BERT-Defense: A Probabilistic Model Based on BERT to Combat Cognitively Inspired Orthographic Adversarial Attacks

Yannik Keller[†], **Jan Mackensen**[†], **Steffen Eger**[‡]
[†] Center of Cognitive Science
[‡] Natural Language Learning Group
Technische Universität Darmstadt
{yannik.keller,jan.mackensen}@stud.tu-darmstadt.de
eger@aiphes.tu-darmstadt.de

Abstract

Adversarial attacks expose important blind spots of deep learning systems. While word- and sentence-level attack scenarios mostly deal with finding semantic paraphrases of the input that fool NLP models, character-level attacks typically insert typos into the input stream. It is commonly thought that these are easier to defend via spelling correction modules. In this work, we show that both a standard spellchecker and the approach of Pruthi et al. (2019), which trains to defend against insertions, deletions and swaps, perform poorly on the character-level benchmark recently proposed in Eger and Benz (2020) which includes more challenging attacks such as visual and phonetic perturbations and missing word segmentations. In contrast, we show that an untrained iterative approach which combines context-independent character-level information with context-dependent information from BERT's masked language modeling can perform on par with human crowd-workers from Amazon Mechanical Turk (AMT) supervised via 3-shot learning.

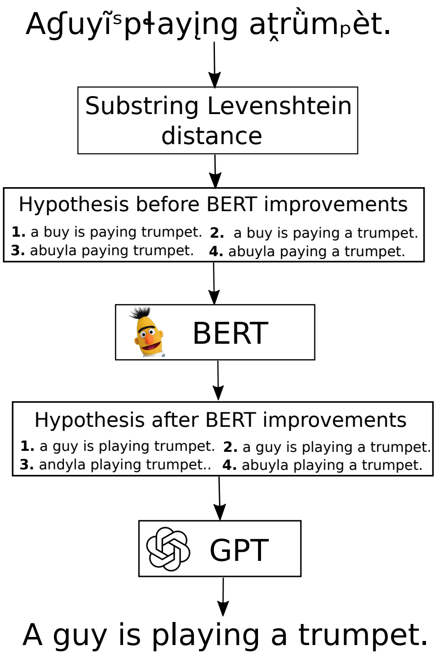
Aɠuyĩˢpɫayįng aṱrȕmₚèt.
Substring Levenshtein distance
Hypothesis before BERT improvements
1. a buy is paying trumpet. 2. a buy is paying a trumpet.
3. abuyla paying trumpet. 4. abuyla paying a trumpet.
BERT
Hypothesis after BERT improvements
1. a guy is playing trumpet. 2. a guy is playing a trumpet.
3. andyla playing trumpet.. 4. abuyla playing a trumpet.
GPT
A guy is playing a trumpet.

Figure 1: A high-level overview of the processing of an example sentence in our adversarial-defense pipeline. The sentences shown for the hypothesis have been created by choosing the maximum of their associated probability distributions over words.

1 Introduction

Adversarial attacks to machine learning systems are malicious modifications of their inputs designed to fool machines into misclassification but not humans (Goodfellow et al., 2015). One of their goals is to expose blind-spots of deep learning models, which can then be shielded against. In the NLP community, typically two different kinds of attack scenarios are considered. "High-level" attacks paraphrase (semantically or syntactically) the input sentence (Iyyer et al., 2018; Alzantot et al., 2018; Jin et al., 2020) so that the classification label does not change, but the model changes its decision. Often, this is framed as a search problem where the attacker has at least access to model predictions (Zang et al., 2020). "Low-level" attackers operate on the level of characters and may consist of adversarial typos (Belinkov and Bisk, 2018; Ebrahimi et al., 2018a; Pruthi et al., 2019; Jones et al., 2020) or replacement of characters with similarly looking ones (Eger et al., 2019; Li et al., 2020a). Such attacks may also be successful when the attacker operates in a blind mode, without having access to model predictions, and they are arguably more realistic, e.g., in social media. However, Pruthi

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 1616–1629
August 1–6, 2021. ©2021 Association for Computational Linguistics

et al. (2019) showed that orthographic attacks can be addressed by placing a spelling correction module in front of a downstream classifier, which may be considered a natural solution to the problem.[1]

In this work, we apply their approach to the recently proposed benchmark *Zéroe* of Eger and Benz (2020), illustrated in Table 1, which provides an array of cognitively motivated orthographic attacks, including missing word segmentation, phonetic and visual attacks. We show that the spelling correction module of Pruthi et al. (2019), which has been trained on simple typo attacks such as character swaps and character deletions, fails to generalize to this benchmark. This motivates us to propose a novel technique to addressing various forms of orthographic adversaries that does not require to train on the low-level attacks: first, we obtain probability distributions over likely true underlying words from a dictionary using a context-independent extension of the Levenshtein distance; then we use the masked language modeling objective of BERT, which gives likelihoods over word substitutions in context, to refine the obtained probabilities. We iteratively repeat this process to improve the word context from which to predict clean words. Finally, we apply a source text independent language model to produce fluent output text.

Our contributions: **(i)** We empirically show that this approach performs much better than the trained model of Pruthi et al. (2019) on the Zéroe benchmark. Furthermore, **(ii)** we also evaluate human robustness on Zéroe and **(iii)** demonstrate that our iterative approach, which we call BERT-Defense, sometimes even outperforms human crowd-workers trained via 3-shot learning.

Attacker	Sentence
inner-shuffle	A man is drnviig a car.
full-shuffle	A amn is ginvdir a acr.
disemvowel	A mn is drvng a cr.
intrude	A ma#n i*s driving a caˆr.
keyboard-typo	A mwn is dricing a caf.
natural-typo	A wan his driving as car.
truncate	A man is drivin a car.
segmentation	Aman isdriving a car.
phonetic	Ae man izz dreyvinn a cahar.
visual	A māᴎ ǐs ḋʀǹʋɪŋɢ ɐ ƈāŕ

Table 1: Examples for the adversarial attacks from the Zéroe benchmark. The phonetic and visual examples show our modified implementations (see appendix A.2).

