Using Punctuation as an Adversarial Attack on Deep Learning Based NLP Systems: An Empirical Study

Brian Formento[1,2], Chuan Sheng Foo[2,3], Luu Anh Tuan[4] and See Kiong Ng[1]

[1]Institute of Data Science, National University of Singapore
[2]Institute for Infocomm Research, A*STAR
[3]Centre for Frontier AI Research, A*STAR
[4]Nanyang Technological University
brian.formento@u.nus.edu, foo_chuan_sheng@i2r.astar.edu.sg

Abstract

This work empirically investigates punctuation insertions as adversarial attacks on NLP systems. Data from experiments on three tasks, five datasets, and six models with four attacks show that punctuation insertions, when limited to a few symbols (apostrophes and hyphens), are a superior attack vector compared to character insertions due to 1) a lower after-attack accuracy ($A_{aft-atk}$) than alphabetical character insertions; 2) higher semantic similarity between the resulting and original texts; and 3) a resulting text that is easier and faster to read as assessed with the Test of Word Reading Efficiency (TOWRE)). The tests also indicate that 4) grammar checking does not mitigate punctuation insertions and 5) punctuation insertions outperform word-level attacks in settings with a limited number of word synonyms and queries to the victim's model. Our findings indicate that inserting a few punctuation types that result in easy-to-read samples is a general attack mechanism. In light of this threat, we assess the impact of punctuation insertions, potential mitigations, the mitigation's tradeoffs, punctuation insertion's worst-case scenarios and summarize our findings in a qualitative casual map, so that developers can design safer, more secure systems.

1 Introduction

The goal of an attack is to disrupt a natural language processing (NLP) model's classification accuracy. The motivation behind researching these adversarial attacks is to create a toolbox of methods to attack systems while also pointing out flaws to improve the models' robustness. Previous work on adversarial research showed that deep learning-based NLP models are sensitive to slight changes in the input (Ebrahimi et al., 2018) such as character perturbations or word substitutions. However, these attack vectors have three major flaws: 1) letter perturbations can be detected by grammar checkers; 2) these attacks may change the meaning or, worse, the human label of the sentence (e.g., 'she' to 'he' with a character deletion for a gender classification system (Zang et al., 2020)); 3) they can make a sample unreadable. Although word-level attacks that change words to a perturbing synonym make these perturbations almost invisible to humans, the cumulative effect of multiple synonym substitutions in a sentence can make the sample harder to understand. Furthermore, the attack must find perturbing word synonyms when attacking samples in a specific domain, such as biology or law, which may be challenging if the algorithm uses general word embeddings with no domain knowledge.

Punctuation insertions, on the other hand, may be a feasible attack vector that is unaffected by the limitations of character perturbations/word substitutions, since it is hard for grammar checkers to detect punctuation (Section 5.5) while also not drastically changing the meaning of the sentence (Sections 5.7, 5.8). Removing punctuation causes deep learning models to perform worse (Ek et al., 2020), as punctuation contains critical information that models require to function correctly (Jones, 1994). Furthermore, punctuation can hold adversarial downstream information (Formento et al., 2021) that may be exploited by malicious users. Punctuation attacks remain an understudied area: Previous works on the topic (Hosseini et al., 2017; Eger and Benz, 2020; Formento et al., 2021) only casually explored punctuation and ignored whether it can generalize or show which punctuation symbols are best suited for intrusion attacks.

Contributions: Through extensive empirical studies, we have determined that punctuation insertions can outperform, in terms of $A_{aft-atk}$, alphabetical character insertions (as shown in Section 5.1) and, under certain conditions, word substitution (5.2), allowing for a user-controllable tradeoff between after-attack accuracy ($A_{aft-atk}$), sample quality, and attack time efficiency when used together in a multi-level attack (5.3). Specifically,

Findings of the Association for Computational Linguistics: EACL 2023, pages 1–34
May 2-6, 2023

hyphen (Hy) and apostrophe (Ap) insertions are the most effective at avoiding straightforward defense mechanisms (as shown in Sections 5.4, 5.5) while preserving the original meaning, as evidenced by achieving 100% semantic similarity in our tests (as shown in Section 5.7). Additionally, by using the TOWRE test, we have demonstrated that inserting only one punctuation type significantly increases attack readability by increasing reading speeds by 800% compared to character insertions, and 96.8% compared to using multiple types of punctuation (as shown in Section 5.8) without compromising the attack performance (Section 5.9). To aid in the understanding of our findings, we have also introduced a casual map in Figure 5.

2 Related Work

Adversarial attacks on NLP systems can be categorized in terms of the level of granularity of the perturbation. **Character-level** attacks (Ebrahimi et al., 2018; Eger and Benz, 2020; Eger et al., 2019; Belinkov and Bisk, 2018; Sun et al., 2020; Boucher et al., 2021) modify individual characters in words to force the tokenizer to process multiple unrelated embeddings instead of the original, resulting in decreased performance. **Word-level** attacks (Jin et al., 2020; Li et al., 2020; Maheshwary et al., 2020) employ a search algorithm to locate useful perturbing embeddings (Jin et al., 2020; Li et al., 2020; Maheshwary et al., 2020) or operations (Tan et al., 2020; Li et al., 2021) that are clustered close to the candidate attack word's embedding given a similarity constraint (such as the Universal Sentence Encoder (Cer et al., 2018)). **Multi-level** attacks combine multiple types of perturbations, making the attack cumulative. Textbugger (Li et al., 2019), which uses both character-level and word-level attacks, is an example of a multi-level attack.

Although previous research has investigated character and word-level attacks, few have studied the use of punctuation attacks. To the best of our knowledge, only Zéroe (Eger and Benz, 2020), Prespective Atk (Hosseini et al., 2017) and SSTA (Formento et al., 2021) have researched punctuation as an attack vector. While the former two randomly insert symbols within a word, the third revealed that symbols contain adversarial information and can be inserted as padding with little further optimization. Zéroe, in particular, is a benchmark of ten different character attacks. Out of these ten, Zéroe Intrude is the only one focusing on punctuation and is thus used as one of the gold standards in this paper.

Our work builds on these previous works by further exploring Zéroe Intrude and the concept, introduced initially in SSTA, that model-specific symbols can attack binary classifiers when used as padding. Our work contributes to the discussion on punctuation symbols being a general mechanism to attack deep learning models while also improving readability through the novel use of the TOWRE metric, which tracks how quickly someone can read the adversarial text.

3 Methodology

3.1 Overview

Suppose we have a sequence classifier $f : \mathcal{X} \mapsto \mathcal{Y}$, that takes an input sequence of words $x = (\tau_1, \ldots, \tau_n) \in \mathcal{X}$ with ground truth label y and outputs a prediction $\hat{y} = f(x)$. An adversarial attack on input x and classifier f would perturb τ, for example, using character manipulations or word substitutions, to produce a new adversarial sample $\hat{x}$ that is misclassified by f such that $f(\hat{x}) \neq y$.

We investigate punctuation and multi-level attacks in gray-box and black-box settings. Specifically, we explore the effects of inserting punctuation when the victim's model classification logit is leaked (**gray-box**) and when it is not (**black-box**). We use a variation of DeepWordBug (DWB) and the original Zéroe Intrude (ZI) attack in these settings. In addition, we combine punctuation together with word substitutions in a gray-box setting (**multi-level**) to evaluate if punctuation can augment word-level attacks. We provide a more detailed description of the respective attacks used in the following sections.

3.2 Attack foundations and baselines

We build upon and compare our results to the following four attack baselines: 1) *Zéroe Intrude (ZI)*, a simple black-box attack (see Section 3.5 (Eger and Benz, 2020)); 2) *DeepWordBug (DWB)*, which uses four-character level perturbations including delete, swap, insert, and nearby character swap (Gao et al., 2018); 3) *TextFooler*, a popular baseline that uses word synonyms from counterfeited embeddings to perturb the sample perturbation (Jin et al., 2020); and 4) *SememePSO*, a recent method that uses a seme (e.g., a morpheme) to create a word substitution together with PSO (Zang et al., 2020).

3.3 Gray-box punctuation attack

As a representative gray-box punctuation attack, we implement a variant of DWB through the TextAttack framework that performs only punctuation insertions instead of alphabetical insertions, swaps, deletions, and substitutions. We denote this punctuation variant as DeepWordBugPunc (DWBP). DWBP has three main steps:

- Step 1: Determine the essential words with set $\tau_R = \{\tau_1 ... \tau_k\}$ for an NLP model f using a word delete schema, ranking them from highest to lowest in terms of output logit change. A delete schema, popularized by BERT-Attack (Li et al., 2020), analyzes the logit change when a word is removed from a sample.
- Step 2: Use user-defined set γ (e.g. γ = {-'}) and the RPos (Random Position) and RPunc (Random Punctuation) flags to return a set of transformations $\{\tau_k\}$ from highest-ranking word τ_k from Step 1.
- Step 3: Search over the attack space by querying the victim's model with samples modified with the transformations from Step 2. Keep the best transformation with regard to the logit and semantic similarity score. The next word from τ_R is then perturbed. This is repeated until either $f(x) \neq f(\hat{x})$ or the algorithm iterates through τ_R. This process is called Greedy Search with Word Replacement (GSWR).

In summary, for a sample x, the algorithm identifies the top words in τ_R. It gradually modifies them by inserting one punctuation symbol and making calls to the victim's model through the GSWR Algorithm 1. Optimizing over τ_R results in GSWR being a time-efficient query alternative to the greedy search algorithm. It gradually replaces τ_R in x with transformations from Step 2 by calling Algorithm 2.

Algorithm 2 takes a word and decides the location and punctuation type to insert with the **RPos** and **RPunc** flags. These two flags, when set to false, allow the algorithm to explore the entire attack space. This in turn creates many transformation variations with γ, therefore allowing GSWR to check the adversarial performance of each symbol in γ at each position within the word τ_k. GSWR keeps the transformation if the change creates a successful reduction in logit score. After an adversarial candidate $\hat{x}$ is found, the semantic similarity between x and $\hat{x}$ with $S^{'} = Sim(x, \hat{x})$ is calculated with a deep learning model (Cer et al., 2018). GSWR will reject all perturbations that miss a semantic similarity threshold, set at 0.8, which ensures a good tradeoff between sample quality and adversarial strength (Li et al., 2019). The algorithm repeats this procedure until the end condition.

The difference between DWBP and DWB is that DWB transforms a word with a composition of transformations (letter substitution, deletion, swap, or insertion), and all the transformed words are added to $\{\hat{\tau_k}\}$. Appendix D.1 gives an extended description for the three steps. We tested all variants of RPos and RPunc when applicable.

Algorithm 1 τ_k Transform Function with GSWR

Input: Word ranking τ_R, Sample x, Symbols γ
Output: Adversarial sentence $\hat{x}$

Initialize $\hat{x} = x$
for each τ_k in τ_R **do**
  if len(τ_k) < 2 or τ_k = Stop-Word **then**
    skip
  else
    Transformations Set $\{\hat{\tau_k}\} = TF\gamma(x(\tau_k), \gamma)$
    for $\hat{\tau_k}$ in Transformations Set $\{\hat{\tau_k}\}$ **do**
      $\hat{x} \leftarrow \hat{\tau_k}$
      $\hat{x}^{Adv}$, $\hat{x}^{Adv}_{Score} = f(\hat{x})$
      if Perturbation successful **then**
        Keep best $\hat{\tau_k}$
      else
        Don't keep change → next word
return $\hat{x}$

Algorithm 2 Step2: τ_k Transform Function *TF*

Input: Word τ_k, Symbols γ, Bool: RPos/RPunc
Output: Adversarial word $\hat{\tau_k}$

Transformations = $\emptyset$
if RPos **then**
  if RPunc **then**
    i = RandInt($Start_{Idx}, End_{Idx}$)
    Transformations $\leftarrow \hat{\tau_k} = \tau_k[:i] + \gamma_{random} + \tau_k[i:]$
  else
    i = RandInt($Start_{Idx}, End_{Idx}$)
    for j in γ **do**
      Transformations $\leftarrow \hat{\tau_k} = \tau_k[:i] + \gamma_j + \tau_k[i:]$
else
  if RPunc **then**
    for i in $|Start_{Idx} - End_{Idx}|$ **do**
      Transformations $\leftarrow \hat{\tau_k} = \tau_k[:i] + \gamma_{random} + \tau_k[i:]$
  else
    for i in $|Start_{Idx} - End_{Idx}|$ **do**
      for j in γ **do**
        Transformations $\leftarrow \hat{\tau_k} = \tau_k[:i] + \gamma_j + \tau_k[i:]$
Return Transformations

3.4 Gray-box multi-level attack

We also evaluated the performance of punctuation insertions when used in conjunction with word-level attacks. To conduct this assessment, we employed two baselines TextFooler and SememePSO.

- TextFooler/DWBP: This variant uses the same word scoring function and the GSWR search algorithm. However, $\hat{\tau_k}$ will be a mix of word synonym and punctuation insertion transformations of τ_k.
- SememePSO/DWBP: This variant uses the same word scoring function but with particle swarm optimization (PSO) as a search technique. PSO uses a population-based evolutionary algorithm that exploits the interactions between individuals in a population to find a solution in a search space. $\hat{\tau_k}$ will be a mix of sememes (a type of word substitution) and punctuation insertion transformations of τ_k.

See Section D.5 in the Appendix for details on TextFooler/DWBP and SememePSO/DWBP.

3.5 Black-box punctuation attack

As a representative black-box attack, we implement a variant of the ZI algorithm instead of DWB, as the latter requires access to logits that are absent in this setting. ZI is a simple black-box attack that randomly perturbs a word in a sample with probability p. It then adds a random symbol from this list—!"#$%&'()*+,-./:;<=>?@[\]^`{|}—between two letters with the same probability p, which we define as baseline ZI. In our variant, ZI perturbs a word with probability p (defined as ZIP) but uses the same predefined symbol.

4 Experimental Setup

4.1 Backbone models and tasks

We evaluated the *BERT* (Devlin et al., 2019), *RoBERTa* (Liu et al., 2019), *XLNet* (Yang et al., 2019), *DistilBERT* (Sanh et al., 2020) models on classification (*MR*), entailment (*MNLI*, *SNLI*), and question answering (*QNLI*, *QQP*) tasks. We also used a *CNN* and *LSTM* for *MR* (details are provided in Appendix C).

4.2 Evaluation metrics

We use the evaluation framework previously proposed in (Morris et al., 2020), where an evaluation set is perturbed and out of the Total Attacked Samples (TAS) set the Number of Successful Attacks ($N_{succ-atk}$), Number of Failed Attacks ($N_{fail-atk}$) and Number of Skipped Attacks ($N_{skp-atk}$) are recorded. *After attack accuracy* ($A_{aft-atk} = \frac{N_{fail-atk}}{TAS}$), the most important metric, represents how well the attacker can fool the model across a dataset. Lower values of $A_{aft-atk}$ indicate that the attacker can fool the model better. *After success rate* ($A_{succ-rte} = \frac{N_{succ-atk}}{TAS-N_{skp-atk}}$), is similar to $A_{aft-atk}$ but ignores previously misclassified samples. *Percentage of perturbed words* refers to the percentage of words the algorithm perturbs out of the number of words in the sample. This metric should be as low as possible, as perturbing more words makes the sample's perturbation more detectable. *Semantic similarity* (Jin et al., 2020; Maheshwary et al., 2020) is an automatic similarity index that describes the visual difference between two samples using a deep learning model. In this case, the Universal Sentence Encoder (Cer et al., 2018) is used, along with a cosine similarity measure between the output embeddings. A value of 1 indicates that the two inputs are semantically equivalent, while 0 represents no similarity. *Average number of queries* represents the number of times the algorithm must invoke the model to perform inference. This metric should be kept low to avoid detection.

4.3 Human evaluation

To evaluate the quality of adversarial samples, we conducted four human studies. The first three are the same tests used in TextFooler (Jin et al., 2020), and Hard-Label (Maheshwary et al., 2020). These tests analyzed the adversarial sample for 1) *grammatical correctness*, where reviewers rate the grammatical correctness of the original and adversarial samples on a scale from 1–5, where 1: many grammatical mistakes and 5: no grammatical mistakes; 2) *reviewer classification accuracy*, where reviewers predict the label of each sample; and 3) *similarity*, where reviewers rate if the two samples are similar (1), dissimilar (0), or ambiguous (0.5); 4) *readability*, where the novel application of TOWRE (Tarar et al., 2015) was used to analyze the quality of adversarial words in character-level black-box attacks. TOWRE is a widely used test that measures an individual's reading accuracy and speed. We adapted TOWRE to record the quality of adversarial examples. Specifically, the reviewer pronounces a list of words, where each word was modified with

one out of four different perturbation types introduced with the ZI algorithm. We record the words per minute (WPM) and error rates. All tests had two reviewers who reviewed 100 samples in the first three tests and 36 in the fourth. Agreement between the reviewers was assessed with Krippendorff's alpha, where a score of 1 indicates complete agreement and -1 indicates complete disagreement. Further implementation details on the human testing method and details on Krippendorff's alpha are in Appendix F.

