the applicability of these models in scenarios with safety concerns. However, one key aspect that has been overlooked is evaluating and developing defense mechanisms against adversarial attacks. In this work, we systematically study how LLMs react to different adversarial defense strategies. We also propose defenses tailored for LLMs that can significantly improve their robustness: First, we develop prompting methods to alert the LLM about potential adversarial contents; Second, we use neural models such as the LLM itself for typo correction; Third, we propose an effective fine-tuning scheme to improve robustness against corrupted inputs. Extensive experiments are conducted to evaluate the adversarial defense approaches. We show that by using the proposed defenses, robustness of LLMs can increase by up to 20%. Our code will be publicly available.

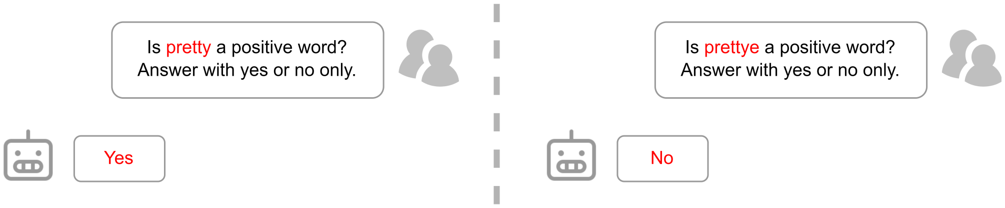
1 Introduction

Large language models (LLMs) such as the decoder-only GPT family (Chowdhery et al., 2022; OpenAI, 2023; Touvron et al., 2023a;b) and the encoder-decoder T5 family (Raffel et al., 2020; Chung et al., 2022) have demonstrated superior performance in various tasks, including natural language understanding (Chung et al., 2022), dialogue generation (Bubeck et al., 2023), logical reasoning (Bang et al., 2023), and even solving mathematical problems (Frieder et al., 2023). These models contain billions of parameters, and their emergent abilities (Wei et al., 2022) facilitate effective zero-shot learning. That is, LLMs are adept in performing tasks in different fields (Choi et al., 2023) using proper prompts without prior exposure to task-specific data.

LLMs are double-edged swords. Despite the successful applications, these models are not *robust*. Figure 1 demonstrates behavior of ChatGPT under input typos. In the example, ChatGPT is asked whether the word “pretty” is of positive sentiment. Without any typos, the LLM correctly answers the question. However, when there is a typo in the input, i.e., “prettye” instead of “pretty”, ChatGPT draws an opposite conclusion. The lack of robustness issue exists beyond typos in input data. For example, prompts are also subject to attack: we can inject backdoor triggers (Xu et al., 2022) or adversarial demonstrations (Wang et al., 2023b) into prompts to trick LLMs in drawing wrong conclusions. These findings raise a serious safety concern: can LLMs be reliably used?

We consider plausible scenarios in real-world applications: **input typos**. In practice, user inputs are often noisy and contain typos, which is undesirable for applications such as dense retrieval and search. For example, Zhuang & Zuccon (2021; 2022) show that retrieval recall can decrease by more than 20% when dealing with input typos. The study of robustness plays a crucial role in mitigating such a performance drop. In this work, we simulate input typos via character-level adversarial attacks (see Figure 1 as an example), and we term the attacks **adversarial typos**. The adversarial typos represent the *worst case input typos*, since they are intentionally created to fool the LLMs. We remark that there are other types of adversarial inputs, such as distracting samples (Wang et al., 2021) and automatically computed adversarial samples (Goodfellow et al., 2015). However, most of these adversarial attacks are not human interpretable and implausible in actual applications.

In this work, we focus on evaluating and developing **defense mechanisms** against adversarial typos, which is a key aspect overlooked by existing works on benchmarking robustness of LLMs (Chen


Is pretty a positive word? Answer with yes or no only.

Yes

Is prettye a positive word? Answer with yes or no only.

No

Figure 1: Behavior of ChatGPT under user input typos.

et al., 2023; Wang et al., 2023a). In practice, somehow surprisingly, we observe that many existing adversarial defense methods do not work well for LLMs (e.g., the rule-based defenses in Table 3 in the experiments). Therefore, adversarial defenses that are tailored for LLMs are needed.

Adversarial defense methods fall into two categories: **black-box defenses** and **white-box defenses**. The former describes the scenario where we do not have access to the model weights (e.g, GPT-4); while in the latter case, we have full access to the model (e.g., Llama).

For black-box defenses, we treat defense models as pre-processors: given a potentially corrupted input, we first run the pre-processor, and then feed the processed results to the LLM. We evaluate and develop several categories of methods: 1) *Rule-based defense* methods employ clustering algorithms to learn mapping rules, which are used to convert potential typos (Jones et al., 2020). For example, if the word "hallo" is in the cluster centered around the word "hello", then all the "hallo" in the inputs will be converted to "hello". 2) *Prefix defense* is a simple strategy where we modify prompts to alert LLMs about potentially adversarial contents. 3) *Neural network defense* methods are models trained to denoise typos. For example, Pruthi et al. (2019) train a sequence-to-sequence model to spell-check inputs. 4) *Self defense* methods are similar to the neural network defense, except that the same LLM is used for both typo correction and inference.

We also evaluate white-box defense approaches. With the development of LLMs, more models are becoming fully available (e.g., Llama). This paves the way for designing stronger defense techniques by utilizing the model weights. Specifically, we fine-tune LLMs on both clean data and corrupted (adversarial) data. The intuition is that once exposed to adversarial typos, models should yield better robustness against them.

We evaluate effectiveness of defense approaches on two families of LLMs: decoder-only models such as Llama, and encoder-decoder models such as Flan (Chung et al., 2022). There has long been debates about the suitable structures of Transformer-based models (Vaswani et al., 2017), even before the era of LLMs. For example, the discussion about BERT vs. GPT vs. T5 continued until the advancement of ChatGPT. In this work, we systematically investigate how different LLMs behave when facing adversarial typos and defense mechanisms.