2 Related work

Zeng et al. (2020) classify adversarial attack scenarios in terms of the *accessibility* of the victim model to the attacker:[2] *white-box* attackers (Ebrahimi et al., 2018b) have full access to the victim model including its gradient to construct adversarial examples. In contrast, *black-box* attackers have only limited knowledge of the victim models: score- (Alzantot et al., 2018; Jin et al., 2020) and decision-based attackers (Ribeiro et al., 2018) require access to the victim models' prediction scores (classification probabilities) and final decisions (predicted class), respectively. A score-based black-box attacker of particular interest in our context is BERT-ATTACK (Li et al., 2020b). BERT-ATTACK uses the masked language model (MLM) of BERT to replace words with other words that fit the context. BERT-ATTACK is related to our approach because it uses BERT's MLM in an attack-mode while we use it in defense-mode. Further, in our terminology, BERT-ATTACK is a high-level attacker, while we combine BERT with an edit distance based approach to restore low-level adversarial attacks. *Blind* attackers make fewest assumptions and have no knowledge of the victim models at all. Arguably, they are most realistic, e.g., in the context of online discussion forums and other forms of social media where users may not know which model is employed (e.g.) to censor toxic comments and users may also not have (large-scale) direct access to model predictions.

In terms of blind attackers, Eger et al. (2019) design the visual perturber VIPER which replaces characters in the input stream with visual nearest neighbors, an operation to which humans are seemingly very robust.[3] Eger and Benz (2020) propose a canon of 10 cognitively inspired orthographic character-level blind attackers. We use this benchmark, which is illustrated in Table 1, in our application scenario. While Eger et al. (2019) and Eger and Benz (2020) are only moderately successful in

[1] One could argue that such a pipeline solution is not entirely satisfactory from a more theoretical perspective, and that downstream classifiers should be innately robust to attacks in the same way as humans.

[2] Another recent survey of adversarial attacks in NLP is provided by Roth et al. (2021).

[3] Basing text processing on visual properties was also recently explored in Wang et al. (2020) and Salesky et al. (2021).

defending against their orthographic attacks with *adversarial learning* (Goodfellow et al., 2015) (i.e., including perturbed instances at train time), Pruthi et al. (2019) show that placing a word recognition (correction) module in front of a downstream classifier may be much more effective. They use a correction model trained to recognize words corrupted by random adds, drops, swaps, and keyboard mistakes. Zhou et al. (2019) also train on the adversarial attacks (insertion, deletion, swap as well as word-level) against which they defend. In contrast, we show that an *untrained* attack-agnostic iterative model based on BERT may perform competitively even with humans (crowd-workers) and that this correction module may further be improved by leveraging attack-specific knowledge. Jones et al. (2020) place an encoding module—which should map orthographically similar words to the same (discrete) 'encoding'—before the downstream classifier to improve robustness against adversarial typos. However, in contrast to Pruthi et al. (2019) and BERT-Defense, their model does not restore the attacked sentence to its original form so that it is less desirable in situations where knowing the underlying surface form may be relevant (e.g., for human introspection or in tasks such as spelling normalization).

In contemporaneous work, Hu et al. (2021) use BERT for masked language modeling together with an edit distance to correct a misspelled word in a sentence. They assume a single misspelled word that they correct by selecting from a set of edit distance based hypotheses using BERT. In contrast, in our approach we assume that multiple or even all words in the sentence have been attacked using adversarial attacks and that we do not know which ones. Then, we use an edit distance and integrate its results probabilistically with context information obtained by BERT, rather than using edit distance only for candidate selection.

3 Methods

Our complete model, which is outlined in Figure 1 on a high level, has three intuitive components. The first component is context-independent and tries to detect the tokens in a sentence from their given (potentially perturbed) surface forms. This makes sense, since we assume orthographic low-level attacks on our data. The second component uses context, via masked language modeling in BERT, to refine the probability distributions obtained from the first step. The third component uses a language model (in our case, GPT) to make a choice between multiple hypotheses. In the following, we describe each of the three components.

3.1 Context-independent probability

In the first step of our sentence restoration pipeline, we use a modified Levenshtein distance to convert the sentence into a list of probability distributions over word-piece tokens from a dictionary D. For the dictionary, we choose BERT's (Devlin et al., 2019) default word-piece dictionary.

We begin by splitting the attacked sentence S at spaces into word tokens $\tilde{w}_i$. However, to be able to use our word-piece dictionary D, we need to find the appropriate segmentation of the tokens into word-pieces.

Modified Levenshtein distance. We developed a modified version of the Wagner–Fischer algorithm (Wagner and Fischer, 1974) that calculates a Levenshtein distance to substrings of the input string and keeps track of start as well as end indices of matching substrings. For each $\tilde{w}_i$ in S, this algorithm (which is described in Appendix A.1) calculates the substring Levenshtein distance *dist* to every word-piece w_d in D.

Segmentation hypothesis. We store the computed distances $\mathit{dist}(\tilde{w}_i, w_d)$ in a dictionary C_i that maps each start-index s and end-index e to a list of distances, i.e., C_i associates

$$(s, e) \mapsto \big(\mathit{dist}(\tilde{w}_i, w_d)\big)_{w_d \in D'}$$

Here, D' selects the subset of all word-pieces in D that match $\tilde{w}_i$ at the substring between s and e. Using C_i, we can then perform a depth-first search to compose $\tilde{w}_i$ from start and end-indices in C_i. For example, a 10 character word $\tilde{w}_i$ could be segmented into two words-pieces that match the substrings from positions 1-5 and 6-10, respectively, or a single word that matches from 1-10. Let $\mathbf{c}_i$ be the set of all segmentations of $\tilde{w}_i$ from start and end indices. For example, $\mathbf{c}_i$ could be $\big\{\big((1,5),(6,10)\big),\ \big((1,10)\big)\big\}$. For each segmentation $\mathbf{c}_{i,\alpha} \in \mathbf{c}_i$, we then calculate a *total distance* $d(\mathbf{c}_{i,\alpha})$ as a sum of the minimum distances of all parts:

$$d(\mathbf{c}_{i,\alpha}) = \sum_{k=1}^{\text{len}(\mathbf{c}_{i,\alpha})} \min(C_i[\mathbf{c}_{i,\alpha,k}]) \qquad (1)$$

Using the total distances to segment each token $\tilde{w}_i$, we can now create hypotheses $\mathbf{H}$ about how the whole sentence S consisting of n tokens should be segmented into word-pieces. For this, we calculate the Cartesian product between the sets of possible segmentations for each word $\tilde{w}_i$, $i = 1, \ldots, n$:

$$\mathbf{H} = \mathbf{c}_1 \times \mathbf{c}_2 \times \cdots \times \mathbf{c}_n$$

We set the *loss* of one hypothesis $\mathbf{h} = (\mathbf{c}_{1,\alpha_1}, \ldots, \mathbf{c}_{n,\alpha_n}) \in \mathbf{H}$ as the sum of the total distances of its parts that we calculated in Eq. (1)

$$\mathit{loss}(\mathbf{h}) = \sum_{i=1}^{n} d(\mathbf{c}_{i,\alpha_i})$$

By evaluating the softmax on the negative total distances of the hypothesis, we calculate probabilities if a hypothesis $\mathbf{h}_v \in \mathbf{H}$ is equal to the true (unknown) segmentation $\mathbf{h}^*$ of the n tokens:

$$\begin{aligned} \mathbf{d}_{\mathbf{H}} &= (\mathit{loss}(\mathbf{h}))_{\mathbf{h} \in \mathbf{H}} \\ \mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S) &= [\mathrm{softmax}(-\mathbf{d}_{\mathbf{H}})]_v \end{aligned} \tag{2}$$

We will refer back to these probabilities in §3.3.