4.4 Defense Baselines

We evaluate fine-tuning and adversarial training as baseline defenses. In detail, it is possible to remove all punctuation during training, fine-tune the model for further epochs on this new punctuationless dataset, and at inference, always strip all punctuation (Table 18). We also experimented with adversarial training (Table 15 in Section 5.6) by using a standard technique (Morris et al., 2020) that is further described in Appendix E.3.

5 Experiments

We use the methodology in Section 3 and experimental setup to explore how punctuation insertions compare to character manipulations (Section 5.1). In Section 5.4, 5.5 and 5.6 we demonstrate how straightforward defence techniques fail and succeed and Sections 5.7, 5.8, and 5.9 highlight the advantages of punctuation insertions where no defence technique is present. The γ choices for each test are summarized and justified in Appendix E.2.

5.1 Punctuation vs character manipulations

How does an attack change when using punctuation insertions instead of letter manipulations? Punctuation insertions can degrade NLP model performance while preserving semantic similarity. The system's $A_{aft-atk}$ is overall reduced (see Figure 1) while semantic similarity remains at 0.96–1.00 when using punctuation insertions (DWBP) compared to 0.87–0.90 when using DWB. Each DWBP box represents the $A_{aft-atk}$ for a dataset across all models with RPos = False. The lower the $A_{aft-atk}$ the more perturbing the attack.

Hyphen, apostrophe, full stop, or comma insertions lower $A_{aft-atk}$ more than any other letter in the alphabet (Figure 2). Values in Figure 2 reflect the after-attack difference [%] between using a letter or punctuation type in an intrusion attack.

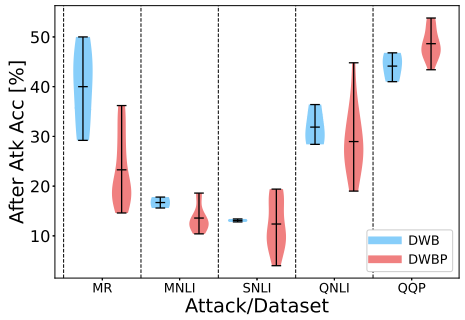

Figure 1: Gray box attack performance

Green/positive values represent an improvement and purple/negative values a decrease between the punctuation symbol on the x-axis and the letters on the y-axis when using DWBP. Each attack in Figure 2 has a constant number of queries, [%] of perturbed words, and query time. The extended results are in Appendix L.

Observation This experiment clarifies that if **any** internal punctuation is present, the system **is vulnerable** and that it is more susceptible to such insertions than other character manipulations and alphabet insertions. We limit our reporting to BERT on MR because other model results are consistent. Full tabular results for other Models and datasets for Figure 1 are in Appendix I in Tables 6 and 7 ("Without Grammar"). While Figure 2 has the other model's results in Appendix L.

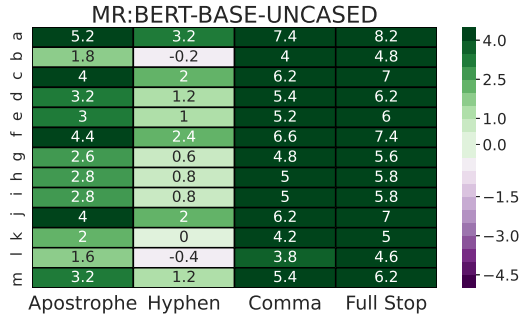

Figure 2: Punctuation vs character insertions. Green indicates positive values; purple indicates negative values

5.2 Punctuation vs word-level attacks

Are there advantages in using punctuation insertions instead of word substitutions? DWBP can also be compared to TextFooler since, with DWBP, a punctuation symbol is mapped to an embedding when using a word piece tokenizer. Figures 3 and 4 each show increasing numbers of unique punctuation symbols from γ (DWBP) or synonyms per word (TextFooler), ranging from 1 to 10. For

DWBP, γ is set to . for $N = 1$, .' for $N = 2$, up to .'-"[,](:) for $N = 10$. In TextFooler, N represents the number of synonyms per word. Figure 3 displays the relationship between N (represented by the points and the x-axis) and improvement in $A_{succ-rte}$ (y-axis). Figure 4 displays the relationship between N (represented by the points), number of queries (x-axis), and the effect on $A_{aft-atk}$ (y-axis). Both experiments used all variations of RPunc/RPos on BERT–MR.

Observation The effectiveness of punctuation insertions is demonstrated DWBP when constrained on N and queries, as seen by the higher $A_{succ-rte}$ with low N in Figure 3 and the low $A_{aft-atk}$ with few queries in Figure 4. Similar results for MNLI can be found in Appendix H.

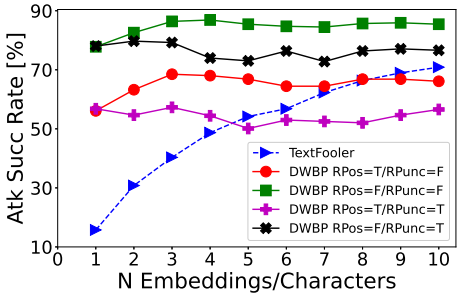

Figure 3: Punctuation embedding efficiency.

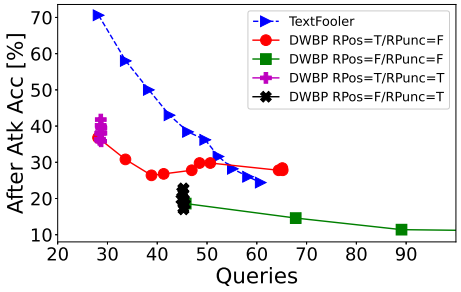

Figure 4: Punctuation query efficiency.

5.3 Punctuation as a multi-level attack

We investigate a composite experiment where τ_k is composed of word substitutions and punctuation insertions. The methodology is introduced in Section 3.4 and the details are given in Appendix D.5. We set RPunc = False, RPos = False, and γ = -'.

Observation: The results in Table 1 indicate that incorporating punctuation insertions into the optimization process enhances TextFooler and SememePSO on BERT trained on MR. The additional findings in Section G.1 of the appendix present results for all tasks and models, and provide further observations.

Dataset	Model (Orig Acc)	Method	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Time Taken [s]	Avg Number Queries
MR	BERT (83.8)	DWBP	17.4	18.32	1	0.721	74.7
		TextFooler	9.4	**17.54**	0.82	1.3072	118.5
		TextFooler/DWBP	**7.6**	18.31	**0.89**	**1.122**	**105.35**
		SememePSO	7	16.52	0.81	16.1811	4950.71
		SememePSO/DWBP	**6**	**9.99**	**0.89**	**7.3252**	**988.44**

Table 1: Multi-level DWBP. Full results in Appendix J

5.4 Removing punctuation as a defense

How does removing punctuation perform as a defense? In this section, we sought to evaluate the effectiveness of simple defenses in countering punctuation attacks by examining the impact of various forms of punctuation removal on attack performance. To aid in this assessment, we employed the use of a casual map in Figure 5, which allows for tracking of the defender's behavior in response to the attacker's changing strategy.

The casual map, presented in the blue quadrant, begins with the "Base Model" on the right-hand side, representing the unchanged finetuned model from Hugging Face, in this instance, specifically BERT finetuned on MR. Adjacent to this model is a large red table, which represents the significant performance drop when utilizing punctuation insertions. For the sake of simplicity, in this map, we limited ourselves to the use of full stops (FS), commas (Co), which are common external punctuation types, and apostrophes (Ap) and hyphens (Hy), which are common internal punctuation types. Given this threat, we identified and explored three options for the defender to take. Beneath the "Base Model," the first option is to remove all punctuation ("All"), which secures the system but leads to an original performance drop of -2.6%. The second option, just beneath "All" is to remove all punctuation found inside of words. While this approach solves the problem, it becomes challenging to identify if a punctuation was inserted by mistake by a user or to prevent the attacker from inserting a whitespace before or after the punctuation insertion. If the attacker adds a whitespace, the attack defaults to the large red table. Furthermore, removing all internal punctuation has a noticeable original performance drop of -1.2%. An alternative to this is to remove all internal punctuation but make an exception for Hy and Ap, reducing the original performance drop to 0%, however, the system remains vulnerable to Ap and Hy. Given the persistent vulnerability to Ap and Hy, the defender may employ a grammar checker to reject all

samples that do not meet a certain grammatical correctness level. When implemented, the robustness of the model increases dramatically, resulting in a semi-secure model.

The blue quadrant also shows "Finetune with punctuation." This base model was further trained and then compared to further training the model when all the punctuation is removed (See "All + Finetune"). As previously highlighted, removing all punctuation can secure the system, the reduced performance drop of -0.6% now indicates that this approach has less of a trade-off between securing the system and original accuracy drop.

In addition, we also explored adversarial training. We discuss the findings of this quadrant in Section 5.6 and it's experimental setup in Section E.3 in the Appendix.

Observation Using a grammar checker increases the robustness of this task. However, as pointed out in the next section in Figure 6, the red candlesticks representing DWBP have a large attack variance depending on the dataset, symbol used, and model. Hence, for a task that results in a semi-secure model, another task may result in a semi-broken model. This reasoning also applies to black box punctuation attacks with ZIP, as pointed out by the large variance in the red candlesticks in Figure 7. Another aspect to consider is the original accuracy drop in performance experienced in the yellow boxes. Depending on the application, this may be acceptable/negligible or unacceptable/too high.

5.5 Grammar checkers as a defense

If a grammar checker preprocesses an input, how does the attack performance change? Another common idea is that character-level attacks are easy to defend against using a grammar checker (Zang et al., 2020). Although adding a grammar checker before processing the input lowers the effectiveness of the attack, punctuation is nonetheless a successful insertion technique with RPos = False, particularly when compared to DWB (Figure 6). Punctuation insertions are also effective in black-box settings (ZIP) and are **as competitive** as alphabetical character manipulations in gray-box settings (DWB) (Figure 7). The high variance of ZIP means that inserting some symbols can lower performance comparably, if not more than any character manipulation technique introduced in DWB. For example, ZIP Ap achieves a 7.8% lower $A_{aft-atk}$ than DWB. The full results can be found in Appendix I (column "With Grammar" in Tables 6 and 7, and the performance of ZIP in Table 10).

Observation DWBP is more successful with the attack, except on the [%] of perturbed words. These results show a curious property of punctuation attacks by highlighting that the [%] of perturbed words is not necessarily aligned with semantic similarity. Therefore, it is possible to have a highly perturbed sample (in terms of [%] of perturbed words) that is nonetheless readable and potentially preserves the original information.

5.6 Adversarial training as a defense

How does adversarial training benefit learning? In this section, we aimed to robustify the model by experimenting with adversarial training on the MR dataset. To test this, we employed the use of the DWBP with hyphens and apostrophes (Hy and Ap). Our findings suggest that adversarial training for language models improves $A_{aft-atk}$. Specifically, $A_{aft-atk}$ increased by 7.4% with Hy and 6.4% with Ap on BERT, as shown in Figure 5. This is demonstrated in the "Adv Training" quadrant, where this model was further finetuned for 4 epochs on the base dataset, while the models beneath it were trained for 4 epochs where the base dataset was extended by 20% with adversarial samples containing either apostrophes or hyphens. The effects of adversarial training were minimal, but did result in an improvement to the model not undergoing any adversarial training. This can be observed by comparing the values in the Broken model to the left of "Adv Training" and to the Broken models that have been adversarially trained beneath "Adv Training". On LSTM, $A_{aft-atk}$ increased by 2.4% with Hy and 1.6% with Ap, with negligible drops/increases in original accuracy, as shown in Table 15 in the appendix.

Observation Our findings are in agreement with previous works, which highlight that adversarial training on large language models, such as BERT or LSTMs, can improve both original and adversarial accuracy (Zhu et al., 2020; Miyato et al., 2017; Cheng et al., 2019; Yoo and Qi, 2021). However, other studies suggest that robustness and generalization may be at odds with one another (Li et al., 2021; Eger and Benz, 2020; Meng and Wattenhofer, 2020). Our experiments also indicate that although adversarial training improves the $A_{aft-atk}$, there is still a large drop in performance.

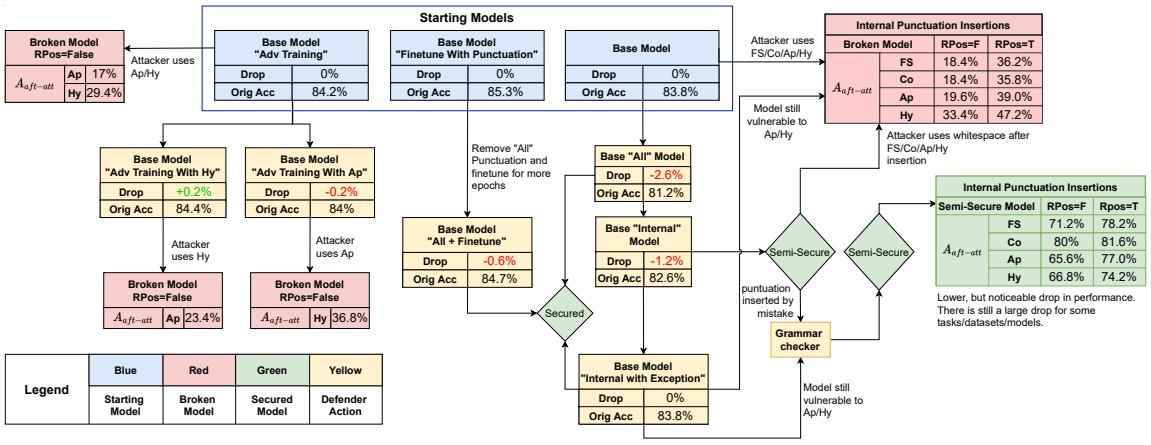

Figure 5: Qualitative casual map for defender/attacker strategy (Section 5.4), values represent BERT–MR.

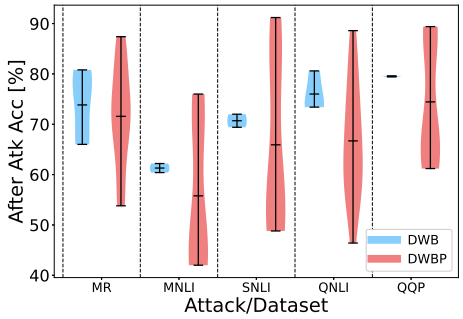

Figure 6: Gray-box attacks against grammar checker

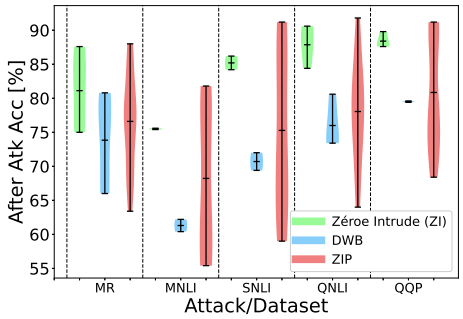

Figure 7: Black-box attacks against grammar checker

5.7 Semantic similarity of punctuation attacks

How similar are samples that have been perturbed with punctuation to the originals? Earlier tests concluded that removing punctuation is an impractical defense technique. We now evaluate the perturbation quality. Apostrophe and hyphen insertions attained a perfect score of 1 for similarity across all samples (in both human and automatic evaluations), a 98% on reviewer classification accuracy (nominal Krippendorff's alpha = 0.960), and a grammatical correctness score difference of 1.29 between the original samples (3.14/5) and adversarial samples (4.43/5), with ordinal Krippendorff's alphas of 0.459 and -0.004 for the original and adversarial samples, respectively. We provide qualitative examples in Table 2 to highlight how the sample changes with punctuation insertions.

MR (Negative Sentiment)	A dark comedy that goes for sick and demented humor simply to do so . the movie is without intent .
TextFooler (Positive Sentiment)	A dark comedy that goes for **psychopathic** and **coot** humor **honestly** to do so . the **film** is without **object** .
DWBP (Positive Sentiment)	A dark comedy that goes for sick and demented humor simply to do so . the movie is **withou't** intent .

Table 2: Qualitative examples of DWBP and TextFooler. **Bold** words represent a perturbed word

Observation The grammar test is widely used (Jin et al., 2020; Maheshwary et al., 2020). However, the low Krippendorff's alphas for grammatical correctness suggest the low reliability of the test in indicating grammatical correctness. Analyzing the visual effect of inserting punctuation makes it possible to observe that the semantics remained unchanged. However, such changes are very noticeable to a human (Table 2).

