CHARACTER-LEVEL ROBUSTNESS SHOULD BE REVISITED

Elias Abad Rocamora[1,*], **Yongtao Wu**[1], **Fanghui Liu**[2]
Grigorios G. Chrysos[3], **Volkan Cevher**[1]
[1]LIONS - École Polytechnique Fédérale de Lausanne, [2]University of Warwick,
[3]University of Wisconsin-Madison

ABSTRACT

Adversarial attacks in Natural Language Processing apply perturbations in the character or token levels. *Token-level* attacks, gaining prominence for their use of gradient-based methods, are susceptible to altering sentence semantics, leading to invalid adversarial examples. While *character-level* attacks easily maintain semantics, they have received less attention as they cannot easily adopt popular gradient-based methods, and are thought to be easy to defend. Challenging these beliefs, we introduce `Charmer`, an efficient query-based adversarial attack capable of achieving high attack success rate (ASR) while generating highly similar adversarial examples. Our method successfully targets both small (BERT) and large (Llama 2) models. Specifically, on BERT with SST-2, `Charmer` improves the ASR in 4.84% points and the USE similarity in 8% points with respect to the previous art.

1 INTRODUCTION

Language Models (LMs) have rapidly become the go-to tools for Natural Language Processing (NLP) tasks like language translation (Sutskever et al., 2014), code development (Chen et al., 2021) and even general counseling via chat interfaces (OpenAI, 2023). However, several failures concerning robustness to natural and adversarial noise have been discovered (Belinkov and Bisk, 2018; Alzantot et al., 2018). Adversarial attacks have been widely adopted in the computer vision community to discover the worst-case performance of Machine Learning models (Szegedy et al., 2014; Goodfellow et al., 2015) or be used to defend against such failure cases (Madry et al., 2018; Zhang et al., 2019).

The application of adversarial attacks in LMs is not straight-forward due to algorithmic (Guo et al., 2021) and imperceptibility constraints (Morris et al., 2020a). Unlike the computer vision tasks, where inputs consist of tensors of real numbers, in NLP tasks, we work with sequences of discrete non-numerical inputs. This results in adversarial attacks being an NP-hard problem even for convex classifiers (Lei et al., 2019). This fact also hardens the use of popular gradient-based methods for obtaining adversarial examples (Guo et al., 2021). To tackle this problem, attackers adopt gradient based strategies in the embedding space, restricting the attack to the token vocabulary (Ebrahimi et al., 2018; Liu et al., 2022; Hou et al., 2023) or the *black-box* setting, where only input-output access to the model is assumed (Alzantot et al., 2018; Gao et al., 2018; Jin et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020; Wallace et al., 2020).

Another difficult analogy to make with the computer vision world is imperceptibility. Adversarial examples should be by definition imperceptible, in the sense that the attack should not modify the human prediction or allow to think that an attack has been done (Szegedy et al., 2014). Given an input $\boldsymbol{x} \in \mathbb{R}^d$, in the numerical-input setting, imperceptibility is controlled by looking for an adversarial example $\hat{\boldsymbol{x}} \in \mathbb{R}^d$ in an ℓ_p ball centered at $\boldsymbol{x}$ with perturbation radius ϵ, i.e., $||\boldsymbol{x} - \hat{\boldsymbol{x}}||_p \leq \epsilon$, where ϵ can be set arbitrarily small.

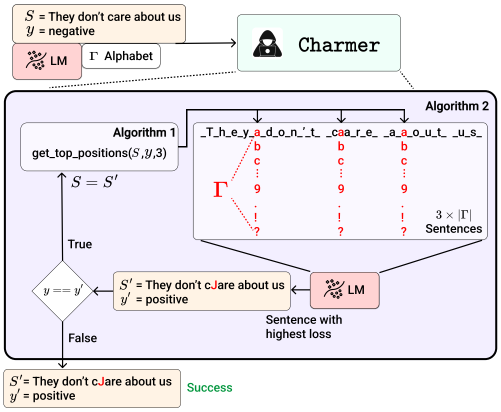
In NLP, Morris et al. (2020a) suggest different strategies for controlling imperceptibility according to the attack level:
Character: Constrain the attack to have a low Levenshtein (edit) distance. However, character-level

[*] Correspondence: `elias.abadrocamora@epfl.ch`

Attack level	Token	Char	
		Previous	Charmer (This work)
ASR(%)	95.16	0.96*	100.00*
High efficiency	✓	✓	✓
Semantics preserving	✗**	✓	✓

*Defended with (Jones et al., 2020).
**According to (Hou et al., 2023; Dyrmishi et al., 2023).

(a) Attack desiderata & state-of-the-art.


S = They don't care about us
y = negative
LM
Γ Alphabet
Charmer
Algorithm 2
Algorithm 1
get_top_positions(S,y,3)
_T_h_e_y_a_d_o_n_'_t_ _caa_r_e_ _a_a_o_u_t_ _u_s_
b
c
9
!
?
Γ
3 × |Γ|
Sentences
$S = S'$
True
$y == y'$
S' = They don't cJare about us
y' = positive
LM
Sentence with
highest loss
False
S'= They don't cJare about us
y' = positive
Success

(b) Schematic of the proposed method, `Charmer`

Figure 1: Desiderata and example of our attack in the sentiment classification task with the positions subset size $n = 3$. At each iteration, our attack computes the most important positions in the sentence via Algorithm 1. Then, we generate all possible sentences replacing a character in the top positions, to get the one with the highest loss. If this sentence is misclassified, the process is finished.

attacks have lost relevance due to the strength of robust word recognition defenses (Pruthi et al., 2019; Jones et al., 2020).
Token: Constrain the embedding similarity[1] of replaced words and of the overall sentence to be high. Nevertheless, Dyrmishi et al. (2023) conclude that state-of-the-art attacks do not produce imperceptible attacks in practice. To be specific, Hou et al. (2023) report 56.5% of their attacks change the semantics of the sentence.

The overall attack desiderata is summarized in Fig. 1a. Existing defenses against character-level attacks rely on robust word recognition modules, which assume the attacker adopts unrealistic constraints, not allowing simple modifications such as insertion or deletion of blank spaces, which are adopted in practice (Li et al., 2019). In this work, we revisit character-level adversarial attacks as a practical solution to imperceptibility. Our attack, `Charmer`, is based on a greedy approach combined with a position subset selection to further speed-up the attack, while minimally affecting performance. Our attack is able to achieve $> 95\%$ ASR in every studied TextAttack benchmark and LLMs Llama-2 and Vicuna, obtaining up to a 23%-point ASR improvement with respect to the runner-up method. We show that existing adversarial training based defenses (Hou et al., 2023) degrade character-level robustness, i.e., increasing the ASR in 3.32% points when compared to standard training. Our findings indicate typo-corrector defenses (Pruthi et al., 2019; Jones et al., 2020) are only successful when a set of strict attack constraints is assumed, if just one of these constraints is relaxed, ASR can increase from 0.96% to 98.09%. Overall, we believe character-level robustness a more consistent measure than token-level robustness.

2 Method

Let us now introduce our method (`Charmer`). In Appendices F.2 and F.3 we present our attack for both standard classifiers and LLM-based classifiers. To circumvent the exponential dependence of the number of nearby sentences on k as indicated by Corollary S6, we propose to greedily select the single-character perturbation with the highest loss Algorithm 2. Furthermore, we reduce the search space for single-character perturbations by considering a subset of locations where characters can be replaced, see Algorithm 1. In Definitions 2.1 and 2.2 we define the key operators employed in our attack.

Definition 2.1 (Expansion and contraction operators)**.** Let $\mathcal{S}(\Gamma)$ be the space of sentences with alphabet Γ and the special character $\xi \notin \Gamma$, the pair of expansion-contraction functions $\phi : \mathcal{S}(\Gamma) \to \mathcal{S}(\Gamma \cup \{\xi\})$ and $\psi : \mathcal{S}(\Gamma \cup \{\xi\}) \to \mathcal{S}(\Gamma)$ is defined as:

[1] Similarity is commonly measured via the cosine similarity of the USE embeddings (Cer et al., 2018).

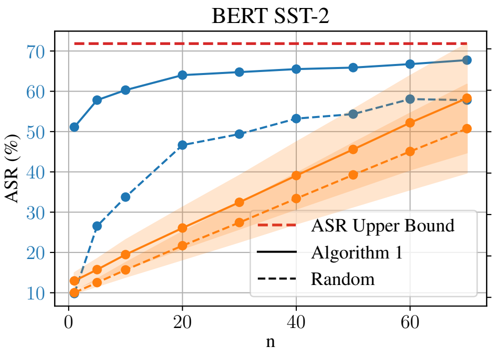


Figure 2: **Selection of the number of candidate positions:** Attack Success Rate (ASR) at $k = 1$ (● left axis) and runtime (● right axis) for our candidate position selection strategy (Algorithm 1, bold lines) and a random selection (Random, dotted lines). Our strategy improves the random baseline at a small cost ($\approx 0.25s$).

Algorithm 1 Top-n position selection.

```
Inputs: model $f$, sentence $S$, test char. $t$, special char. $\xi$, number of positions $n$, loss $\mathcal{L}$ and label $y$.
$\boldsymbol{l} = \mathbf{0}$          ▷ Initialize losses with zeros
for $i = 1, \ldots, 2|S| + 1$ do
    $P = \phi(S)$          ▷ Expand sentence
    if $P_i = t$ then $P_i \leftarrow \xi$
    else $P_i \leftarrow t$
    $l_i = \mathcal{L}(f(\psi(P)), y)$          ▷ Eval. loss
return Top-$n(\boldsymbol{l})$          ▷ Id. of Top $n$ values
```

Algorithm 2 `Charmer` Adversarial Attack

```
Inputs: model $f$, sentence $S$, alphabet of characters $\Sigma$, max Levenshtein distance $k$, candidate positions $n$, loss function $\mathcal{L}$ and label $y$.
$S' = S$          ▷ Initialize attack
for $i = 1, \ldots, k$ do
    $Z = \text{get\_top\_locations}(f, S', y, n)$          ▷ Algorithm 1
    $\mathcal{S}' = \{\psi(\phi(S') \overset{j}{\leftarrow} c),\ \forall j \in Z, \forall c \in \Sigma \cup \{\xi\}\}$          ▷ All sentences with modifications in $Z$
    $\boldsymbol{l} = \mathcal{L}(f(\mathcal{S}'), y)$          ▷ Batch of $n \cdot (|\Sigma| + 1)$ sentences
    $j^* = \arg\max_{j \in [|\mathcal{S}'|]} l_j$
    $S' = \mathcal{S}'_{j^*}$          ▷ Sentence with highest loss in the batch
    if $\arg\max_{\hat{y} \in [o]} f(S') \neq y$ then return $S'$          ▷ Successful
return $S'$          ▷ Unsuccessful
```

$$\phi(S) := \begin{cases} \xi & \text{if } |S| = 0 \\ \xi, S_1 \oplus \phi(S_{2:}) & \text{otherwise}\,. \end{cases} \quad \psi(S) := \begin{cases} \emptyset & \text{if } |S| = 0 \\ \psi(S_{2:}) & \text{if } S_1 = \xi \\ S_1 \oplus \psi(S_{2:}) & \text{otherwise}\,. \end{cases}$$

Clearly, $\phi(S)$ inserts ξ into S in all possible positions between characters and at the beginning and end of the sentence. Similarly, $\psi(S)$ aims to remove all ξ occurred in S.

Definition 2.2 (Replacement operator). Let $S \in \mathcal{S}(\Gamma \cup \{\xi\})$, the integer $i \in [|S|]$ and the character c, the replacement operator is defined as: $S \overset{i}{\leftarrow} c := S_{:i-1} \oplus c \oplus S_{i+1:}$.

3 Experiments

Our experiments are conducted in the publicly available[2] TextAttack models (Morris et al., 2020b) and open-source large language models including Llama 2-Chat 7B (Touvron et al., 2023) and Vicuna 7B (Chiang et al., 2023). We evaluate our attack in the text (or text pair) classification datasets SST-2 (Socher et al., 2013), RTE (Dagan et al., 2006; Wang et al., 2019), QNLI (Rajpurkar et al., 2016), MNLI-m (Williams et al., 2018) and AG-News (Gulli, 2005; Zhang et al., 2015). We provide additional experiments in the supplementary material, see Appendix E.

[2] `https://huggingface.co/textattack`

Table 1: **Attack evaluation in the TextAttack BERT and RoBERTa models:** Token-level and character-level attacks are highlighted with 🔵 and 🔴 respectively. For each metric, the best method is highlighted in **bold** and the runner-up is $\underline{\text{underlined}}$. `Charmer` consistently achieves the highest Attack Success Rate (ASR), while achieving the the smallest Levenshtein distance (d_{lev}) in every case. Additionally, the similarity between the original and attacked sentences is the runner-up in all cases.

		BERT				RoBERTa			
	Method	ASR (%) ↑	$d_{\text{lev}}(S, S')$ ↓	Sim(S, S') ↑	Time (s) ↓	ASR (%) ↑	$d_{\text{lev}}(S, S')$ ↓	Sim(S, S') ↑	Time (s) ↓
AG-News	GBDA 🔵	42.09	$17.76_{\pm(9.33)}$	$0.93_{\pm(0.05)}$	$13.86_{\pm(3.14)}$	-	-	-	-
	BAE-R 🔵	17.09	$15.07_{\pm(10.59)}$	$\mathbf{0.97}_{\pm(0.02)}$	$1.61_{\pm(1.36)}$	18.27	$15.29_{\pm(10.34)}$	$\mathbf{0.97}_{\pm(0.02)}$	$2.14_{\pm(1.81)}$
	BERT-attack 🔵	29.90	$20.66_{\pm(16.91)}$	$0.93_{\pm(0.05)}$	$5.58_{\pm(12.92)}$	27.55	$16.96_{\pm(12.95)}$	$0.94_{\pm(0.04)}$	$\underline{1.44}_{\pm(1.76)}$
	DeepWordBug 🔴	60.51	$11.75_{\pm(8.00)}$	$0.78_{\pm(0.18)}$	$\mathbf{0.81}_{\pm(0.52)}$	56.81	$11.81_{\pm(7.69)}$	$0.79_{\pm(0.16)}$	$\mathbf{0.69}_{\pm(0.35)}$
	TextBugger 🔴	50.85	$19.79_{\pm(17.93)}$	$0.90_{\pm(0.06)}$	$\underline{1.53}_{\pm(1.13)}$	51.21	$21.42_{\pm(19.28)}$	$0.90_{\pm(0.06)}$	$2.30_{\pm(1.61)}$
	TextFooler 🔵	78.98	$53.18_{\pm(39.30)}$	$0.84_{\pm(0.11)}$	$3.75_{\pm(2.76)}$	84.48	$52.45_{\pm(36.97)}$	$0.84_{\pm(0.11)}$	$3.84_{\pm(2.77)}$
	TextGrad 🔵	85.85	$55.38_{\pm(30.33)}$	$0.77_{\pm(0.11)}$	$7.98_{\pm(9.24)}$	78.75	$31.94_{\pm(15.57)}$	$0.86_{\pm(0.07)}$	$9.86_{\pm(9.74)}$
	CWBA 🔴	86.72	$15.71_{\pm(7.17)}$	$0.65_{\pm(0.19)}$	$174.15_{\pm(130.91)}$	81.39	$13.73_{\pm(11.24)}$	$0.86_{\pm(0.11)}$	$55.33_{\pm(43.19)}$
	(Pruthi et al., 2019) 🔴	90.02	$6.25_{\pm(4.69)}$	$0.86_{\pm(0.14)}$	$49.47_{\pm(48.26)}$	88.91	$6.55_{\pm(5.13)}$	$0.86_{\pm(0.14)}$	$29.75_{\pm(24.53)}$
	`Charmer-Fast` (Ours) 🔴	$\underline{95.86}$	$\underline{4.85}_{\pm(3.96)}$	$0.92_{\pm(0.08)}$	$3.12_{\pm(3.88)}$	$\underline{91.87}$	$\underline{4.87}_{\pm(4.07)}$	$0.91_{\pm(0.09)}$	$3.15_{\pm(3.83)}$
	`Charmer` (Ours) 🔴	**98.51**	$\mathbf{3.68}_{\pm(3.08)}$	$\underline{0.95}_{\pm(0.06)}$	$8.74_{\pm(11.10)}$	**96.88**	$\mathbf{3.73}_{\pm(3.07)}$	$\underline{0.95}_{\pm(0.05)}$	$9.45_{\pm(11.20)}$
SST-2	GBDA 🔵	83.37	$12.20_{\pm(6.94)}$	$0.85_{\pm(0.11)}$	$9.32_{\pm(1.78)}$	-	-	-	-
	BAE-R 🔵	66.38	$10.10_{\pm(7.00)}$	$0.83_{\pm(0.18)}$	$1.24_{\pm(0.86)}$	63.16	$10.22_{\pm(6.33)}$	$0.85_{\pm(0.16)}$	$0.73_{\pm(0.62)}$
	BERT-attack 🔵	69.57	$12.19_{\pm(9.55)}$	$0.87_{\pm(0.09)}$	$239.80_{\pm(1763.30)}$	64.21	$11.26_{\pm(7.18)}$	$0.86_{\pm(0.10)}$	$18.12_{\pm(32.34)}$
	DeepWordBug 🔴	81.39	$3.74_{\pm(2.95)}$	$0.80_{\pm(0.17)}$	$\mathbf{0.22}_{\pm(0.12)}$	84.27	$4.61_{\pm(3.47)}$	$0.75_{\pm(0.20)}$	$\mathbf{0.28}_{\pm(0.16)}$
	TextBugger 🔴	68.49	$5.97_{\pm(5.87)}$	$\mathbf{0.91}_{\pm(0.06)}$	$1.75_{\pm(0.91)}$	61.10	$6.85_{\pm(6.54)}$	$\mathbf{0.90}_{\pm(0.05)}$	$1.82_{\pm(0.97)}$
	TextFooler 🔵	$\underline{95.16}$	$17.17_{\pm(12.51)}$	$0.82_{\pm(0.15)}$	$0.90_{\pm(0.57)}$	95.00	$17.76_{\pm(12.45)}$	$0.82_{\pm(0.15)}$	$1.16_{\pm(0.76)}$
	TextGrad 🔵	94.04	$21.61_{\pm(11.30)}$	$0.75_{\pm(0.13)}$	$19.94_{\pm(22.32)}$	95.49	$17.07_{\pm(9.57)}$	$0.81_{\pm(0.10)}$	$3.75_{\pm(2.83)}$
	CWBA 🔴	72.92	$8.55_{\pm(3.78)}$	$0.53_{\pm(0.26)}$	$33.81_{\pm(33.86)}$	49.84	$8.88_{\pm(3.94)}$	$0.65_{\pm(0.17)}$	$56.35_{\pm(46.42)}$
	(Pruthi et al., 2019) 🔴	90.94	$2.22_{\pm(1.35)}$	$0.85_{\pm(0.14)}$	$4.86_{\pm(4.02)}$	92.93	$2.52_{\pm(1.57)}$	$0.84_{\pm(0.14)}$	$5.29_{\pm(4.68)}$
	`Charmer-Fast` (Ours) 🔴	**100.00**	$\underline{1.74}_{\pm(1.02)}$	$0.89_{\pm(0.13)}$	$\underline{0.34}_{\pm(0.31)}$	$\underline{99.39}$	$\underline{2.29}_{\pm(1.53)}$	$0.84_{\pm(0.15)}$	$\underline{0.47}_{\pm(0.49)}$
	`Charmer` (Ours) 🔴	**100.00**	$\mathbf{1.47}_{\pm(0.74)}$	$\underline{0.90}_{\pm(0.11)}$	$1.27_{\pm(0.84)}$	**99.51**	$\mathbf{1.76}_{\pm(1.12)}$	$\underline{0.89}_{\pm(0.12)}$	$1.52_{\pm(1.25)}$