Word probability distributions. In a hypothesis $\mathbf{h} \in \mathbf{H}$, a token $\tilde{w}_i$ has a single segmentation of start and end indices associated with it, $\mathbf{c}_{i,\alpha}$. For all start- and end-indices (s, e), $C_i[s, e]$ stores the distances of the words that match $\tilde{w}_i$ between s and e. Let D' again be the dictionary containing all those words. Let w_d be a word-piece in D' and let $w^* \in D'$ be the true match for the substring between s and e of $\tilde{w}_i$. Then, we can compute a context-independent probability that w_d is equal to w^*, by evaluating the softmax on the negative distances stored in C_i:

$$\mathbf{P}_{\mathbf{h}}(w_d = w^* \mid \tilde{w}) = [\mathrm{softmax}(-C_i[s, e])]_d$$

When we do this for all words in $\mathbf{h}$ and concatenate the results, we get a vector $\mathbf{V}_{\mathbf{h}}$ of probability distributions over dictionary word-pieces. This is illustrated in Figure 2. We introduce the following notation to select a probability distribution based on its index in $\mathbf{h}$ using the subscript j:

$$\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}) \coloneqq \mathbf{V}_{\mathbf{h},j}$$

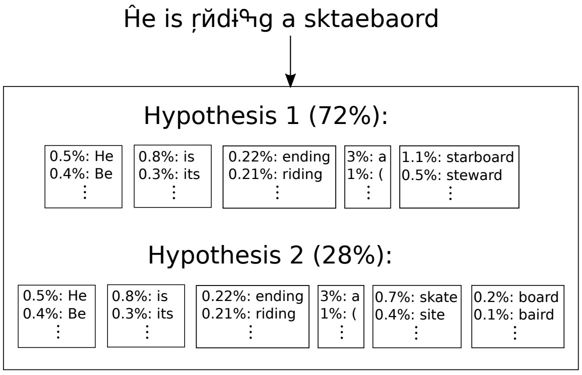

Figure 2: A context independent probability distribution over words calculated for an example input sentence. There are multiple segmentation-hypothesis associated with the sentence that each consist of a sequence of probability distributions over word-tokens.

Domain-specific distance. In the remainder, we will refer to the way of calculating the substring distance as described above as *attack-agnostic*. Beyond this, we also aim to leverage domain-specific knowledge. We refer to such an augmented distance as the *domain-specific* distance dist_M. Here, we modify the operation costs in the substring Levenshtein distance in certain situations.

1. Edit distance is reduced for visually similar characters. This builds on visual character representations (Eger et al., 2019). See appendix A.3 for details.
2. Addressing intruder attacks, we reduce deletion costs depending on the frequency f of the character in the source word. Our assumption is that the same intruder symbol may be repeated in one word. Thus, we decay the cost exponentially for increasing frequency using the formula 0.75^{f-1}.
3. Vowel insertion cost is reduced to 0.3 for words that contain no vowels.

To address letter-shuffling, we additionally compute an anagram distance of how close the attacked word $\tilde{w}_i$ is to being an anagram to the dictionary word w_d. Let m be the number of characters that are in one of the two words, but not in the other. Then, our anagram distance $\mathit{dist}_{\mathrm{A}}$ computes to

$$\mathit{dist}_{\mathrm{A}}(\tilde{w}_i, w_d) = 2m + 1$$

When two words are permutations of each other, the anagram distance is minimal and otherwise it increases linearly in the number of different characters between the two words. We then take the minimum of the anagram distance and the substring Levenshtein distance with modified operation costs $\mathit{dist}_{\mathrm{M}}$ to obtain the *domain-specific* $\mathit{dist}_{\mathrm{F}}$:

$$\mathit{dist}_{\mathrm{F}}(\tilde{w}_i, w_d) = \min(\mathit{dist}_{\mathrm{A}}(\tilde{w}_i, w_d), \mathit{dist}_{\mathrm{M}}(\tilde{w}_i, w_d))$$

3.2 Context-dependent probability using BERT

In the following, we describe the context-based improvement for a single hypothesis $\mathbf{h} \in \mathbf{H}$. In

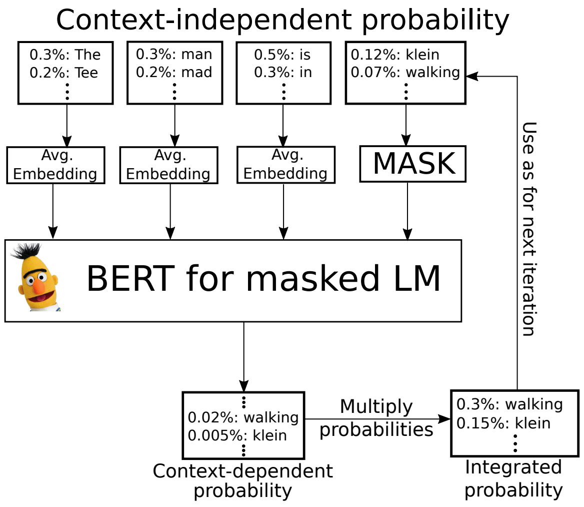

Figure 3: Iterative, context-based improvements of the word predictions using BERT for masked LM. Each iteration, a different token will be masked. We calculate context-dependent probabilities using Eq. (3) and integrate them with our context-independent probabilities in Eq. (4).

Figure 3, the whole process is illustrated for an example sentence. The number of required iterations should scale linearly with the amount of tokens in the hypothesis, so we perform $2 \cdot |\mathbf{h}|$ iterations in total. To perform one improvement iteration, we perform the following steps:

1) Select an index j, of a token, that will be masked for this iteration.

2) For the next part, we slightly modify BERT for masked LM. Instead of using single tokens as inputs as in BERT, we want to use our context-independent probability distributions over word-piece tokens. Thus, for each token $w_{\mathbf{h},j}$ in $\mathbf{h}$, we embed all relevant tokens w_d from the context-independent process described above using BERT's embedding layer and combine them into a weighted average embedding using weights $\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w})$.