5.8 Limiting punctuation and readability

Does limiting the punctuation types improve readability? Our tests suggest that focusing on a few types of punctuation facilitate meaning preservation (Section 5.7). Another reason to limit punctuation insertions is to improve readability. To test readability, we used TOWRE, where a reviewer pronounces a list of words with four different perturbations in the test using the Zéroe algorithm with $p = 0.8$ (high perturbation strength). The four types

are: 1) no perturbation (original); 2) ZIP with apostrophe (ZIP Ap), 3) ZI; and 4) character insertions. ZI uses all punctuation symbols from Section 3.5 and character insertions uses all alphabetic characters. Our ZIP Ap method has the fastest reading speeds. Specifically, Table 4 shows an improvement in WPM from 7 WPM (chracter insertion) to 32 WPM (ZI) to 63 WPM (ZIP Ap), with a ratio Krippendorff's alpha of 0.977 and a consistent error rate reduction from character insertions (71.43) to ZI (6.17) to ZIP Ap insertions (1.43). In terms of reading speeds, ZIP Ap is an improvement over character insertions by 800% and by 96% over ZI.

Observation Compared to character insertions and ZI, apostrophe insertions by ZIP Ap are easier and faster to read, as seen with the perfect semantic similarity, WPM improvement, and error reduction. We also show, for the first time, that an attacker cannot use alphabetical character insertions in a high perturbation black-box setting as the samples become too scrambled (Table 3).

MNLI		
Original	Premise	Not only that but they don't pay the money either
	Hypothesis	They also do not contribute financially.
Character Insertions	Hypothesis	**Thzezy** also do not contribute **fdiunlavnyckiwaulvlwyv**.
Zéroe Intrude	Hypothesis	**Th]ely** also do not contribute **f)i^n>a]n{c-i{a}l∗l{y'**.
Insertions (Full Stop)	Hypothesis	**Th.e.y** also do not contribute **f.i.n.a.n.c.i.a.l.l.y.**.

Table 3: Qualitative examples of FS Insertions vs ZI vs Ch. **Bold** words represent a perturbed word

TOWRE	**Method**			
	Original	**ZIP Ap**	**Zéroe Intrude**	**Character Insertions**
Time [s]	24.54	33.60	60.00	60.00
WPM	86.64±9.6	63.35±7.3	32.50±1.5	7.00±0
Errors	0.00±0	0.50±0.5	2.00±0	5.00±1
Error Rate [%]	0.00±0	1.43±1.4	6.17±0.2	71.43±14.3
Self-Corrections	1.00±0	0.50±0	1.50±0	0.50±0.5
Self-Correction Rate [%]	2.86±0	1.43±1.4	4.55±1.3	7.14±7.1

Table 4: Reading efficiency for the four perturbations

5.9 Limiting punctuation types

Is the attack still effective when using a limited punctuation set? Despite limiting the types of punctuation, ZIP performs similarly to ZI and better than character insertions (Figure 8). The test in Figure 8 explores the ability of ZIP with Ap (apostrophe), Hy (hyphen), Co (comma), and FS (full stop) insertions to generalize to black-box attacks. We compare these ZIP intrusions to ZI with all punctuation types and character insertions (Ch) with all alphabet letters using the ZI algorithm on MR-BERT. Figure 8 shows the delta change in $A_{aft-atk}$ for each attack technique against the others for $p = 0.8$. Each square represents the $A_{aft-atk}$ from the x-axis attack method minus the $A_{aft-atk}$ from the y-axis attack method. Table 14 in the Appendix displays the $A_{aft-atk}$ [%] and semantic similarity (S) values for Figure 8. Limiting punctuation with ZIP also avoids grammar checking better than using all punctuation types with ZI (Figure 7) as ZIP can focus on one highly perturbing symbol.

Observation ZIP Ap achieved comparable results to that of Ch and ZI (Figure 8) where the difference is even smaller when comparing with other models. Using character insertions can thus be deemed counterproductive and should be avoided. Attacks should instead focus on only one punctuation type, such as Ap, since, compared to Ch and ZI as Section 5.8 highlighted, readability is preserved.

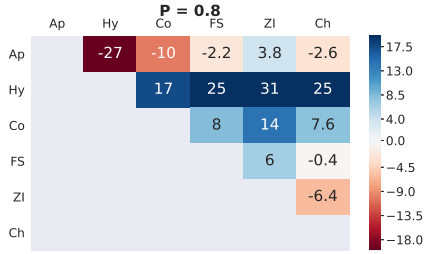

Figure 8: X-Axis $A_{aft-atk}$ minus Y-Axis $A_{aft-atk}$. ZIP (Ap, Hy, Co, Fs) vs ZI, Ch

6 Conclusion

Researching adversarial attacks aims to create a toolbox to identify flaws and improve model robustness. Results show that punctuation insertions as an attack are more effective than character manipulations (Figure 1) and alphabetical character insertions (Figure 2) and better evade grammar checkers (Figures 6, 7). Punctuation insertions preserve more information and are faster to read (Section 5.7, 5.9). Simple defenses and adversarial training are not necessarily effective (Section 5.4, 5.6). The information-preserving characteristic of this attack could potentially evade censorship. Conversely, this highlights that a system deployed to combat fake news and offensive language propagation can potentially be compromised by this use of punctuation. Our defense findings are summarized in Figure 5. We hope this inspires further research in the under-explored area of punctuation and how to process it. The code is available[1].

[1]Provided at EmpiricalPunctuationInsertionAttacks

7 Limitations

This work considers only classification tasks, which raises questions on whether such punctuation types can generalize to research tasks such as fake news, offensive content detection, and seq-to-seq tasks such as translation. From our experiments, we can conclude that punctuation insertion attacks (DWBP) with one symbol (apostrophe or hyphen), given our evaluation metrics work better in terms of after-attack accuracy, readability, and defense avoidance than alphabetical character insertions. However, We've found some limitations and cases where punctuation insertions with apostrophes or hyphens don't work better than the alternative. For example, ZI with all punctuation symbols can achieve on some datasets and models a lower after-attack accuracy, therefore, a better attack success rate Figure8 than using ZIP with an apostrophe of 3.8%. This increase in performance, however, has a cost since the sample will be harder to read. We only tested on English language, punctuation insertions on other languages are mostly unexplored.

References

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

Nicholas P. Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2021. Bad characters: Imperceptible nlp attacks. *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004.

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. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Yong Cheng, Lu Jiang, and Wolfgang Macherey. 2019. Robust neural machine translation with doubly adversarial inputs. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4324–4333, Florence, Italy. Association for Computational Linguistics.

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, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. 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 and Yannik Benz. 2020. From hero to z'eroe: 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, 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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Adam Ek, Jean-Philippe Bernardy, and Stergios Chatzikyriakidis. 2020. How does punctuation affect neural models in natural language inference. In *Proceedings of the Probability and Meaning Conference (PaM 2020)*, Gothenburg. Association for Computational Linguistics.

Brian Formento, See-Kiong Ng, and Chuan Sheng Foo. 2021. Special symbol attacks on nlp. *International (Joint) Conference on Neural Networks*.

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

Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. corr abs/1702.08138 (2017). *arXiv preprint arxiv:1702.08138*.

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. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8018–8025.

Bernard E. M. Jones. 1994. Exploring the role of punctuation in parsing natural text. In *COLING 1994*

Volume 1: The 15th International Conference on Computational Linguistics, Kyoto, Japan.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069, Online. Association for Computational Linguistics.

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

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. 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.

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

Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2020. Generating natural language attacks in a hard label black box setting. In *AAAI Conference on Artificial Intelligence*.

Zhao Meng and Roger Wattenhofer. 2020. A geometry-inspired attack for generating natural language adversarial examples. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6679–6689, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Takeru Miyato, Andrew M. Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *International Conference on Learning Representations*.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. TextAttack: A framework for adversarial attacks, data augmentation, and adversarial training in NLP. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Lichao Sun, Kazuma Hashimoto, Wenpeng Yin, Akari Asai, Jia Li, Philip Yu, and Caiming Xiong. 2020. Adv-bert: Bert is not robust on misspellings! generating nature adversarial samples on bert.

Samson Tan, Shafiq Joty, Min-Yen Kan, and Richard Socher. 2020. It's morphin' time! combating linguistic discrimination with inflectional perturbations. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

Jessica M. Tarar, Elizabeth B. Meisinger, and Rachel H. Dickens. 2015. Test review: Test of word reading efficiency–second edition (towre-2) by torgesen, j. k., wagner, r. k., & rashotte, c. a. *Canadian Journal of School Psychology*, 30(4):320–326.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Neural Information Processing Systems*.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of NLP models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 945–956, Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. Freelb: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations*.

A Future Work

How punctuation attacks can augment an effective adversarial learning schema is still an open question. Our punctuation insertion serve as a foundation for future punctuation manipulation. Hyphens, for example, can be used to s-p-e-l-l out words, syl-la-bi-fi-ca-tion, or to indicate s-stammering or so-so-sobbing in a sentence. There is no research exploring whether stammering or sobbing punctuation perturbations could generate a high-quality adversarial attack on NLP without compromising the meaning.

Exploring whether the identified punctuation types and attacks generalize to more complex prediction tasks like fake news, offensive content detection, and seq-to-seq tasks such as translation is an interesting topic for future work.

B Appendix: Ethics Statement

This research was conducted in accordance with the ACM Code of Ethics.

C Appendix: Task and Datasets

MR: The Rotten Tomatoes movie review dataset holds a sentiment classification task with positive/negative reviews. **QNLI**: Is a question-answering dataset where an annotator extracts the answer from a reference text. The task is to allow the model to predict whether the context sentence holds the answer to the question. **QQP**: Is a duplicate question detection task, where the model is required to detect if the two questions are asking the same thing. **SNLI**: Is composed of human-written sentence pairs where each annotator generates an entailing for each given premise. **MNLI**: It is similar to SNLI but covers multiple genres.

Task	Dataset	Train	Test	Avg Len (Test)	Classes
Sentiment Classification	MR	8.5k	1k	18.7	2
Entailment	MNLI	392k	9.8k	29.2	3
	SNLI	550k	10k	21.4	3
Question Answering	QNLI	105k	5.4k	37.6	2
Duplicate Question	QQP	363k	40k	22.2	2

Table 5: Overview of datasets used in experiments

D Appendix: Methodology Details

Each attack composition has three components, a word scoring function, a set of transformation functions, and a search algorithm.

D.1 Gray-box

D.2 Step 1: Word scoring function

The original DeepWordBug paper introduces four word scoring functions: Replace-1 Score, Temporal Head Score, Temporal Tail Score, and Combination Score. All of which are now outdated. Therefore, for the Gray-Box tests, we use the same schema as that of TextFooler, popularized by BERT-Attack (Li et al., 2020). BERT-Attack records the original sentence's inference logit. Then for each word in the input, the word is deleted. BERT-Attack then extracts a new logit with the remainder of the sample, tracking the difference in value between the original logit and the new logit for each word. It then regards the words with the most significant output change as the most important to f. The original input sentence x with τ_n words is turned to z_L tokens through a tokenizer function F_t in $z \in (z_1 ... z_L)$ tokens. To find the set of most important words, which we call $\tau_R = \{\tau_1 ... \tau_k\}$, that need to be perturbed to attack f, the delete schema associates a rank value R_k for each x_{τ_k}, or sample x without top word τ_k. R_k. Calculate τ_k with:

$$R_i = f(F_t(x))_{x_{Score}} - f(F_t(x_{x \cap x_{\setminus \tau_k}}))_{x_{Score}} \quad (1)$$

where $x_{\setminus \tau_k} = (\tau_1, ..., \tau_{k-1}, \tau_{k+1}, ..., \tau_k)$. Thereafter $\forall R_k \in R$ each word τ_i are ranked highest to lowest, resulting in τ_R, x_{Score} represents the output logit from model f.

D.3 Step 2: Transformation

For every top word in step 1, the second step finds 'transformation' candidates.

DeepWordBug returns four total candidates. The first candidate has a random letter character inserted in a random position. The second has a random letter deleted. The third has a random letter substituted with another, and the fourth changes the position of two adjacent letters.

DeepWordBugPunc adds punctuation symbols in the sentence to create candidates. The number of candidates depends on γ, RPos, and RPunc. γ is user-specific and is the punctuation types that can be inserted. An example is γ = { - ' }. With RPos and RPunc It is possible to choose whether to insert a γ_{random} punctuation symbol at a random location, or return candidates for all possible punctuation insertions and position of such insertions.

D.4 Step 3: Optimization

For every word, Algorithm 2 returns a set of transformations. To choose which transformation is best and whether to keep it, we explore Greedy Search with Word Replacement (GSWR). GSWR is a time-efficient query modification applied to the greedy search algorithm. It replaces with a transformation only words strongly correlated with a change in the output when removed from the input. GSWR keeps the transformation if the change creates a successful perturbation. After an adversarial candidate is found the semantic similarity is calculated, with a deep learning model (Cer et al., 2018), between x and $\hat{x}$ with $S^{'} = Sim(x, \hat{x})$. GSWR will reject all perturbations that do not meet a semantic similarity threshold (set at 0.8).

If $|\tau_k| = 1$ or the word is part of a predefined set of stop words, the algorithm does not do the operation. As the algorithm perturbs top words τ_k, it checks for: if the i perturbation was successful at reducing the logit score, if so, the algorithm keeps the perturbation with γ_i, we define this new sample as $\hat{x}$. This is repeated until either $f(x) \neq f(\hat{x})$ or the algorithm runs out of τ_k.

D.5 Multi-level extension

We also evaluated the performance of punctuation insertions when used in conjunction with word-level attacks. To conduct this assessment, we employed two baselines: 1) TextFooler, a popular method that utilizes word synonyms from counterfeited embeddings to perturb the sample (Jin et al., 2020); and 2) SememePSO, a recent approach that employs a sememe (e.g., a morpheme) to create a word substitution, in conjunction with the use of PSO (Zang et al., 2020).

D.5.1 Gray-box multi-level attack

We explored two multi-level attacks based on TextFooler and SememePSO respectively:

- TextFooler/DWBP: This variant uses the same word scoring function and the GSWR search algorithm. However, $\hat{\tau_k}$ will be a mix of word synonym and punctuation insertion transformations of τ_k.

- SememePSO/DWBP: This variant uses the same word scoring function but with particle swarm optimization (PSO) as a search technique. PSO uses a population-based evolutionary algorithm that exploits the interactions between individuals in a population to find a solution in a search space. $\hat{\tau_k}$ will be a mix of sememes (a type of word substitution) and punctuation insertion transformations of τ_k.

We performed multi-level attacks to explore their effect on deep learning models. The TextFooler/DWBP and SememePSO/DWBP methods result in $\{\hat{\tau_k}\}$ having both word substitutions and punctuation insertion candidates. For TextFooler/DWBP, TextFooler returns 20-word substitutions, and since RPunc and RPos are both false, DWBP returns K transformations. K is proportionate to the number of letters in the word and the length of γ. In our tests, γ = {-'}. Appendix D.5 gives an extended description for the two multi-level attacks and the TextFooler/SememePSO baselines.

To be clear, although we change SememePSO in the SememePSO/DWBP test and TextFooler in the TextFooler/DWBP, we compare SememePSO/DWBP and TextFooler/DWBP to their **unaltered baselines**.

D.5.2 TextFooler

Where Line 6 returns only TextFooler's word synonym substitutions. For τ_k, the algorithm will return 50-word substitutions. This baseline uses GSWR.

D.5.3 TextFooler/DWBP

Line 6 in Algorithm 2 is changed to both call TextFooler's word substitution and DeepWordBug-Punc's punctuation insertion functionality and concatenating the resulting transformations in Transformation Set $\hat{\tau_k}$. For TextFooler/DWBP, TextFooler returns 20-word substitutions, and since RPunc and RPos are both False, DWBP returns N number of transformations. N is proportionate to the number of letters in the word and the length of γ. In our Tests γ = { ' - }. This baseline uses GSWR. Hyperparameter-wise, we reduce TextFooler/DeepWordBugPunc word embeddings for TextFooler from 50 to 20 on all tasks.

D.5.4 SememePSO

uses word substitutions based on sememes together with a different search algorithm based on particle swarm optimization (PSO). We use an existing implementation of SememePSO from the TextAttack library. PSO exploits a swarm composed of individual samples called particles that interact within a space to find a solution iteratively. Every particle,

which in the case of SememePSO is a sample with a sememe word substitution, has a position in the search space and a velocity. Multiple samples with a sememe word substitution form a swarm. Each particle in the swarm is initialized with a random velocity and position. PSO, after that, records for each particle its own best position in the search space and a global best position. This best position is calculated using an optimization score, which is the victim's output logit for a classification task. If one of the samples achieves the desired optimization score, the algorithm is terminated since this sample can attack the model. Otherwise, each particle has its position and velocity updated with values from the individual best position, global best position, the inertia weight, two acceleration coefficients, and two random coefficients. The PSO components would replace lines 10-13 in Algorithm 2.