Table 2: **Attack evaluation in Llama 2-Chat 7B.** `Charmer -Fast` outperforms baselines in terms of attack success rate, Levenshtein distance, and achieves comparable similarity and speed.

	SST-2				QNLI				RTE			
Method	ASR (%)	$d_{\text{lev}}(S, S')$	Sim(S, S')	Time	ASR (%)	$d_{\text{lev}}(S, S')$	Sim(S, S')	Time	ASR (%)	$d_{\text{lev}}(S, S')$	Sim(S, S')	Time
BAE-R 🔵	60.13	10.55	0.82	2.31	47.16	10.36	**0.95**	2.46	66.02	6.96	**0.88**	1.39
BERT-attack 🔵	57.86	12.05	$\underline{0.86}$	1.61	60.30	14.07	0.91	2.72	$\underline{90.78}$	8.77	0.82	1.41
DeepWordBug 🔴	50.82	$\underline{5.24}$	0.73	**1.01**	49.93	$\underline{3.86}$	0.88	**1.24**	50.97	$\underline{2.67}$	0.76	**0.61**
TextBugger 🔴	41.89	8.99	**0.89**	1.63	58.92	10.59	0.91	$\underline{2.44}$	79.61	7.76	0.80	$\underline{1.19}$
TextFooler 🔵	$\underline{85.79}$	20.91	0.79	3.54	$\underline{64.04}$	18.03	0.91	4.05	86.41	8.92	$\underline{0.84}$	1.73
`Charmer-Fast` 🔴	**95.47**	**2.55**	0.83	$\underline{1.47}$	**93.51**	**2.40**	$\underline{0.93}$	5.66	**97.10**	**1.68**	0.82	2.06

3.1 SELECTING THE NUMBER OF POSITIONS

To select the appropriate number of candidate positions n for Algorithm 1, we evaluate the ASR and runtime of the attack with $n \in \{1, 5, 10, 20, 30, 40, 50, 60, 70\}$. We conduct the experiment with the fine-tuned BERT on SST-2 from TextAttack at $k = 1$. For the SST-2 test sentences, the maximum number of positions across the dataset is 489 and the average is 213.72. We would like a value of n much smaller than these values. As a comparison, we report the ASR computed by exploring all possible positions (ASR Upper Bound). Additionally, to test the effect of our heuristic, we evaluate the performance when randomly selecting n positions (Random).

In Sec. 1, we can observe that the ASR consistently grows when increasing the number of candidate positions. However, the increase is less noticeable for $n > 20$, therefore, the increase in runtime does not pay off the increase in ASR. This leads us to choose $n = 20$ for the rest of our experiments. When compared with the random position selection, our method greatly improves the ASR for all the studied n, while introducing a minor time increase of 0.25 seconds on average.

3.2 COMPARISON AGAINST STATE-OF-THE-ART ATTACKS

In Tables 1 and 2, we can observe `Charmer` consistently achieves the highest ASR with $> 93\%$ in every benchmark. At the same time, our method obtains the lowest Levenshtein distance (d_{lev}). Regarding the similarity (Sim), our `Charmer` attains the runner up similarity in all cases, proving its ability to generate highly similar adversarial examples. With respect to time, `Charmer` is not as fast as the simple DeepWordBug, however, the runtime is comparable to previous state-of-the-art token-level TextGrad. If speed is preferred to adversarial example quality, we can set $n = 1$ (`Charmer-Fast`), which attains a runtime closer to DeepWordBug at the cost of a higher d_{lev} and lower ASR. This phenomenon is aligned with the results of Sec. 3.1, as the ASR at $k = 1$ is lower when n is lower.

3.3 ADVERSARIAL TRAINING

In this section, we analyze the performance of models trained with adversarial training defenses (Madry et al., 2018). Following the insights of Hou et al. (2023), we use the TRADES objective (Zhang et al., 2019). We compare the use of a token-level attack, TextGrad, v.s. a character-level attack, `Charmer`, for solving the inner maximization problem. We use TextGrad with the recommended hyperparameters for training and `Charmer` with the standard hyperparameters and $k = 1$. Every 100 training steps, we measure the clean, TextFooler and `Charmer` ($k = 1$) adversarial accuracies. We train on 5 random initializations of BERT-base (Devlin et al., 2019) for 1 epoch in SST-2.

Table 3: **Adversarial Training defenses:** `Charmer` is an effective defense against character-level attacks, minimally affects clean accuracy and does not improve token-level robustness. On the contrary, TextGrad hinders character-level robustness and clean accuracy to improve token-level robustness.

Method	Acc. (%) ↑	ASR-Char (%) ↓	ASR-Token (%) ↓
Standard	**92.43**	64.02	95.16
`Charmer` ●	87.20$_{\pm(1.34)}$	**20.34**$_{\pm(1.17)}$	95.17$_{\pm(1.15)}$
TextGrad ●	80.94$_{\pm(0.60)}$	67.34$_{\pm(4.87)}$	**71.36**$_{\pm(3.63)}$

In Table S7 we can firstly observe that both the TextGrad and `Charmer` defenses improve the token-level and character-level robustness respectively when compared with the standard training baseline. This was expected as this is the objective each method is targetting. Interestingly, `Charmer` does not improve the token-level robustness and TextGrad hinders the character-level robustness. This observation is confirmed when looking at the training evolution in Fig. S4. It remains open to know if we should aim at character or token level robustness, nevertheless our results indicate character-level robustness is less conflicted with clean accuracy.

3.4 BYPASSING TYPO CORRECTORS

We analyze the performance of our attack with and without the `PJC` constraints, see Appendix E. We train the strongest typo-corrector (Pruthi et al., 2019) and use it in front of the BERT-base model from TextAttack. For the robust encoding defense we train a BERT-base model over the agglomerative clusters (Jones et al., 2020).

Table 4: **Robust word recognition defenses:** `Charmer` is able to break the studied defenses with 100% ASR. Robust word recognition defenses are effective only when considering `PCJ` constraints.

Defense	Acc. (%)	`PJC`?	ASR (%)	$d_{\text{lev}}(S, S')$	Sim(S, S')
None	92.43	✗	100.00	1.47$_{\pm(0.74)}$	0.90$_{\pm(0.11)}$
		✓	96.65	1.86$_{\pm(1.14)}$	0.87$_{\pm(0.14)}$
(Pruthi et al., 2019)	88.53	✗	100.00	1.28$_{\pm(0.51)}$	0.90$_{\pm(0.11)}$
		✓	70.34	2.08$_{\pm(1.49)}$	0.85$_{\pm(0.14)}$
(Jones et al., 2020)	83.94	✗	100.00	1.43$_{\pm(0.71)}$	0.88$_{\pm(0.11)}$
		✓	0.96	1.14$_{\pm(0.38)}$	0.92$_{\pm(0.06)}$

In Table 4, we can observe that `Charmer` attains 100% ASR when not considering the `PJC` constraints. It is only when considering `PJC` that robust word recognition defenses are effective. In Table S15 we analyze the effect of relaxing each of the `PJC` constraints while keeping the rest. We observe that by relaxing any of the `LowEng`, `End` or `Start` constraints, performance grows considerably for both defenses, e.g., from 0.96% to 98.09% ASR when relaxing `LowEng` in the robust encoding case. This result indicates that robust word recognition defenses provide a false sempsation of robustness. Together with the observations in Appendix E.1, we believe adversarial training based methods suppose a more promising avenue towards achieving character-level robustness.

4 CONCLUSION

We have proposed an efficient character-level attack based on a novel strategy to select the best positions to perturb at each iteration. Our attack (`Charmer`) is able to obtain close to 100% ASR both in BERT-like models and LLMs like Llama-2. `Charmer` defeats both token-based adversarial training defenses (Hou et al., 2023) and robust word recognition defenses (Pruthi et al., 2019; Jones et al., 2020). When integrated within adversarial training, our attack is able to improve the robustness against character-level attacks. We believe defending agains character-level attacks is an interesting open problem, with adversarial training posing as a promising avenue for defenses.

Acknowledgments

Authors acknowledge the constructive feedback of reviewers and the work of SET LLM @ ICLR 2024 workshop organizers. We thank Zulip[3] for their project organization tool. ARO - Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-24-1-0048. Hasler AI - This work was supported by Hasler Foundation Program: Hasler Responsible AI (project number 21043) SNF project – Deep Optimisation - This work was supported by the Swiss National Science Foundation (SNSF) under grant number 200021_205011. Fanghui Liu is supported by the Alan Turing Institute under the UK-Italy Trustworthy AI Visiting Researcher Programme.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

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

Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. A neural probabilistic language model. *Advances in neural information processing systems (NeurIPS)*, 2000.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 2017.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 *Advances in neural information processing systems (NeurIPS)*, 2020.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, 2017.

Nicholas Carlini, Milad Nasr, Christopher A. Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei Koh, Daphne Ippolito, Florian Tramèr, and Ludwig Schmidt. Are aligned neural networks adversarially aligned? In *Advances in neural information processing systems (NeurIPS)*, 2023.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2018.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[3] `https://zulip.com`

Minhao Cheng, Jinfeng Yi, Pin-Yu Chen, Huan Zhang, and Cho-Jui Hsieh. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *AAAI Conference on Artificial Intelligence*, 34, 2020.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The pascal recognising textual entailment challenge. In Joaquin Quiñonero-Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché Buc, editors, *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*. Springer Berlin Heidelberg, 2006.

Matt Davis. Psycholinguistic evidence on scrambled letters in reading, 2003. URL `https://www.mrc-cbu.cam.ac.uk/people/matt.davis/cmabridge/`.

Ambra Demontis, Marco Melis, Maura Pintor, Matthew Jagielski, Battista Biggio, Alina Oprea, Cristina Nita-Rotaru, and Fabio Roli. Why do adversarial attacks transfer? explaining transferability of evasion and poisoning attacks. In *28th USENIX security symposium (USENIX security 19)*, pages 321–338, 2019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019.

Salijona Dyrmishi, Salah Ghamizi, and Maxime Cordy. How do humans perceive adversarial text? a reality check on the validity and naturalness of word-based adversarial attacks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2018.

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

Siddhant Garg and Goutham Ramakrishnan. BAE: BERT-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015.

Antonio Gulli. Ag's corpus of news articles, 2005. URL `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`.

Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. Gradient-based adversarial attacks against text transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

Michael Held, Philip Wolfe, and Harlan P Crowder. Validation of subgradient optimization. *Mathematical programming*, 6:62–88, 1974.

Bairu Hou, Jinghan Jia, Yihua Zhang, Guanhua Zhang, Yang Zhang, Sijia Liu, and Shiyu Chang. Textgrad: Advancing robustness evaluation in NLP by gradient-driven optimization. In *International Conference on Learning Representations (ICLR)*, 2023.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. *AAAI Conference on Artificial Intelligence*, 2020.

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*, 2020.

Taku Kudo and John Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2018.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

Deokjae Lee, Seungyong Moon, Junhyeok Lee, and Hyun Oh Song. Query-efficient and scalable black-box adversarial attacks on discrete sequential data via bayesian optimization. In *International Conference on Machine Learning*, pages 12478–12497. PMLR, 2022.

Qi Lei, Lingfei Wu, Pin-Yu Chen, Alex Dimakis, Inderjit S Dhillon, and Michael J Witbrock. Discrete adversarial attacks and submodular optimization with applications to text classification. *Proceedings of Machine Learning and Systems*, 1:146–165, 2019.

Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union, 1966.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. *Network and Distributed Systems Security (NDSS) Symposium*, 2019.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020.

Aiwei Liu, Honghai Yu, Xuming Hu, Shu'ang Li, Li Lin, Fukun Ma, Yawen Yang, and Lijie Wen. Character-level white-box adversarial attacks against transformers via attachable subwords substitution. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022.

Xiaogeng Liu, Nan Xu, Muhao Chen, and Chaowei Xiao. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv preprint arXiv:2310.04451*, 2023.

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 arXiv:1907.11692*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

Stoyan Mihov and Klaus U. Schulz. Fast approximate search in large dictionaries. *Computational Linguistics*, 30(4):451–477, 2004. doi: 10.1162/0891201042544938. URL https://aclanthology.org/J04-4003.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems (NeurIPS)*, 2013.

Petar Nikolaev Mitankin. Universal levenshtein automata. building and properties. *Sofia University St. Kliment Ohridski*, 2005.

John Morris, Eli Lifland, Jack Lanchantin, Yangfeng Ji, and Yanjun Qi. Reevaluating adversarial examples in natural language. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, 2020a.

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

Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. Scalable extraction of training data from (production) language models, 2023.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

David D Palmer. Tokenisation and sentence segmentation. *Handbook of natural language processing*, 2000.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018.

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*, 2019.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL `https://aclanthology.org/D16-1264`.

Graham Rawlinson. *The Significance of Letter Position in Word Recognition*. Phd thesis, Nottingham University, 1976.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Sahar Sadrizadeh, AmirHossein Dabiri Aghdam, Ljiljana Dolamic, and Pascal Frossard. Targeted adversarial attacks against neural machine translation. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023a.

Sahar Sadrizadeh, Ljiljana Dolamic, and Pascal Frossard. Transfool: An adversarial attack against neural machine translation models. *Transactions on Machine Learning Research*, 2023b. ISSN 2835-8856. URL `https://openreview.net/forum?id=sFk3aBNb81`.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

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 (EMNLP)*, 2013.

Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast WordPiece tokenization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems (NeurIPS)*, 27, 2014.

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

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 arXiv:2307.09288*, 2023.

Hélène Touzet. On the levenshtein automaton and the size of the neighbourhood of a word. In *Language and Automata Theory and Applications*, pages 207–218. Springer, 2016.

Eric Wallace, Mitchell Stern, and Dawn Song. Imitation attacks and defenses for black-box machine translation systems. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online, 2020. Association for Computational Linguistics.