3) We now bypass BERT's embedding layer and feed the weighted average embeddings and the embedding for the mask token directly into the next layers of BERT[1]. As a result, BERT provides us with a vector of scores $\mathbf{S}_{\text{BERT}}$ for how well the words from the word-piece dictionary D fit into the position of the masked word.

4) By applying the softmax on these scores, we obtain a new probability distribution over word-pieces which is dependent on the context c of the token at position j:

$$\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid c) = [\text{softmax}(\mathbf{S}_{\text{BERT}})]_d \qquad (3)$$

5) We make the simplifying assumption that each word is attacked independently from the other words. Thus, the context c is independent of the attack on the word $\tilde{w}$. This means that the following equality holds:

$$\begin{aligned}\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}, c) = \\ \mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w})\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid c)\end{aligned} \qquad (4)$$

6) We go back to step 1) and use $\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}, c)$ from Eq. (4) instead of $\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w})$ to create the average embedding at position j.

3.3 Selecting the best hypothesis with GPT

After performing the context-based improvements, we are left with multiple hypothesis $\mathbf{h} \in \mathbf{H}$. Each of them has a hypothesis probability $\mathbf{P}(\mathbf{h}{=}\mathbf{h}^* \mid S)$ and a list of word-piece probabilities of length $|\mathbf{h}|$ over dictionary words associated with it. Now, we finally collapse the probability distributions by taking the argmax to form actual sentences $S_\mathbf{h}$:

$$\begin{aligned}w_{\mathbf{h},j} &\leftarrow \operatorname*{argmax}_{w_d}(\mathbf{P}_{\mathbf{h},j}(w_d = w^* \mid \tilde{w}, c)) \\ S_\mathbf{h} &\leftarrow (w_{\mathbf{h},j})_{j=1}^{|\mathbf{h}|}\end{aligned} \qquad (5)$$

This allows us to use GPT (Radford et al., 2018) to calculate a language modeling (perplexity) score $\text{LM}_{S_\mathbf{h}}$ for each sentence. Using softmax, we again transform these scores into a probability distribution that describes the probability of a segmentation hypothesis $\mathbf{h}_v \in \mathbf{H}$ being the correct segmentation $\mathbf{h}^*$, based on the restored sentences $S_\mathbf{H}$:

$$\begin{aligned}S_\mathbf{H} &\leftarrow (S_\mathbf{h})_{\mathbf{h}\in\mathbf{H}} \\ \text{LM}_{S_\mathbf{H}} &\leftarrow (\text{LM}_{S_\mathbf{h}})_{\mathbf{h}\in\mathbf{H}} \\ \mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_\mathbf{H}) &= [\text{softmax}(\text{LM}_{S_\mathbf{H}})]_v\end{aligned}$$

The original probability $\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S)$ assigned to each hypothesis is only based on the results of the Levenshtein distance for the attacked sentence S. Thus, as $\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S)$ only depends on the character-level properties of the attacked sentence and $\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_\mathbf{H})$ only depends on the semantic properties of the underlying sentences, it makes sense to assume that these distributions are independent. This allows us to simply multiply them, to get a probability distribution that captures semantic as well as character-level properties:

$$\begin{aligned}\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_\mathbf{H}, S) = \\ \mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S_\mathbf{H})\mathbf{P}(\mathbf{h}_v{=}\mathbf{h}^* \mid S)\end{aligned} \qquad (6)$$

[1] Although BERT has only been trained on the single token embeddings, we empirically found that feeding in averaged embeddings produces very sensible results.

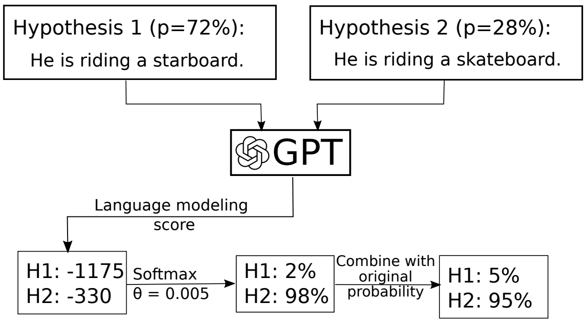

Figure 4: An example of how we use OpenAI GPT to decide on which hypothesis to choose as our final sentence prediction. The original probability of the segmentation hypothesis calculated in Eq. (2) is multiplied with a probability calculated from the language modeling score using Eq. (6).

In Figure 4, we visualize the above described process for a specific example with only 2 hypotheses.

4 Experimental Setup

To obtain adversarially attacked sentences against which to defend, we use the Eger and Benz (2020) benchmark Zéroe of low-level adversarial attacks. This benchmark contains implementations for a wide range of cognitively inspired adversarial attacks such as letter shuffling, disemvoweling, phonetic and visual attacks. The attacks are parameterized by a perturbation probability $p \in [0, 1]$ that controls how strongly the sentence is attacked.

We decided to slightly modify two of the attacks in Zéroe, the phonetic and the visual attacks. On close inspection, we found the phonetic attacks to be too weak overall, with too few perturbations per word. The visual attacks in Zeroé are based on pixel similarity which is similar to the visual similarity based defense in our *domain-specific* model. Thus, to avoid attacking with the same method we defend with, we decided to switch to a description based visual attack model (DCES), just like in the original paper (Eger et al., 2019).[4] Our modifications are described in Appendix A.2.

Evaluation Instead of evaluating on a downstream task, we evaluate on the task of restoring the original sentences from the perturbed sentences. This allows us to easier compare to human performances. It also provides a more difficult test case, as a downstream classifier may infer the correct solution even with part of the input destroyed or omitted. Finally, being able to correct the input is also important when the developed tools would be used for humans, e.g., in spelling correction.

We evaluate the similarity of the sentences to the ground-truth sentences with the following metrics:

1. **Percent perfectly restored** (PPR). The percent of sentences that have been restored perfectly. This is a coarse-grained sentence-level measure.
2. **Editdistance**. The Levenshtein (edit) distance measures the number of insertions, deletions, and substitutions necessary (on character-level) to transform one sequence into another.
3. **MoverScore** (Zhao et al., 2019). MoverScore measures the semantic similarity between two sentences using BERT. It has been show to correlate highly with humans as a semantic evaluation metric.

For all of the metrics, letter case was ignored.

Attack scenarios. We sampled 400 sentences from the GLUE (Wang et al., 2018) STS-B development dataset for our experiments. We use various attack scenarios to attack the sentences:

i) Each of the attack types of the Zéroe benchmark (see Table 1). We set $p = 0.3$ throughout.

ii) To evaluate how higher perturbation levels influence restoration difficulty, we create 5 attack scenarios for one attack scenario (we randomly chose phonetic attacks) with perturbation levels p from 0.1 to 0.9.

iii) We add combinations of attacks: these are performed by first attacking the sentence with one attack and then with another.

iv) In Random attack scenarios (rd:0.3, [rd:0.3,rd:0.3], [rd:0.6,rd:0.6]), one or two attack types from the benchmark are randomly chosen for each sentence. These constitute stronger attack situations and may be seen as more challenging test cases.