D.5.5 SememePSO/DWBP

uses both sememe word substitutions and punctuation insertions to construct $\{\tau_k\}$ and uses PSO to find the best substitution out of this set. Hyperparameter-wise for SememePSO/DWBP, the attack is changed by reducing the SememePSO population size from 60 to 5 (MR, QNLI) to 2 (MNLI, SNLI, and QQP) and reducing the number of iterations from 20 to 2 for all tasks.

E Appendix: Implementation Details

E.1 Attack detail

All tests were carried out with the TextAttack (Morris et al., 2020) framework to ensure repeatability, standardization, and ease of future integration. The DeepWordBug baseline, for fairness comparison, has a cosine semantic similarity constraint set to 0.8 with (Cer et al., 2018) to ensure the perturbed sample does not differ too much from the original sample and is comparable to other baselines. For TextFooler, SememePSO we keep the default implementation from TextAttack when comparing them with DWBP in Table 1, Tables in Appendix J and Figure 4,3.

For each sample, we keep the After attack accuracy, the number of queries, semantic similarity, and [%] of perturbed words. These metrics are then averaged across 500 samples to complete each test. All data was sourced from the test set of their respective dataset and sampled under a common/standard seed $\in$ 755, which is the standard seed used in the TextAttack framework. For DeepWordBugPunc the tests in Section 5.2 have been conducted on one punctuation symbol and with RPos = False while γ = {' -}, RPos = True and RPunc = True for tests in section 5.3.

We used BERT, XLNET, and RoBERTa with 110 million parameters and DistilBERT with 66 million parameters. Every test has been run on a 32GB NVIDIA Tesla V100. The TextFooler and DeepWordBugPunc tests took approximately between 30 min and 1 hour to run, while PSO took between 5 and 10 hours. Regarding the human studies, the participants were not paid and were sourced from a lab at a university. All the participants were made aware verbally of how the data would be used. All scientific artifacts from this paper will be made available on GitHub under an MIT license.

It is possible to keep the perturbed words %, semantic similarity, the average time taken, and the average number of queries to concentrate on changes in $A_{aft-atk}$ by adding a word limit constraint on the % of words perturbed in the input. We use this strategy to construct Figure 2.

E.2 Choice of symbols

The experiments in 5 narrow down a choice for γ. We focus on the most frequent punctuation for each dataset (Table 16 in the Appendix) and find that the distribution of common punctuation is similar across datasets. We therefore use all punctuation for Section 5.1 and 5.5 and the ten most popular symbols for Section 5.2, while the other tests focus on apostrophes, hyphens, commas, and full stops (the two most common internal and non-internal symbols; Figure 2, Section 5.9; Figure 8). We use the results from Sections 5.1 and 5.2 to justify multi-level attacks with apostrophes and hyphens. γ = {-'} is a good choice since they are internal punctuation and create added problems to the defender (see Sections 5.4 and 5.5). The human studies in Sections 5.7 and 5.8 tested γ = {-'} and γ = {'}; we did not do human tests on other punctuation insertions as they are visually similar. Nonetheless, we believe the results will be similar regardless of the punctuation type inserted (full stop, comma, apostrophe, or hyphen).

E.3 Adversarial training details

The standard adversarial technique in (Morris et al., 2020; Yoo and Qi, 2021) works by, at each epoch, finding the adversarial sample for each datapoint

(if it exists). It then extends the base dataset by 20% using the adversarial data. For MR, we do fine-tuning and adversarial training for 4 epochs with a batch size of 16 and a learning rate of $2e^{-5}$. We compare this by fine-tuning the same model using the same hyperparameters but on the base dataset.

F Appendix: Human Evaluation Details

F.1 Appendix: Gray-box and multi-level human evaluation

We follow the evaluation strategy used in TextFooler (Jin et al., 2020) and Hard-Label (Maheshwary et al., 2020). Therefore evaluate the quality of the generated samples across three metrics; Grammatical Correctness: Measures in the Likert scale, between 1 and 5. The reviewer compares the adversarial sentence to the grammar of the original as a reference. Classification: Asks the reviewer to classify the sample. We then check if the human classification matches the true label, Similarity: The user inputs a number representing one of three choices where dissimilar is 0, ambiguous 0.5, and 1 similar. The three tests were conducted with two native English-speaking students from India and the UK who have a tertiary university education. They were trained using 3 test samples. We sampled 100 samples at random from MR targeting BERT for this test.

F.2 Appendix: Black-box human evaluation

Finally, we introduce a novel application of TOWRE (Tarar et al., 2015). To generate the word list, we extract all words from the NLTK python package and pick six words randomly for each word length between 4 and 9. The reviewer pronounces 36 words as accurately and fast as possible. The test reports the number of words correctly pronounced, the number of errors, self-corrections, and the time to pronounce the 36 words or the number correctly pronounced in 1 minute. The WPM (Words Per Minute) metric extrapolates from the time or the correct number of words. The reviewers conducting TOWRE are from Singapore and Brazil. Both hold tertiary education. All the tests were conducted in one sitting and took 15 minutes each. To ensure no duplicates existed in the word list, we manually checked the 145 words across the 4 tests and found no duplicates. TOWRE was initially introduced to measure sight word reading fluency. It is widely used in clinical practices to diagnose dyslexia or reading difficulties in children.

F.3 Krippendorff's alpha

We use the Krippendorff's Alpha reliability metric to detect whether a test has statistical significance. Krippendorff's Alpha extracts a value between -1 and 1 after highlighting the agreement between multiple reviewers in a trial. This metric can calculate statistical reliability for nominal (classification, semantic similarity), ordinal (grammar test), and ratio (WPM) data types. A value close to -1 represents complete disagreement between reviewers normalizing by chance, 0 represents neither statistical agreement nor disagreement, and 1 is perfect agreement.

G Extra Findings

G.1 Punctuation as a multi-level attack

Extra Observation We find an interesting trade-off between $A_{aft-atk}$, sample quality, and attack time efficiency depending on the influence of punctuation insertions over the word-level attacks. Hyperparameter-wise, the changes in Section D.5.1 increase the attack effectiveness of punctuation insertions by decreasing classification accuracy after the attack ($A_{aft-atk}$) while increasing the quality/meaning/readability of the text. These changes are also more efficient compared to other hyperparameters because the number of queries and amount of time taken to optimize the sample are decreased.

Other hyperparameters can achieve lower $A_{aft-atk}$ but at the cost of time, queries, and sample quality. We hypothesize that this interesting behavior derives from punctuation insertions being unconstrained by a similarity constraint. These attacks can inject information from different parts of the embedding space by inserting punctuation and avoiding word substitutions. Analyzing the visual effect of inserting punctuation makes it possible to observe that the semantics remained unchanged. However, such changes are more noticeable than word substitutions (Table 2).

H Appendix: Extended Budget Study

Figure 9 illustrates how the $A_{succ-rte}$ improves as each word in the sample can be either replaced with N synonyms (TextFooler) or have one of N punctuation characters inserted in the word (DWBP) in four different ways according to how the flags RPos/RPunc are set. The behavieour of RPos and

RPunc changes DWBP, as previously explained in Section 3.3.

Increasing the number of word synonyms in TextFooler or potential punctuation symbols in DWBP results in more transformations ($\hat{\tau}_k$) that the GSWR algorithm needs to evaluate by performing queries to the victim model, therefore seaching for the optimal transformation. The query response is shown in Figure 10. Both tests suggest that DWBP performs better with limited word synonyms and limited queries from the attacker. On the other hand, TextFooler performs better when the algorithm has many synonym candidates to chose from for each word.

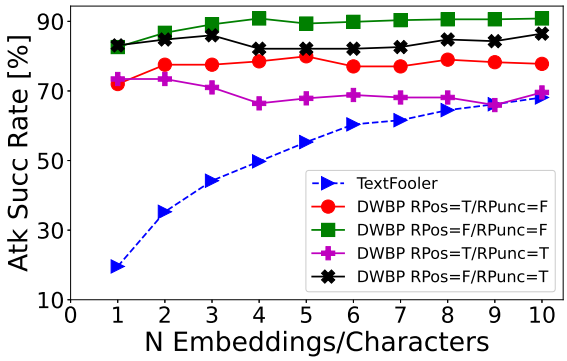

Figure 9: After Success Rate (higher is better) as the number of characters in γ is increased for DWBP vs the number N of synonyms is increased for TextFooler

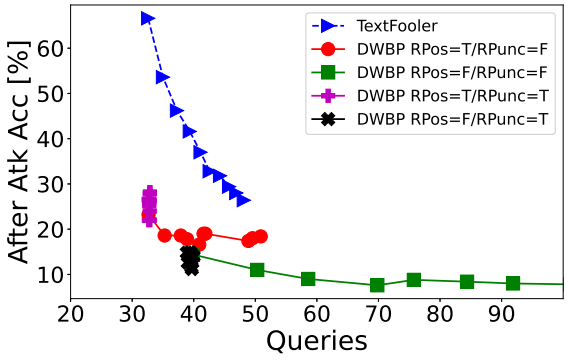

Figure 10: After Attack Accuracy (Lower is better) vs the number of queries required to find an adversarial solution. Each point represents the number of unique punctuation symbols (for DWBP) or synonyms (for TextFooler) from 1 to 10

I Appendix: Extended Non-Grammar/Grammar Checker Attack Results

The extended results of DWBP when a model employs a grammar checker (Language Tool) as a defense technique are in Table 6. The table is with RPos = False. With RPos = True (Table 8), although it requires less queries the attack is not as effective, especially when there is a grammar checker (Figure 11 and 12). We also report the results for the most frequent non internal punctuation with RPos = False (Table 7) and with RPos = True (Table 9). Limiting punctuation is also effective against a grammar checker. The findings in fact generalize to a black box attack (Table 10). This table shows that Zeroe with all characters is ineffective and limiting punctuation is competitive with a gray-box character attack technique.

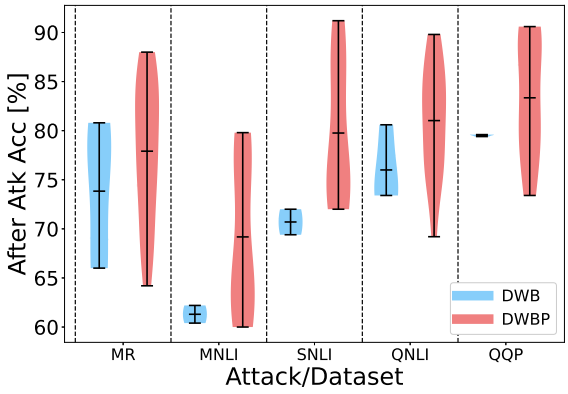

Figure 11: Summary of 'With Grammar Checker' (Table 8 and 9) $A_{aft-atk}$ across datasets with RPos = True

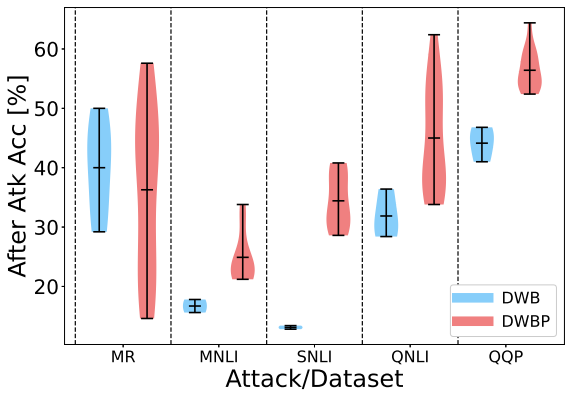

Figure 12: Summary of 'Without Grammar Checker' (Table 8 and 9) $A_{aft-atk}$ across datasets with RPos = True

J Appendix: Extended Multi-level Attack Results

The results for multi-level DWBP and DWBP on MR, MNLI, SNLI, QQP and QNLI across all models is shown in Table 11,12,13

Dataset	Model (Orig Acc)	Method	Without Grammar Checker				With Grammar Checker			
			After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries
MR	CNN (76.6)	DWB	34.2	9.86	0.87	32.3	66.4	7.41	0.88	26.23
		DWBP -	26.6	14.24	**1**	44.09	54.6	9.61	**1**	29.9
		DWBP '	**14.8**	16.86	**1**	43.23	**53.8**	10.3	**1**	28.22
	LSTM (77)	DWB	29.2	10.16	0.87	32.05	66	7.5	0.88	26.32
		DWBP -	26.4	13.71	**1**	43.1	**53.8**	10.8	**1**	30.02
		DWBP '	**19.2**	14.91	**1**	42.2	55.6	10.33	**1**	28.44
	BERT (83.8)	DWB	43.2	10.79	0.87	35.5	77.8	8.4	0.89	26.95
		DWBP -	33.4	14.59	**1**	46.51	66.8	11.75	**1**	31.58
		DWBP '	**19.6**	17.89	**1**	47.02	**65.6**	12.54	**1**	29.52
	RoBERTa (88)	DWB	50	11.58	0.87	35.17	80.8	8.64	0.87	26.73
		DWBP -	36.2	16.44	**1**	45.88	**71.2**	12.66	**1**	31.25
		DWBP '	**18.8**	19.64	**1**	47.53	72.6	13.14	**1**	30.02
	XLNet (87)	DWB	43.4	11.06	0.86	34.92	78.2	7.77	0.88	26.89
		DWBP -	35.8	15.77	**1**	46.29	**70**	12.69	**1**	31.58
		DWBP '	**20.2**	19.63	**1**	47.86	71	12.02	**1**	30.11
MNLI	BERT (82.8)	DWB	15.6	7.67	0.9	38.98	62.2	5.94	0.91	31.89
		DWBP -	18.4	7.61	**1**	39.07	46	7.42	**1**	33.82
		DWBP '	**12.2**	8.62	**1**	40.36	**42.6**	8.62	**1**	33.85
	DistilBERT (80.6)	DWB	17.8	7.24	0.9	38.84	60.4	6.15	0.9	31.82
		DWBP -	18.6	7.54	**1**	38.93	42.4	7.25	**1**	33.68
		DWBP '	**12**	8.22	**1**	40.19	**42**	8.39	**1**	33.7
SNLI	BERT (91.2)	DWB	13.4	8.45	0.88	29.58	69.4	6.28	0.89	23.83
		DWBP -	18.8	7.44	**1**	29.71	**51.2**	7.41	**1**	25.41
		DWBP '	**10.2**	8.28	**1**	30.23	52.4	7.7	**1**	25.12
	DistilBERT (86.6)	DWB	12.8	8.51	0.89	29.92	72	6.13	0.89	24.25
		DWBP -	19.4	7.48	**1**	29.73	52.6	7.14	**1**	25.65
		DWBP '	**7.4**	8.79	**1**	30.56	**48.8**	8.4	**1**	25.45
QNLI	BERT (91.2)	DWB	30.8	8.98	0.9	65.04	74	6.37	0.92	47.69
		DWBP -	38	7.7	**1**	70.61	59.8	7.4	**1**	51.46
		DWBP '	**27.6**	9.54	**1**	75.12	**55.4**	8.56	**1**	51.86
	RoBERTa (92)	DWB	36.4	9.79	0.9	65.86	80.6	5.99	0.93	47.94
		DWBP -	44.8	9.39	**1**	76.46	71.6	7.08	**1**	52.79
		DWBP '	**32**	11.36	**1**	80.47	**66.6**	8.86	**1**	53.72
	DistilBERT (86.2)	DWB	28.4	9.23	0.91	63.68	73.4	6.23	0.92	47.96
		DWBP -	35.4	7.87	**1**	71.42	58.6	6.55	**1**	51.5
		DWBP '	**26.2**	9.41	**1**	74.53	**55.4**	7.56	**1**	51.94
QQP	BERT (90.4)	DWB	46.8	8.42	0.9	39.42	79.6	7.48	0.9	25.79
		DWBP -	50.6	7.4	**1**	39.24	**61.2**	8.07	**1**	28.58
		DWBP '	47	8.44	**1**	41.98	62.8	8.95	**1**	28.68
	DistilBERT (90.8)	DWB	41	9.77	0.89	37.87	79.4	7.01	0.91	25.97
		DWBP -	53.2	7.35	**1**	38.67	62.8	8.11	**1**	28.53
		DWBP '	45	9.16	**1**	41.62	**61.4**	9.37	**1**	28.7
	XLNet (91.2)	DWB	44.6	9.58	0.89	38.93	79.6	7.44	0.9	25.95
		DWBP -	53.2	8.85	**1**	38.91	**68.8**	9.65	**1**	28.6
		DWBP '	47.6	10.18	**1**	42.8	70.2	10.46	**1**	29.05

Table 6: Results without (Original) and when using the LanguageTool grammar checker with RPos=False and internal punctuation

K Appendix: Black-Box Heatmaps

The extended results for the performance difference between ZIP (apostrophe (Ap), hyphen (Hy), comma (Co), full stop (FS) and for QQP question mark (Qu)), character insertions, Zéroe on MR, MNLI, SNLI, QQP and QNLI are in Figure14 for LSTM on MR, Figure 13 for BERT on MR, Figure 16 for DistilBERT on MNLI, Figure 15 for BERT on MNLI,18 for DistilBERT on SNLI, Figure 17 for BERT on SNLI, 20 for DistilBERT on QNLI, Figure 19 for BERT on QNLI, 22 for DistilBERT on QQP, Figure 21 for BERT on QQP. For BERT on MR we present the values to construct figure 13 in Table 14 as an example.