In summary, in this work we consider input typos, a practical but under-explored scenario that hinders performance of LLMs. We systematically study how LLMs react to different defense approaches against adversarial typos. We summarize our contributions and findings as follows:

- ◇ We can adopt the LLM itself or train another smaller model to denoise the typos. This black-box defense approach improves robustness of all LLMs.
- ◇ We can modify the prompt to alert the LLM in paying attention to potential adversarial contents. Such a black-box defense strategy improves robustness when the LLM is adept in following human instructions, e.g., Llama2-Chat.
- ◇ For white-box defenses, we can fine-tune the LLM on both clean and corrupted data. This simple strategy is extremely effective in improving model robustness.

2 Background

2.1 Large Language Models

Large language models have become the de facto standard solution for many natural language processing tasks. These models are a step forward from smaller language models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), DeBERTa (He et al., 2021b;a) and T5 (Raffel et al., 2020). LLMs have extremely large number of parameters compared with their predecessors. For example, DeBERTa-xxl, the largest publicly available encoder-only model in its series, contains about 1.3B parameters, whereas LLMs rarely have less than 10B parameters.

The debate about the best architecture of Transformer-based models (Vaswani et al., 2017) continues in the era of LLMs. One commonly used structure is the decoder-only Transformer. The most famous example is the GPT series: starting from GPT (Radford et al., 2018), GPT-2 Radford et al. (2019), GPT-3 (Brown et al., 2020) and InstructGPT (Ouyang et al., 2022), the series has now progressed to the evolutionary GPT-4 (OpenAI, 2023). Other examples include Llama (Touvron et al., 2023a;b), PaLM (Chowdhery et al., 2022), OPT (Zhang et al., 2022) and BLOOM (Scao et al., 2022). Another widely applied structure is the encoder-decoder Transformer, evolving from T5 (Raffel et al., 2020) and BART (Lewis et al., 2020) to the Flan family (Chung et al., 2022) including Flan-T5 and Flan-ul2. Many existing works on benchmarking robustness of LLMs focus on the decoder-only GPT family (Wang et al., 2023a; Chen et al., 2023). We bridge this gap by evaluating both decoder-only and encoder-decoder models.

2.2 Adversarial Attacks

In this work, we focus on models' robustness to input typos, which is a practical scenario that hinders model performance in real-world applications such as dense retrieval and search (Zhuang & Zuccon, 2021; 2022). Specifically, we use character-level adversarial attacks to simulate the *worse case input typos*. That is, the modifications to the input (i.e., the typos) are intentionally designed so that the LLM draws wrong conclusions. Existing works have designed other character level (Belinkov & Bisk, 2018; Gao et al., 2018; Eger et al., 2019), word level (Papernot et al., 2016; Zhao et al., 2018; Alzantot et al., 2018) and sentence level (Iyyer et al., 2018) attacks to mislead the models.

We note that in parallel to typos in the input data, LLMs are also not robust to adversarially constructed prompts (Xu et al., 2022; Wu & Shi, 2022; Si et al., 2022; Zhu et al., 2023) and demonstrations (Wang et al., 2023b).

2.3 Adversarial Defenses

Adversarial defenses fall into two categories: black-box defenses and white-box defenses. In the black-box scenario, we do not have access to model weights (e.g., GPT-4), but we can only observe the outputs of the model. In the white-box scenario, we have full access to the model (e.g., Llama), namely gradient-based defense methods are feasible in this case.

In black-box defenses, defense models are usually treated as pre-processors (Gong et al., 2019; Jones et al., 2020). That is, we first run the defense model to denoise the potentially corrupted input, and then the denoised result is passed to the downstream model for inference.

The white-box defense scheme brings more possibility when designing algorithms since we have full control of the model. However, many existing works define adversarial samples in the continuous embedding space instead of the discrete input space, such that the attacks are not human interpretable (Liu et al., 2020; Cheng et al., 2021; Zuo et al., 2021a;b). Instead, we focus on inputs typos that are plausible in real-world applications.

3 Setups, Attacks, and Defenses

3.1 Datasets, Models, and Inference

Datasets. We adopt six datasets from the GLUE benchmark (Wang et al., 2019). For each input sample, we first construct corresponding adversarial samples to simulate input typos. Then, we evaluate LLMs' performance on both the clean data and the corrupted data.

⋄ *RTE* (Dagan et al., 2006; Bar-Haim et al., 2006; Giampiccolo et al., 2007; Bentivogli et al., 2009) is a fusion of several annual textual entailment challenges on news articles and headlines. The corresponding task is to determine, given a pair of sentences, whether the meaning of the second one can be inferred from the first one.

◇ *MRPC* (Dolan & Brockett, 2005) is a set of sentence pairs extracted from news articles. The underlying task is to determine whether two sentences are semantically equivalent.

◇ *SST-2* (Socher et al., 2013) is a corpus of sentences extracted from online movie reviews, and we need to classify each review as positive or negative.

◇ *QNLI* (Rajpurkar et al., 2016) is a binary question entailment task. Based on a question-sentence pair, the goal is to determine whether the sentence provides a suitable answer to the question.

◇ *QQP* (Wang et al., 2019) is a collection of question pairs collected from Quora. The associated task is to determine whether two given questions are paraphrases of each other.

◇ *MNLI* (Williams et al., 2018) is a crowd-sourced collection of sentence pairs with textual entailment annotations. Given a premise and a hypothesise, the language task is to determine whether the premise entails, contradicts or is neutral with respect to the hypothesis.

Models. We adopt two families of Transformer-based (Vaswani et al., 2017) LLMs covering both decoder-only and encoder-decoder architectures.

◇ *Llama2-Chat* (Touvron et al., 2023b) is a decoder-only model. It is the "chat" version of Llama2 that is trained to follow human instructions (i.e., prompts). We consider two mode sizes: 7B and 13B. Different from the previous version (Touvron et al., 2023a) and many closed-source models, the training of Llama2-Chat emphasizes on model safety.

◇ *Flan-T5* (Chung et al., 2022) is an encoder-decoder model. We adopt two sizes of Flan-T5: 3B and 11B. We note that Flan-T5 is multi-task instruction fine-tuned on a large number of datasets similar to the natural language understanding tasks we consider.

Inference. To evaluate black-box defense methods, we inference LLMs in a zero-shot setting. We list the prompts we adopted to inference Flan-T5 and Llama2-Chat in Appendix A. In the prompts, we instruct the models to choose from a set of possible answers. For example, we instruct the model to choose from *[positive, negative]* for sentiment classification tasks such as SST-2.