Each of the 19 attack scenarios is applied to all 400 sentences individually to create 19 test cases of attacked sentences.

Baselines and upper bounds. To evaluate how well BERT-Defense restores sentences, we compare its sentence restoration ability to two base-

[4] Using description based defense and pixel based attacks would have been possible just as well, but we believe doing it reversely is consistent with the original specification in Eger et al. (2019).

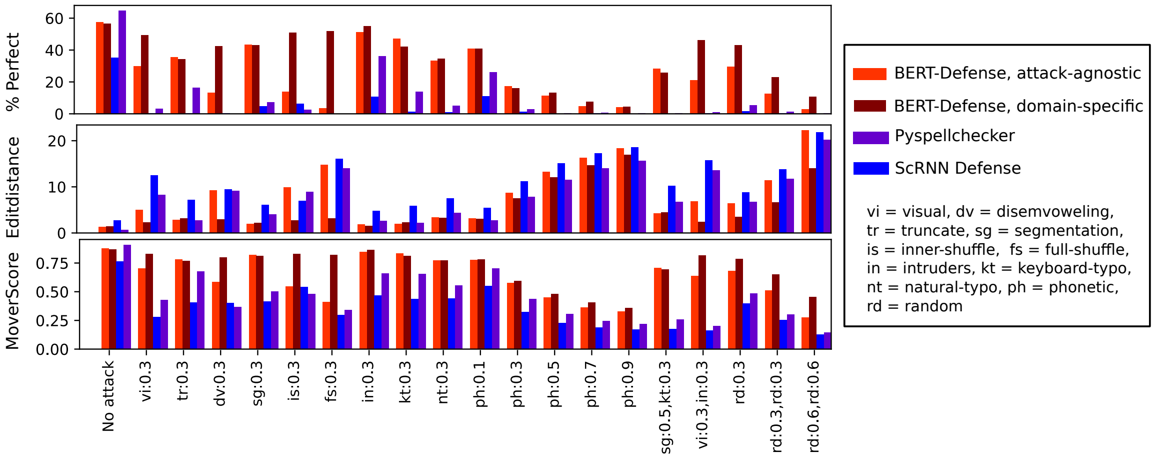

Figure 5: Comparison between BERT-Defense, and the two baseline adversarial defense tools "pyspellchecker" and "ScRNN defense". The x-labels describe the attack and perturbation level the sentences were attacked with, before applying on of the adversarial defense methods. For conditions with two attack types, the perturbations were applied in order. For edit distance, lower is better. For the other metrics, higher is better. Exact values for the results are included in the appendix in Table 6.

lines: (a) the *Pyspellchecker* (Barrus, 2020), a simple spellchecking algorithm that uses the Levenshtein distance and word frequency to correct errors in text; (b) "ScRNN defense" from the Pruthi et al. (2019) paper. This method uses an RNN that has been trained to recognize and fix character additions, swaps, deletions and keyboard typos. Further, as we use Zeroé, a cognitively inspired attack benchmark supposed to fool machines but not humans, it is especially interesting to see how BERT-Defense compares to human performance. Thus, (c) we include human performance, obtained from a crowd-sourcing experiment on Amazon Mechanical Turk (AMT). Note that humans are often considered upper bounds in such settings.

Human experiment. Twenty-seven subjects were recruited using AMT (21 male, mean age 38.37, std age 10.87) using PsiTurk (Gureckis et al., 2016). Participants were paid $3 plus up to $1 score based bonus (mean bonus 0.56, std bonus 0.40) for restoring about 60 adversarially attacked sentences. The task took on average 43.9 minutes with a standard deviation of 20.1. Twenty of the subjects where native English speakers, seven where non-native speakers. The two groups did not significantly differ regarding their edit distances to the true underlying sentences (unequal variance t-test, $p = .85$).

We sampled 40 random sentences from nine of our attack scenarios plus 40 random (non-attacked) sentences from the original document. Each sentence was restored by four different humans. The whole set of 1600 sentences (10 scenarios times 40 sentences each times 4 repetitions) was then randomly split into 27 sets of about 60 sentences. No split contained the same sentence multiple times. Each of the 27 participants got one of these sets assigned. After a short instruction text, the participants where shown three examples of how to correctly restore a sentence ("3-shot learning"). Then they were shown the sentences in their set sequentially and entered their attempts at restoring the sentences into a text-field.

5 Results and Discussion

Comparison with baselines. Figure 5 visualizes the results (full results are in the appendix). BD_{agn} (BERT-Defense, attack-agnostic) significantly outperforms both baselines regarding *MoverScore* and *PPR* for all random attack scenarios (p $\ll$ 0.01, equal variance t-test). However, only BD_{spec} (BERT-Defense, domain-specific) achieves a lower edit distance than the baselines. This discrepancy between the measures is explained by the fact that, by taking context into account, BERT-Defense searches for the best restoration in the space of sensible sentences, while *Pyspellchecker* searches the best restoration for each word individually. Although *ScRNN defense* uses an RNN and is able to take context into account, we found that it also mainly seems to restore the words individually and rarely produces grammatically correct sentences for strongly attacked inputs. Table 3, which illustrates failure cases of all models, sup-

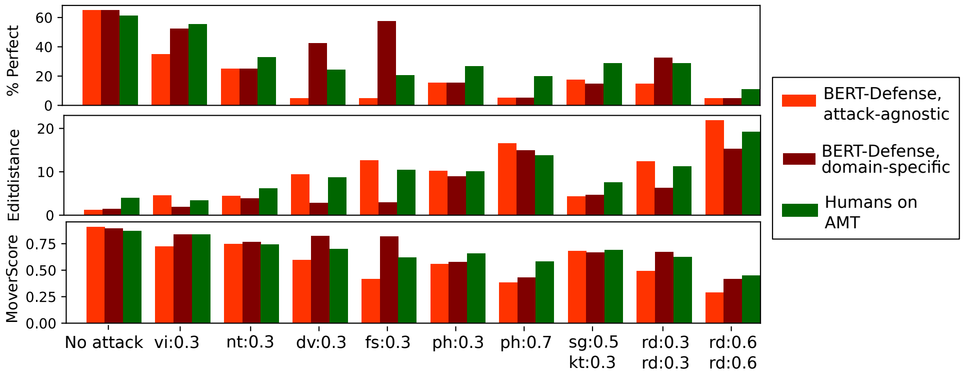

Figure 6: Comparison between BERT-Defense and humans on Amazon Mechanical Turk.

ports this. In the failure case when BERT-Defense fails to recognize the correct underlying sentence, BERT-Defense outputs a mostly different sentence that usually does make some sense, but has little in common with the ground-truth sentence. This results in much higher edit distances than the failure cases of the baselines which produce grammatically wrong sentences, while restoring individual words the best they can (this sometimes means not trying at all). Interestingly, humans tend to produce similar failure cases as BERT-Defense.