L Appendix: Punctuation vs Characters

The extended results for the performance increase in terms of after attack accuracy between inserting letters and punctuation (apostrophe, hyphen, comma, full stop), character insertions, Zéroe on MR, MNLI, SNLI, QQP and QNLI are in Figure 23 for LSTM on MR, Figure 24 for BERT on MR, Figure 25 for DistilBERT on MNLI, Figure 26 for BERT on MNLI,27 for DistilBERT on SNLI, Figure 28 for BERT on SNLI, 29 for DistilBERT on QNLI, Figure 30 for BERT on QNLI, 31 for DistilBERT on QQP, Figure 32 for BERT on QQP.

The results show a constant improvement across all tasks except for QQP when inserting punctuation. Interestingly the strongest punctuation inser-

Dataset	Model (Orig Acc)	Method	Without Grammar Checker				With Grammar Checker			
			After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries
MR	CNN (76.6)	DWBP .	15.4	16.61	0.97	44.1	62.2	9.43	0.99	27.42
		DWBP ,	14.6	16.84	1	43.2	71	6.14	1	24.27
		DWBP "	27.2	14.12	1	44.2	75.8	4.47	1	22.11
	LSTM (77)	DWBP .	19.4	14.86	0.97	42.93	63.4	8.73	0.99	27.31
		DWBP ,	19.2	14.91	1	42.2	72.6	5.76	1	24.28
		DWBP "	26.6	13.62	1	43.16	76.2	4.7	1	22.01
	BERT (83.8)	DWBP .	18.4	16.72	0.97	45.57	71.2	10.1	0.98	27.81
		DWBP ,	18.4	16.64	1	45.78	80	9.1	1	24.57
		DWBP "	29.4	14.43	1	44.94	83.2	6.15	1	22.32
	RoBERTa (88)	DWBP .	19.4	19.35	0.96	48.81	79.2	9.96	0.98	28.22
		DWBP ,	18.6	19.53	1	47.12	85	5.06	1	24.6
		DWBP "	34.6	16.5	1	46.49	87.4	6.24	1	22.34
	XLNet (87)	DWBP .	17.8	19.85	0.97	48.34	75.6	10.44	0.99	28.31
		DWBP ,	18	19.68	1	47.46	84.4	5.85	1	24.7
		DWBP "	34	15.89	1	45.69	87	0	0	22.41
MNLI	BERT (82.8)	DWBP .	14	7.94	1	39.98	47.8	7.46	1	32.62
		DWBP ,	12.6	7.77	1	39.59	76	4.1	1	30.09
		DWBP)	11.6	8.08	1	40.04	71.2	5.38	1	30.4
	DistilBERT (80.6)	DWBP .	12.8	7.36	1	39.47	45.4	7.28	1	32.37
		DWBP ,	13.2	7.41	1	39.28	74	5.28	1	30.07
		DWBP)	10.4	7.95	1	39.49	70.4	6.22	1	30.4
SNLI	BERT (91.2)	DWBP .	10	7.88	1	29.98	57	6.92	1	24.2
		DWBP ,	10.6	8.31	1	30.08	85.6	5.92	1	22.54
		DWBP "	17	7.75	1	29.79	91.2	0	0	22.13
	DistilBERT (86.6)	DWBP .	9.6	8.14	1	30.26	54	7.55	1	24.6
		DWBP ,	4	8.84	1	30.19	80.6	5.91	1	23.01
		DWBP "	16.8	7.36	1	29.58	85.8	5.81	1	22.58
QNLI	BERT (91.2)	DWBP ,	25	9.36	1	73.9	83.6	5.12	1	42.49
		DWBP .	26.8	9	1	74.66	66.6	7.27	1	48.93
		DWBP ?	25	9.92	1	74.08	58.2	8.33	1	50.53
	RoBERTa (92)	DWBP ,	28.6	12.23	1	79.46	88.6	4.62	1	42.47
		DWBP .	32.2	11.6	1	80.69	76.4	7.69	1	49.69
		DWBP ?	31.4	11.84	1	81.15	71	8.39	1	52.09
	DistilBERT (86.2)	DWBP ,	19	10.09	1	70.73	79.2	4.93	1	42.5
		DWBP .	19.2	9.66	1	70.2	63	7.11	1	48.87
		DWBP ?	23.2	7.66	1	71.08	46.4	7.78	1	49.13
QQP	BERT (90.4)	DWBP ?	46.2	8.41	1	41.96	64.4	8.33	1	28.03
		DWBP ,	48.6	8.29	1	42.31	86.8	6.28	1	23.5
		DWBP "	49.4	7.78	1	39.01	88.6	8.86	1	23.28
	DistilBERT (90.8)	DWBP ?	43.4	9.39	1	41.27	64.8	9.39	1	28.04
		DWBP ,	46.2	8.98	1	41.69	87.2	6.27	1	23.54
		DWBP "	50.4	7.82	1	37.95	88.2	6.98	1	23.26
	XLNet (91.2)	DWBP ?	47.4	10.19	1	42.83	70.8	10.65	1	28.33
		DWBP ,	47.6	10.34	1	42.64	89.2	7.12	1	23.6
		DWBP "	53.8	9.45	1	38.69	89.4	7.64	1	23.32

Table 7: Results without (Original) and when using the LanguageTool grammar checker with RPos=False and most frequent non internal punctuation from Table 16

tion appears to vary between tasks. For example, the comma is the strongest in MNLI for BERT, while the full stop is strongest for SNLI on BERT. Moreover, whether there are character insertions or punctuation insertions in the QQP task seems to have little to no difference; at times, character insertions are better, for example, when inserting a hyphen in QQP when trained on BERT. We speculate that QQP is hard to attack, whether using character or punctuation insertions. It could be hard to attack because a model is sensitive to samples with similar question pairs. Hence, it is easy to perturb them to become unsimilar by adding character or punctuation symbols. However, to perturb a nonsimilar question pair to become similar is harder, and neither character nor punctuation symbols can do this. Future research to prove this can investigate this phenomenon by plotting the ROC and Precision/Recall graphs. However, the high $A_{aft-atk}$ in 13 and Table 6 in the Appendix, especially compared to other tasks, is a good indication of this theory being correct. Exploring the reasons behind these phenomena, and introducing a novel attack that can further decrease the $A_{aft-atk}$ of QQP, could be an interesting entry point for future research.

M Appendix: Adversarial training results

See Table 15 for the results of adversarial training using DWBP with hyphen insertions.

Dataset	Model (Orig Acc)	Method	Without Grammar Checker				With Grammar Checker			
			After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries
MR	CNN (76.6)	DWB	34.2	9.86	0.87	32.3	66.4	7.41	0.88	26.23
		DWBP -	26.8	14.18	1	26.54	67	9.42	1	24.39
		DWBP ’	14.8	16.85	1	26.37	68.4	7.11	1	23.99
	LSTM (77)	DWB	29.2	10.16	0.87	32.05	66	7.5	0.88	26.32
		DWBP -	26.6	13.57	1	26.23	64.2	9.43	1	24.29
		DWBP ’	19.2	14.91	1	26.05	67.2	7.87	1	23.87
	BERT (83.8)	DWB	43.2	10.79	0.87	35.5	77.8	8.4	0.89	26.95
		DWBP -	47.2	15.49	1	28.67	74.2	11.05	1	24.86
		DWBP ’	39	18.21	1	28.94	77.4	9.61	1	24.37
	RoBERTa (88)	DWB	50	11.58	0.87	35.17	80.8	8.64	0.87	26.73
		DWBP -	55.2	16.24	1	28.58	81.2	11.28	1	24.85
		DWBP ’	43.6	19.12	1	29.43	83.8	9.48	1	24.45
	XLNet (87)	DWB	43.4	11.06	0.86	34.92	78.2	7.77	0.88	26.89
		DWBP -	57.6	14.6	1	28.79	79.2	9.35	1	25.02
		DWBP ’	47.4	19.22	1	29.72	82.4	9.04	1	24.55
MNLI	BERT (82.8)	DWB	15.6	7.67	0.9	38.98	62.2	5.94	0.91	31.89
		DWBP -	33.8	7.99	1	32.58	63.4	6.84	1	31.42
		DWBP ’	26	9.54	1	33.04	68	6.56	1	31.21
	DistilBERT (80.6)	DWB	17.8	7.24	0.9	38.84	60.4	6.15	0.9	31.82
		DWBP -	31.4	8.25	1	32.49	60	6.9	1	31.29
		DWBP ’	24.4	9.5	1	32.88	62.8	6.89	1	31.24
SNLI	BERT (91.2)	DWB	13.4	8.45	0.88	29.58	69.4	6.28	0.89	23.83
		DWBP -	39.8	8.43	1	24.63	72.6	7.02	1	23.43
		DWBP ’	33.8	10.12	1	24.98	76.4	6.9	1	23.14
	DistilBERT (86.6)	DWB	12.8	8.51	0.89	29.92	72	6.13	0.89	24.25
		DWBP -	40.8	7.8	1	24.89	72.8	6.45	1	23.74
		DWBP ’	28.6	9.7	1	25.2	72	6.43	1	23.52
QNLI	BERT (91.2)	DWB	30.8	8.98	0.9	65.04	74	6.37	0.92	47.69
		DWBP -	48.4	7.95	1	47.62	76.4	6.07	1	43.75
		DWBP ’	39.6	9.55	1	48.67	77.8	6.34	1	43.71
	RoBERTa (92)	DWB	36.4	9.79	0.9	65.86	80.6	5.99	0.93	47.94
		DWBP -	62.4	8.56	1	49.13	83.4	5.86	1	43.96
		DWBP ’	52.6	10.83	1	50.83	84.4	6.18	1	43.75
	DistilBERT (86.2)	DWB	28.4	9.23	0.91	63.68	73.4	6.23	0.92	47.96
		DWBP -	48.6	7.85	1	47.87	75.2	6.15	1	43.89
		DWBP ’	39.8	9.57	1	48.88	75	6.43	1	43.88
QQP	BERT (90.4)	DWB	46.8	8.42	0.9	39.42	79.6	7.48	0.9	25.79
		DWBP -	54.2	8.53	1	27.46	73.4	7.95	1	25.1
		DWBP ’	52.4	9.22	1	28.01	77.6	7.72	1	25.05
	DistilBERT (90.8)	DWB	41	9.77	0.89	37.87	79.4	7.01	0.91	25.97
		DWBP -	57.4	8.1	1	27.44	73.4	7.67	1	25.12
		DWBP ’	52.4	10.39	1	28.1	78.8	7.68	1	25.15
	XLNet (91.2)	DWB	44.6	9.58	0.89	38.93	79.6	7.44	0.9	25.95
		DWBP -	59.8	9.91	1	27.53	81.4	8.07	1	25.26
		DWBP ’	60.2	11.18	1	28.46	84.4	8.75	1	25.32

Table 8: Results without (Original) and when using the LanguageTool grammar checker with RPos=True and internal punctuation

N Appendix: Analysis

N.1 Empirical punctuation counts across datasets

The variance of symbols within each dataset is high. Table 16 shows the number of punctuation symbols and their proportion as a percentage of other characters for each dataset. The table is subdivided into ‘Total Punctuation’ and ‘Internal punctuation’ or the punctuation only appearing within words, such as apostrophes and hyphens. This distinction is essential, as Section 5 empirically motivates why punctuation can be used as an attack vector and cannot be easily defended.

N.2 Removing Punctuation

Table 17 shows the impact on all models across all datasets of removing either all punctuation, only internal punctuation, or just removing internal punctuation except the apostrophe and hyphen, which the two punctuation characters over-represented within words, as seen from Table 16. On the other hand, Table 18 shows how the original accuracy changes if the models are finetuned on data with no punctuation.

N.3 Most frequent punctuation in dataset attack

Table 19 highlights the drop in performance by the type of punctuation symbol used in the attack. The attack uses the most frequent symbols in a sample

Dataset	Model (Orig Acc)	Method	Without Grammar Checker				With Grammar Checker			
			After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Number Queries
MR	CNN (76.6)	DWBP .	16.2	16.41	0.97	26.34	70.2	7.21	0.99	23.51
		DWBP ,	14.6	16.83	1	26.35	74	4.55	1	22.47
		DWBP "	27	14.15	1	26.56	76.2	5	1	21.89
	LSTM (77)	DWBP .	20	14.68	0.98	26.03	70.2	7.21	0.99	23.34
		DWBP ,	19.2	14.91	1	26.05	74.8	6.37	1	22.34
		DWBP "	26.6	13.58	1	26.23	76.6	4.67	1	21.77
	BERT (83.8)	DWBP .	36.2	16.26	0.97	28.47	78	7.74	0.98	23.73
		DWBP ,	35.8	16.19	1	28.5	81.6	8.65	1	22.64
		DWBP "	44.4	13.91	1	28.33	83.2	6.15	1	22.05
	RoBERTa (88)	DWBP .	46	18.44	0.98	29.42	86	8.91	1	23.85
		DWBP ,	42.8	18.9	1	29.25	87.4	5.94	1	22.66
		DWBP "	55.2	14.86	1	28.55	88	0	0	22.09
	XLNet (87)	DWBP .	43.6	18.52	0.97	29.4	84.4	7.22	0.99	23.95
		DWBP ,	45.6	18.53	1	29.53	85.2	6.39	1	22.79
		DWBP "	56.4	15.3	1	28.73	87	0	0	22.16
MNLI	BERT (82.8)	DWBP .	22	9.63	1	32.86	63.8	5.92	1	30.76
		DWBP ,	23.2	8.76	1	32.79	79.8	3.76	1	29.71
		DWBP)	22.8	9.64	1	32.9	77.2	5.11	1	29.95
	DistilBERT (80.6)	DWBP .	21.8	8.77	1	32.65	62	5.76	1	30.75
		DWBP ,	22.4	9.01	1	32.67	77.6	5.09	1	29.67
		DWBP)	21.2	9.15	1	32.67	77.2	4.52	1	29.93
SNLI	BERT (91.2)	DWBP .	33.4	9.17	1	24.87	76.4	6.76	1	22.85
		DWBP ,	31	9.68	1	24.85	89	6.53	1	22.21
		DWBP "	39.8	9.23	1	24.69	91.2	0	0	22.09
	DistilBERT (86.6)	DWBP .	29.4	9.68	1	25.21	75.8	5.86	1	23.25
		DWBP ,	30.4	9.33	1	25.15	85.2	4.71	1	22.64
		DWBP "	37.2	8.34	1	24.87	86.2	5.16	1	22.52
QNLI	BERT (91.2)	DWBP ,	37.6	9.79	1	48.5	88.6	4.99	1	40.14
		DWBP .	40.6	8.98	1	48.59	81.6	5.01	1	42.52
		DWBP ?	36.2	9.99	1	48.36	81.4	5.05	1	43.1
	RoBERTa (92)	DWBP ,	52.4	11.43	1	50.71	89.8	3.76	1	39.97
		DWBP .	56.4	9.99	1	51	87	4.62	1	42.45
		DWBP ?	54.8	10.6	1	51.06	85.8	5.22	1	43.05
	DistilBERT (86.2)	DWBP ,	36.8	10.06	1	48.7	82.8	4.97	1	40.1
		DWBP .	33.8	9.62	1	47.88	77	5.09	1	42.46
		DWBP ?	35	8.13	1	47.85	69.2	5.27	1	42.67
QQP	BERT (90.4)	DWBP ?	54.8	9.45	1	28.08	78.4	8.18	1	24.79
		DWBP ,	53.6	9	1	28.01	88.8	8.2	1	23.08
		DWBP "	55.4	8.41	1	27.45	90.2	5	1	23
	DistilBERT (90.8)	DWBP ?	53.6	9.86	1	28.07	80.4	7.6	1	24.81
		DWBP ,	54.4	10.1	1	28.17	89	6.4	1	23.11
		DWBP "	56.4	8.82	1	27.38	89.8	7.36	1	23.03
	XLNet (91.2)	DWBP ?	58.8	11.63	1	28.51	83.6	7.68	1	24.88
		DWBP ,	58.4	11.57	1	28.46	90.6	7.82	1	23.15
		DWBP "	64.4	10.14	1	27.63	90.4	6.98	1	23.1

Table 9: Results without (Original) and when using the LanguageTool grammar checker with RPos=True and most frequent non internal punctuation from Table 16

for each task in Table 16.