3.2 Adversarial Typos

We focus on input typos, which is a practical scenario that hinders model performance in real-world applications. In more details, we simulate *worst case input typos* using character-level adversarial attacks. Because our goal is to simulate plausible inputs in actual applications, the corrupted samples should be human interpretable. To facilitate this, for each input sentence, we modify at most 4 words; where in each word, we modify at most one character. The character-level modification can be *insertion*, *deletion* and *substitution*.

Algorithm 1: Simulate input typos via character-level adversarial attack.

```
Input: Input sample $(x, y)$, where $x$ is the input with $L$ words $x = (x_1, \cdots, x_L)$ and $y$ is the
       label; Maximum number of words to change $N_{\text{change}}$; Maximum number of tries $N_{\text{try}}$.
for $t = 1, \cdots, N_{change}$ do
    Randomly select a word $x_n$ from the sentence $x$ that has not been changed yet;
    for $i = 1, \cdots, N_{try}$ do
        Change $x_n$ to $x_n^i$ by randomly perturbing one character in $x_n$;
        Replace word $x_n$ in sentence $x$ by $x_n^i$, call the corrupted sentence $x^i$;
        Record $p(y|x^i)$, the probability of the ground-truth label $y$;
    Update the sentence $x \leftarrow x^i$, where $i$ is selected that yields the lowest probability $p(y|x^i)$;
Output: The corrupted sentence.
```

Algorithm 1 summarizes the attack algorithm we employed. Note that we access the probability of the ground-truth label as a metric to select attacks. For example, in SST-2, the labels are either *positive* or *negative*. Then for a sentence x, we retrieve two logits (or logprobs) corresponding to the LLM generating *positive* and *negative*, respectively; after which the logits are normalized to obtain $p(\textit{positive}|x)$ and $p(\textit{negative}|x)$. The above procedure is straightforward for open-source LLMs;

and for closed-source LLMs such as GPT-4, publicly available APIs are also provided to access the logprob of the most likely tokens.

Table 1 demonstrates an example of constructing input typos. Initially, the LLM outputs the correct answer with very high confidence. However, after three minor modifications to the text, the LLM makes a wrong prediction with high confidence.

Table 1: An example of constructing a corrupted sample for Flan-T5-3B. *Action* is the action taken in the previous step; *Confidence* is the probability of the ground-truth label (see Algorithm 1).

Input	**Action**	**Prediction**	**Label**	**Confidence**
that's pure pr hype	—	negative	negative	1.0
tha~~t~~'s pure pr hype	deletion	negative	negative	1.0
tha~~t~~'cs pure pr hype	insertion	negative	negative	0.7
tha~~t~~'cs pure pr hyp~~e~~	deletion	positive	negative	0.3

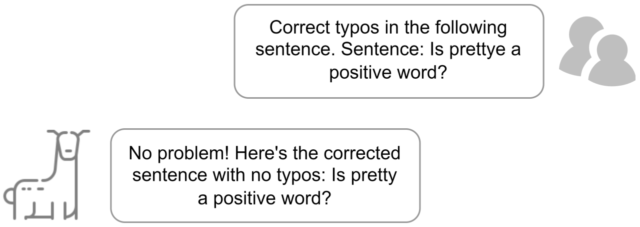

Correct typos in the following sentence. Sentence: Is prettye a positive word?
No problem! Here's the corrected sentence with no typos: Is pretty a positive word?

Figure 2: An example of using Llama for typo correction.

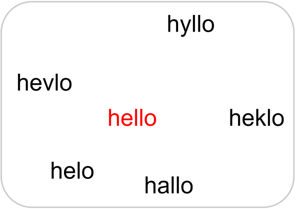


Figure 3: An illustration of a rule-based word cluster.

3.3 Black-Box Defenses

In the black-box scenario, defense methods are usually pre-processors. That is, given a potentially corrupted input, we first run the pre-processor to denoise the input, and the result is subsequently fed to the LLM. We consider several categories of black-box defense methods:

◇ **Rule-based defense.** These methods do not use neural networks for denoising purposes. Instead, they learn mapping rules to convert typos. In this work, we use the most representative rule-based defense: Robust Encoding (Jones et al., 2020). The underlying idea is to systematically assign each word a class representative, and subsequently replace each word in the input by its corresponding representative. An example of a word cluster is shown in Figure 3. Note that in the example, "hello" is the center of the cluster, such that all the other words in the cluster (e.g., "hallo") will be converted to "hello" when running the defense algorithm. Clusters are built based on word frequencies and edit-distance (the number of edits), and exist in two distinct variants.

- *Rule-CC*: Clusters are derived from a connected component clustering on the vocabulary graph, where edges are drawn between edit-distance one words.
- *Rule-Agg*: Clusters are built in an agglomerative manner to balance stability (inclusion of most typos in clusters) and fidelity (consistency within clusters).

It is worth pointing out that there are fewer clusters than words, implying that two real words might be associated to the same representative. Therefore, rule-based defenses may change the meaning of the input (see Table 4 in the experiments as an example).

◇ **Prefix defense.** We modify the prompts in Appendix A by prepending a warning: *In the following question, please pay close attention to typos before answering*. Even though typos will not be directly corrected using this simple strategy, the additional instruction can alert the LLM about potential adversarial contents.

◇ **Neural network (NN) defense.** In this approach, a BART-base (Lewis et al., 2020) model is trained for denoising purposes. BART-base is a sequence-to-sequence Transformer model that contains about 140M parameters, making it much lighter than LLMs. The defense model is trained to recover the initial text (e.g., *pretty*) from the corrupted text (e.g., *prettye*).

◇ **Self defense.** LLMs are extremely powerful and can fulfill most tasks if properly prompted. We investigate whether we can use LLMs themselves to defend against adversarial typos. Figure 2 illustrates that Llama can successfully identify and correct typos. Therefore, in this approach, we inference a LLM twice for a potentially corrupted input. In the first round, we ask the LLM to correct typos, and then we inference the LLM again using the typo-corrected sentence.