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 *International Conference on Learning Representations (ICLR)*, 2019.

Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in NLP. In *COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics*, 1992.

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)*. Association for Computational Linguistics, 2018.

Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *Journal of Machine Learning Research*, 21(43):1–36, 2020. URL `http://jmlr.org/papers/v21/19-569.html`.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper_files/paper/2015/file/250cf8b51c773f3f8dc8b4be867a9a02-Paper.pdf`.

Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. Autodan: Automatic and interpretable adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*, 2023.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

Contents of the Appendix

In Appendix A we discuss the societal impact of our work. In Appendix D we provide additional background in NLP and the employed datasets. In Appendix E we provide additional experimental validation of `Charmer`. In Appendix G, we provide the proof of Corollary S6.

Notation: Sentences are sequences of characters in the set Γ. Sentences are denoted with uppercase letters S. For sets of sentences we use caligraphic uppercase letters $\mathcal{S}$. We denote the concatenation operator of two sentences as $\oplus$. The empty character $\emptyset$ is defined so that sentences remain invariant to concatenations with it, i.e., $S \oplus \emptyset = S$. We use the shorthand $[n]$ for $\{1, 2, \dots, n\}$ for any positive integer n. We use bold lowercase letters for vectors $\boldsymbol{x} \in \mathbb{R}^d$, with the i^{th} position being $x_i \in \mathbb{R}$.

A Broader impact

In this work, we revisit character-level adversarial attacks and improve upon the prior art performance. We believe that showing that character-level attacks cannot easily be defended, is important to warn about the need of defenses. Otherwise, malicious individuals or organizations could take advantage of this unawareness. However, we note that our algorithm could empower individuals to achieve malicious purposes. We will release our code to allow defenders to assess their performance against our attack.

B Related work

We provide an overview of adversarial attacks in NLP. Adversarial attacks have been devised for producing missclassifications (Alzantot et al., 2018), generating unfaithful machine translations (Cheng et al., 2020; Sadrizadeh et al., 2023a;b; Wallace et al., 2020), providing malicious outputs (jailbreaking) (Zou et al., 2023; Zhu et al., 2023; Carlini et al., 2023) or even extracting training data (Nasr et al., 2023). We distinguish these methods in two main branches: *token-level* and *character-level* attacks.

Token based: Early token-based attacks rely in black-box token replacement/insertion strategies based on heuristics for estimating the importance of each position, and the candidate tokens for the operation (Ren et al., 2019; Jin et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020; Lee et al., 2022). Ebrahimi et al. (2018) and Li et al. (2019) consider the token gradient information to select which token to replace. Guo et al. (2021) propose GBDA, the first, but inefficient, full gradient based text adversarial attack. TextGrad Hou et al. (2023) is a more efficient variante proposed to be integrated within Adversarial Training.

Character based: Belinkov and Bisk (2018) showcase that character-level Machine Translation models are sensible to natural character level perturbations (typos) and adversarially chosen ones. (Pruthi et al., 2019) propose to iteratively change the best possible character until success. However, this strategy can be inefficient for lengthy sentences. Addressing this issue, other methods propose pre-filtering the most important words/tokens in the sentence, to then introduce a random typo (Gao et al., 2018), or the best typo among a random sample (Li et al., 2019). In (Liu et al., 2022), a token-based attack with character-level Levenshtein distance constraints is considered. However, considering token-level information for assesing character-level importance can be suboptimal. Ebrahimi et al. (2018) propose involving embedding gradient information for evaluating the importance of characters, making the strategy only valid for character-level models. (Yang et al., 2020) evaluate the relevance of each character by masking and evaluating the loss in each position, to then modify the top positions. This strategy does not consider character insertions, and does not take into acount the effect of indivudual changes in the importance of positions. Similarly to (Yang et al., 2020), our method measures the importance of every position plus insertions. After a perturbation is done, importances are updated to consider the interaction between perturbations.

C Problem setting

In this section, we summarize the setting, Levenshtein distance and the operators used in our attack.

C.1 The sentence space

Let Γ be the alphabet set. A sentence S with l characters (i.e., the length $|S| = l$) in Γ is defined with $S = c_1 c_2 \cdots c_l \in \Gamma^l$. For notational simplicity, we denote $S_i = c_i$ as the character in the i^{th} position and $S_{i:} = c_i c_{i+1} \cdots c_l$ ($S_{:i} = c_1 c_2 \cdots c_i$) as the sequence obtained by taking the characters after (before) the i^{th} position included. We denote $\mathcal{S}(\Gamma)$ as the set of (all possible) sequences with characters in Γ with length less than L. Let $d_{\text{lev}} : \mathcal{S}(\Gamma) \times \mathcal{S}(\Gamma) \to \mathbb{R}^+$ be the Levenshtein distance (Levenshtein et al., 1966), also known as the edit distance. To be specific, for any two sentences $S, S' \in \mathcal{S}(\Gamma)$, the Levenshtein distance is defined as:

$$d_{\text{lev}}(S, S') := \begin{cases} |S| & \text{if } |S'| = 0 \\ |S'| & \text{if } |S| = 0 \\ d_{\text{lev}}(S_{2:}, S'_{2:}) & \text{if } S_1 = S'_1 \\ 1 + \min \begin{Bmatrix} d_{\text{lev}}(S_{2:}, S'_{2:}) \\ d_{\text{lev}}(S_{2:}, S') \\ d_{\text{lev}}(S, S'_{2:}) \end{Bmatrix} & \text{otherwise} . \end{cases}$$

Example S1 (d_{lev} form $S = \text{Hello}$ to several modifications.).

$$\begin{array}{cc} d_{\text{lev}}(\text{Hello}, \text{Helo}) = 1 & d_{\text{lev}}(\text{Hello}, \text{Hallo}) = 1 \\ d_{\text{lev}}(\text{Hello}, \text{Helloo}) = 1 & d_{\text{lev}}(\text{Hello}, \text{Haloo}) = 2 . \end{array}$$

Note that d_{lev} represents the cost in number of character *insertions*, *deletions* and *replacements* needed for S to become equal to S' or vice-versa.

C.2 Adversarial robustness

In this work, we tackle robustness restricted by the Levenshtein distance. This enables the search of highly similar, hard to detect and semantics-preserving adversarial examples (Morris et al., 2020a).

Definition S2 (k-robustness at S)**.** Denote the set of sentences at distance up to k as

$$\mathcal{S}_k(S, \Gamma) = \{S' \in \mathcal{S}(\Gamma) : d_{\text{lev}}(S, S') \leq k\} .$$

A learning model (e.g., neural networks) $f : \mathcal{S}(\Gamma) \to \mathcal{Y}$, where $\mathcal{Y}$ is the label space, is called k-robust at S if $f(S) = f(S'), \ \forall S' \in \mathcal{S}_k(S, \Gamma)$. If $f(S) \neq f(S')$ for some $S' \in \mathcal{S}_k(S, \Gamma)$, we say S' is an *adversarial example*.

Without loss of generality, we focus on the classification task. In the adversarial robustness community, adversarial examples are usually sought by solving some optimization problem (Carlini and Wagner, 2017). Given a data sample $(S, y) \in \mathcal{S}(\Gamma) \times [o]$ and a classification model $\boldsymbol{f} : \mathcal{S}(\Gamma) \to \mathbb{R}^o$, with o classes, we solve:

$$\max_{S' \in \mathcal{S}_k(S, \Gamma)} \mathcal{L}\left(\boldsymbol{f}(S'), y\right) , \tag{1}$$

where $\mathcal{L}$ is some loss function, e.g., the cross entropy loss. In the following, we elaborate on how Eq. (1) is solved.

C.3 Characterizing the perturbations

To explore adversarial examples in $\mathcal{S}_k(S, \Gamma)$, we make use of the contraction, expansion (Definition 2.1) and replacement operators (Definition 2.2) to characterize this set.

Thanks to Definitions 2.1 and 2.2, it is easy to check the following proposition.

Proposition S3 (Characterization of d_{lev}-1 operations)**.** *Let $S \in \mathcal{S}(\Gamma)$ be a non-empty sentence, and S' be another sentence satisfying $d_{lev}(S, S') = 1$. Then we can find $i \in [2|S| + 1]$ and a character $c \in \Gamma \cup \{\xi\}$ such that*

$$S' = \psi\left(\phi\left(S\right) \overset{i}{\leftarrow} c\right) .$$

Remark S4 (Non-uniqueness). The parametrization of the transformation from S to S' might not be unique. For example, for $S = \text{Hello}$ and $S' = \text{Helo}$, both pairs $(i = 6, c = \xi)$ and $(i = 8, c = \xi)$ are valid parametrizations.

Remark S5 (Intuition). Replacing a character in Γ for ξ, and applying ψ results in a deletion. Similarly, replacing a ξ by a character in Γ and applying ξ results in an insertion.

Corollary S6 (Generating $\mathcal{S}_k$)**.** *Let S be a non-empty sentence in the volcabulary, with $|\Gamma| > 1$, for any $k \geq 1$, the set $\mathcal{S}_k(S, \Gamma)$ (see Definition S2) can be obtained by the following recursion:*

$$\mathcal{S}_k(S,\Gamma) = \begin{cases} \left\{ \psi\left(\phi\left(S\right) \overset{i}{\leftarrow} c\right) \begin{array}{c} \forall i \in [2|S|+1] \\ \forall c \in \Gamma \cup \{\xi\} \end{array} \right\} & , \textit{if}\, k = 1, \\ \left\{ \psi\left(\phi\left(\hat{S}\right) \overset{i}{\leftarrow} c\right) \begin{array}{c} \forall i \in [2|\hat{S}|+1] \\ \forall c \in \Gamma \cup \{\xi\} \\ \forall \hat{S} \in \mathcal{S}_{k-1}(S,\Gamma) \end{array} \right\} & , \textit{if}\, k > 1\,. \end{cases}$$

The size of these sets is bounded as:

$$\frac{|\Gamma|^{k+1} - 1}{|\Gamma| - 1} \leq |\mathcal{S}_k(S, \Gamma)| \leq (|\Gamma| + 1)^k \cdot \prod_{j=1}^{k} (2(|S| + k) - 1)\,.$$

Remark S7. In the case $|\Gamma| = 1$, for any $S \in \mathcal{S}(\Gamma) : |S| \geq k$, we trivially have that $|\mathcal{S}_k(S, \Gamma)| = 2k + 1$.

Proof. Refer to Appendix G.

Note that exactly computing $|\mathcal{S}_k(S, \Gamma)|$ is non-trivial and complex dynamic programming algorithms have been proposed for this task (Mihov and Schulz, 2004; Mitankin, 2005; Touzet, 2016). The exponential dependence of $|\mathcal{S}_k(S, \Gamma)|$ on k makes it unfeasible to evaluate every sentence in the set, therefore, smarter strategies are needed.

D Additional background

We introduce additional information of the employed datasets in Appendix D.1 and about Neural Networks (NNs) for NLP in Appendix D.2.

D.1 Datasets

In Table S6 we provide the size, classes, alphabet and examples for all the studied datasets. All of our datasets are publicly available in `https://huggingface.co/datasets`.

D.2 NN architectures for NLP

Unlike Computer Vision applications, where images can be directly fed into the NN, some pre-processing is needed in order to feed text into our models. A common practice is grouping characters into *tokens* (Webster and Kit, 1992; Palmer, 2000; Sennrich et al., 2015; Kudo and Richardson, 2018; Song et al., 2021) and assigning a vector representation (*embedding*) to each token in the text (Bengio et al., 2000; Mikolov et al., 2013; Pennington et al., 2014; Bojanowski et al., 2017). After a sequence of vector representations is obtained, an appropriate NN architecture can be used, e.g., RNNs or Transformers in an encoder and/or decoder fashion Sutskever et al. (2014); Peters et al. (2018); Devlin et al. (2019); Brown et al. (2020). Overall, the architecture will be:

$$f(S) = \hat{f}\left(\mathbf{G}\left(S\right)\boldsymbol{T}\right)\,,$$

where $\boldsymbol{E} = \mathbf{G}\left(S\right)\boldsymbol{T}$ is the embedding representation of the sequence, $\mathrm{G} : \mathcal{S}(\Gamma) \to \mathcal{S}(V_{\text{tok}})$ is the tokenizer with V_{tok} as the token vocabulary and $\boldsymbol{T} \in \mathbb{R}^{|V_{\text{tok}}| \times d}$ is the matrix containing the embeddings of each token row-wise.

Table S5: **Attack transferability:** Adversarial examples are generated in the *Source Model* to be evaluated in the *Target Model*. For both `Charmer` and TextFooler, the ASR is consideraby lower when the target model is different from the source model. We observe no clear difference in transfer attack performance between TextFooler and `Charmer`. MNLI-m and AG-News are the easiest and hardest datasets for generating transfer attacks respectively.

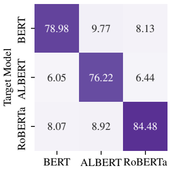
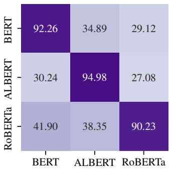
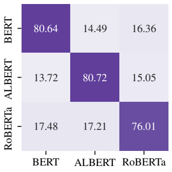
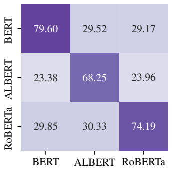
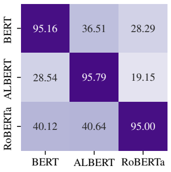
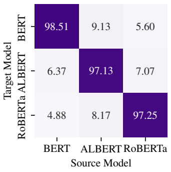
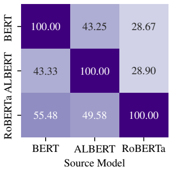
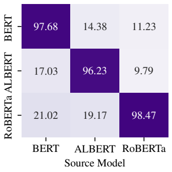
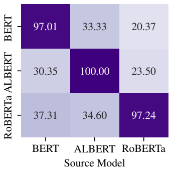
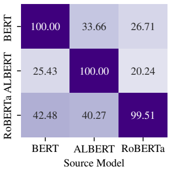
Attack	AG-News	MNLI-m	QNLI	RTE	SST-2
TextFooler					
`Charmer`					

Table S6: **Description of the employed datasets:** When a character is not printable in LaTeX, we default to its Unicode encoding.

AG-News

Test Size	1,000
Classes	4 (World, Sports, Business, Sci/Tech)
Alphabet $\lvert\Gamma\rvert = 82$	$\Gamma =${’ ’, ’!’, ’"’, ’#’, ’$’, ’&’, ’’’, ’(’, ’)’, ’*’, ’,’, ’-’, ’.’, ’/’, ’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’:’, ’;’, ’=’, ’?’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’G’, ’H’, ’I’, ’J’, ’K’, ’L’, ’M’, ’N’, ’O’, ’P’, ’Q’, ’R’, ’S’, ’T’, ’U’, ’V’, ’W’, ’X’, ’Y’, ’Z’, ’\’, ’_’, ’a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’}
Example	$S =$’Fears for T N pension after talks Unions representing workers at Turner Newall say they are ’disappointed’ after talks with stricken parent firm Federal Mogul.’, $y = 3$ (Business)

MNLI-m

Test Size	1,000
Classes	3 (Entailment, Neutral, Contradiction)
Alphabet $\lvert\Gamma\rvert = 81$	$\Gamma =${’ ’, ’!’, ’"’, ’$’, ’%’, ’&’, ’’’, ’(’, ’)’, ’,’, ’-’, ’.’, ’/’, ’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’:’, ’;’, ’?’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’G’, ’H’, ’I’, ’J’, ’K’, ’L’, ’M’, ’N’, ’O’, ’P’, ’Q’, ’R’, ’S’, ’T’, ’U’, ’V’, ’W’, ’X’, ’Y’, ’Z’, ’a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’, ’£’, ’é’, ’ô’}
Example	$S_{\text{premise}} =$’The new rights are nice enough’, $S_{\text{hypothesis}} =$’Everyone really likes the newest benefits’, $y = 2$ (Neutral)