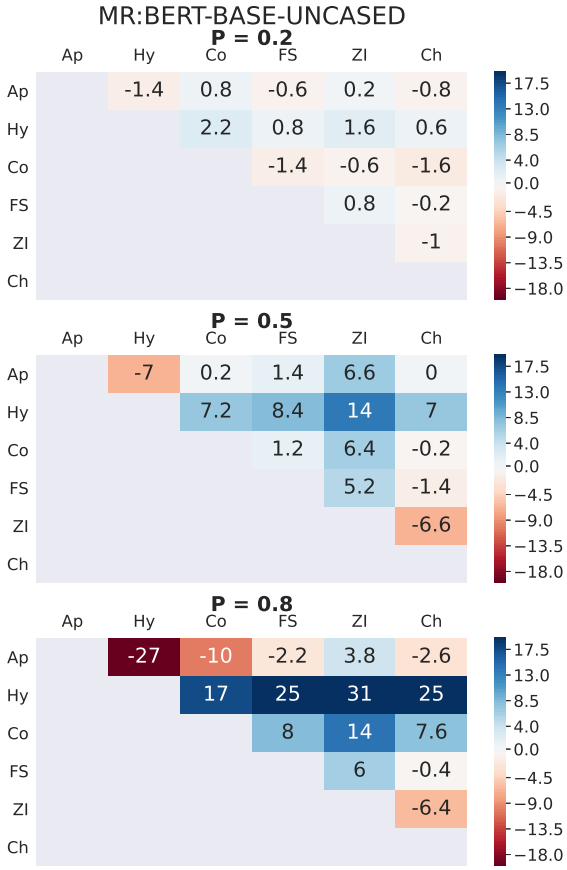

Figure 13

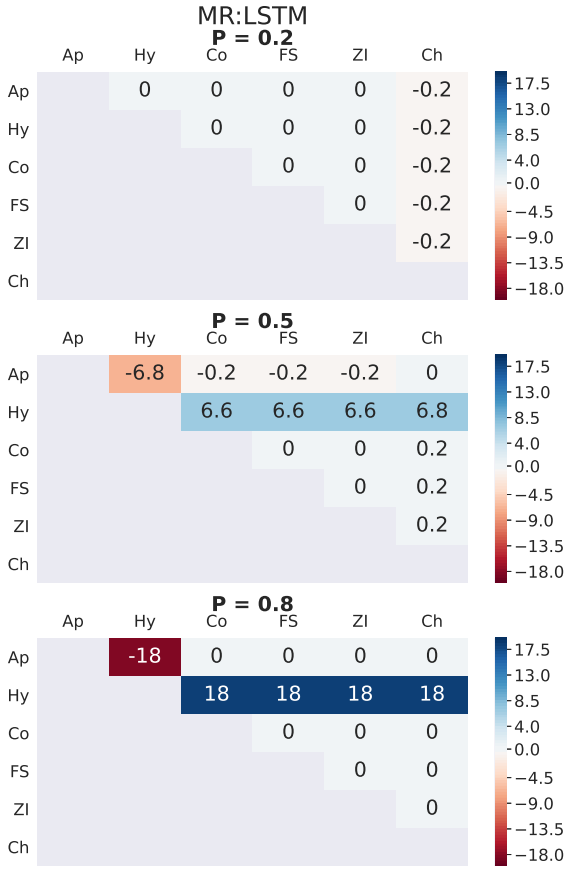

Figure 14

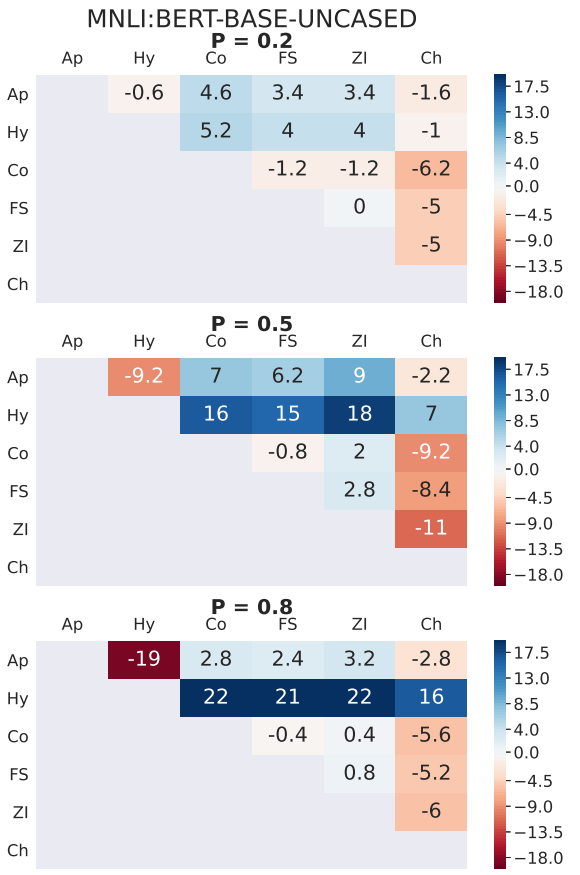

Figure 15

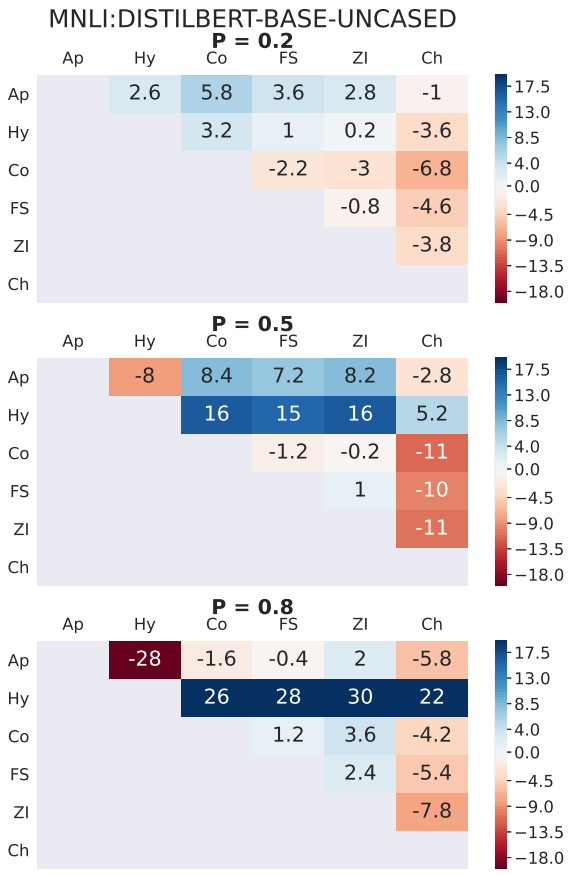

Figure 16

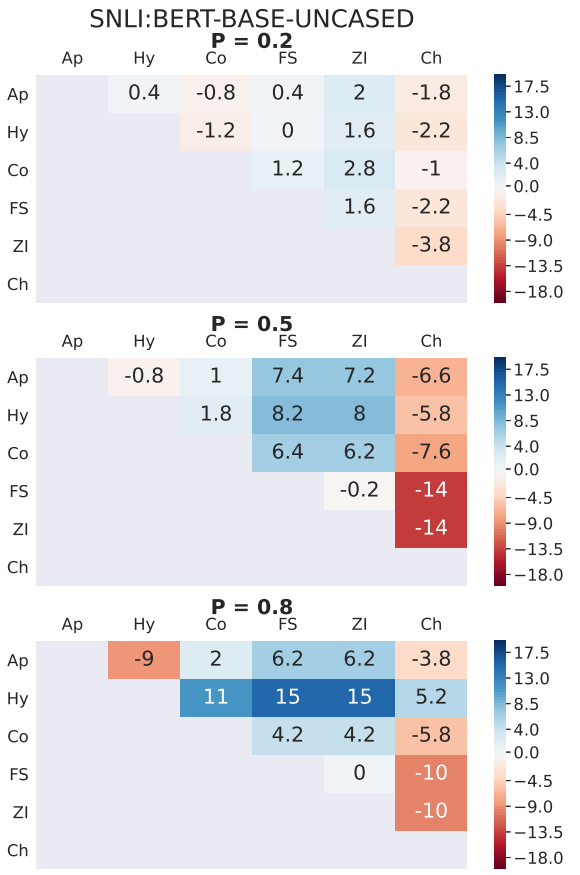

Figure 17

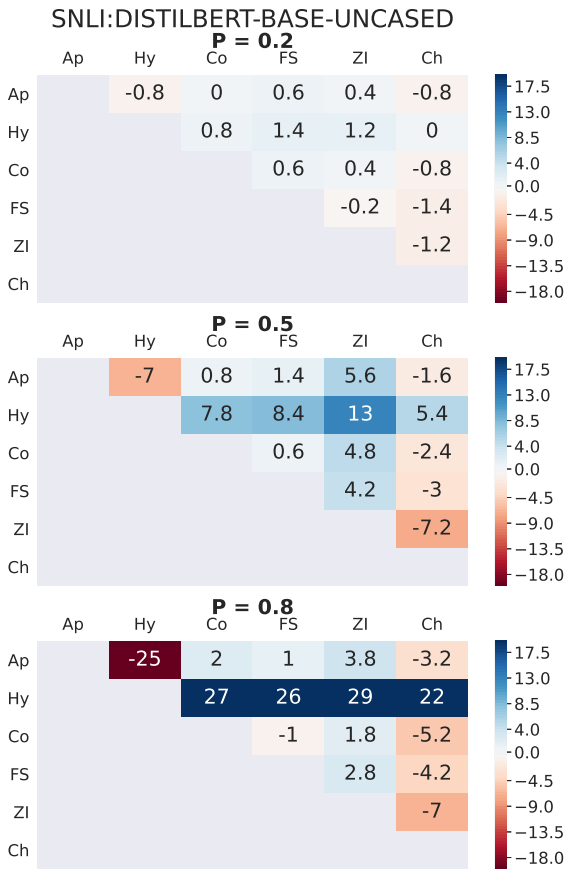

Figure 18

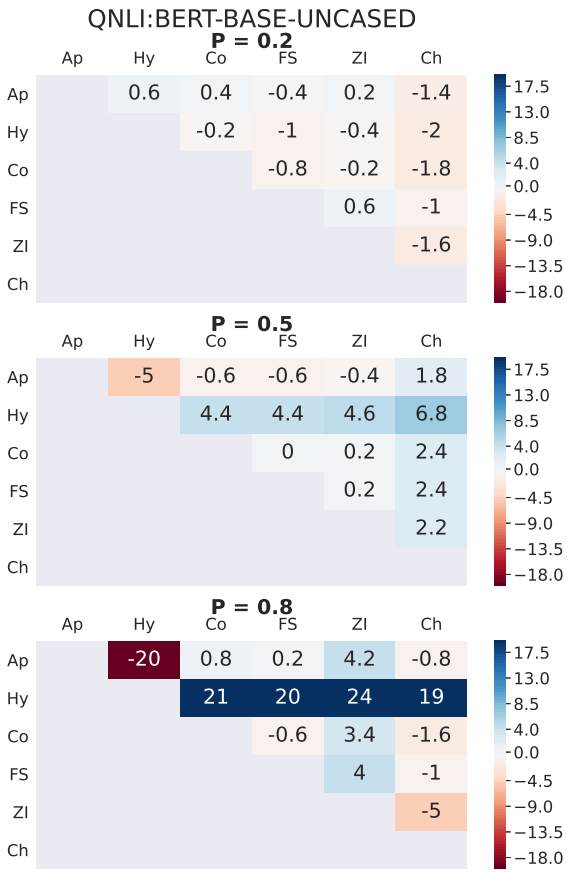

Figure 19

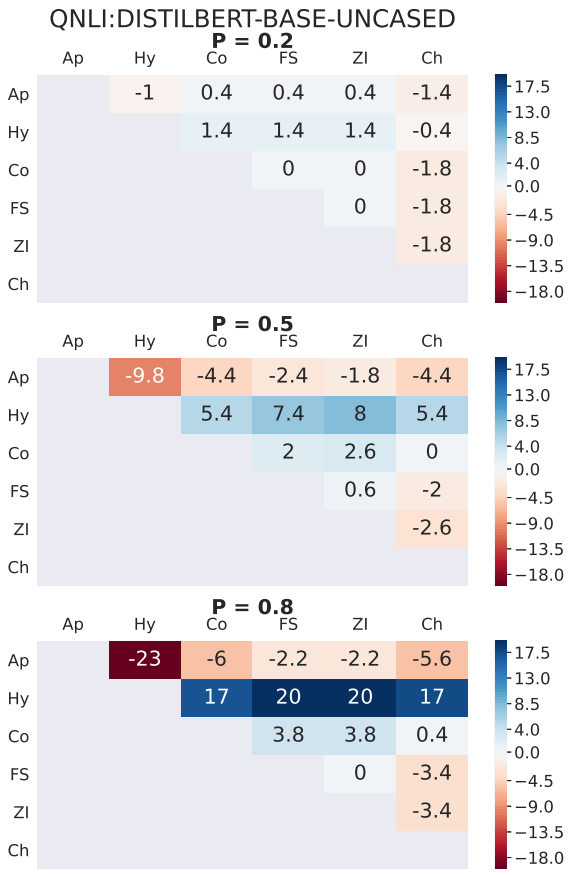

Figure 20

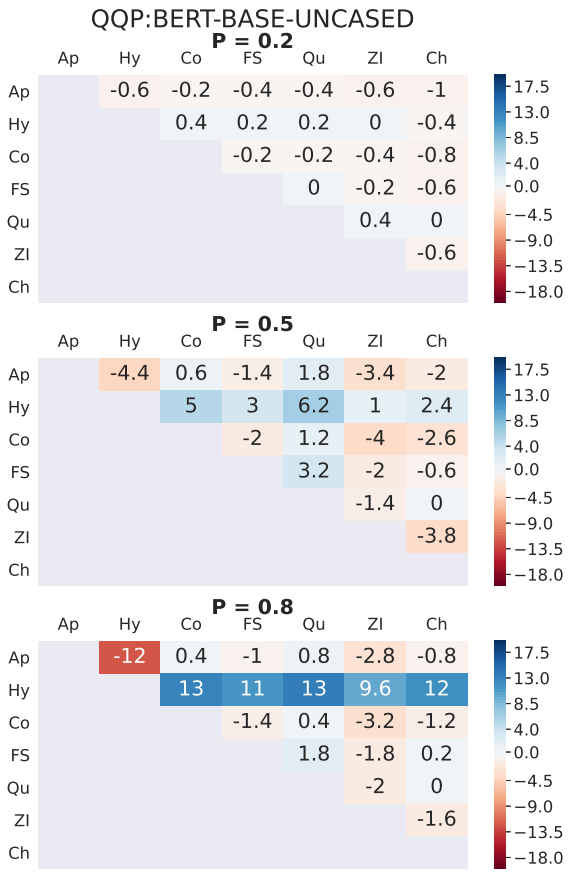

Figure 21

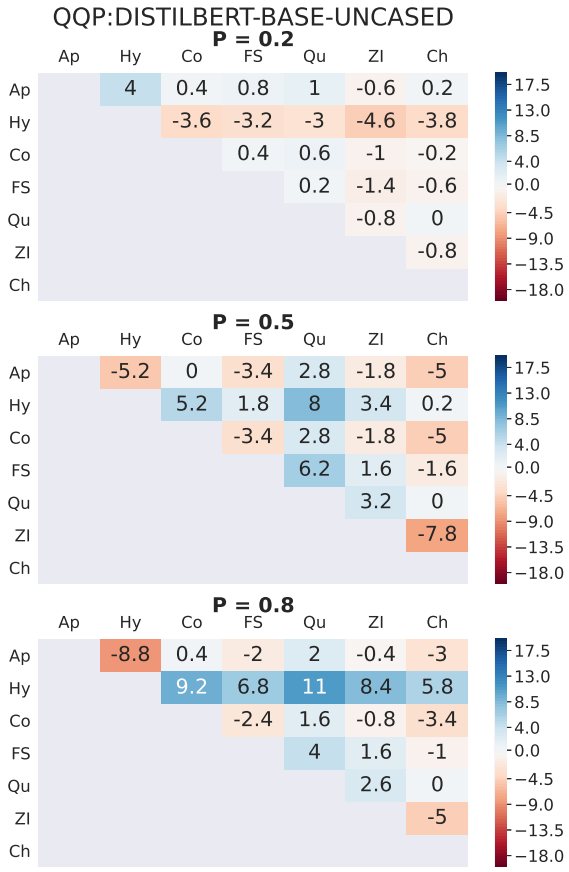

Figure 22

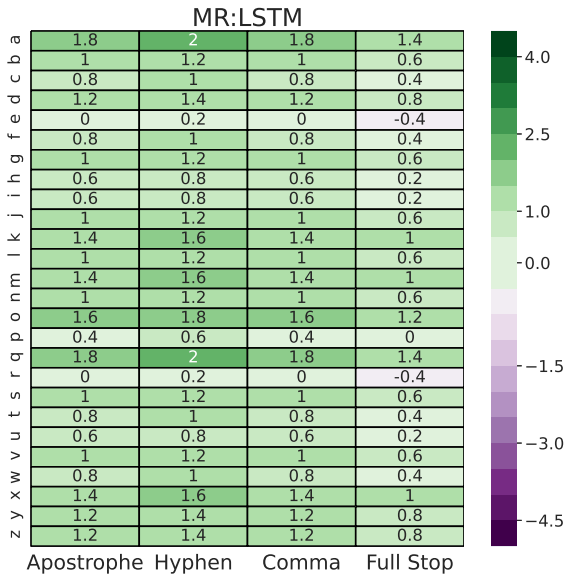

Figure 23

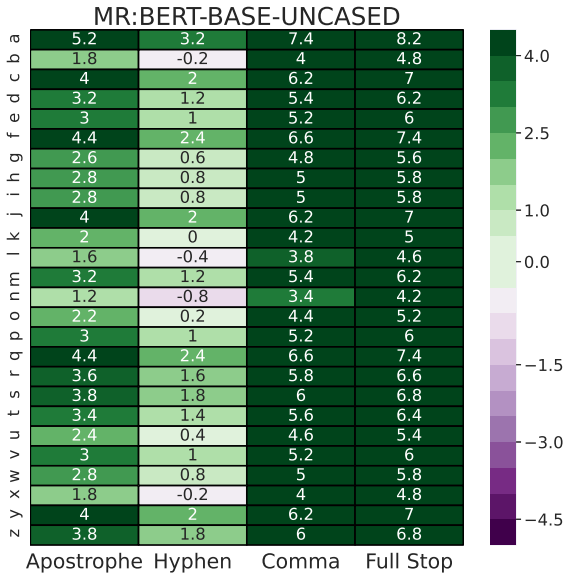

Figure 24

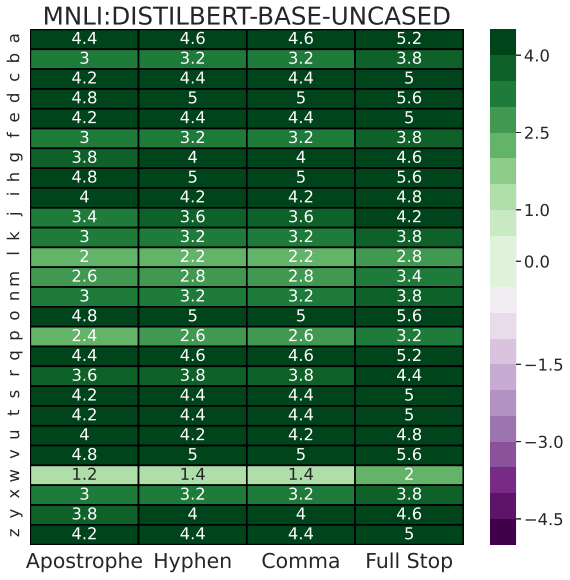

Figure 25

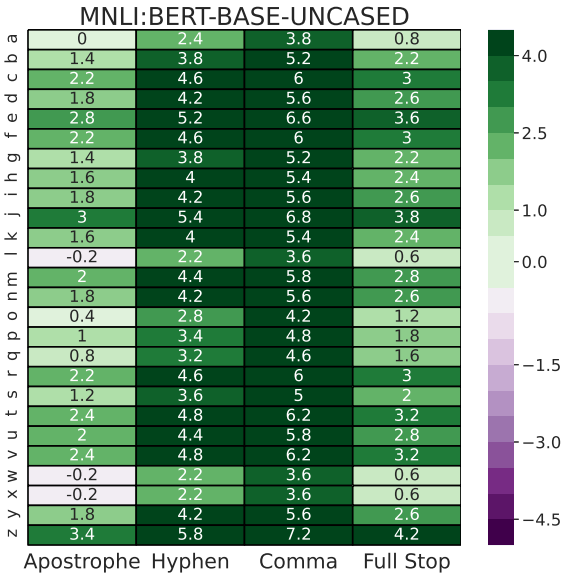

Figure 26

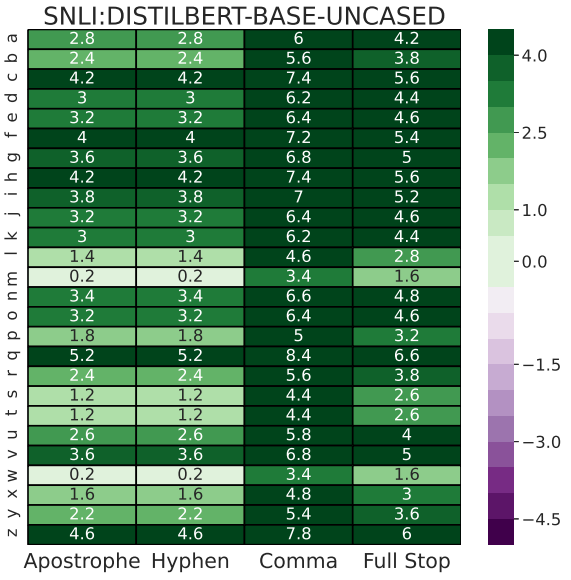

Figure 27

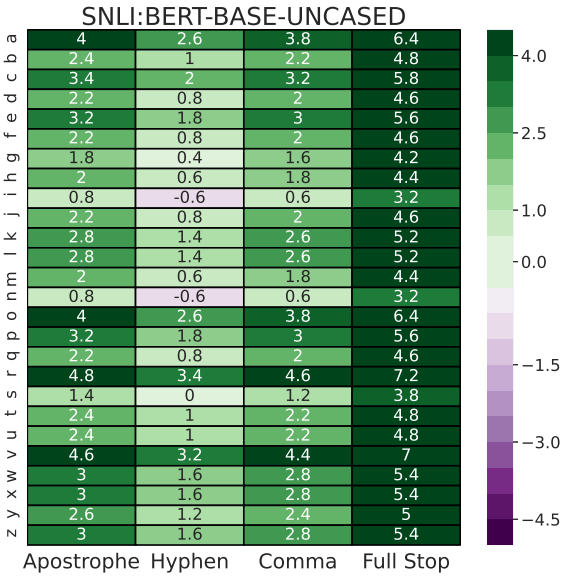

Figure 28

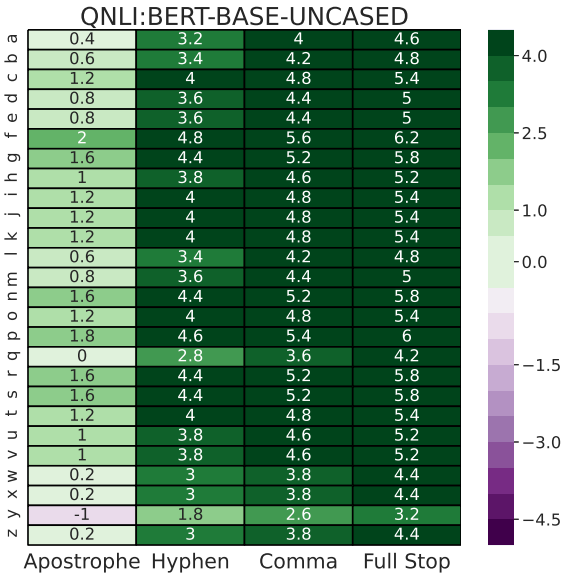

Figure 30

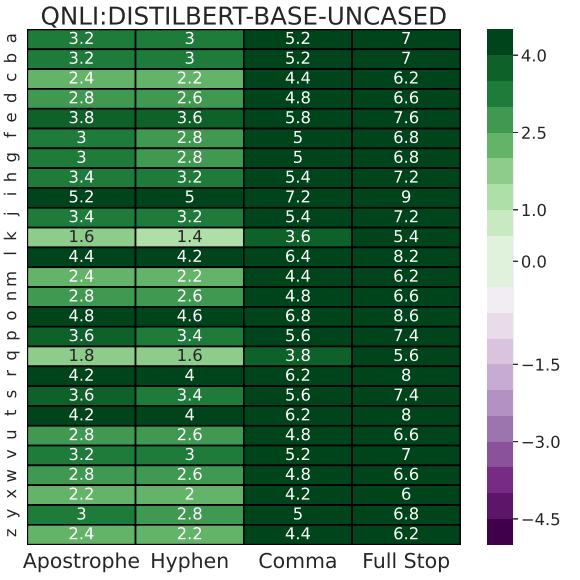

Figure 29

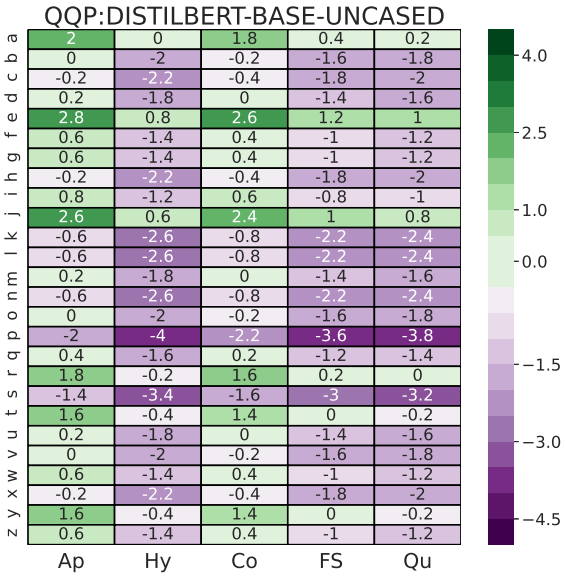

Figure 31

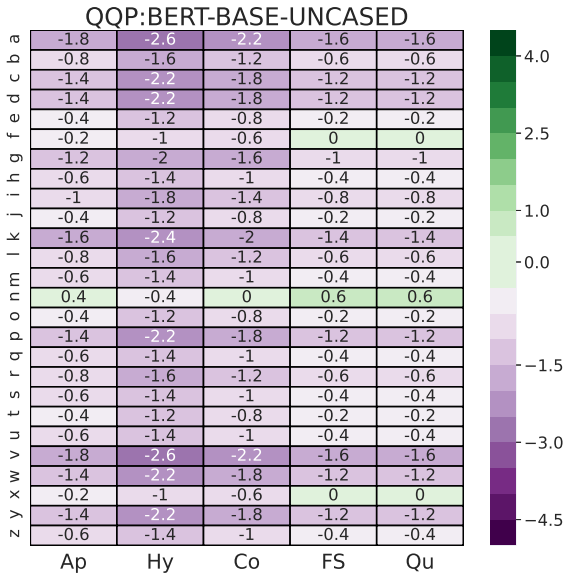
QQP:BERT-BASE-UNCASED
a -1.8 -2.6 -2.2 -1.6 -1.6
b -0.8 -1.6 -1.2 -0.6 -0.6
c -1.4 -2.2 -1.8 -1.2 -1.2
d -1.4 -2.2 -1.8 -1.2 -1.2
e -0.4 -1.2 -0.8 -0.2 -0.2
f -0.2 -1 -0.6 0 0
g -1.2 -2 -1.6 -1 -1
h -0.6 -1.4 -1 -0.4 -0.4
i -1 -1.8 -1.4 -0.8 -0.8
j -0.4 -1.2 -0.8 -0.2 -0.2
k -1.6 -2.4 -2 -1.4 -1.4
l -0.8 -1.6 -1.2 -0.6 -0.6
m -0.6 -1.4 -1 -0.4 -0.4
n 0.4 -0.4 0 0.6 0.6
o -0.4 -1.2 -0.8 -0.2 -0.2
p -1.4 -2.2 -1.8 -1.2 -1.2
q -0.6 -1.4 -1 -0.4 -0.4
r -0.8 -1.6 -1.2 -0.6 -0.6
s -0.6 -1.4 -1 -0.4 -0.4
t -0.4 -1.2 -0.8 -0.2 -0.2
u -0.6 -1.4 -1 -0.4 -0.4
v -1.8 -2.6 -2.2 -1.6 -1.6
w -1.4 -2.2 -1.8 -1.2 -1.2
x -0.2 -1 -0.6 0 0
y -1.4 -2.2 -1.8 -1.2 -1.2
z -0.6 -1.4 -1 -0.4 -0.4
Ap Hy Co FS Qu
4.0
2.5
1.0
0.0
−1.5
−3.0
−4.5

Figure 32

Dataset	Model	Method	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Average Time Taken [s]	Avg Number Queries	Drop [%]
MR	CNN (76.6)	Zeroe	75	11.33	0.88	0.4007	0	1.6
		DWB	66.4	7.41	0.88	0.8542	26.23	10.2
		ZIP -	69	15	1	0.2901	0	7.6
		ZIP '	68.8	25.94	1	0.5539	0	7.8
	LSTM (77)	Zeroe	75.2	8.49	0.92	0.4879	0	1.8