When comparing the performance on specific attacks, we see a consistent margin of about 0.2 MoverScore and 15-35 percentage points PPR between BD_{agn} and the baselines across all attacks. Exceptions include inner-shuffle, for which ScRNN-Defense is on par with BD_{agn} and segmentation attacks, which hurt the performance of the baselines far more than the performance of BERT-Defense, which includes segmentation hypothesis as an essential part of its restoration pipeline. For BD_{spec}, we see gains for attacks where we leverage domain-specific knowledge. The biggest gains of around 0.25 MoverScore are achieved against full-shuffle, inner-shuffle and disemvoweling attacks.

In the *No attack* condition, we checked if the adversarial defense methods introduce mistakes when presented with clean sentences. Indeed, all models introduce some errors: all three evaluation metrics show that BERT-Defense introduces a few more errors than *Pyspellchecker* but less than *ScRNN defense*.

Comparison with humans. As stated before, we evaluate human performance on 40 random sentences for each of nine attacks and the no attack condition (see appendix). For each of the sentences, we obtain restorations from 4 crowd-workers. For each attack scenario, we evaluate our metrics on all restorations of these 40 sentences and averaged the results. The results on the 40 attacked sentences are shown in Figure 6. While BD_{agn} performs slightly worse than humans, BD_{spec} matches human performance with respect to all three evaluation metrics. Regarding performance on specific attacks, humans are still better than BERT-Defense when it comes to defending phonetic attacks, while they have a hard time defending full-shuffle attacks. The evaluations for the *No attack* setting reveal that the crowd-workers in our experiment do make quite a few copying mistakes. In fact, they introduce slightly more mistakes than BERT-Defense.

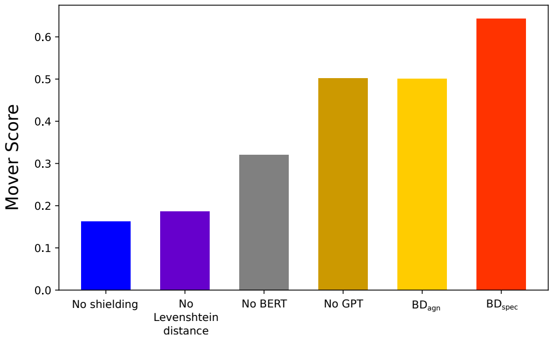

Figure 7: Ablation study for BERT-Defense. The MoverScore metric is shown for BERT-Defense with exactly one single component left out, respectively, on the rd:0.3,rd:0.3 attack. For comparison, we also show the MoverScore without shielding and after shielding with BD_{agn} or BD_{spec} using all components.

Ablation Study. We perform an ablation study to asses the contribution of each individual component of BERT-Defense. For the *No Levenshtein distance* condition, we created the context-independent probability distribution by setting the probability of known words (words in the dictionary) in the attacked dataset to one and using a uniform random distribution for all unknown words.

Attacked Sentence	ScRNN	BD$_{\text{agn}}$
To lorge doog's wronsing in sum grass.	to lorge doog's wronsing in sum grass.	two large dogs rolling in the grass.
Two large dogs runningin some grass.	two large dogs runningin some grass.	two large dogs running in some grass.
Tw large dogs rnnng in some grss.	throw large dogs running in some grss.	two large dogs running in some grass.
Two larg dog runnin in some grass.	two larg dog runnin in some grass.	a large dog running in some grass.
Twolarge dogs running income graas.	twolarge dogs running income graas.	two large dogs running into grass.
To lrg doog's rntng in sm gras ..	to long dogs ring in sm gras.	to the dogs running in the grass.

Table 2: Various illustrative attacks on the sentence "Two large dogs running in some grass." and restorations by ScRNN and BD$_{\text{agn}}$. The attacked sentences are attacked with the following attacks (top to bottom): Phonetic-0.7, Segmentation-0.3, Disemvowel-0.3, Truncate-0.3, Segmentation-0.5 & Keyboard-Typo-0.3, Random-0.3 & Random-0.3 (the last two are double attacks).

Ground-truth	china gives us regulators access to audit records
Attacked (rd:0.6,rd:0.6)	hainc gcive us regulafors essacf to tufai rsxrdeo
Bert-Defense (attack-agnostic)	haine gave us regulators space to turn us over
Human	hain gives us regulators escape to dubai suborder
ScRNN Defense	hainc give us regulafors essacf to tufai rsxrdeo
Pyspellchecker	hain give us regulators essay to tufa rsxrdeo

Table 3: Failure cases for BERT-Defense, humans and the baseline methods. Note that in the failure case, BERT-Defense and Humans restore sentences that are grammatically correct, but are mostly different from the ground-truth. On the other hand, Pyspellchecker and ScRNN Defense (Pruthi et al., 2019) either refuse to try at all for strongly attacked words or create grammatically nonsensical sentences.

When using BERT-Defense without BERT, we directly select the best hypothesis from the context-independent probability distribution using GPT. To run BERT-Defense without GPT, we select the hypothesis with the highest probability according to the results from the modified Levenshtein distance and improve it using context-dependent probabilities obtained with BERT. We evaluate on the rd:0.3,rd:0.3 attack scenario, because we think that it is the most challenging attack.

The results are shown in Figure 7. They indicate that the most important component of BERT-Defense is the Levenshtein distance, as BERT often does not have enough context to meaningfully restore the sentences, given the difficult attacks from Zeroé that typically modify many words in each sentence. Removing BERT also considerably decreases the performance of the defense model. Finally, BERT-Defense without GPT performs on par with BD$_{agn}$ in these experiments, suggesting that BERT-Defense can also be used without GPT for hypothesis selection.

More illustrating examples. To give an impression of the dataset and how the models cope with the adversarial attacks, we show more illustrating examples in Tables 2 and 5 (appendix). These indicate the superiority of our approach in that it typically generates semantically adequate sentences.