QNLI

Test Size	1,000
Classes	2 (Entailment, Not entailment)
Alphabet $\lvert\Gamma\rvert = 233$	$\Gamma =${[’21513’, ’21068’, ’25104’, ’8722’, ’8211’, ’8212’, ’8216’, ’8217’, ’8220’, ’8221’, ’24605’, ’27166’, ’ ’, ’!’, ’"’, ’#’, ’$’, ’%’, ’&’, ’’’, ’(’, ’)’, ’8230’, ’+’, ’,’, ’-’, ’.’, ’/’, ’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’:’, ’;’, ’8243’, ’=’, ’>’, ’?’, ’<’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’G’, ’H’, ’I’, ’J’, ’K’, ’L’, ’M’, ’N’, ’O’, ’P’, ’Q’, ’R’, ’S’, ’T’, ’U’, ’V’, ’W’, ’X’, ’Y’, ’Z’, ’[’, ’8260’, ’]’, ’601’, ’_’, ’‘’, ’a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’, ’’, ’|’, ’’, ’ ’, ’20094’, ’642’, ’8838’, ’21129’, ’650’, ’38498’, ’7841’, ’£’, ’7845’, ’7847’, ’8364’, ’20140’, ’8366’, ’°’, ’7857’, ’±’, ’·’, ’½’, ’38081’, ’Å’, ’Ç’, ’712’, ’É’, ’7879’, ’Î’, ’720’, ’40657’, ’7889’, ’Ö’, ’×’, ’Ü’, ’ß’, ’à’, ’á’, ’ä’, ’1063’, ’æ’, ’ç’, ’è’, ’é’, ’ê’, ’í’, ’ï’, ’ð’, ’ñ’, ’ó’, ’38515’, ’õ’, ’ö’, ’ø’, ’ù’, ’8801’, ’û’, ’ü’, ’65279’, ’7940’, ’263’, ’268’, ’269’, ’272’, ’1072’, ’275’, ’27735’, ’281’, ’283’, ’30494’, ’30495’, ’8478’, ’1075’, ’287’, ’22823’, ’26408’, ’299’, ’26413’, ’815’, ’305’, ’1079’, ’1080’, ’321’, ’322’, ’20803’, ’324’, ’333’, ’1085’, ’27491’, ’1089’, ’34157’, ’7547’, ’1093’, ’379’, ’626’, ’928’, ’592’, ’37941’, ’39340’, ’941’, ’942’, ’943’, ’594’, ’945’, ’946’, ’947’, ’948’, ’949’, ’432’, ’951’, ’952’, ’953’, ’954’, ’955’, ’956’, ’596’, ’950’, ’959’, ’957’, ’961’, ’962’, ’964’, ’966’, ’23494’, ’8134’, ’969’, ’973’, ’603’, ’8172’, ’8242’]}
Example	$S_{\text{premise}} =$’What came into force after the new constitution was herald?’, $S_{\text{hypothesis}} =$’As of that day, the new constitution heralding the Second Republic came into force.’, $y = 1$ (Entailment)

RTE

Test Size	277
Classes	2 (Entailment, Not entailment)
Alphabet $\lvert\Gamma\rvert = 72$	$\Gamma =${’ ’, ’"’, ’$’, ’%’, ’&’, ’’’, ’(’, ’)’, ’,’, ’-’, ’.’, ’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’A’, ’B’, ’C’, ’D’, ’E’, ’F’, ’G’, ’H’, ’I’, ’J’, ’K’, ’L’, ’M’, ’N’, ’O’, ’P’, ’Q’, ’R’, ’S’, ’T’, ’U’, ’V’, ’W’, ’Y’, ’Z’, ’a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’}
Example	$S_{\text{premise}} =$’Dana Reeve, the widow of the actor Christopher Reeve, has died of lung cancer at age 44, according to the Christopher Reeve Foundation.’, $S_{\text{hypothesis}} =$’Christopher Reeve had an accident.’, $y = 2$ (Not entailment)

SST-2

Test Size	872
Classes	2 (Negative, Positive)
Alphabet $\lvert\Gamma\rvert = 55$	$\Gamma =${’æ’, ’à’, ’é’, ’ ’, ’!’, ’$’, ’%’, ’’’, ’(’, ’)’, ’,’, ’-’, ’.’, ’/’, ’0’, ’1’, ’2’, ’3’, ’4’, ’5’, ’6’, ’7’, ’8’, ’9’, ’:’, ’;’, ’?’, ’‘’, ’a’, ’b’, ’c’, ’d’, ’e’, ’f’, ’g’, ’h’, ’i’, ’j’, ’k’, ’l’, ’m’, ’n’, ’o’, ’p’, ’q’, ’r’, ’s’, ’t’, ’u’, ’v’, ’w’, ’x’, ’y’, ’z’, ’ö’}
Example	$S =$’it ’s a charming and often affecting journey .’, $y = 2$ (Positive)

E Additional experimental validation and details

In the text pair classification tasks (MNLI-m, RTE, and QNLI), we perturb only the *hypothesis* sentence. If the length of the test dataset is more than $1,000$, we restrict to the first $1,000$ samples. If a test dataset is not available for a benchmark, we evaluate in the validation dataset, this is a standard practice (Morris et al., 2020b). For comparison with other attacks, we use the default hyperparameters of those methods. For `Charmer` we use $n = 20$ positions (see Algorithm 1) and $k = 10$ except for AG-news where we use $k = 20$ because of the much longer sentences present in the dataset. `Charmer-Fast` simply takes $n = 1$ to speed-up the attack. For the alphabet Σ, in order to not introduce out-of-distribution characters, we take the characters present in each evaluation dataset. All of our experiments were conducted in a machine with a single NVIDIA A100 SXM4 GPU. For better illustration between token-level and character-level attacks, we mark them with 🔵 and 🔴 respectively.

We compare against the following state-of-the-art attacks **(i) token level:** BAE-R (Garg and Ramakrishnan, 2020), TextFooler (Jin et al., 2020), BERT-attack (Li et al., 2020), GBDA (Guo et al., 2021) and TextGrad (Hou et al., 2023), **(ii) character level:** DeepWordBug Gao et al. (2018), TextBugger (Li et al., 2019) and CWBA Liu et al. (2022). For each attack method, we evaluate the attach success rate (ASR), the average Levenshtein distance measured at character level ($d_{\text{lev}}(S, S')$) and the cosine similarity (Sim(S, S')) measured as in Guo et al. (2021), i.e., computing the cosine similarity of the USE encodings (Cer et al., 2018). We evaluate the performance of the attacks in the finetuned BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) from TextAttack. Additional experiments with ALBERT (Lan et al., 2020) can be found in Appendix E.

We additionally test the performance of the proposed method in Llama 2-Chat 7B (Touvron et al., 2023). Additional results on Vicuna 7B (Chiang et al., 2023) are deferred to Appendix E.6. Note that in the case of LLMs, the inference process is extremely costly. As a result, we only use the fast version of `Charmer`, i.e., $n = 1$. Moreover, we perform an additional position selection framework to further accelerate. Specifically, we first tokenize the input sentence and mask each token to determine the most important one based on the loss. Next, we perform Algorithm 1 for the position in these important tokens. An ablation study of such token selection procedure can be found in Appendix E.6.

We analyze the performance of our attack when attaking models defended by a typo-corrector (Pruthi et al., 2019), or a robust encoding module (Jones et al., 2020). We notice the success of these defenses can be attributed by the properties of the considered attacks. In Pruthi et al. (2019); Jones et al. (2020), the studied attacks are constrained to[4]:

- `NoRepeat`: Not perturb the same word twice.
- `First`: Not perturb the first character of a word.
- `Last`: Not perturb the last character of a word.
- `Length`: Not perturb words with less than 4 chars.
- `LowEng`: Only insert or replace for lowercase characters in the English alphabet.

A word is anything between blank spaces. We denote these as the *Pruthi-Jones Constraints* (`PJC`). While these constraints aim at preserving the meaning of *every individual word* (Rawlinson, 1976; Davis, 2003), in sentence classification, we might sacrifice the meaning of a word during the attack, if the global meaning of the sentence is preserved.

E.1 Adversarial Training

In this section, we analyze the performance of models trained with adversarial training defenses (Madry et al., 2018). Following the insights of Hou et al. (2023), we use the TRADES objective (Zhang et al., 2019). We compare the use of a token-level attack, TextGrad, v.s. a character-level attack, `Charmer`, for solving the inner maximization problem. We use TextGrad with the recommended hyperparameters for training and `Charmer` with the standard hyperparameters and $k = 1$. Every 100 training steps, we measure the clean, TextFooler and `Charmer` ($k = 1$) adversarial accuracies. We train on 5 random initializations of BERT-base (Devlin et al., 2019) for 1 epoch in SST-2.

[4] Pruthi et al. (2019), further constrain the attack by only considering replacements of nearby characters in the English keyboard.

Table S7: **Adversarial Training defenses:** `Charmer` is an effective defense against character-level attacks, minimally affects clean accuracy and does not improve token-level robustness. On the contrary, TextGrad hinders character-level robustness and clean accuracy to improve token-level robustness.

Method	Acc. (%) ↑	ASR-Char (%) ↓	ASR-Token (%) ↓
Standard	**92.43**	64.02	95.16
`Charmer` ●	87.20$_{\pm(1.34)}$	**20.34**$_{\pm(1.17)}$	95.17$_{\pm(1.15)}$
TextGrad ●	80.94$_{\pm(0.60)}$	67.34$_{\pm(4.87)}$	**71.36**$_{\pm(3.63)}$

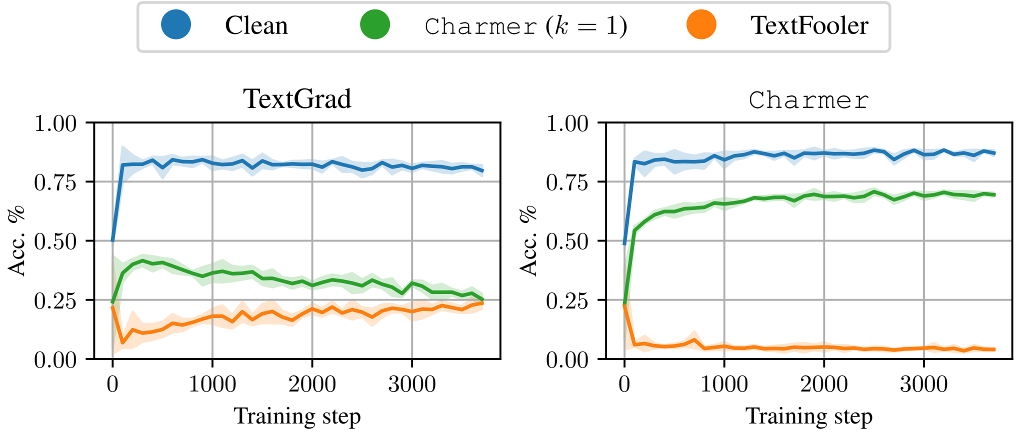


Figure S3: **Adversarial Training Evolution:** When employing `Charmer` as a defense, clean and character-level accuracies grow consistently through training steps, while token-level (TextFooler) accuracy is unimproved. The TextGrad defense consistently improves the token-level accuracy at the cost of hindering clean and character-level accuracy, which grow in the first ≈ 400 steps to then start decreasing.

In Table S7 we can firstly observe that both the TextGrad and `Charmer` defenses improve the token-level and character-level robustness respectively when compared with the standard training baseline. This was expected as this is the objective each method is targetting. Interestingly, `Charmer` does not improve the token-level robustness and TextGrad hinders the character-level robustness. This observation is confirmed when looking at the training evolution in Fig. S4. It remains open to know if we should aim at character or token level robustness, nevertheless our results indicate character-level robustness is less conflicted with clean accuracy.

E.2 Qualitative analysis of `Charmer`

In this section we analyze the characteristics of the perturbations introduced by `Charmer`.

Firstly, we display the most common operations for each dataset when attacking the corresponding TextAttack BERT model. In Fig. S5 we can observe that across datasets, the most common operations correspond to insertions of punctuation marks such as paranthesis, dots, commas, question marks, percentages or dollar symbols.

Secondly, we analyze the distribution of the location of perturbations across the sentences. From Fig. S6 on the one hand, we can conclude that there is no clear region where attacks are more common for the MNLI-m, RTE and SST-2 datasets. On the other hand, for AG-News and QNLI, perturbations appear to be more common closer to the beginning of the sentence. This inclination towards perturbations in AG-News, could be explained by the fact that most sentences in AG-News start with the news header, therefore, the model might be biased towards classifying based on the header.

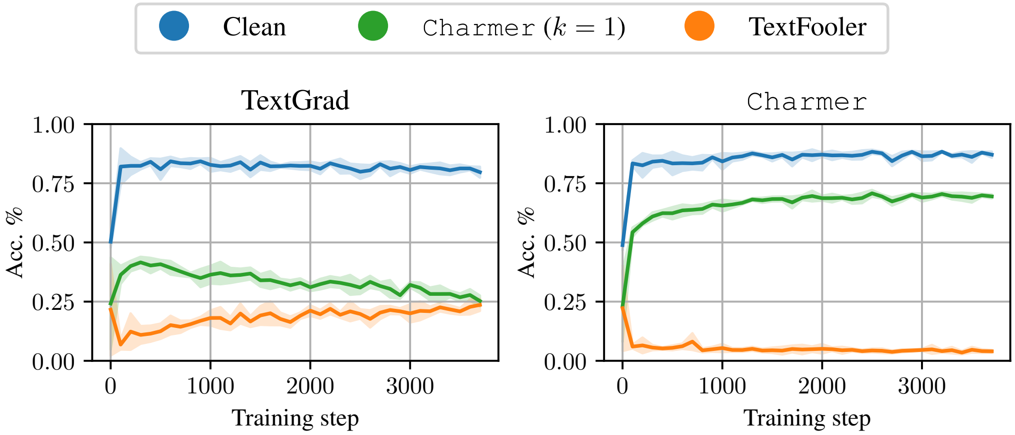


Figure S4: **Adversarial Training Evolution:** When employing `Charmer` as a defense, clean and character-level accuracies grow consistently through training steps, while token-level (TextFooler) accuracy is unimproved. The TextGrad defense consistently improves the token-level accuracy at the cost of hindering clean and character-level accuracy, which grow in the first ≈ 400 steps to then start decreasing.

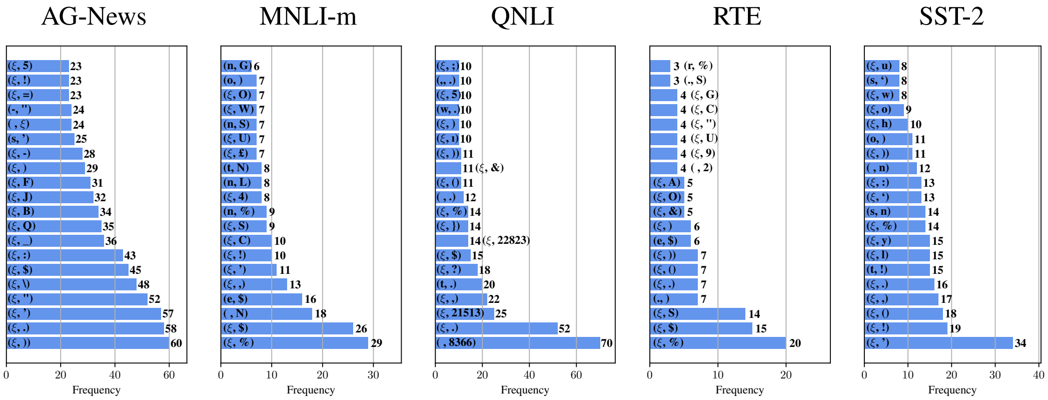


Figure S5: **Top 20 most common replacements with `Charmer`:** The pair of characters (c_1, c_2) indicates that c_1 is replaced by c_2 in the sentence. If $c_1 = \xi$, the replacement represents an insertion and if $c_2 = \xi$ the operation represents a deletion. The special character is denoted as ξ as the green character ξ did not appear in any of the most common operations. The most common operations are insertions of punctuation and special characters.

For completeness, we provide in Tables S8 to S12 the BERT adversarial examples provided by every attack in Table 1 in the first 3 correctly classified sentences for each dataset.

E.3 TEXTATTACK BASELINE

In this section we complement the analysis in Sec. 3.2 by reporting results for the TextAttack ALBERT models (Lan et al., 2020). Due to space reasons, we complement Table 1 with MNLI-m and RTE in Table S13

In Table S14, we can observe `Charmer` consistently attains the highest ASR among all the studied methods, while obtaining the lowest Levenshtein distance in $4/5$ cases and highest similarity in $3/5$ cases.

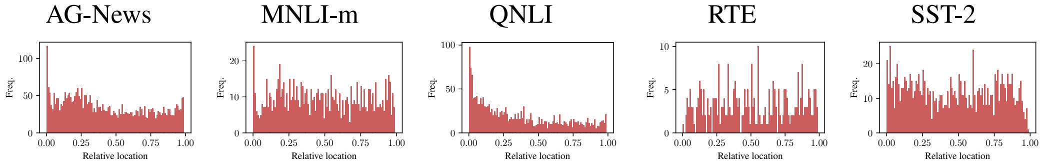


Figure S6: **Distribution of the relative location of perturbations in the sentence with Charmer:** 0 and 1 represent an insertion before the first character and after the last character in the sentence respectively. We do not observe any tendency in the QNLI, RTE and SST-2 datasets. For AG-News and QNLI, the perturbations in locations closer to 0 appear to be more common.