		DWB	66	7.5	0.88	1.0247	26.32	11
		ZIP -	66.8	14.02	1	0.3308	0	10.2
		ZIP '	65.4	24.26	1	0.5739	0	11.6
	BERT (83.8)	Zeroe	82.2	7.43	0.89	0.3426	0	1.6
		DWB	77.8	8.4	0.89	1.0962	26.95	6
		ZIP -	78	13.46	1	0.4014	0	5.8
		ZIP '	70	25.12	1	0.5532	0	13.8
	RoBERTa (88)	Zeroe	87.6	12.27	0.95	0.2185	0	0.4
		DWB	80.8	8.64	0.87	0.9447	26.73	7.2
		ZIP -	80.6	14.26	1	0.3216	0	7.4
		ZIP '	78.2	24.09	1	0.4729	0	9.8
	XLNet (87)	Zeroe	85.6	6.64	0.95	0.7718	0	1.4
		DWB	78.2	7.77	0.88	1.9823	26.89	8.8
		ZIP -	79.6	13.6	1	0.5011	0	7.4
		ZIP '	76.8	24.36	1	0.6806	0	10.2
MNLI	BERT (82.8)	Zeroe	75.6	5.14	0.93	0.329	0	7.2
		DWB	62.2	5.94	0.91	0.8279	31.89	20.6
		ZIP -	64.6	6.9	1	0.2379	0	18.2
		ZIP '	57	11.2	1	0.3272	0	25.8
	DistilBERT (80.6)	Zeroe	75.4	4.73	0.95	0.3012	0	5.2
		DWB	60.4	6.15	0.9	0.7645	31.82	20.2
		ZIP -	69.6	7.49	1	0.1913	0	11
		ZIP '	57.6	11.99	1	0.3292	0	23
SNLI	BERT (91.2)	Zeroe	86.2	6.32	0.94	0.2505	0	5
		DWB	69.4	6.28	0.89	0.5844	23.83	21.8
		ZIP -	61	6.16	1	0.1865	0	30.2
		ZIP '	65.8	10.33	1	0.2095	0	25.4
	DistilBERT (86.6)	Zeroe	84.2	6.61	0.92	0.2391	0	2.4
		DWB	72	6.13	0.89	0.434	24.25	14.6
		ZIP -	74.4	6.37	1	0.165	0	12.2
		ZIP '	65.2	11.37	1	0.2254	0	21.4
QNLI	BERT (91.2)	Zeroe	88.6	5.94	0.94	1.6294	0	2.6
		DWB	74	6.37	0.92	1.7717	47.69	17.2
		ZIP -	81.2	10.69	1	0.5898	0	10
		ZIP '	73.4	21.69	1	1.0907	0	17.8
	RoBERTa (92)	Zeroe	90.6	3.19	0.96	0.9367	0	1.4
		DWB	80.6	5.99	0.93	1.7655	47.94	11.4
		ZIP -	82	10.59	1	0.7899	0	10
		ZIP '	77	22.08	1	1.0432	0	15
	DistilBERT (86.2)	Zeroe	84.4	7.36	0.94	0.9437	0	1.8
		DWB	73.4	6.23	0.92	1.6188	47.96	12.8
		ZIP -	76	10.14	1	0.5882	0	10.2
		ZIP '	66.2	20.63	1	1.0017	0	20
QQP	BERT (90.4)	Zeroe	87.6	6.56	0.92	0.3038	0	2.8
		DWB	79.6	7.48	0.9	0.6998	25.79	10.8
		ZIP -	74.4	8.79	1	0.2272	0	16
		ZIP '	68.4	14.55	1	0.3317	0	22
	DistilBERT (90.8)	Zeroe	87.8	7.26	0.93	0.3298	0	3
		DWB	79.4	7.01	0.91	0.6158	25.97	11.4
		ZIP -	72.8	8.33	1	0.1927	0	18
		ZIP '	69.8	14.29	1	0.2581	0	21
	XLNet (91.2)	Zeroe	89.8	7.26	0.92	0.8158	0	1.4
		DWB	79.6	7.44	0.9	1.6554	25.95	11.6
		ZIP -	78	8.77	1	0.3762	0	13.2
		ZIP '	75.8	14.34	1	0.455	0	15.4