6 Conclusion

We introduced BERT-Defense, a model that probabilistically combines context-independent word level information obtained from edit distance with context-dependent information from BERT's masked language modeling to combat low-level orthographic attacks. Our model does not train on possible error types but still substantially outperforms a spell-checker as well as the model of Pruthi et al. (2019), which has been trained to shield against edit distance like attacks, on a comprehensive benchmark of cognitively inspired attack scenarios. We further show that our model rivals human crowd-workers supervised in a 3-shot manner. The generality of our approach allows it to be applied to a variety of different "normalization" problems, such as spelling normalization or OCR post-correction (Eger et al., 2016) besides the adversarial attack scenario considered in this work, which we will explore in future work.

We release our code and data at `https://github.com/yannikkellerde/BERT-Defense`.

Acknowledgments

We thank the anonymous reviewers for their helpful comments.

References

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

Tyler Barrus. 2020. pyspellchecker.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018a. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018b. 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)*, pages 31–36, Melbourne, Australia. Association for Computational Linguistics.

Steffen Eger. 2015. Do we need bigram alignment models? On the effect of alignment quality on transduction accuracy in G2P. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1175–1185.

Steffen Eger and Yannik Benz. 2020. From Hero to Zéroe: A Benchmark of Low-Level Adversarial Attacks. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 786–803, Suzhou, China. Association for Computational Linguistics.

Steffen Eger, Tim vor der Brück, and Alexander Mehler. 2016. A comparison of four character-level string-to-string translation models for (OCR) spelling error correction. *The Prague Bulletin of Mathematical Linguistics*, 105(1):77.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text Processing Like Humans Do: Visually Attacking and Shielding NLP Systems. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1634–1647.

Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*.

Todd M. Gureckis, Jay Martin, John McDonnell, Alexander S. Rich, Doug Markant, Anna Coenen, David Halpern, Jessica B. Hamrick, and Patricia Chan. 2016. psiturk: An open-source framework for conducting replicable behavioral experiments online. *Behavior Research Methods*, 48(3):829–842.

Yifei Hu, Xiaonan Jing, Youlim Ko, and Julia Taylor Rayz. 2021. Misspelling correction with pretrained contextual language model. *arXiv preprint arXiv:2101.03204*.

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

Sittichai Jiampojamarn, Grzegorz Kondrak, and Tarek Sherif. 2007. Applying many-to-many alignments and hidden markov models to letter-to-phoneme conversion. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 372–379, Rochester, New York. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Jinfeng Li, Tianyu Du, Shouling Ji, Rong Zhang, Quan Lu, Min Yang, and Ting Wang. 2020a. Textshield: Robust text classification based on multimodal embedding and neural machine translation. In *29th*

USENIX Security Symposium (USENIX Security 20), pages 1381–1398. USENIX Association.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020b. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202, Online. Association for Computational Linguistics.

Fredrik Lundh and Alex Clark. 2020. Pillow.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

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

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 856–865, Melbourne, Australia. Association for Computational Linguistics.

Tom Roth, Yansong Gao, Alsharif Abuadbba, Surya Nepal, and Wei Liu. 2021. Token-modification adversarial attacks for natural language processing: A survey. *arXiv preprint arXiv:2103.00676*.

Elizabeth Salesky, David Etter, and Matt Post. 2021. Robust open-vocabulary translation from visual text representations. *CoRR*, abs/2104.08211.

Carnegie Mellon University. 2014. The CMU pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`. Accessed: 2020-11-25.

Robert A. Wagner and Michael J. Fischer. 1974. The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Haohan Wang, Peiyan Zhang, and Eric P Xing. 2020. Word shape matters: Robust machine translation with visual embedding. *arXiv preprint arXiv:2010.09997*.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080, Online. Association for Computational Linguistics.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2020. Openattack: An open-source textual adversarial attack toolkit. *arXiv preprint arXiv:2009.09191*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4904–4913, Hong Kong, China. Association for Computational Linguistics.

A Appendices

A.1 Modified Wagner-Fischer algorithm

The modified Wagner-Fischer algorithm gets the source word S of length n and the target word T of length m as inputs and performs the following operations in a run-time $O(mn)$.

1. Initialize distance matrix $\mathbf{D}$ of size $(m+1) \times (n+1)$ with zeros
2. For $i \in [1, m+1]$ do: $\mathbf{D_{i,1}} \leftarrow i - 1$
3. Initialize a set-valued start matrix $\mathbf{M}$ of the same size as $\mathbf{D}$ with empty sets.
4. For $j \in [1, n+1]$ do: $\mathbf{M_{1,j}} \leftarrow$ Set$\{j-1\}$
5. For $i \in [1, m+1]$ and $j \in [1, n+1]$ do:
   - Use previous entries of D to calculate total cost of getting to (i, j) with the edit distance operations:
     - Insertion: $\mathbf{D_{i-1,j}} + 1$
     - Substitution: $\mathbf{D_{i-1,j-1}} + 1$
     - Deletion: $\mathbf{D_{i,j-1}} + 1$
     - Swap: $\mathbf{D_{i-2,j-2}} + 1$
     - If $T_i = S_j$ then no operation cost: $\mathbf{D_{i-1,j-1}}$
   - Enter the lowest cost from the edit distance operations into $\mathbf{D_{i,j}}$
   - Update $\mathbf{M_{i,j}}$ by merging the set with the set-valued entries of $\mathbf{M}$ that led to (i, j) with lowest cost
6. Initialize empty list L
7. Store lowest entry of $\mathbf{D_{m+1}}$ as c and for $j \in [1, n+1]$ do
   - If $\mathbf{D_{m+1,j} = c}$ do: For $m \in \mathbf{M_{m+1,j}}$ do: Add a 2-tuple (m, j) into L.
8. Return c,L

A.2 Visual and Phonetic attacks

Visual attacks. In the BERT-Defense full-distance pipeline, we exploit visual similarity (see appendix A.3). The visual attacks implemented in (Eger and Benz, 2020) are also based on visual similarity. To avoid attacking with the same method that we defend with, we decided to use VIPER-DCES (Eger et al., 2019) instead. VIPER-DCES exchanges characters based on similarity of the descriptions from the Unicode 11.0.0 final names list (e.g. LATIN SMALL LETTER A for the character 'a').

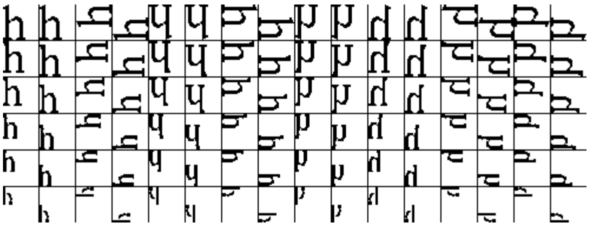

Figure 8: Different orientations/scales used for the letter h. The version that matches a Unicode character the best is used to calculate their similarity.