Table S8: **Attack examples in the first 3 sentences of AG-News.**

Method	Sentence	Prediction
Original	Fears for T N pension after talks Unions representing workers at Turner Newall say they are 'disappointed' after talks with stricken parent firm Federal Mogul.	2
TextBugger 🔴	Fears for T percent pension after conversationAssociations representing workers at Turner Newall say they are 'dsappointed' after talks with stricken parent firm Federal Mogul.	3
TextGrad 🔵	Fears for t n pension after chatcustomers representing hours at turner newall complain they are'disappointed'after chat with stricken parent provider federal mogul.	3
TextFooler 🔵	Fears for T percent pension after debateSyndicatesportrayal worker at Turner Newall say they are 'disappointed' after chatter with bereaved parenting corporationsCanada Mogul.	0
DeepWordBug 🔴	Fears for T N pension after alks Unions representing workers at Turner Newall say they are 'disppointed' after taclks with stricken parent firm GFederal Mogul.	2
BAE-R 🔵	Fears for T pl pension after talks Unions representing workers at Turner controls say they are 'disappointed' after talks with stricken parent firm Federal Mogul.	3
BERT-attack 🔵	Fears for T e pension after talks Unions representing workers at Turner Newall say they are 'disappointed, after discussion with stricken parent firm global Mogul.	3
GBDA 🔵	fears for t n pension after talks unions representing workers at turner newall say they are'disappointed'after talks with stricken parent knesset federal mogul'	0
CWBA 🔴	fe±rs for t n pen8314ion af1408er ta322ks un1603ons representing wo248yers at tu650ner newall say they are'disaxacalanted'after ta38525ks with stfucken par12459nt firm federal mogul.	3
(Pruthi et al., 2019) 🔴	Fears for T N pension after talks Unions representing workers at Tuurner Neqall say they are 'disappointed' after talks with stricken parent firm Federal Mogul.	0
Charmer 🔴	Fears for T E pension :fter talks Unions representing workers at Turner Newall say they are 'disappointed' after talks with stricken parent firm Federal Mogul.	3
Original	The Race is On: Second Private Team Sets Launch Date for Human Spaceflight (SPACE.com) SPACE.com - TORONTO, Canada - A second \team of rocketeers competing for the #36;10 million Ansari X Prize, a contest for\privately funded suborbital space flight, has officially announced the first\launch date for its manned rocket.	3
TextBugger 🔴	The Race is On: SegundoOwn Team Sets InitiateStardate for Humanitarian Spaceflight (SPACE.com) SAPCE.com - VANCOUVER, Canadian - per secs\team of rocketeers compete for the #36;10 m1110llion Asari X Prize, a contest for\secretly funded suborbital space flight, has solemn announced the first\lau1400ch date for its manned rocket.	3
TextGrad 🔵	The election is on: second private party sets eva date for human spacecapsule (moon.com) space.com - paris, canada -- a fourth\trio of rocketeers seeking for the #36;10 million ansari x prize, a contest for\privately dollar suborbitapollo space jump, has openly announced the first\launch date for its young child.	0
TextFooler 🔵	The Race is Around: Second Privy Remit Set Lanza Timeline for Humanitarian Spaceflight (SEPARATION.com) SEPARATION.com - CANADIENS, Countries - para second\squad of rocketeers suitors for the #36;10 billion Ansari X Nobel, a contestant for\covertly championed suborbital spaceship plane, had solemnly proclaim the first\begantimeline for its desolatebomb.	0
DeepWordBug 🔴	The Race is On: Second Private Tam Sets ZLaunch Date for HumJan Spaceflight (SPACE.com) SPACE.col - TORONTO, Canada - A second\team of rocketeers competing for the #3;1 million Asari X Priz, a contest for\privately ufnded suborbitIal space flight, has officially announced the first\launch Bate for its mannwed rocket.	3
BAE-R 🔵	The Race is On: current Private Team Sets launches Date for Human Spaceflight (SPACE.com) SPACE.com - TORONTO, sa--canada second\jury of rocketeers competing for the #36;10 million Ansari X s, a contest for\privately financed suborbital space flight, has just announced the first\launch date for its proposed rocket.	3
BERT-attack 🔵	The Race is On: Second Private Team Sets landing Date for earth Spaceflight (mars.com) SPACE.com - TORONTO, Canada - is current\team of rocketeers competing for the #36;10 million Ansari X quest, a contest for\privately financed suborbital space launch, has officially announced the first\landingnumber for its apollo rocket.	3
GBDA 🔵	the race is on : second private team sets launch date for human spaceflight (barack. com) continents. com - newsweek, cuba - - a second \team of rocketeers competing for the # 36 ; 10 million ansari x prize, a contest for \privately funded suborbital space flight, has officially announced the first \launch date for its manned rocket.	0
CWBA 🔴	t20234e rauee is on : sec601nd private te4536m sets lau1410ch da1746e for human space969light (sp*ce. c2m) sp64257ce. c699m - tor8482nto, can8482da - - a second \team of rockiaeers com0sting for the # 36 ; 10 million ansari x pr3936ze, a contest for \privately fun24179ed suborbital space flight, has offeldally announced the first \launch date for its man2488ed roc20986et.	3
(Pruthi et al., 2019) 🔴	The Race is On: Second Privqte Team Sets Launch Dajte for Himan Spacefight (SPADE.com) SPADE.com - TODONTO, Cahada - A sdcond\tam of rocketees copmeting for the #36;10 million Anqari X Pfize, a cntest for\privately funedd sublorbital space fight, has offically announced the first\launch date for its mabned rocket.	3
Charmer 🔴	The Race is On: SeZond Private Team Sets LaunchDate for ?uQan Spaceflight (SPACE/com) SPDACE.0om - TORONTO, Canada U-A second"team f rocketeers competing for the #36;10 million Ansari X Prize, a contest forDrivatelB funded suborbital space flight, has officially announced 'he first$launch date for its manned rocket.	0
Original	Ky. Company Wins Grant to Study Peptides (AP) AP - A company founded by a chemistry researcher at the University of Louisville won a grant to develop a method of producing better peptides, which are short chains of amino acids, the building blocks of proteins.	3
TextBugger 🔴	Ky. Compnay Wins Subsidies to Examine Ppetides (AP) APS - A company based by a chemist research1077r at the Academia of Indianapolis won a grant to develop a methodology of production best peptides, which are brief string of amino aids, the building block of prote1110ns.	3
TextGrad 🔵	Ky. company wins awarded to treat peptides (ab) ao - a company founded by a chemistry researcher at the time of louisville won a lead to develop a treatment of producing greater peptides, which are short chains of fatty acids, the basis blocks of muscle.	2
TextFooler 🔵	Ky. Businesses Wins Grant to Study Peptides (HAS) HAS - A company founded by a chemistry researcher at the University of Louisville won a grant to develop a method of producing better peptides, which are short chains of amino acids, the building blocks of proteins.	2
DeepWordBug 🔴	Ky. Comapny Wins Grant to SWtudy Peptieds (A) AP - A company foRnded by a cFhemistry researchfer at the Univrsity of Louisvlle won a grant to devIelop a Qethod of proucing bette peptides, which are short Shains of amino acids, the bunlding blocks of profteins.	2
BAE-R 🔵	Ky. mit Wins Grant to Study Peptides (AP) AP - biotechnology company founded by a chemistry researcher at the University of Louisville won a grant to study a method of producing better peptides, which are short chain of amino acids, the building blocks of genes.	3
BERT-attack 🔵	Ky. university Wins commission to research Peptides (AP) AP - A company owned by a chemistry lecturer at the University of ky won a grant to research a method of research better peptides, which are short chains of amino acids, the building blocks of protein.	3
GBDA 🔵	ky. company wins grant to study peptides (ap) ap - a company founded by a chemistry researcher at the university of louisville won a grant to develop a method of producing better peptides, which are short chains of amino acids, the building blocks of proteins.	3
CWBA 🔴	ky. com1405any wi2327s gr9824nt to stjdy pep65293ides (ap) ap - a company fzonded by a chehritry ressefcher at the unfeitsity of louis^{d}ille w^{a}n a gr12449nt to devefop a met12369od of producing better pep1110ides, which are short chains of am1074no ac1495ds, the building blocks of profcins.	1
(Pruthi et al., 2019) 🔴	Ky. Compahy Wins Grant to Stuwy Peptdies (AP) AP - A company founded by a chemistry researcher at the UQniversity of Louisville won a grant to develop a method of producing better pepgides, which are short chains of amino aids, the building blocks of proheins.	1
Charmer 🔴	Ky. Company Wins Grant to StuJdy Peptides (AP) AFP - A company founded by a chemistry researcher at the University of Louisville won a grant to develop a method of producing better peptides, which are short chains of amino acids, the building blocks of proteins.	0

Table S9: **Attack examples in the first 3 sentences of MNLI-m.**

Method	Sentence	Prediction
Original	`Everyone really likes the newest benefits`	2
TextBugger 🔴	`Somebody really lies the newest benefits`	0
TextGrad 🔵	`Everyone really hates the newest benefits`	0
TextFooler 🔵	`Nobody really likes the newest benefits`	0
DeepWordBug 🔴	`Everyone really yikes the newest benefits`	0
BAE-R 🔵	`nobody really likes the newest benefits`	0
BERT-attack 🔵	`Everyone really hates the newest benefits`	0
GBDA 🔵	`everyone really likes the newest misery`	0
(Pruthi et al., 2019) 🔴	`Everyone really lies the newest benefits`	0
`Charmer` 🔴	`Everyone really Yikes the newest benefits`	0
Original	`The Government Executive articles housed on the website are not able to be searched.`	0
TextBugger 🔴	`The Government Executive articles housed on the websites are not able to be searched.`	2
TextGrad 🔵	`The government executive articles housed on the website are not able to be destroyed.`	2
TextFooler 🔵	`The Government Executive articles housed on the website are not incapable to be searched.`	2
DeepWordBug 🔴	`The Government Executive articles housed on the website are not able to be sarched.`	2
BAE-R 🔵	`The Government cabinet articles housed on the website are not likely to be searched.`	2
BERT-attack 🔵	`The Government Executive articles housed on the website are not to to be visited.`	2
GBDA 🔵	`the government executive articles housed on the website are not sure to be raided.`	2
(Pruthi et al., 2019) 🔴	`The Government Executive articles housed on the website are not able to be sarched.`	2
`Charmer` 🔴	`The Government Executive articles housed on the website are Got able to be searched.`	1
Original	`I like him for the most part, but would still enjoy seeing someone beat him.`	1
TextBugger 🔴	`I like him for the most part, but would still enjoy seeing someone beat him.`	1
TextGrad 🔵	`I like him for the most cent, but would never enjoy seeing someone beat him.`	0
TextFooler 🔵	`I like him for the most portion, but would still cherishes seeing someone conquering him.`	2
DeepWordBug 🔴	`I like him for the most art, but would still enjoy seeing someone beat him.`	2
BAE-R 🔵	`I like him for the most people, but would still enjoy seeing someone beat him.`	2
BERT-attack 🔵	`I like him for the most all, but would still enjoy seeing someone beat him.`	2
GBDA 🔵	`i like him for the most part, but howard always regrets seeing someone beat him.`	2
(Pruthi et al., 2019) 🔴	`I like him for the most part, but would still enjoy sewing someone beat him.`	2
`Charmer` 🔴	`I like him for the most p4art, but would still enjoy seeing someone beat him.`	2

Table S10: **Attack examples in the first 3 sentences of QNLI.**

Method	Sentence	Prediction
Original	As of that day, the new constitution heralding the Second Republic came into force.	0
TextBugger 🔴	As of that day, the new constitution heralding the Second Republics came into for1010e.	1
TextGrad 🔵	As of that day, the new constitution heralding the second republic registered into real.	1
TextFooler 🔵	As of that day, the new constitution heralding the Second Republics went into troupes.	1
DeepWordBug 🔴	As of that day, the new constitution heralding the Second Republic came into ofrce.	1
BAE-R 🔵	As of that document, the new constitution heralding the Second republic came into existence.	0
BERT-attack 🔵	As of that day, the new constitution heraldof the Second Republic came into real.	1
GBDA 🔵	as of that day, the new constitution heralding the second republic came into force.	0
(Pruthi et al., 2019) 🔴	As of that day, the new constitution heralding the Second Republic came into forde.	1
Charmer 🔴	As of that day, the new constitution heralding the Second Republic came into for$ce.	1
Original	The most important tributaries in this area are the Ill below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	1
TextBugger 🔴	The most important tributaries in this areas are the Ill below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	0
TextGrad 🔵	The most important tributaries in this area are the ill below of strasbourg, the neckar in mannheim and the main across from cincinnati.	0
TextFooler 🔵	The most important tributaries in this areas are the Ill below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	0
DeepWordBug 🔴	The most important tributaries in this area are the IAl below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	0
BAE-R 🔵	The most important tributaries in this sector are the Ill below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	0
BERT-attack 🔵	The most important tributaries in this area are the far below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	0
GBDA 🔵	the most important tributaries in this area are the ill below of strasbourg, the neckar in mannheim and the jedi across from mainz.	0
(Pruthi et al., 2019) 🔴	The most important tributaries in this area are the Ill below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	1
Charmer 🔴	The most iðportant tributaries in this area are the Ill below of Strasbourg, the Neckar in Mannheim and the Main across from Mainz.	0
Original	In most provinces a second Bachelor's Degree such as a Bachelor of Education is required to become a qualified teacher.	1
TextBugger 🔴	In most provinces a s1077cond Bac1392elor's Degrees such as a Bachelo11397 of Education is required to become a qualified teacher.	0
TextGrad 🔵	In most provinces a minimum bachelor's degree such as a bachelor of education is required to become a qualified teacher.	0
TextFooler 🔵	ing most provinces a second Bachelor's Grades such as a Diplomas of Tuition is required to become a qualified teacher.	0
DeepWordBug 🔴	I most provinces a qecond Bachelor's Wegree such as a BFchelor of ducation is required to beome a qualified teachr.	0
BAE-R 🔵	In most provinces a basic Bachelor's Degree such as a bachelor of studies is needed to become a qualified teacher.	1
BERT-attack 🔵	In most canadian a diploma bachelorthethe Degree such as a major of Education is required to become a qualified teacher.	0
GBDA 🔵	in most provinces a second bachelor's degree such as a bachelor of education is minimum to become a qualified teacher.	0
(Pruthi et al., 2019) 🔴	In most provinces a second Bachelor's Degree such as a Bachelor of Education is required to become a qualified teacher.	1
Charmer 🔴	In most provinces a2second Bachelor's Degree such as a Bachelor of Education is required to become a qualified teacher.	0

Table S11: **Attack examples in the first 3 sentences of RTE.**

Method	Sentence	Prediction
Original	`Christopher Reeve had an accident.`	1
TextBugger 🔴	`Christopher Reeve had an accident.`	1
TextGrad 🔵	`Christopher reeve had an career.`	0
TextFooler 🔵	`Valeria Reeve was an collisions.`	0
DeepWordBug 🔴	`Christopher Reeve had an accidnt.`	1
BAE-R 🔵	`karen Reeve had an accident.`	1
BERT-attack 🔵	`david Reeve had an stroke.`	0
GBDA 🔵	`christopher reeve had an stroke.`	0
(Pruthi et al., 2019) 🔴	`Christopher Reeve had an acciSdent.`	1
`Charmer` 🔴	`Christopher Reeve had an accidentS`	0
Original	`Pennsylvania has the biggest Amish community in the U.S.`	1
TextBugger 🔴	`Penn has the largest Amish community in the U.S.`	0
TextFooler 🔵	`Pennsylvania has the wide Amish community in the U.S.`	0
DeepWordBug 🔴	`Pennsylvania has the bigges Amish community in the U.S.`	1
BAE-R 🔵	`Pennsylvania has the large Amish community in the U.S.`	0
BERT-attack 🔵	`Pennsylvania has the huge Amish community in the U.S.`	1
GBDA 🔵	`pennsylvania has the strongest amish community in the state. nara geographical`	0
(Pruthi et al., 2019) 🔴	`Pennsylvania has the biggeat Amish community in the U.S.`	0
`Charmer` 🔴	`Pennsylvania has the biggeAt Amish community in the U.S.`	0
Original	`Security forces were on high alert after a campaign marred by violence.`	0
TextBugger 🔴	`Security forces were on high alert after a countryside marred by violence.`	1
TextGrad 🔵	`Security families were on high alert after a month marred by violence.`	1
TextFooler 🔵	`Security forces were on high alert after a countryside marred by violence.`	1
DeepWordBug 🔴	`Security forces were on high alert after a cmpaign marred by violence.`	1
BAE-R 🔵	`ransport force were on high alert after a campaigning marred by violence.`	1
GBDA 🔵	`security forces were on high alert after a campaign marred by marches.`	1
(Pruthi et al., 2019) 🔴	`Security forces were on high alert after a campairn marred by violence.`	1
`Charmer` 🔴	`Security forces were on high alert after a2campaign marred by violence.`	1

E.4 ATTACK TRANSFERABILITY

In this section we study the transferability of `Charmer` attacks. This is a widely studied setup in the computer vision community (Demontis et al., 2019). For each dataset, attack and model, we generate the attacked sentences and evaluate the ASR when using them for attacking other models. As a reference we take the best token-level method from Table 1, i.e., TextFooler.