3.4 White-Box Defenses

The white-box scenario brings more possibility since we have full control of the model. Existing works (Liu et al., 2020; Cheng et al., 2021) rely on the concept of adversarial regularization to improve model robustness. Specifically, adversarial samples in the continuous embedding space are constructed to augment the training data. However, construction of the adversarial samples is extremely slow since multiple forward/backward passes are needed. Such a computational burden is even more severe in the era of LLMs.

We adopt a much simpler yet effective strategy for white-box defense: we fine-tune the LLM on both clean data and corrupted data. The intuition is that once exposed to data with typos, the LLM should recognize the typos' patterns and become robust to them.

4 Experiments

In all the experiments, for a clean input sentence, we use Algorithm 1 to build a corrupted version of the sentence. We note that the corrupted sentences are model-specific, i.e., the same input sentence will be attacked differently for each model. For black-box defense methods, we first denoise the corrupted sentence by running defense model, and then we feed the denoised output to the LLM to generate the final prediction. For white-box defenses, we directly inference the LLM since it is fine-tuned to be robust to adversarial typos.

4.1 Robustness of LLMs

Table 2: Experimental results of Flan-T5 and Llama2-Chat. In the results, *corrupted* means models are evaluated on corrupted data, and *clean* means models are evaluated on clean data without typos.

	Flan-T5-3B		**Flan-T5-11B**		**Llama2-Chat-7B**		**Llama2-Chat-13B**	
	clean	corrupted	clean	corrupted	clean	corrupted	clean	corrupted
RTE	93.1	81.2	89.5	81.5	77.5	65.6	76.8	66.3
MRPC	82.3	71.9	82.6	73.6	65.0	62.2	69.7	38.4
SST-2	94.8	81.9	96.1	86.2	94.1	77.2	94.8	78.4
QNLI	94.7	86.7	94.5	87.9	76.6	57.5	73.6	54.8
QQP	90.9	73.4	88.7	75.8	58.6	34.9	71.4	55.6
MNLI	91.8	72.9	90.3	75.4	51.1	31.0	50.5	38.9
Average	91.3	78.0	90.3	80.1	70.5	54.7	72.8	55.4

Table 2 shows experimental results on two versions of Flan-T5 and two versions of Llama2-Chat. We report evaluation results on both the clean data without typos and the corrupted data with typos. From the results, we observe the following regarding model robustness:

- ◇ LLMs are not robust to adversarial typos. For example, average performance of Flan-T5-11B decreases by 10.2%, and that of Llama2-Chat-13B decreases by 17.4%. The results show that the attack algorithm in Algorithm 1 can successfully identify cases where even small changes to the inputs can drastically change the outputs.
- ◇ Flan-T5 models are more robust than Llama2-Chat models. For example, average performance of Flan-T5-3B drops by 13.3%, and average performance of Llama2-Chat-7B drops by 15.8% when evaluating on the corrupted data.

We also observe that larger model sizes do not necessarily translate to better performance on all the tasks. For example, on the SST-2 dataset without typos, performance of Llama2-Chat-7B is only

0.7% lower than Llama2-Chat-13B (94.1 vs. 94.8), although the former is about two times smaller. Also, we see that performance of Flan-T5-3B is 91.3% averaged across the six tasks, which is even 1.0% higher than the performance of Flan-T5-11B (90.3%). Such a phenomenon is also observed in existing works (Chung et al., 2022).

4.2 Black-Box Defense Results

Table 3: Effectiveness of different black-box defense methods on different models. We report the average performance of the six tasks.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Prefix**	**NN**	**Self**
Flan-T5-3B	91.3	78.0	59.3	65.2	77.4	82.3	79.9
Flan-T5-11B	90.3	80.1	64.4	69.6	78.6	83.3	84.1
Llama2-Chat-7B	70.5	54.7	57.3	58.9	56.8	66.9	66.5
Llama2-Chat-13B	72.8	55.4	51.1	54.9	57.7	67.3	69.0

Table 4: An example of applying defense methods to a corrupted sentence. Here, ✓ means the model (Llama2-Chat-7B) makes a correct prediction, and ✗ means otherwise.

Type	**Sentence**	**Correct?**
Clean	has all the depth of a wading pool	✓
Corrupted	has all t**y**e dep~~t~~h of a wading pool	✗
Rule-CC	**his** all the **death** of a **working personal**	✗
Rule-Agg	**his** all the **death** of a **working paul**	✗
NN	**Has** all the **depth** of a wading pool	✓
Self	**It has** all the **depth** of a wading pool	✓

Table 3 demonstrates effectiveness of different black-box defense methods, where we report the average performance of Flan-T5 and Llama2-Chat on the six tasks in Table 2. More details are deferred to Appendix B. We have the following observations:

First, rule-based methods do not work well for defending against adversarial typos. Recall that in rule-based defenses, each word belongs to a specific cluster and the word is converted to its cluster representation. However, because there are much fewer clusters than words, rule-based defense methods can change the meaning of a sentence. We show an example in Table 4 to better understand the rule-based approaches. In the example, "wading pool" in the corrupted sentence (note that these two words are not changed by the adversarial attack) is mapped to "working personal" or "working paul" by rule-based defenses. Such a mapping renders the entire sentence nonsense, and we observe that indeed the LLM makes wrong predictions on the sentence's sentiment. The large model capacities of LLMs enable them to infer the correct words to some extent even without any defense mechanisms. Therefore, after applying rule-based typo correction, when the contextual information loss outweighs the gain brought by typo correction, performance of LLMs drop.

Second, NN-defense and self-defense works well for both Flan-T5 and Llama2-Chat. For example, in Table 3, we see that applying NN-defense brings 3.2% accuracy gain for FlanT5-11B (from 80.1 to 83.3); while applying self-defense brings 4.0% accuracy gain (from 80.1 to 84.1). The performance gain is brought by the fact that both of these defense approaches can successfully correct the typos while maintaining the semantic meaning of the original sentence (see Table 4 for an example).

Third, prefix-defense is effective for LLaMa2-Chat. Recall that in prefix-defense, we prepend an instruction *"In the following question, please pay close attention to typos before answering"* to the prompt. The intuition is that even though typos can not be directly corrected like the other defense methods, the additional instruction can alert the LLM about potential adversarial contents. From the results in Table 3, we see that for Llama2-Chat, model robustness increases by about 2%.