Phonetic attacks. The phonetic embeddings implemented in Eger and Benz (2020) do not consistently produce phonetic attacks of sufficient quality. Thus, we used a many-to-many aligner (Jiampojamarn et al., 2007; Eger, 2015) together with the CMU Pronouncing Dictionary (cmudict) (University, 2014) and a word frequency list to calculate statistics for the correspondence between letters and phonemes. To attack a word, we convert the word to phonemes using cmudict and then convert it back to letters by sampling from the statistics. The perturbation probability p for this attack controls the sampling temperature which describes how likely it is to sample letters that less frequently correspond to the phoneme in question. Using this method, we generate high-quality phonetically attacked sentences such as the one in Table 1.

A.3 Visual similarity

We calculate the visual similarity of 30000 Unicode characters to 26 letters and 10 numbers. Each glyph is drawn with Python's pillow library (Lundh and Clark, 2020) in 20pt using a fitting font from the google-Noto font collection. The bitmap is then cropped to contain only the glyph. Then the image is resized and padded on the right and bottom to be of size $30px \times 30px$. When comparing the bitmap of a unicode glyph image and a letter/number glyph, multiple versions of the letter/number bitmap are created. For letters, the lowercase as well as the uppercase versions of each letter are taken. The bitmap gets downsized to 5 different sizes between $30px \times 30px$ and $15px \times 15px$, rotated and flipped in all 8 unique ways and then padded to $30px \times 30px$ again, such that the glyph is either placed at the top-left or the bottom left. See Figure 8 for an example. The percentage of matching black pixels between bitmaps are calculated and the highest matching percentage of all version becomes the similarity score S. The substitution cost between two characters will then be calculated based on the similarity with the equation *cost* $= \max(0, \min(1, (0.8 - S) * 3))$. The parameters of this equation have been tuned, so that highly similar characters have a in very low substitution costs while weakly similar characters have next to no reduced in substitution cost.

A.4 Parameters, runtime and computing infrastructure

All experiments where run on a single machine using an Intel(R) Core(TM) i7-4790K processor and a Nvidia GeForce GTX 1070 Ti graphics card. The restoration of a single sentence in the experiments took on average 0.1 seconds using ScRNN Defense, 1.34 seconds for Pyspellchecker and 8 seconds for BERT-defense. In total, BD$_{agn}$ includes 5 free parameters, most of them controlling the temperature of the used softmax operation to ensure good relative weighting of the probability distributions. The parameter values are shown in Table 4. All additional parameters for BD$_{spec}$ have been described in §3.1.

Parameter	**Value**
Softmax temperature for context-independent hypothesis	10
Softmax temperature for context-independent word-probabilities	1
Softmax temperature for BERT	0.25
Softmax temperature for GPT	0.005
Max number of hypothesis	10

Table 4: Parameters for BERT-Defense.

Attacked	theensuing battls abd airstrikes killed at peast 10 militqnts.
Ground-truth	the ensuing battle and airstrikes killed at least 10 militants.
BD$_{agn}$	the ensuing battle and air strikes killed at least 10 militants.
BD$_{spec}$	the ensuing battle and air strikes killed at least 10 militants.
ScRNN Defense	tunney battls and airstrikes killed at past 10 militqnts.
Pyspellchecker	theensuing battle abd airstrips killed at past 10 militants
Attacked	Ǹo, yōư ᵈo tn' ndee ṭo eʰva knaet acslses oɾ nḏəřeā̆ a degrɜe ïn youȓ areă.
Ground-truth	No, you don't need to have taken classes or earned a degree in your area.
BD$_{agn}$	no, you do ,' nee ,' not besides of never a degree, you are.
BD$_{spec}$	no, you do no' need to have taken classes or have a degree in your area.
ScRNN Defense	, yu so to nerve to era knaet access of need a degree ïn your areă.
Pyspellchecker	of you to an' dee to eva knew aisles of nḏəřeā̆ a degree in you are.
Attacked	A man ix riding ;n s voat.
Ground-truth	A man is riding on a boat.
BD$_{agn}$	a man is riding in a boat.
BD$_{spec}$	a man is riding in a boat.
ScRNN Defense	a man imax riding on s voat.
Pyspellchecker	a man ix riding in s vote

Table 5: Additional adversarial shielding examples on the rd:0.3,rd:0.3 dataset.

Dataset	$\mathrm{BD}_{\mathrm{agn}}$		$\mathrm{BD}_{\mathrm{spec}}$		**Pyspellchecker**		**ScRNN Defense**	
Metric	Mover	Editdist	Mover	Editdist	Mover	Editdist	Mover	Editdist
vi:0.3	0.696	5.04	0.830	2.267	0.387	8.54	0.257	12.54
tr:0.3	0.778	2.91	0.767	3.14	0.605	3.34	0.386	7.25
dv:0.3	0.574	9.27	0.794	2.995	0.335	9.53	0.379	9.48
sg:0.3	0.820	2.02	0.808	2.22	0.459	4.52	0.4	6.19
is:0.3	0.539	9.91	0.842	2.767	0.44	9.26	0.520	7.04
fs:0.3	0.399	14.78	0.688	3.227	0.310	14.41	0.277	16.12
in:0.3	0.845	1.9	0.861	1.597	0.588	3.14	0.445	4.92
kt:0.3	0.832	1.96	0.562	2.36	0.596	2.8	0.416	5.89
nt:0.3	0.764	3.38	0.512	3.322	0.504	4.91	0.423	7.59
ph:0.1	0.776	3.21	0.779	3.082	0.632	3.35	0.535	5.50
ph:0.3	0.569	8.77	0.587	7.55	0.397	8.29	0.302	11.26
ph:0.5	0.437	13.25	0.469	12.062	0.275	11.78	0.208	15.19
ph:0.7	0.350	16.37	0.395	14.735	0.218	14.33	0.167	17.34
ph:0.9	0.314	18.45	0.341	16.967	0.194	15.81	0.152	18.63
sg:0.5,kt:0.3	0.701	4.29	0.537	4.47	0.23	7.07	0.158	10.25
vi:0.3,in:0.3	0.627	6.48	0.679	2.467	0.172	13.73	0.14	15.75
rd:0.3	0.676	6.48	0.650	3.485	0.44	7.25	0.375	8.91
rd:0.3,rd:0.3	0.501	11.49	0.451	6.657	0.269	12.0425	0.232	13.92
rd:0.6,rd:0.6	0.257	22.30	0.441	14.0	0.12	20.46	0.104	21.90

Table 6: Exact scores for the results shown in Figure 5.