In Table S5 we can observe both TextFooler and `Charmer` fail to produce high ASRs in the transfer attack setup. As a reference, the highest transfer ASR was 55.48% and was attained by `Charmer` in the MNLI-m dataset, with BERT as a Source Model and RoBERTa as the target model. We notice in the AG-News dataset it is considerably harder to produce transfer attacks, with the highest transfer ASR being 9.77% among all setups. We believe improving the ASR in the transfer attack setup is an interesting avenue.

E.5 ROBUST WORD RECOGNITION DEFENSES

To complete the analysis, we repeat the experiments in Sec. 3.4 in the RTE, MNLI-m and QNLI datasets[5].

In Table S15 we can observe a similar phenomenon as in Table S15, i.e., robust word encoding defenses only work when assuming the attacker adopts the `PJC` constraints. When either the `LowEng`, `Start` or `End` constrants are relaxed, the ASR considerably grows close to 100%. It is worth mentioning that in the RTE dataset, defending with Jones et al. (2020) results in `Charmer` without any constraints achieving only 65.93% ASR. Nevertheless, this defense degrades the clean accuracy to less than 50%. If the dataset is balanced, as RTE approximately is[6], a constant classifier can achieve 50% clean accuracy and 0% ASR. This fact shows the little value of the RTE defended model in Jones et al. (2020).

[5] The AG-News dataset is not studied in Pruthi et al. (2019); Jones et al. (2020).

[6] `https://huggingface.co/datasets/glue/viewer/rte/validation`

Table S12: **Attack examples in the first 3 sentences of SST-2.**

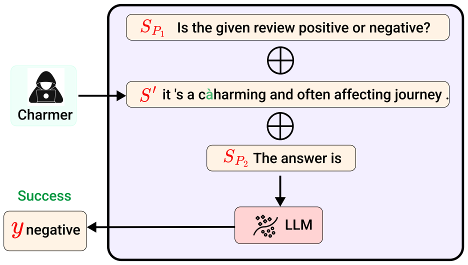
Method	Sentence	Prediction
Original	it 's a charming and often affecting journey .	1
TextBugger 🔴	it 's a ch593rming and often affecting voyage .	1
TextGrad 🔵	it's a dangerous and often affecting travelling.	0
TextFooler 🔵	it 's a cutie and often afflicts journey .	0
DeepWordBug 🔴	it 's a Wcharming and otfen affceting journey .	0
BAE-R 🔵	it 's a dark and often winding journey .	0
BERT-attack 🔵	it 's a one and often another journey .	1
GBDA 🔵	it's a colourful and not affecting journey.	0
CWBA 🔴	it's a char640ing a→d ofʷen affe37070ting jo1657ney.	0
(Pruthi et al., 2019) 🔴	it 's a chrming and often acfecting journey .	0
Charmer 🔴	it 's a %harming and often affecting journey .	0
Original	unflinchingly bleak and desperate	0
TextBugger 🔴	unflinchingly somber and desperate	1
TextGrad 🔵	unflinchingly dark and desperate	1
TextFooler 🔵	unflinchingly eerie and desperate	1
DeepWordBug 🔴	unflinchingly blak and despertae	1
BAE-R 🔵	unflinchingly happy and desperate	1
BERT-attack 🔵	unflinchingly dark and desperate	1
GBDA 🔵	unflinchingly picturesque and desperate	1
CWBA 🔴	unfl30340aringly byaak ayod deshilarate	1
(Pruthi et al., 2019) 🔴	unflinchingly beak and deseprate	1
Charmer 🔴	unflinchingly àbleak and desperate	1
Original	allows us to hope that nolan is poised to embark a major career as a commercial yet inventive filmmaker .	1
TextBugger 🔴	allows nous to hope that nolan is poised to embark a major career as a commercial however invntive cinematographers .	0
TextGrad 🔵	allows us to argue that nolan is ineligible to embark a major career as a commercial fails inventive filmmaker.	0
TextFooler 🔵	allows ourselves to hope that nolan is poised to embarked a severe career as a commercial yet noveltysuperintendent .	0
DeepWordBug 🔴	allows Gs to hope that nolan is Loised to embark a major career as a commercial yet invewntive filmmaker .	0
BAE-R 🔵	allows it to hope that nolan is poised to assume a major career as a commercial yet amateur filmmaker .	1
BERT-attack 🔵	allows to to hope that nolan is eligible to embark a major career as a commercial yet inventexperienced filmmaking .	0
GBDA 🔵	allows us to doubt that nolan is poised to embark a major career as a commercial lower inventive writer.	0
CWBA 🔴	all21335ws us to ho26954e that nolan is po2313sed to embark a major career as a commbecial y1705t inventive filmmaker.	0
(Pruthi et al., 2019) 🔴	allows us to hope that nolan is poised to ebark a major career as a commercial yet infentive filmmaker .	0
Charmer 🔴	allows us to hope that no$an is poised to embark a major career as a commercial yet inventive filmmaker .	0

Table S13: **Attack evaluation in the TextAttack BERT and RoBERTa models in QNLI, MNLI-m and RTE.**

		BERT				RoBERTa			
	Method	ASR (%) ↑	$d_{\text{lev}}(S, S')$ ↓	$\text{Sim}(S, S')$ ↑	Time (s) ↓	ASR (%) ↑	$d_{\text{lev}}(S, S')$ ↓	$\text{Sim}(S, S')$ ↑	Time (s) ↓
MNLI-m	GBDA 🔵	$\underline{97.97}$	$11.45_{\pm(6.52)}$	$0.73_{\pm(0.16)}$	$11.68_{\pm(3.23)}$	-	-	-	-
	BAE-R 🔵	70.00	$6.46_{\pm(3.32)}$	$\underline{0.83}_{\pm(0.15)}$	$0.53_{\pm(0.44)}$	67.39	$6.47_{\pm(3.31)}$	$\mathbf{0.84}_{\pm(0.14)}$	$0.54_{\pm(0.37)}$
	BERT-attack 🔵	92.41	$6.95_{\pm(6.57)}$	$\underline{0.83}_{\pm(0.13)}$	$26.53_{\pm(204.23)}$	$\underline{97.62}$	$6.56_{\pm(4.16)}$	$\underline{0.83}_{\pm(0.14)}$	$2.60_{\pm(15.58)}$
	DeepWordBug 🔴	84.88	$2.30_{\pm(1.68)}$	$0.75_{\pm(0.18)}$	$\underline{0.23}_{\pm(0.12)}$	78.41	$2.79_{\pm(2.02)}$	$0.71_{\pm(0.21)}$	$\underline{0.27}_{\pm(0.15)}$
	TextBugger 🔴	85.36	$4.17_{\pm(4.33)}$	$\underline{0.83}_{\pm(0.13)}$	$0.44_{\pm(0.32)}$	86.36	$5.41_{\pm(5.40)}$	$0.80_{\pm(0.15)}$	$0.51_{\pm(0.38)}$
	TextFooler 🔵	92.26	$9.83_{\pm(6.87)}$	$0.82_{\pm(0.14)}$	$0.52_{\pm(0.41)}$	90.23	$10.50_{\pm(7.79)}$	$0.81_{\pm(0.14)}$	$0.54_{\pm(0.42)}$
	TextGrad 🔵	93.69	$9.98_{\pm(5.66)}$	$0.75_{\pm(0.13)}$	$2.50_{\pm(1.97)}$	95.44	$9.10_{\pm(5.30)}$	$0.79_{\pm(0.12)}$	$3.56_{\pm(2.83)}$
	(Pruthi et al., 2019) 🔴	57.62	$1.32_{\pm(0.64)}$	$0.83_{\pm(0.12)}$	$4.48_{\pm(3.73)}$	52.84	$\underline{1.36}_{\pm(0.63)}$	$0.82_{\pm(0.13)}$	$7.48_{\pm(6.49)}$
	`Charmer-Fast` (Ours) 🔴	**100.00**	$\underline{1.23}_{\pm(0.58)}$	$\mathbf{0.85}_{\pm(0.14)}$	$\mathbf{0.21}_{\pm(0.17)}$	**100.00**	$\underline{1.36}_{\pm(0.78)}$	$0.82_{\pm(0.15)}$	$\mathbf{0.23}_{\pm(0.19)}$
	`Charmer` (Ours) 🔴	**100.00**	$\mathbf{1.14}_{\pm(0.42)}$	$\mathbf{0.85}_{\pm(0.13)}$	$1.45_{\pm(0.81)}$	**100.00**	$\mathbf{1.17}_{\pm(0.46)}$	$\mathbf{0.84}_{\pm(0.13)}$	$1.49_{\pm(0.82)}$
ONLI	GBDA 🔵	47.16	$11.88_{\pm(7.02)}$	$0.93_{\pm(0.06)}$	$13.85_{\pm(2.94)}$	-	-	-	-
	BAE-R 🔵	40.04	$11.44_{\pm(8.30)}$	$\mathbf{0.95}_{\pm(0.07)}$	$2.31_{\pm(2.36)}$	41.66	$10.37_{\pm(9.05)}$	$\mathbf{0.96}_{\pm(0.04)}$	$2.18_{\pm(2.77)}$
	BERT-attack 🔵	70.21	$16.21_{\pm(12.44)}$	$0.90_{\pm(0.08)}$	$239.86_{\pm(1395.15)}$	70.65	$17.78_{\pm(13.28)}$	$0.89_{\pm(0.12)}$	$2.70_{\pm(12.58)}$
	DeepWordBug 🔴	71.57	$4.52_{\pm(4.04)}$	$0.86_{\pm(0.15)}$	$\mathbf{0.50}_{\pm(0.34)}$	64.34	$5.07_{\pm(4.67)}$	$0.85_{\pm(0.17)}$	$\mathbf{0.59}_{\pm(0.41)}$
	TextBugger 🔴	75.77	$8.16_{\pm(9.93)}$	$0.89_{\pm(0.10)}$	$\underline{0.99}_{\pm(0.78)}$	67.39	$9.08_{\pm(10.31)}$	$0.90_{\pm(0.10)}$	$\underline{0.90}_{\pm(0.72)}$
	TextFooler 🔵	80.64	$23.42_{\pm(21.56)}$	$0.87_{\pm(0.12)}$	$1.90_{\pm(1.70)}$	76.01	$25.74_{\pm(28.53)}$	$0.87_{\pm(0.12)}$	$2.00_{\pm(2.09)}$
	TextGrad 🔵	77.35	$30.03_{\pm(20.41)}$	$0.82_{\pm(0.10)}$	$4.54_{\pm(3.82)}$	76.80	$21.56_{\pm(15.74)}$	$0.87_{\pm(0.08)}$	$5.80_{\pm(4.58)}$
	(Pruthi et al., 2019) 🔴	17.70	$\mathbf{1.57}_{\pm(0.81)}$	$0.93_{\pm(0.07)}$	$7.22_{\pm(4.91)}$	17.45	$\mathbf{1.54}_{\pm(0.88)}$	$\underline{0.93}_{\pm(0.08)}$	$7.46_{\pm(5.33)}$
	`Charmer-Fast` (Ours) 🔴	$\underline{94.69}$	$2.21_{\pm(1.69)}$	$0.93_{\pm(0.09)}$	$1.33_{\pm(1.55)}$	$\underline{96.95}$	$2.73_{\pm(2.15)}$	$0.90_{\pm(0.12)}$	$1.72_{\pm(2.27)}$
	`Charmer` (Ours) 🔴	**97.68**	$\underline{1.94}_{\pm(1.48)}$	$\underline{0.94}_{\pm(0.07)}$	$9.19_{\pm(9.60)}$	**97.86**	$\underline{2.20}_{\pm(1.69)}$	$0.92_{\pm(0.08)}$	$10.55_{\pm(9.69)}$
RTE	GBDA 🔵	76.62	$8.99_{\pm(4.74)}$	$0.78_{\pm(0.13)}$	$16.71_{\pm(7.29)}$	-	-	-	-
	BAE-R 🔵	64.68	$6.98_{\pm(3.39)}$	$\underline{0.87}_{\pm(0.09)}$	$0.84_{\pm(0.65)}$	64.06	$6.11_{\pm(3.84)}$	$\mathbf{0.89}_{\pm(0.08)}$	$0.76_{\pm(0.69)}$
	BERT-attack 🔵	68.00	$10.06_{\pm(9.75)}$	$0.78_{\pm(0.19)}$	$23.67_{\pm(51.78)}$	31.34	$6.00_{\pm(3.96)}$	$0.86_{\pm(0.08)}$	$5.36_{\pm(20.47)}$
	DeepWordBug 🔴	65.67	$1.64_{\pm(0.82)}$	$0.85_{\pm(0.10)}$	$\mathbf{0.12}_{\pm(0.03)}$	62.67	$1.83_{\pm(1.09)}$	$0.82_{\pm(0.14)}$	$\mathbf{0.13}_{\pm(0.04)}$
	TextBugger 🔴	74.13	$3.38_{\pm(3.48)}$	$\mathbf{0.88}_{\pm(0.09)}$	$0.35_{\pm(0.18)}$	71.43	$3.91_{\pm(3.73)}$	$\mathbf{0.89}_{\pm(0.08)}$	$\underline{0.38}_{\pm(0.40)}$
	TextFooler 🔵	79.60	$7.42_{\pm(6.10)}$	$\mathbf{0.88}_{\pm(0.09)}$	$0.47_{\pm(0.59)}$	74.19	$7.68_{\pm(5.94)}$	$\underline{0.88}_{\pm(0.09)}$	$0.47_{\pm(0.59)}$
	TextGrad 🔵	81.77	$10.02_{\pm(5.69)}$	$0.76_{\pm(0.13)}$	$2.44_{\pm(1.06)}$	73.97	$6.40_{\pm(3.30)}$	$0.84_{\pm(0.08)}$	$3.57_{\pm(3.61)}$
	(Pruthi et al., 2019) 🔴	62.19	$\mathbf{1.18}_{\pm(0.41)}$	$0.86_{\pm(0.08)}$	$8.45_{\pm(6.25)}$	49.31	$\mathbf{1.21}_{\pm(0.54)}$	$0.87_{\pm(0.08)}$	$12.23_{\pm(8.84)}$
	`Charmer-Fast` (Ours) 🔴	$\underline{89.55}$	$\underline{1.36}_{\pm(0.93)}$	$\underline{0.87}_{\pm(0.12)}$	$\underline{0.29}_{\pm(0.27)}$	$\underline{91.71}$	$1.78_{\pm(1.71)}$	$0.82_{\pm(0.15)}$	$0.41_{\pm(0.54)}$
	`Charmer` (Ours) 🔴	**97.01**	$1.55_{\pm(1.42)}$	$0.86_{\pm(0.13)}$	$2.50_{\pm(2.33)}$	**97.24**	$\underline{1.61}_{\pm(1.39)}$	$0.85_{\pm(0.13)}$	$2.74_{\pm(2.87)}$

E.6 ATTACK OF LLM CLASSIFIER

The prompt design in attacking different LLMs is summarized in Table S18. A schematic of attacking LLMs is present in Fig. S7. In this experiment, we use token-based position selection to further accelerate the process of attack. Specifically, we mask each token in the inputs and select the top ten tokens with the highest loss. Since some tokens consist of many characters, we only use 40 positions of these tokens to perform Algorithm 1. The remaining step is the same as in Algorithm 2. Notably, in Table S16, we see that such a process can significantly accelerate the attack while maintaining the performance of ASR and other metrics. The result on Vicuna 7B is present in Table S17, where we can see the proposed `Charmer` achieves much higher ASR than other baselines with less edit distance.



S_{P_1} Is the given review positive or negative?
⊕
Charmer
S' it 's a càharming and often affecting journey .
⊕
S_{P_2} The answer is
LLM
Success
y negative

Figure S7: Schematic of the proposed `Charmer` in attacking LLM-classifiers. `Charmer` modifies the input to S' with a small perturbation (annotated in green color) to the original input so that the model produces the desired output y. S_{P_1} and S_{P_2} are auxiliary prompts that remain unchanged during the attack.

Table S14: **Attack evaluation in the TextAttack ALBERT models:** Token-level and character-level attacks are highlighted with 🔵 and 🔴 respectively. for each metric, the best method is highlighted in **bold** and the runner-up in underlined. `Charmer` consistently achieves highest Attack Success Rate (ASR).