4.3 White-Box Defense Results

Existing white-box defense methods leverage adversarial regularization (Cheng et al., 2021; Liu et al., 2020). Specifically, these methods construct continuous adversarial samples in the embedding space to augment the training data. However, such a process is extremely slow since multiple forward and backward passes are needed to construct an adversarial sample. Instead, we propose a simple strategy for white-box defense: we fine-tune LLMs on both clean training data and corrupted training data derived from Algorithm 1.

However, fine-tuning models with billions of parameters can be computationally prohibitive. Therefore, we use an off-the-shelf parameter efficient fine-tuning method: LoRA (Hu et al., 2022). For a weight matrix $W \in \mathbb{R}^{d\times k}$, full fine-tuning computes its gradient with respect to the loss, and updates this weight matrix accordingly. In LoRA, the update of the weight matrix is $W \leftarrow W + BA$, where $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$ are low-rank such that $r \ll \min(d, k)$. We note that the weight matrix W is frozen during fine-tuning, and we only update the two low-rank matrices A and B.

In the experiments, we freeze all the weights in the LLM, and we add the LoRA components (i.e., A and B) to the query and value matrices in all the attention layers. This is the same strategy as Hu et al. (2022) and has shown to the empirically effective. As a result, for both Flan-T5-3B and Llama2-Chat-7B, we only fine-tune about 5M parameters. We defer training details to Appendix D.

Table 5: Performance of Flan-T5-3B after fine-tuning. We consider two fine-tuning settings: tuning on clean training data only; and tuning on clean and corrupted training data. We evaluate model performance on both the clean development set and the corrupted development set.

Setting	Test data	RTE	MRPC	SST-2	QNLI	QQP	MNLI	Average
Zero-shot	clean	93.1	82.3	94.8	94.7	90.9	91.8	91.3
	corrupted	81.2	71.9	81.9	86.7	73.4	72.9	78.0
Fine-tune Clean	clean	92.7	90.7	94.4	94.8	90.9	91.9	92.6
	corrupted	86.5	88.0	88.7	90.9	81.0	79.9	85.8
Fine-tune Clean + Corr.	clean	93.1	90.7	96.6	94.8	91.1	91.9	93.0
	corrupted	88.0	88.8	94.0	92.6	88.6	85.0	89.5

Table 6: Performance of Llama2-Chat-7B after fine-tuning. We consider two fine-tuning settings: tuning on clean training data only; and tuning on clean and corrupted training data. We evaluate model performance on both the clean development set and the corrupted development set.

Setting	Test data	RTE	MRPC	SST-2	QNLI	QQP	MNLI	Average
Zero-shot	clean	77.5	65.0	94.1	76.6	58.6	51.1	70.5
	corrupted	65.6	62.2	77.2	57.5	34.9	31.0	54.7
Fine-tune Clean	clean	89.5	87.9	96.2	92.6	90.9	87.8	90.8
	corrupted	85.5	86.5	89.2	89.1	86.3	83.3	86.7
Fine-tune Clean + Corr.	clean	89.5	88.4	96.4	92.6	90.8	87.8	90.9
	corrupted	87.7	87.3	93.8	90.5	89.1	85.8	89.0

Table 5 demonstrates experimental results of Flan-T5-3B; and Table 6 demonstrates experimental results of Llama2-Chat-7B. We consider two fine-tuning schemes: 1) we only fine-tune LLMs on the clean training data; and 2) we fine-tune LLMs on both the clean training data and the corrupted training data that contain typos. From the results, we see that

- ◇ Fine-tuning on task-specific clean data improves both model performance and robustness. For example, after fine-tuning, performance of Flan-T5-3B increases by 1.3% on the clean test data and 7.8% on the corrupted test data. As another example, performance of Llama2-Chat-7B increases by 20.3% on the clean data, and robustness increases by 32.0%.
- ◇ Fine-tuning on task-specific clean and corrupted data can further improve model robustness. From Table 5, we see that performance of Flan-T5-3B is 89.5% on the corrupted test data, which is more than 20% higher than that in the zero-shot setting.

4.4 Attacking Adversarial Defenses

Table 3 demonstrates that black-box defenses can indeed improve model robustness. A natural question to ask is: how robust are the defenses against stronger attacks?

Table 7: An example of applying NN-defense to corrupted sentences. Here, *weak attack* refers to the result of applying the attack method in Algorithm 1.

Type	**Sentence**
Clean	as vulgar as it is banal
Weak Attack **NN-defense**	a**b** vulgar as **bt** **s**s banal **c** **A** vulgar as **it is** banal **c**
Strong Attack **NN-defense**	a**b** vulgar as **p**it **b**is banal **c** **A** vulgar as **p**it **b**is banal **c** **18**

Recall that in Algorithm 1, for an input pair (x, y), where x is the input and y is its label, we find an attack x' that empirically minimizes $p(y|x')$. To construct strong attacks, we expose the defense method to the attacker. That is, in the strong attack algorithm, we empirically minimize $p(y|\text{defense}(x'))$, where $\text{defense}(\cdot)$ is the defense method, e.g., a neural typo corrector. In this way, we can build a tailored attack algorithm for each defense approach. Table 7 demonstrates an example of applying strong attacks. We see that with the attack method in Algorithm 1 (i.e., *weak attack*), the NN-defense model can resolve most of the typos. However, when we expose the defense model to the attacker, the resulting attacks are much stronger (i.e., *strong attack*). We see that with strong attacks, the NN-defense model can no longer recover the original semantic meaning.

Table 8: Effectiveness of different black-box defense methods on different models under the strong attack scheme. We report the average performance of the six tasks.

	Rule-CC	**Rule-Agg**	**Prefix-defense**	**NN-defense**	**Self-defense**
Flan-T5-3B	47.3	52.0	72.4	77.1	80.1
Flan-T5-11B	50.3	55.7	71.2	77.0	82.0
Llama2-Chat-7B	42.8	45.4	44.7	61.2	61.3
Llama2-Chat-13B	42.7	44.0	51.3	61.2	64.7

Table 8 summarizes model performance under strong attacks. Model performance on individual tasks are deferred to Appendix C. We see that the attacks are indeed stronger compared with the weak attacks (see results in Table 3). We also note that even with the strong attacks, performance of NN-defense and self-defense are still satisfactory. For example, performance of the self-defense method on Flan-T5-3B when dealing with the strong attacks is on par with the performance when dealing with the weak attacks.