		ALBERT			
	Method	ASR (%) ↑	$d_{\text{lev}}(S, S')$ ↓	$\text{Sim}(S, S')$ ↑	Time (s) ↓
AG-News	CWBA 🔴	57.96	$22.69_{\pm(21.07)}$	$0.64_{\pm(0.22)}$	$205.69_{\pm(162.00)}$
	BAE-R 🔵	18.26	$15.15_{\pm(11.28)}$	$\mathbf{0.97}_{\pm(0.02)}$	$1.84_{\pm(1.70)}$
	BERT-attack 🔵	37.23	$21.34_{\pm(15.29)}$	$0.93_{\pm(0.05)}$	$2.41_{\pm(2.14)}$
	DeepWordBug 🔴	56.90	$9.77_{\pm(6.77)}$	$0.83_{\pm(0.14)}$	$\mathbf{0.73}_{\pm(0.40)}$
	TextBugger 🔴	71.76	$17.48_{\pm(16.82)}$	$0.91_{\pm(0.06)}$	$\underline{1.38}_{\pm(0.98)}$
	TextFooler 🔵	76.22	$46.51_{\pm(35.21)}$	$0.87_{\pm(0.10)}$	$3.89_{\pm(3.11)}$
	TextGrad 🔵	75.37	$42.43_{\pm(19.05)}$	$0.85_{\pm(0.07)}$	$8.23_{\pm(9.15)}$
	(Pruthi et al., 2019) 🔴	88.00	$5.50_{\pm(4.74)}$	$0.89_{\pm(0.13)}$	$29.17_{\pm(25.58)}$
	`Charmer-Fast` 🔴	$\underline{95.44}$	$\underline{3.25}_{\pm(2.88)}$	$\underline{0.95}_{\pm(0.06)}$	$2.38_{\pm(3.27)}$
	`Charmer` 🔴	**97.13**	$\mathbf{2.45}_{\pm(2.31)}$	$\mathbf{0.97}_{\pm(0.04)}$	$6.94_{\pm(12.33)}$
MNLI-m	BAE-R 🔵	71.57	$6.28_{\pm(3.27)}$	$0.84_{\pm(0.14)}$	$0.54_{\pm(0.34)}$
	BERT-attack 🔵	$\underline{97.50}$	$7.14_{\pm(5.17)}$	$0.84_{\pm(0.12)}$	$2.96_{\pm(16.16)}$
	DeepWordBug 🔴	85.90	$2.31_{\pm(1.63)}$	$0.77_{\pm(0.17)}$	$\underline{0.27}_{\pm(0.13)}$
	TextBugger 🔴	88.05	$4.86_{\pm(4.64)}$	$0.82_{\pm(0.13)}$	$0.55_{\pm(0.38)}$
	TextFooler 🔵	94.98	$9.49_{\pm(6.45)}$	$0.82_{\pm(0.13)}$	$0.55_{\pm(0.40)}$
	TextGrad 🔵	94.15	$8.96_{\pm(4.90)}$	$0.79_{\pm(0.12)}$	$2.33_{\pm(1.63)}$
	(Pruthi et al., 2019) 🔴	58.18	$1.26_{\pm(0.57)}$	$0.84_{\pm(0.11)}$	$5.14_{\pm(5.25)}$
	`Charmer-Fast` 🔴	**100.00**	$\underline{1.17}_{\pm(0.42)}$	$\underline{0.85}_{\pm(0.13)}$	$\mathbf{0.22}_{\pm(0.15)}$
	`Charmer` 🔴	**100.00**	$\mathbf{1.08}_{\pm(0.28)}$	$\mathbf{0.86}_{\pm(0.11)}$	$1.53_{\pm(0.70)}$
QNLI	BAE-R 🔵	45.97	$10.94_{\pm(7.57)}$	$\mathbf{0.95}_{\pm(0.06)}$	$2.52_{\pm(2.46)}$
	BERT-attack 🔵	73.40	$16.74_{\pm(16.94)}$	$0.90_{\pm(0.10)}$	$760.09_{\pm(5904.67)}$
	DeepWordBug 🔴	74.07	$5.00_{\pm(4.36)}$	$0.85_{\pm(0.16)}$	$\mathbf{0.57}_{\pm(0.43)}$
	TextBugger 🔴	76.03	$9.59_{\pm(11.48)}$	$0.90_{\pm(0.10)}$	$\underline{1.15}_{\pm(0.99)}$
	TextFooler 🔵	80.72	$22.56_{\pm(20.96)}$	$0.88_{\pm(0.11)}$	$2.13_{\pm(2.00)}$
	TextGrad 🔵	74.78	$28.47_{\pm(17.98)}$	$0.84_{\pm(0.09)}$	$5.78_{\pm(6.02)}$
	(Pruthi et al., 2019) 🔴	26.47	$\underline{1.85}_{\pm(1.18)}$	$0.93_{\pm(0.09)}$	$10.12_{\pm(8.49)}$
	`Charmer-Fast` 🔴	$\underline{96.19}$	$2.26_{\pm(1.74)}$	$0.93_{\pm(0.08)}$	$1.58_{\pm(2.36)}$
	`Charmer` 🔴	**96.23**	$\mathbf{1.78}_{\pm(1.11)}$	$\underline{0.94}_{\pm(0.07)}$	$9.60_{\pm(8.10)}$
RTE	BAE-R 🔵	61.14	$6.69_{\pm(3.43)}$	$\underline{0.88}_{\pm(0.09)}$	$0.82_{\pm(0.53)}$
	BERT-attack 🔵	9.80	$5.20_{\pm(2.95)}$	$0.86_{\pm(0.16)}$	$21.39_{\pm(34.86)}$
	DeepWordBug 🔴	59.24	$1.54_{\pm(0.84)}$	$0.84_{\pm(0.13)}$	$\mathbf{0.13}_{\pm(0.05)}$
	TextBugger 🔴	70.62	$4.45_{\pm(5.24)}$	$0.87_{\pm(0.11)}$	$0.44_{\pm(0.33)}$
	TextFooler 🔵	68.25	$7.60_{\pm(5.61)}$	$\mathbf{0.89}_{\pm(0.09)}$	$0.52_{\pm(0.65)}$
	TextGrad 🔵	70.70	$7.07_{\pm(3.25)}$	$0.83_{\pm(0.10)}$	$2.56_{\pm(2.28)}$
	(Pruthi et al., 2019) 🔴	48.34	$\mathbf{1.22}_{\pm(0.41)}$	$0.86_{\pm(0.09)}$	$11.56_{\pm(7.69)}$
	`Charmer-Fast` 🔴	$\underline{97.16}$	$1.68_{\pm(1.32)}$	$0.83_{\pm(0.14)}$	$\underline{0.42}_{\pm(0.44)}$
	`Charmer` 🔴	**100.00**	$\underline{1.29}_{\pm(0.65)}$	$0.87_{\pm(0.10)}$	$2.49_{\pm(2.13)}$
SST-2	CWBA 🔴	77.88	$11.18_{\pm(4.58)}$	$0.55_{\pm(0.25)}$	$58.28_{\pm(50.83)}$
	BAE-R 🔵	62.77	$10.25_{\pm(7.24)}$	$0.85_{\pm(0.16)}$	$0.78_{\pm(0.77)}$
	BERT-attack 🔵	72.34	$11.57_{\pm(6.89)}$	$0.85_{\pm(0.10)}$	$148.11_{\pm(1077.71)}$
	DeepWordBug 🔴	84.78	$3.37_{\pm(2.47)}$	$0.82_{\pm(0.16)}$	$\mathbf{0.23}_{\pm(0.12)}$
	TextBugger 🔴	72.52	$5.61_{\pm(5.51)}$	$\mathbf{0.91}_{\pm(0.06)}$	$1.85_{\pm(0.90)}$
	TextFooler 🔵	95.79	$15.79_{\pm(11.12)}$	$0.83_{\pm(0.14)}$	$1.11_{\pm(0.74)}$
	TextGrad 🔵	96.28	$18.67_{\pm(9.73)}$	$0.80_{\pm(0.11)}$	$2.95_{\pm(1.65)}$
	(Pruthi et al., 2019) 🔴	95.05	$1.98_{\pm(1.28)}$	$0.87_{\pm(0.13)}$	$4.66_{\pm(3.98)}$
	`Charmer-Fast` 🔴	$\underline{99.88}$	$\underline{1.62}_{\pm(0.88)}$	$\underline{0.89}_{\pm(0.12)}$	$\underline{0.38}_{\pm(0.34)}$
	`Charmer` 🔴	**100.00**	$\mathbf{1.38}_{\pm(0.67)}$	$\mathbf{0.91}_{\pm(0.10)}$	$1.38_{\pm(0.93)}$

Table S15: **Effect of each `PJC` constraint:** `Charmer` ASR when individually removing each constraint while keeping the rest. Performance with no constraints (`None`) put as reference. The ASR drastically increases when removing the `LowEng`, `End` or `Start` constraints, proving the fragility of existing robust word recognition defenses.

		SST-2		RTE		MNLI-m		QNLI	
Defense	Attack constraint	Acc. (%)	ASR	Acc. (%)	ASR (%)	Acc. (%)	ASR (%)	Acc. (%)	ASR (%)
(Pruthi et al., 2019)	`None`	88.53	100.00	60.36	92.17	76.33	100.00	73.55	86.38
	`PJC`		70.34		42.17		87.39		43.05
	`-LowEng`		99.22		70.48		97.90		65.53
	`-Length`		74.61		45.78		92.51		46.19
	`-End`		93.91		63.86		96.19		57.63
	`-Start`		98.58		69.88		97.11		57.08
	`-NoRepeat`		74.09		42.17		89.36		42.51
(Jones et al., 2020)	`None`	83.94	100.00	48.74	65.93	68.44	100.00	76.20	98.56
	`PJC`		0.96		2.96		2.93		1.05
	`-LowEng`		98.09		57.04		96.63		94.62
	`-Length`		0.96		2.96		2.93		0.79
	`-End`		71.72		44.44		75.26		54.99
	`-Start`		88.93		47.41		82.87		67.32
	`-NoRepeat`		5.46		6.67		9.81		3.81

Table S16: Ablation study on the time efficiency of different methods of position selection in Llama 2-Chat 7B. We choose the fast version of `Charmer` with $n = 1$ and $k = 10$. The result shows that combing Algorithm 1 with a token-based pre-selection procedure can notably improve the efficiency of the proposed `Charmer`.

	Method	ASR (%)	$d_{\text{lev}}(S, S')$	$\text{Sim}(S, S')$	Time
SST-2	`Charmer-Fast` (Token-based Algorithm 1)	95.47	**2.55**	0.83	**1.47**
	`Charmer-Fast` (Algorithm 1)	**95.60**	2.56	**0.85**	3.32
QNLI	`Charmer-Fast` (Token-based Algorithm 1)	93.51	2.40	0.93	**5.66**
	`Charmer-Fast` (Algorithm 1)	**96.82**	**2.34**	**0.94**	10.30
RTE	`Charmer-Fast`(Token-based Algorithm 1)	97.10	1.68	0.82	**2.06**
	`Charmer-Fast`(Algorithm 1)	**98.07**	**1.64**	**0.84**	2.63

E.7 Jailbreaking LLM

Disclaimer: this attack can lead to harmful content

In this section, we showcase that the proposed method can also be applied to jailbreaking LLMs, which refers to designing prompts to allow LLMs to output harmful content. We compare against the recent jailbreaking attacks GCG (Zou et al., 2023) and AutoDAN (Liu et al., 2023) in the AdvBench banchmark (Zou et al., 2023). First, we overview the formulation of jailbreaking LLM. Given a harmful query S, the goal of jailbreaking is to make the model answer certain query S even when the safety system prompt S_{system} is provided, see Fig. 1b for an illustration of S_{system}. Formally, our attacker aims to find an adversarial example S' based on S such that:

$$\max_{S' \in \mathcal{S}_k(S,\Sigma)} \mathcal{L}\left(f(S_{\text{system}} \oplus S'), y\right) := \max_{S' \in \mathcal{S}_k(S,\Sigma)} \mathbb{P}(y | S_{\text{system}} \oplus S')$$

where $\mathcal{L}$ is defined as the probability of generating the target $y :=$ "Sure, here is" $\oplus$ S. For example, if S is "how to make a bomb", then y is "Sure, here is how to make a bomb".

In this experiment, we use the Advbench dataset proposed by Zou et al. (2023), which consists of harmful and toxic sentences across different topics. Due to the costly computational budget of attacking LLMs, we only use the first 50 sentences in the dataset. We compare the proposed `Charmer` against the white-box attack GCG Zou et al. (2023) and black-box attack AutoDAN Liu et al. (2023). We adopt the same hyper-parameters of LLM and criterion of success as in Zou et al. (2023). The result in Table S19 shows that `Charmer` can pass the safety alignment process of LLMs with much less change in terms of Levenshtein distance.

Table S17: **Attack evaluation in Vicuna 7B:** We choose the fast version of `Charmer` with $n = 1$ and $k = 10$. `Charmer` outperforms baselines in terms of attack success rate, Levenshtein distance , and achieves comparative similarity and speed.

	Method	ASR (%)	$d_{\text{lev}}(S, S')$	$\text{Sim}(S, S')$	Time
QNLI	BAE-R 🔵	40.66	12.36	**0.96**	3.11
	BERT-attack 🔵	56.67	16.61	0.91	4.03
	DeepWordBug 🔴	43.77	3.63	0.91	**1.32**
	TextBugger 🔴	53.11	9.08	0.93	2.56
	TextFooler 🔵	51.28	20.70	0.92	4.76
	`Charmer-Fast` 🔴	**98.35**	**2.04**	0.94	4.89
RTE	BAE-R 🔵	64.11	5.96	**0.89**	1.23
	BERT-attack 🔵	82.78	8.92	0.82	1.60
	DeepWordBug 🔴	50.97	2.67	0.76	**0.61**
	TextBugger 🔴	71.77	6.63	0.84	1.12
	TextFooler 🔵	78.47	7.94	0.86	1.64
	`Charmer-Fast` 🔴	**89.05**	**1.56**	0.85	1.98
SST-2	BAE-R 🔵	43.22	14.29	0.75	3.28
	BERT-attack 🔵	32.31	13.21	0.85	2.42
	DeepWordBug 🔴	30.72	4.22	0.76	**0.88**
	TextBugger 🔴	23.01	9.97	0.88	1.73
	TextFooler 🔵	64.04	18.03	**0.91**	4.05
	`Charmer-Fast` 🔴	**91.89**	**2.47**	0.85	1.66

Table S18: Prompting in different LLMs and datasets. The sentences outside “[Input]” are considered as auxiliary prompts S_{P_1} and S_{P_2}, as demonstrated in Fig. S7.

Model	**Dataset**	**Prompt design**
Llama 2-Chat 7B	SST-2	Is the given review positive or negative? [Input] The answer is
	RTE	[Input premise] Based on the paragraph above can we conclude the following sentence, answer with yes or no. [Input hypothesis] The answer is
	QNLI	Does the sentence answer the question? Answer with yes or no. Question: [Input premise] Sentence: [Input hypothesis] The answer is
Vicuna 7B	SST-2	Analyze the tone of this statement and respond with either positive or negative: [Input] The answer is:
	RTE	[Input premise] Based on the paragraph above can we conclude the following sentence, answer with yes or no. [Input hypothesis] The answer is
	QNLI	[Input premise] Based on the question above, does the following sentence answer the question? [Input hypothesis] Answer with yes or no. The answer is

F Additional method details

F.1 Pre-selection of replacement locations

In Proposition S3, we consider all possible locations ($i \in [2 \cdot |S| + 1]$), leading to $|\mathcal{S}_1(S, \Gamma)| \leq (|\Gamma| + 1) \cdot (2 \cdot |S| + 1)$ by Corollary S6, which can be relatively big for lengthy sentences (e.g., $|S|$ can be up to 844 for AG-News). We propose considering a subset of n locations in order to remove the dependency on the length of the sentence. To select the top n locations, we propose testing the relevance of each position by replacing each character with a “test" character and looking at the

[7] In Vicuna 7B and Guanaco 7B, AutoDAN uses the initialized handcrafted prefix in jailbreaks successfully so that the time is 0. Our method can also work on top of these handcrafted prefixes.

Table S19: Attack evaluation in jailbreaking different large language models.

Model	Method	ASR (%)	$d_{\text{lev}}(S, S')$	Time
Vicuna 7B	GCG	100.00	35.26	56.32
	AutoDAN	100.00	3677	-[7]
	Charmer	100.00	**3.44**	**17.41**
Guanaco 7B	GCG	100.00	55.34	30.06
	AutoDAN	100.00	3677.00	- [7]
	Charmer	100.00	**4.98**	**24.99**
Llama 2-Chat 7B	GCG	86.00	76.74	948.44
	AutoDAN	18.00	3209.33	29.01
	Charmer	94.00	28.02	341.66

change in the loss. In Algorithm 1 we formalize our proposed strategy. Note that if the test character is going to be replaced by itself in a certain position, we replace by the special character ξ (Line 5 in Algorithm 1). In practice we use the white space (U+0020) as the test character. Overall, Algorithm 1 performs $\mathcal{O}(|S|)$ forward passes trough the language model.