5 Conclusion and Discussion

LLMs have demonstrated superior performance in various natural language processing tasks. However, they are not robust to adversarial attacks. In this work, we consider input typos, which is a practical yet under-explored scenario that hinders performance of LLMs. We systematically study the effectiveness of adversarial defense approaches against input typos. Through extensive experiments, we find that many existing rule-based defense methods that work well on conventional language models are not effective for LLMs. Therefore, we design adversarial defenses tailored for LLMs. First, we develop prompting methods (prefix-defense) to alert the LLM about potential adversarial contents. Second, we find that we can denoise the inputs by adopting either the LLM itself (self-defense) or using a separate small language model (NN-defense). Finally, in the white-box setting, we find that fine-tune LLMs on both clean and corrupted data can be extremely beneficial for model robustness.

In this work, we develop several methods that improve model robustness. Among them, NN-defense is particularly favorable. This is because fine-tuning the LLM may not always be feasible, rendering white-box defenses impractical. Among the black-box defense methods, NN-defense and self-defense are equally effective. However, NN-defense is much faster as only a small language model is used to denoise the input typos, making it more attractive. We note that the proposed prefix-defense is faster than NN-defense and can also improve robustness to some extent. We leave further investigation along this direction as future works.

REFERENCES

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2890–2896, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1316.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, et al. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity. *ArXiv preprint*, abs/2302.04023, 2023.

Roy Bar-Haim, Ido Dagan, Bill Dolan, Lisa Ferro, and Danilo Giampiccolo. The second PASCAL recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*, 2006.

Yonatan Belinkov and Yonatan Bisk. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Luisa Bentivogli, Ido Dagan, Hoa Trang Dang, Danilo Giampiccolo, and Bernardo Magnini. The fifth pascal recognizing textual entailment challenge. In *In Proc Text Analysis Conference (TAC'09*, 2009.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *ArXiv preprint*, abs/2303.12712, 2023.

Xuanting Chen, Junjie Ye, Can Zu, Nuo Xu, Rui Zheng, Minlong Peng, Jie Zhou, Tao Gui, Qi Zhang, and Xuanjing Huang. How robust is gpt-3.5 to predecessors? a comprehensive study on language understanding tasks. *ArXiv preprint*, abs/2303.00293, 2023.

Hao Cheng, Xiaodong Liu, Lis Pereira, Yaoliang Yu, and Jianfeng Gao. Posterior differential regularization with f-divergence for improving model robustness. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1078–1089, Online, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.85.

Jonathan H Choi, Kristin E Hickman, Amy Monahan, and Daniel Schwarcz. Chatgpt goes to law school. *Available at SSRN*, 2023.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *ArXiv preprint*, abs/2204.02311, 2022.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *ArXiv preprint*, abs/2210.11416, 2022.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, pp. 177–190, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33427-0, 978-3-540-33427-9. doi: 10.1007/11736790_9.

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)*, pp. 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423.

William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. Text processing like humans do: Visually attacking and shielding NLP systems. 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)*, pp. 1634–1647, Minneapolis, Minnesota, 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1165.

Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Christian Petersen, Alexis Chevalier, and Julius Berner. Mathematical capabilities of chatgpt. *ArXiv preprint*, abs/2301.13867, 2023.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pp. 50–56. IEEE, 2018.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. The third PASCAL recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*, pp. 1–9, Prague, 2007. Association for Computational Linguistics.

Hongyu Gong, Yuchen Li, Suma Bhat, and Pramod Viswanath. Context-sensitive malicious spelling error correction. In Ling Liu, Ryen W. White, Amin Mantrach, Fabrizio Silvestri, Julian J. McAuley, Ricardo Baeza-Yates, and Leila Zia (eds.), *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pp. 2771–2777. ACM, 2019. doi: 10.1145/3308558.3313431.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *ArXiv preprint*, abs/2111.09543, 2021a.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021b.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1875–1885, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1170.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 2752–2765, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.245.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703.

Xiaodong Liu, Hao Cheng, Pengcheng He, Weizhu Chen, Yu Wang, Hoifung Poon, and Jianfeng Gao. Adversarial training for large neural language models. *ArXiv preprint*, abs/2004.08994, 2020.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

OpenAI. Gpt-4 technical report. *arXiv*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. Crafting adversarial input sequences for recurrent neural networks. In *MILCOM 2016-2016 IEEE Military Communications Conference*, pp. 49–54. IEEE, 2016.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 8024–8035, 2019.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 5582–5591, Florence, Italy, 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1561.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *ArXiv preprint*, abs/2211.05100, 2022.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. Prompting gpt-3 to be reliable. *ArXiv preprint*, abs/2210.09150, 2022.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1631–1642, Seattle, Washington, USA, 2013. Association for Computational Linguistics.

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*, abs/2302.13971, 2023a.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023b.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 5998–6008, 2017.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *ArXiv preprint*, abs/2111.02840, 2021.

Jindong Wang, Xixu Hu, Wenxin Hou, Hao Chen, Runkai Zheng, Yidong Wang, Linyi Yang, Haojun Huang, Wei Ye, Xiubo Geng, et al. On the robustness of chatgpt: An adversarial and out-of-distribution perspective. *ArXiv preprint*, abs/2302.12095, 2023a.

Jiongxiao Wang, Zichen Liu, Keun Hee Park, Muhao Chen, and Chaowei Xiao. Adversarial demonstration attacks on large language models. *ArXiv preprint*, abs/2305.14950, 2023b.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *ArXiv preprint*, abs/2206.07682, 2022.

Adina Williams, Nikita Nangia, and Samuel Bowman. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1112–1122, New Orleans, Louisiana, 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1101.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv preprint*, abs/1910.03771, 2019.

Hui Wu and Xiaodong Shi. Adversarial soft prompt tuning for cross-domain sentiment analysis. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2438–2447, Dublin, Ireland, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.174.