F.2 Attack classifier

In the case of using a classifier $\boldsymbol{f} : \mathcal{S}(\Gamma) \to \mathbb{R}^o$, where the predicted class is given by $\hat{y} = \arg\max_{y \in [o]} f(S)_y$ with o classes, we follow Hou et al. (2023) and use the Carlini-Wagner Loss[8] (Carlini and Wagner, 2017):

$$\mathcal{L}(\boldsymbol{f}(S), y) = \max_{\hat{y} \neq y} f(S)_{\hat{y}} - f(S)_y \,. \tag{2}$$

In this case, a sentence S' is an adversarial example when $\mathcal{L}(\boldsymbol{f}(S'), y) \geq 0$. To search the closest sentence in Levenshtein distance that produces a misclassification, we iteratively solve Eq. (1) with $k = 1$ until the adversarial sentence S' is misclassified. Our attack pseudo-code is presented in Algorithm 2.

F.3 Attack LLM

Now we illustrate how to apply our method on attacking LLM-based classifiers. Given a data sample $(S, y) \in \mathcal{S} \times [C]$, the input to LLMs is formulated by concatenating the original sentences with some instructive prompts S_{P_1}, S_{P_2} in the format of $S_{P_1} \oplus S \oplus S_{P_2}$. A schematic for illustration and additional details on the prompting strategy can be found in Appendix E.6. Similar to Eq. (1), we aim to solve:

$$\max_{S' \in \mathcal{S}_k(S, \Sigma)} \mathcal{L}\left(\boldsymbol{f}(S_{P_1} \oplus S' \oplus S_{P_2}), y\right) \,,$$

where the model output $f(S_{P_1} \oplus S' \oplus S_{P_2})_i := \mathbb{P}(i | S_{P_1} \oplus S' \oplus S_{P_2})$ is the conditional probability of the next token i. We can still use the Carlini-Wagner Loss defined in Eq. (2) by considering the next token probability for the classes.

G Proof of Corollary S6

In this section, we provide the technical proof of Corollary S6.

[8] In the original paper, Carlini and Wagner (2017) clip the value of the loss to be 0 at maximum. We do not clip in order to deal with cases where the loss is positive for different adversarial examples.

Proof. Starting with the upper bound, in the base case, we have $|\mathcal{S}_0| = |\{S\}| = 1$. Then, we will prove the relationship between $|\mathcal{S}_k|$ and $|\mathcal{S}_{k-1}|$. For a certain $S' \in \mathcal{S}_{k-1}$, we have:

$$\begin{aligned}
|\mathcal{S}_k(S,\Gamma)| &= \left| \bigcup_{S' \in \mathcal{S}_{k-1}} \{S'' : d_{\text{lev}}(S', S'') \leq 1\} \right| \\
&\leq \sum_{S' \in \mathcal{S}_{k-1}} |\{S'' : d_{\text{lev}}(S', S'') \leq 1\}| \\
\text{[Proposition S3]} &= \sum_{S' \in \mathcal{S}_{k-1}} \left| \left\{ \psi\left(\phi\left(S'\right) \overset{i}{\leftarrow} c\right) \;\; \forall i \in [2|S'|+1], \forall c \in \Gamma \cup \{\xi\} \right\} \right| \\
\text{[\# combinations of } i\text{'s and } c\text{'s]} &\leq \sum_{S' \in \mathcal{S}_{k-1}} (2|S'|+1) \cdot (|\Gamma|+1) \\
[|S'| \leq |S| + k \;\; \forall S' \in \mathcal{S}_k] &\leq \sum_{S' \in \mathcal{S}_{k-1}} (2(|S|+k)-1) \cdot (|\Gamma|+1) \\
&= (2(|S|+k)-1) \cdot (|\Gamma|+1) \cdot |\mathcal{S}_{k-1}| \; .
\end{aligned}$$

Finally, by induction we have

$$|\mathcal{S}_k(S,\Gamma)| \leq (|\Gamma|+1)^k \cdot \prod_{j=1}^{k} (2(|S|+k)-1) \, .$$

For the lower bound, it is enough to compute the size of the set of strings obtained by adding just prefixes:

$$\begin{aligned}
|\mathcal{S}_k(S,\Gamma)| &\geq |\{\psi(P \oplus \phi(S)), \forall P \in \{P' \in \mathcal{S}(\Gamma \cup \{\xi\}),\; |P'| \leq k\}\}| \\
&= |\{\psi(P), \forall P \in \{P' \in \mathcal{S}(\Gamma \cup \{\xi\}),\; |P| \leq k\}\}| \\
&= \left| \bigcup_{i=0}^{k} \{P' \in \mathcal{S}(\Gamma),\; |P'| = i\} \right| \\
\text{[Disjoint sets]} &= \sum_{i=0}^{k} |\{P' \in \mathcal{S}(\Gamma),\; |P'| = i\}| \\
&= \sum_{i=0}^{k} |\Gamma|^i \\
\text{[Geometric series]} &= \begin{cases} \frac{1-|\Gamma|^{k+1}}{1-|\Gamma|} & \text{if } |\Gamma| > 1 \\ k+1 & \text{if } |\Gamma| = 1 \end{cases}
\end{aligned}$$

□

H Alternative attack designs

In this section, we cover alternative algorithmic designs called PGA-`Charmer` for solving Eq. (1) Specifically, we study relaxing the binary constraints in Eq. (BP) in order to perform a Projected Gradient Ascent (PGA) procedure.

H.1 PGA-Charmer

Let $\boldsymbol{E}^{(i)} = \textbf{Token}(S^{(i)})\boldsymbol{T} \in \mathbb{R}^{l_i \times d}$ be the embeddings of tokens for any sentence $S^{(i)} \in \mathcal{S}' \subseteq \mathcal{S}(\Gamma)$ with $i \in [|\mathcal{S}'|]$. The zero-padded embeddings for the sentences in the set $\mathcal{S}'$ become:

$$\hat{\boldsymbol{E}}^{(i)} = \boldsymbol{E}^{(i)} \oplus \boldsymbol{0}_{(\bar{l}-l_i) \times d} \in \mathbb{R}^{\bar{l} \times d}, \quad \forall i \in [|\mathcal{S}'|] \; ,$$

where $\bar{l} = \max\{l_i : i \in [|\mathcal{S}'|]\}$ and $\oplus$ is the concatenation operator along the first dimension.

Remark S1 (Model output after zero padding). Given a function f, the output before and after zero padding is unchanged, i.e., $\boldsymbol{f}(\hat{\boldsymbol{E}}^{(i)}) = \boldsymbol{f}(\boldsymbol{E}^{(i)}) \ \forall \boldsymbol{E}^{(i)} \in \mathcal{S}'$.

We can reformulate the problem in Eq. (1) as:

$$\begin{array}{cc} \max\limits_{\boldsymbol{u} \in \mathbb{R}^{|\mathcal{S}_k(S,\Gamma)|}} & \mathcal{L}\left(\boldsymbol{f}\left(\sum_{i=1}^{|\mathcal{S}_k(S,\Gamma)|} u_i \cdot \hat{\boldsymbol{E}}^{(i)}\right), y\right) \\ \text{s.t.} & u_i \in \{0,1\} \ \forall i \in [|\mathcal{S}_k(S,\Gamma)|], \quad ||\boldsymbol{u}||_1 = 1 \end{array}, \tag{BP}$$

which is a constrained binary optimization problem. Note that given $\boldsymbol{u}^{\text{BP}}$ a maximizer of Eq. (BP) with $i^{\text{BP}} := \arg\max_{i \in [|\mathcal{S}_k(S,\Gamma)|]} u_i^{\text{BP}}$, we have that the sentence $S^{(i^{\text{BP}})} \in \mathcal{S}_k(S,\Gamma)$ is a maximizer of Eq. (1).

However, solving Eq. (BP) is as hard as solving Eq. (1) because of the exponential size of $\mathcal{S}_k(S,\Gamma)$, see Corollary S6. Alternatively, we can relax the binary constraints from the $\boldsymbol{u}$ vector and solve:

$$\begin{array}{cc} \max_{\boldsymbol{u} \in \mathbb{R}^{|\mathcal{S}_k(S,\Gamma)|}} & \mathcal{L}\left(\boldsymbol{f}\left(\sum_{i=1}^{|\mathcal{S}_k|} u_i \cdot \hat{\boldsymbol{E}}^{(i)}\right), y\right) \\ \text{s.t.} & u_i \in [0,1] \ \forall i \in [|\mathcal{S}_k(S,\Gamma)|], \quad ||\boldsymbol{u}||_1 = 1. \end{array} \tag{SP}$$

In this case, given $\boldsymbol{u}^{\text{SP}}$ a maximizer of Eq. (SP), we know:

$$\mathcal{L}\left(\boldsymbol{f}\left(\sum_{i=1}^{|\mathcal{S}_k(S,\Gamma)|} u_i^{\text{SP}} \cdot \hat{\boldsymbol{E}}^{(i)}\right), y\right) \geq \mathcal{L}\left(\boldsymbol{f}\left(\sum_{i=1}^{|\mathcal{S}_k(S,\Gamma)|} u_i^{\text{BP}} \cdot \hat{\boldsymbol{E}}^{(i)}\right), y\right).$$

Note that the embeddings $\sum_{i=1}^{|\mathcal{S}_k(S,\Gamma)|} u_i^{\text{SP}} \cdot \hat{\boldsymbol{E}}^{(i)}$ have no correspondence to any sentence in $\mathcal{S}_k(S,\Gamma)$. However, we can still take $i^{\text{SP}} = \arg\max_{i \in [|\mathcal{S}_k(S,\Gamma)|]} u_i^{\text{SP}}$ and hopefully $S^{(i^{\text{SP}})} \in \mathcal{S}_k(S,\Gamma)$ is an adversarial example. To solve Eq. (SP), we employ projection gradient ascent with step-size η as follows:

$$\boldsymbol{u}^{t+1} = \Pi_{\Delta}(\boldsymbol{u}^t + \eta \nabla \mathcal{L}_{\boldsymbol{u}}(\boldsymbol{u}^t)),$$

where $\Pi_{\Delta}(\cdot)$ is the projection function. Let us denote by $\hat{\boldsymbol{u}} := \boldsymbol{u}^t + \eta \nabla \mathcal{L}_{\boldsymbol{u}}(\boldsymbol{u}^t)$ for notation simplification, then the projection step essentially aims to solve the following quadratic programming problem:

$$\begin{aligned} & \boldsymbol{u}^{t+1} = \arg\min_{\boldsymbol{u}} \frac{1}{2}||\boldsymbol{u} - \hat{\boldsymbol{u}}||_2^2, \\ & \text{subject to} : \sum u_i = 1, \quad u_i \geq 0. \end{aligned} \tag{QP}$$

The Lagrangian associated with Eq. (QP) is as follows:

$$\mathcal{L}(\boldsymbol{u}, \lambda, \boldsymbol{v}) := \frac{1}{2}||\boldsymbol{u} - \hat{\boldsymbol{u}}||_2^2 + \lambda(\boldsymbol{u}^\top \mathbf{1} - 1) - \boldsymbol{v}^\top \hat{\boldsymbol{u}},$$

where $\lambda \in \mathbb{R}, \boldsymbol{v} \in \mathbb{R}^{|S_{\text{all}}|}$ are the Lagrange multipliers. The Karush-Kuhn-Tucker optimality conditions are necessary and sufficient for solving Eq. (QP), that is:

$$\nabla_{\boldsymbol{u}} \mathcal{L}(\boldsymbol{u}, \lambda, \boldsymbol{v}) = \boldsymbol{u} - \hat{\boldsymbol{u}} + \lambda \mathbf{1} - \boldsymbol{v} = \mathbf{0}, \tag{3}$$

$$u_i \geq 0, \tag{4}$$

$$\sum u_i - 1 = 0, \tag{5}$$

$$v_i \geq 0, \tag{6}$$

$$v_i u_i = 0. \tag{7}$$

Clearly, given any λ, if we set $u_i = \max(\hat{u}_i - \lambda, 0), \quad v_i = \max(\lambda - \hat{u}_i, 0)$, then Eqs. (3), (4), (6) and (7) can be satisfied. Therefore, the remaining problem reduces to find a λ that satisfies Eq. (5), i.e.,

$$\sum u_i - 1 = \sum \max(\hat{u}_i - \lambda, 0) - 1 = 0.$$

We employ the algorithm proposed in Held et al. (1974) to solve it, as presented in Algorithm 3. Lastly, we select the $\arg\max_j(u_j^\star)$ element in S_{all} as the attack sentence S'.

Algorithm 3 Projection into simplex (Held et al., 1974)

```
**Input**: $\hat{\boldsymbol{u}} := \boldsymbol{u}^t + \eta\nabla\mathcal{L}_{\boldsymbol{u}}(\boldsymbol{u}^t) \in \mathbb{R}^{|S_{\text{all}}|}$.
Sort $\hat{\boldsymbol{u}}$ such that $\hat{u}_1 \leq \hat{u}_2 \leq \cdots \leq \hat{u}_{|S_{\text{all}}|}$.
Set $J_0 := \max(J : \frac{-1+\sum_{i=J+1}^{|S_{\text{all}}|}\hat{u}_i}{|S_{\text{all}}|-J} > \hat{u}_J)$.
Calculate $\lambda = \frac{-1+\sum_{i=J_0+1}^{|S_{\text{all}}|}\hat{u}_i}{|S_{\text{all}}|-J}$.
Set $u_i^{t+1} = \max(\hat{u}_i - \lambda, 0)$.
**Output**: $\boldsymbol{u}^{t+1}$
```

Table S20: **Comparison between our PGA-`Charmer` and query-based `Charmer` (proposed in the main body) in BERT:** The best method is highlighted in **bold**. While the PGA-`Charmer` strategy can noticeably improve the runtime, the ASR, Levenshtine distance and similarity are considerably degraded.

	Method	ASR (%) ↑	$d_{\text{lev}}(S, S')$ ↓	Sim(S, S') ↑	Time (s) ↓
AG-News	PGA-`Charmer`	86.94	$7.33_{\pm(5.01)}$	$0.87_{\pm(0.11)}$	$\mathbf{8.15}_{\pm(7.04)}$
	`Charmer`	**98.51**	$\mathbf{3.68}_{\pm(3.08)}$	$\mathbf{0.95}_{\pm(0.06)}$	$8.74_{\pm(11.10)}$
MNLI-m	PGA-`Charmer`	99.05	$2.11_{\pm(1.53)}$	$0.79_{\pm(0.19)}$	$\mathbf{0.85}_{\pm(0.72)}$
	`Charmer`	**100.00**	$\mathbf{1.14}_{\pm(0.42)}$	$\mathbf{0.85}_{\pm(0.13)}$	$1.45_{\pm(0.81)}$
QNLI	PGA-`Charmer`	81.19	$3.46_{\pm(2.32)}$	$0.89_{\pm(0.12)}$	$\mathbf{5.15}_{\pm(4.60)}$
	`Charmer`	**97.68**	$\mathbf{1.94}_{\pm(1.48)}$	$\mathbf{0.94}_{\pm(0.07)}$	$9.19_{\pm(9.60)}$
RTE	PGA-`Charmer`	72.64	$\mathbf{1.53}_{\pm(1.45)}$	$\mathbf{0.86}_{\pm(0.11)}$	$\mathbf{0.65}_{\pm(0.71)}$
	`Charmer`	**97.01**	$1.55_{\pm(1.42)}$	$\mathbf{0.86}_{\pm(0.13)}$	$2.50_{\pm(2.33)}$
SST-2	PGA-`Charmer`	97.52	$2.68_{\pm(1.82)}$	$0.84_{\pm(0.16)}$	$\mathbf{1.09}_{\pm(0.89)}$
	`Charmer`	**100.00**	$\mathbf{1.47}_{\pm(0.74)}$	$\mathbf{0.90}_{\pm(0.11)}$	$1.27_{\pm(0.84)}$

H.2 COMPARISON BETWEEN PGA-CHARMER AND QUERY-BASED CHARMER

In this section, we experimentally validate the efficiency of PGA-`Charmer` and compare it against query-based `Charmer`, which is proposed in the main body. The result in Table S20 shows that PGA-`Charmer` can efficiently reduce the runtime as it does not require the forward pass over a mini-batch of sentences after position selection. However, PGA-`Charmer` is worse than `Charmer` on other metrics, e.g., ASR, Levenshtine distance and similarity are degraded. We believe that combining the efficiency of PGA-`Charmer` and high ASR in -`Charmer` holds promise for future research endeavors.