Lei Xu, Yangyi Chen, Ganqu Cui, Hongcheng Gao, and Zhiyuan Liu. Exploring the universal vulnerability of prompt-based learning paradigm. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 1799–1810, Seattle, United States, 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.137.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *ArXiv preprint*, abs/2205.01068, 2022.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, et al. Promptbench: Towards evaluating the robustness of large language models on adversarial prompts. *ArXiv preprint*, abs/2306.04528, 2023.

Shengyao Zhuang and Guido Zuccon. Dealing with typos for BERT-based passage retrieval and ranking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2836–2842, Online and Punta Cana, Dominican Republic, 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.225.

Shengyao Zhuang and Guido Zuccon. Characterbert and self-teaching for improving the robustness of dense retrievers on queries with typos. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1444–1454, 2022.

Simiao Zuo, Chen Liang, Haoming Jiang, Pengcheng He, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. ARCH: Efficient adversarial regularized training with caching. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 4118–4131, Punta Cana, Dominican Republic, 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.348.

Simiao Zuo, Chen Liang, Haoming Jiang, Xiaodong Liu, Pengcheng He, Jianfeng Gao, Weizhu Chen, and Tuo Zhao. Adversarial regularization as stackelberg game: An unrolled optimization approach. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6562–6577, Online and Punta Cana, Dominican Republic, 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.527.

A PROMPTS

Below, we show the prompts we used to inference Flan-T5 and Llama2-Chat models.

Table 9: Prompts used to inference Flan-T5.

Task	Prompt
RTE	Read the following paragraph and determine if the hypothesis is true: *[input$_1$]*. Hypothesis: *[input$_2$]*. OPTIONS: yes, no
MRPC	First sentence: *[input$_1$]*; Second sentence: *[input$_2$]*. Would you say that these sentences have the same meaning?
SST-2	Review: *[input$_1$]*. Is this review sentence negative or positive? OPTIONS: positive, negative
QNLI	Premise: *[input$_1$]*; Hypothesis: *[input$_2$]* Does the premise entail the hypothesis?
QQP	First question: *[input$_1$]*; Second question: *[input$_2$]*. Would you say that these questions have the same meaning?
MNLI	Does the sentence *[input$_1$]* answer the question *[input$_2$]*?

Table 10: Prompts used to inference Llama2-Chat-7B.

Task	Prompt
RTE	*[input$_1$]*. Based on the paragraph above can we conclude that *[input$_2$]*? Answer with yes or no only. Answer:
MRPC	Do these sentences mean the same thing? *[input$_1$]*, *[input$_2$]*. Answer with yes or no only. Answer:
SST-2	Is the following review positive or negative? Answer in a single word. Review: *[input$_1$]*. Answer:
QNLI	Does the sentence answer the question? Answer with yes or no only. Question: *[input$_1$]*. Sentence: *[input$_2$]*. Answer:
QQP	First sentence: *[input$_1$]*. Second sentence: *[input$_2$]*. Would you say that these sentences have close meanings? Answer with yes or no only. Answer:
MNLI	Premise: *[input$_1$]*. Hypothesis: *[input$_2$]*. Is the hypothesis entailed by the premise? Answer in a single word with yes, no or neutral. Answer:

Table 11: Prompts used to inference Llama2-Chat-13B.

Task	Prompt
RTE	*[input$_1$]*. Based on the paragraph above can we conclude that *[input$_2$]*? Answer with yes or no only. Answer:
MRPC	Here are two sentences *[input$_1$]*, *[input$_2$]*. Do they have the same meaning? Answer with yes or no only. Answer:
SST-2	Is the following review positive or negative? Answer in a single word. Review: *[input$_1$]*. Answer:
QNLI	Does the sentence answer the question? Answer with yes or no only. Question: *[input$_1$]*. Sentence: *[input$_2$]*. Answer:
QQP	First sentence: *[input$_1$]*. Second sentence: *[input$_2$]*. Would you say that these sentences have close meanings? Answer with yes or no only. Answer:
MNLI	Premise: *[input$_1$]*. Hypothesis: *[input$_2$]*. Is the hypothesis entailed by the premise? Answer in a single word with yes, no or irrelevant. Answer:

B DETAILS OF BLACK-BOX DEFENSE RESULTS

In Table 3, we demonstrate effectiveness of black-box defenses on different models by showing the average score of six tasks. Below, we show detailed breakdown of model performance when applying each defense method on each task.

Table 12: Effectiveness of black-box defense methods on Flan-T5-3B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Prefix**	**NN**	**Self**
RTE	93.1	81.2	59.4	65.9	81.9	80.1	82.2
MRPC	82.3	71.9	41.5	47.8	65.0	68.4	74.0
SST-2	94.8	81.9	70.0	77.2	83.9	91.4	86.8
QNLI	94.7	86.7	66.0	71.3	87.6	88.6	87.3
QQP	90.9	73.4	66.8	70.6	73.8	81.9	74.5
MNLI	91.8	72.9	52.0	58.2	72.0	83.2	74.8
Average	91.3	78.0	59.3	65.2	77.4	82.3	79.9

Table 13: Effectiveness of black-box defense methods on Flan-T5-11B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Prefix**	**NN**	**Self**
RTE	89.5	81.5	63.8	66.7	79.7	82.2	84.1
MRPC	82.6	73.6	54.6	63.4	66.5	71.6	77.5
SST-2	96.1	86.2	74.3	79.8	88.4	93.3	90.8
QNLI	94.5	87.9	70.1	75.9	89.6	88.0	89.4
QQP	88.7	75.8	69.9	71.9	71.7	81.9	81.3
MNLI	90.3	75.4	53.5	60.1	75.4	82.7	81.5
Average	90.3	80.1	64.4	69.6	78.6	83.3	84.1

Table 14: Effectiveness of black-box defense methods on Llama2-Chat-7B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Prefix**	**NN**	**Self**
RTE	77.5	65.6	60.1	61.2	66.3	73.9	72.5
MRPC	65.0	62.2	59.8	59.7	65.7	65.5	65.7
SST-2	94.1	77.2	67.3	71.2	80.3	89.8	88.1
QNLI	76.6	57.5	56.9	59.9	54.7	64.4	69.2
QQP	58.6	34.9	61.0	60.1	45.4	59.2	56.2
MNLI	51.1	31.0	38.4	41.2	28.4	48.6	47.4
Average	70.5	54.7	57.3	58.9	56.8	66.9	66.5

Table 15: Effectiveness of black-box defense methods on Llama2-Chat-13B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Prefix**	**NN**	**Self**
RTE	76.8	66.3	51.1	55.1	58.7	71.0	69.9
MRPC	69.7	38.4	38.7	45.5	65.9	62.9	66.6
SST-2	94.8	78.4	61.4	68.7	74.2	88.5	90.7
QNLI	73.6	54.8	51.8	53.3	52.2	63.6	69.1
QQP	71.4	55.6	63.9	64.2	60.6	70.3	70.2
MNLI	50.5	38.9	39.9	42.6	34.3	47.5	47.5
Average	72.8	55.4	51.1	54.9	57.7	67.3	69.0

C Details of Defense Against Strong Attacks

In Table 8, we demonstrate effectiveness of black-box defenses against strong attacks. Specifically, for each model, we show the average score of six tasks. Below, we show detailed breakdown of model performance when applying each defense method on each task.

Table 16: Effectiveness of black-box defenses against strong attacks on Flan-T5-3B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Self**	**Prefix**	**NN**
RTE	93.1	81.2	46.7	50.4	81.5	76.8	79.0
MRPC	82.3	71.9	35.7	38.6	74.8	58.2	62.5
SST-2	94.8	81.9	55.3	61.7	85.0	81.3	85.1
QNLI	94.7	86.7	54.5	60.4	88.2	86.6	84.4
QQP	90.9	73.4	58.6	59.8	75.7	66.8	77.1
MNLI	91.8	72.9	33.1	41.1	75.1	64.6	74.2
Average	91.3	78.0	47.3	52.0	80.1	72.4	77.1

Table 17: Effectiveness of black-box defenses against strong attacks on Flan-T5-11B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Self**	**Prefix**	**NN**
RTE	89.5	81.5	54.0	56.5	82.2	72.1	76.1
MRPC	82.6	73.6	40.3	47.7	70.0	54.2	63.5
SST-2	96.1	86.2	55.6	64.9	90.6	82.8	86.6
QNLI	94.5	87.9	55.9	63.5	89.7	86.5	84.9
QQP	88.7	75.8	57.7	58.5	81.5	63.1	75.8
MNLI	90.3	75.4	38.4	43.0	78.2	68.7	75.3
Average	90.3	80.1	50.3	55.7	82.0	71.2	77.0

Table 18: Effectiveness of black-box defenses against strong attacks on Llama2-Chat-7B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Self**	**Prefix**	**NN**
RTE	77.5	65.6	40.9	46.4	64.5	53.3	68.8
MRPC	65.0	62.2	48.7	51.2	65.1	65.7	64.4
SST-2	94.1	77.2	49.4	57.1	85.2	67.0	84.8
QNLI	76.6	57.5	45.7	47.4	63.0	43.4	61.0
QQP	58.6	34.9	49.3	45.4	47.9	22.8	46.4
MNLI	51.1	31.0	22.8	25.1	42.3	16.1	41.7
Average	70.5	54.7	42.8	45.4	61.3	44.7	61.2

D More Details about White-Box Defense

We implement white-box defenses using PyTorch (Paszke et al., 2019) and the HuggingFace codebase (Wolf et al., 2019). In all the experiments, we use a batch size of 64 and we use AdamW (Loshchilov & Hutter, 2019) as the optimizer. The other hyper-parameters are shown in Table 20 and Table 21.

Table 19: Effectiveness of black-box defenses against strong attacks on Llama2-Chat-13B.

	No Defense		**Defense Methods**				
	Clean	**Corrupted**	**Rule-CC**	**Rule-Agg**	**Self**	**Prefix**	**NN**
RTE	76.8	66.3	44.2	48.2	63.8	53.6	68.8
MRPC	69.7	38.4	33.7	32.3	62.2	65.1	52.5
SST-2	94.8	78.4	47.9	53.0	89.2	62.1	81.5
QNLI	73.6	54.8	49.1	48.0	62.2	46.7	57.3
QQP	71.4	55.6	56.0	53.7	67.2	46.5	63.4
MNLI	50.5	38.9	25.3	28.5	43.3	33.6	43.5
Average	72.8	55.4	42.7	44.0	64.7	51.3	61.2

Table 20: Hyer-parameters for fine-tuning Flan-T5-3B. Here, *LR* is the learning rate, and *LoRA-alpha* is an initialization parameter in Hu et al. (2022). We consider two settings: *clean* means we only train on clean training data, and *c+c* means we train on clean and corrupted training data.

	LR		**Epoch**		**Warmup**		**LoRA-rank**		**LoRA-alpha**	
	clean	c+c	clean	c+c	clean	c+c	clean	c+c	clean	c+c
RTE	4e-4	3e-4	30	30	0.1	0.2	32	8	64	16
MRPC	1e-4	3e-4	20	10	0.1	0.1	8	8	16	16
SST-2	5e-4	5e-4	10	10	0	0.1	8	8	16	16
QNLI	1e-3	1e-3	10	10	0.1	0.2	8	8	16	16
QQP	1e-3	7e-4	20	10	0.1	0.1	8	8	16	16
MNLI	7e-4	3e-4	10	10	0.1	0.1	8	8	16	16

Table 21: Hyer-parameters for fine-tuning Llama2-Chat-7B. Here, *LR* is the learning rate, and *LoRA-alpha* is an initialization parameter in Hu et al. (2022). We consider two settings: *clean* means we only train on clean training data, and *c+c* means we train on clean and corrupted training data.

	LR		**Epoch**		**Warmup**		**LoRA-rank**		**LoRA-alpha**	
	clean	c+c	clean	c+c	clean	c+c	clean	c+c	clean	c+c
RTE	2e-4	2e-4	30	30	0.1	0.1	32	32	64	64
MRPC	7e-4	7e-4	20	20	0.1	0.1	8	8	16	16
SST-2	5e-4	5e-4	10	10	0.1	0.1	8	8	16	16
QNLI	7e-4	2e-4	10	10	0.1	0.1	8	8	16	16
QQP	1e-3	7e-4	10	10	0.1	0.1	8	8	16	16
MNLI	5e-4	5e-4	10	10	0.1	0.1	8	8	16	16