Table 10: Results of Zeroe, DWB and ZIP attacks while using the LanguageTool grammar checker

Dataset	Model (Orig Acc)	Method	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Avg Time Taken [s]	Avg Number Queries
MR	CNN (76.6)	DWBP	14.6	16.99	1	0.0513	69.01
		TextFooler	0.4	**11.82**	0.85	0.2144	74.79
		TextFooler/DWBP	**0.2**	13.09	**0.89**	**0.1145**	**69.75**
		SememePSO	2.6	13.73	0.83	0.6824	2711.91
		SememePSO/DWBP	**2**	**10.97**	**0.86**	**0.4516**	**1012.17**
	LSTM (77)	DWBP	19.2	15.13	1	0.066	66.9
		TextFooler	0.8	**11.43**	0.86	0.1943	71.03
		TextFooler/DWBP	**0.4**	12.87	**0.89**	**0.1289**	**67.95**
		SememePSO	2.8	13.17	0.83	0.7235	2342.27
		SememePSO/DWBP	**1.6**	**10.26**	**0.86**	**0.5366**	**923.21**
	BERT (83.8)	DWBP	17.4	18.32	1	0.721	74.7
		TextFooler	9.4	**17.54**	0.82	1.3072	118.5
		TextFooler/DWBP	**7.6**	18.31	**0.89**	**1.122**	**105.35**
		SememePSO	7	16.52	0.81	16.1811	4950.71
		SememePSO/DWBP	**6**	**9.99**	**0.89**	**7.3252**	**988.44**
	RoBERTa (88)	DWBP	14	19.08	1	0.71	72.42
		TextFooler	5.4	**16.21**	0.83	1.1566	106.89
		TextFooler/DWBP	**5.8**	16.92	**0.89**	**0.9861**	**94.63**
		SememePSO	6	17.44	0.8	15.9324	4855.71
		SememePSO/DWBP	**5.8**	**10.96**	**0.88**	**9.4678**	**1225.24**
	XLNet (87)	DWBP	16.2	19.1	1	2.8193	74.35
		TextFooler	7.4	**15.68**	0.83	4.4761	108.5
		TextFooler/DWBP	**5.4**	17.19	**0.88**	**3.9146**	**96.17**
		SememePSO	5.8	16.75	0.81	53.6015	4619.19
		SememePSO/DWBP	**6**	**10.8**	**0.88**	**35.2176**	**1162.83**

Table 11: Results on classification for multi-level DWBP

Dataset	Model (Orig Acc)	Method	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Average Time Taken [s]	Avg Number Queries
MNLI	BERT (82.8)	DWBP	9.6	8.43	1	0.5381	51.26
		TextFooler	12.2	**6.99**	0.9	0.8749	76.18
		TextFooler/DWBP	**4.2**	8.02	**0.96**	**0.706**	**63.68**
		SememePSO	20.2	**5.9**	0.9	**2.0034**	1200.36
		SememePSO/DWBP	**5**	6.16	0.94	2.068	**208.73**
	DistilBERT (80.6)	DWBP	11.4	7.95	1	0.2668	50.55
		TextFooler	12.6	**7.54**	0.9	0.516	77.88
		TextFooler/DWBP	**5.4**	7.95	**0.96**	**0.3899**	**64.85**
		SememePSO	21.6	**6**	0.89	**1.0294**	1146.93
		SememePSO/DWBP	**6.2**	6.44	**0.94**	1.0934	**220.54**
SNLI	BERT (91.2)	DWBP	7.2	7.99	1	0.4037	38.57
		TextFooler	14	**7.46**	0.9	0.6992	64.2
		TextFooler/DWBP	**3.2**	7.72	**0.97**	**0.523**	**47.96**
		SememePSO	16.6	6.9	0.88	2.1876	764.64
		SememePSO/DWBP	**2.8**	**6.63**	**0.93**	**1.3637**	**139.42**
	DistilBERT (86.6)	DWBP	6.6	8.36	1	0.2101	38.77
		TextFooler	10	**7.75**	0.9	0.4072	64.33
		TextFooler/DWBP	**1.6**	7.79	**0.96**	**0.2848**	**48.09**
		SememePSO	14.4	**6.66**	0.88	1.1608	689.18
		SememePSO/DWBP	**2**	6.68	**0.93**	**0.7641**	**151.06**

Table 12: Results on entailment for multi-level DWBP

Dataset	Model	Method	After Attack Acc [%]	Perturbed Words [%]	Semantic Sim	Average Time Taken [s]	Avg Number Queries
QNLI	BERT (91.2)	DWBP	24.4	10.01	1	0.9616	114.79
		TextFooler	22.6	**9.46**	0.9	1.6601	168.88
		TextFooler/DWBP	**18.2**	10.32	**0.95**	**1.413**	**156.15**
		SememePSO	37.4	11.13	0.88	44.0359	12838.6
		SememePSO/DWBP	**27.2**	**5.79**	**0.96**	**18.2188**	**2093.68**
	RoBERTa (92)	DWBP	26.8	11.9	1	1.0311	120.65
		TextFooler	26.8	**9.94**	0.9	1.6683	174.98
		TextFooler/DWBP	**18.8**	11.45	**0.94**	**1.4461**	**159.57**
		SememePSO	41	11.32	0.87	40.4838	14041.7
		SememePSO/DWBP	**32.4**	**6.45**	**0.95**	**28.7531**	**2325.13**
	DistilBERT (86.2)	DWBP	23.8	9.28	1	0.4786	110.45
		TextFooler	22	**10.04**	0.9	0.9577	168.64
		TextFooler/DWBP	**13.8**	10.66	**0.95**	**0.7953**	**145.54**
		SememePSO	37.2	11.28	0.88	18.1074	13254.8
		SememePSO/DWBP	**28.4**	**6.26**	**0.96**	**15.2255**	**2207.65**
QQP	BERT (90.4)	DWBP	45.2	8.31	1	0.388	60.53
		TextFooler	42.2	8.44	0.9	0.7218	116.5
		TextFooler/DWBP	**41.2**	**8.43**	**0.97**	**0.6001**	**102.06**
		SememePSO	50.4	7.94	0.88	**2.2428**	10858
		SememePSO/DWBP	**42.8**	**7.36**	**0.96**	3.8072	**398.73**
	DistilBERT (90.8)	DWBP	45.4	8.62	1	0.2059	60.1
		TextFooler	38.6	**9.48**	0.9	0.4686	114.73
		TextFooler/DWBP	**37.6**	9.9	**0.95**	**0.3919**	**100.76**
		SememePSO	50	8.59	0.88	2.925	11194.4
		SememePSO/DWBP	**40.8**	**7.84**	**0.95**	**1.9978**	**397.52**
	XLNet (91.2)	DWBP	44.8	9.88	1	1.7345	61.85
		TextFooler	38.8	**9.6**	0.89	3.1888	115.37
		TextFooler/DWBP	**37.2**	9.71	**0.94**	**2.6764**	**100.81**
		SememePSO	49.4	8.3	0.88	10.0921	10057.7
		SememePSO/DWBP	**41.4**	**7.91**	**0.93**	**10.092**	**400.73**

Table 13: Results on question answering tasks for multi-level DWBP

P	ZIP Ap		ZIP Hy		ZIP Co	
	$A_{aft-atk}$	S	$A_{aft-atk}$	S	$A_{aft-atk}$	S
0.2	77.8	1.0	79.2	1.0	77.0	1.0
0.5	68.2	1.0	75.2	1.0	68.0	1.0
0.8	47.8	1.0	75.0	1.0	58.0	1.0
P	**ZIP FS**		**ZI Ze**		**ZI Ch**	
	$A_{aft-atk}$	S	$A_{aft-atk}$	S	$A_{aft-atk}$	S
0.2	78.4	0.97	77.6	0.91	78.6	0.83
0.5	66.8	0.94	61.6	0.79	68.2	0.72
0.8	50.0	0.91	44.0	0.69	50.4	0.62

Table 14: Results of black-box insertions on MR/BERT

Dataset	Model	Baseline Orig Acc [%]	Baseline After Attack Acc [%]	Robust Orig Acc [%]	Robust After Attack Acc [%]
MR Hy	LSTM	78.2	29.8	77.6	32.2
	BERT	84.2	29.4	84.4	36.8
MR Ap	LSTM	78.2	19.8	78.8	21.4
	BERT	84.2	17.0	84.0	23.4

Table 15: Adversarial training

Dataset	Punctuation	Counts	Percentage
MR	**Total Punctuation Count**		
	.	2596	1.37E+00%
	,	1934	1.02E+00%
	’	1073	5.68E-01%
	-	1007	5.33E-01%
	"	146	7.72E-02%
	[	58	3.07E-02%
	]	58	3.07E-02%
	Internal Punctuation Count		
	’	922	5.02E-01%
	-	718	3.91E-01%
	/	17	9.26E-03%
	]	4	2.18E-03%
	[	2	1.09E-03%
MNLI	**Total Punctuation Count**		
	.	3499	1.22E+00%
	,	2041	7.13E-01%
	’	1460	5.10E-01%
	-	527	1.84E-01%
	)	156	5.45E-02%
	(	150	5.24E-02%
	?	125	4.37E-02%
	Internal Punctuation Count		
	’	1347	4.81E-01%
	-	496	1.77E-01%
	.	68	2.43E-02%
	,	49	1.75E-02%
	?	15	5.36E-03%
SNLI	**Total Punctuation Count**		
	.	3523	1.99E+00%
	,	598	3.38E-01%
	-	113	6.39E-02%
	’	54	3.06E-02%
	"	28	1.58E-02%
	&	3	1.70E-03%
	/	1	5.66E-04%
	Internal Punctuation Count		
	-	113	6.55E-02%
	’	50	2.90E-02%
	.	5	2.90E-03%
	/	1	5.79E-04%
	,	1	5.79E-04%
QNLI	**Total Punctuation Count**		
	,	3755	9.98E-01%
	.	2328	6.18E-01%
	?	1983	5.27E-01%
	-	734	1.95E-01%
	’	715	1.90E-01%
	"	672	1.79E-01%
	(	562	1.49E-01%
	Internal Punctuation Count		
	-	708	1.93E-01%
	’	611	1.67E-01%
	.	202	5.52E-02%
	,	155	4.23E-02%
	–	55	1.50E-02%
QQP	**Total Punctuation Count**		
	?	4220	2.10E+00%
	,	521	2.60E-01%
	"	470	2.34E-01%
	’	460	2.29E-01%
	.	349	1.74E-01%
	-	162	8.08E-02%
	(	138	6.88E-02%
	Internal Punctuation Count		
	’	397	2.04E-01%
	-	146	7.50E-02%
	.	93	4.78E-02%
	/	89	4.57E-02%
	(	30	1.54E-02%

Table 16: Frequency of total punctuation in samples and frequency of punctuation only found within words

Dataset	Model	Method	New Orig Acc [%]	Drop [%]
MR	CNN (76.6)	All	72.2	**4.4**
		Internal	74.4	**2.2**
		Internal With Exception	76.6	0
	LSTM (77)	All	72.8	**4.2**
		Internal	74.2	**2.8**
		Internal With Exception	77	0
	BERT (83.8)	All	81.2	**2.6**
		Internal	82.6	**1.2**
		Internal With Exception	83.8	0
	RoBERTa (88)	All	86.2	**1.8**
		Internal	87.8	**0.2**
		Internal With Exception	88	0
	XLNet (87)	All	84.6	**2.4**
		Internal	86.4	**0.6**
		Internal With Exception	87	0
MNLI	BERT (82.8)	All	80.8	**2**
		Internal	82.4	**0.4**
		Internal With Exception	82.4	**0.4**
	DistilBERT (80.6)	All	78.4	**2.2**
		Internal	80	**0.6**
		Internal With Exception	80.4	**0.2**
SNLI	BERT (91.2)	All	90.8	**0.4**
		Internal	91.2	0
		Internal With Exception	91.2	0
	DistilBERT (87)	All	86.6	**0.4**
		Internal	87	0
		Internal With Exception	87	0
QNLI	BERT (91.2)	All	87.2	**4**
		Internal	90.6	**0.6**
		Internal With Exception	90.8	**0.4**
	RoBERTa (92)	All	91.2	**0.8**
		Internal	92	0
		Internal With Exception	92	0
	DistilBERT (86.2)	All	84.2	**2**
		Internal	85.8	**0.4**
		Internal With Exception	85.8	**0.4**
QQP	BERT (90.4)	All	88.6	**1.8**
		Internal	90	**0.4**
		Internal With Exception	90	**0.4**
	DistilBERT (90.8)	All	88.6	**2.2**
		Internal	90.6	**0.2**
		Internal With Exception	90.8	0
	XLNet (91.2)	All	89.8	**1.4**
		Internal	91.2	0
		Internal With Exception	91.2	0

Table 17: Results when punctuation is removed

Dataset	Model	Baseline Finetune with punctuation Eval Acc [%]	Finetune with no punctuation Eval Acc [%]	Drop [%]
MR	LSTM	79.8±0.5	78.9±0.4	**0.9**
	BERT	85.3±0.8	84.7±0.6	**0.6**
MNLI	BERT	84.9	83.5	**1.4**
SNLI	BERT	89.9	88.6	**1.3**

Table 18: Finetuning on no punctuation

Dataset	Model (Orig Acc)	Method	After Attack Acc [%]	Average Time Taken [s]	Drop [%]
MR	CNN (76.6)	DWBP .	15.4	0.0412	**61.2**
		DWBP ,	14.6	0.0407	**62**
		DWBP "	27.2	0.0391	**49.4**
	LSTM (77)	DWBP .	19.4	0.0574	**57.6**
		DWBP ,	19.2	0.0564	**57.8**
		DWBP "	26.6	0.0544	**50.4**
	BERT (83.8)	DWBP .	18.4	0.4748	**65.4**
		DWBP ,	18.4	0.462	**65.4**
		DWBP "	29.4	0.4428	**54.4**
	RoBERTa (88)	DWBP .	19.4	0.499	**68.6**
		DWBP ,	18.6	0.4812	**69.4**
		DWBP "	34.6	0.4459	**53.4**
	XLNet (87)	DWBP .	17.8	1.8583	**69.2**
		DWBP ,	18	1.855	**69**
		DWBP "	34	1.7026	**53**
MNLI	BERT (82.8)	DWBP .	14	0.4317	**68.8**
		DWBP ,	12.6	0.4317	**70.2**
		DWBP)	11.6	0.4423	**71.2**
	DistilBERT (80.6)	DWBP .	12.8	0.2232	**67.8**
		DWBP ,	13.2	0.2195	**67.4**
		DWBP)	10.4	0.2231	**70.2**
SNLI	BERT (91.2)	DWBP .	10	0.323	**81.2**
		DWBP ,	10.6	0.3258	**80.6**
		DWBP "	17	0.3175	**74.2**
	DistilBERT (86.6)	DWBP .	9.6	0.1677	**77**
		DWBP ,	4	0.175	**82.6**
		DWBP "	16.8	0.168	**69.8**
QNLI	BERT (91.2)	DWBP ,	25	0.6877	**66.2**
		DWBP .	26.8	0.6731	**64.4**
		DWBP ?	25	0.7068	**66.2**
	RoBERTa (92)	DWBP ,	28.6	0.756	**63.4**
		DWBP .	32.2	0.7564	**59.8**
		DWBP ?	31.4	0.7678	**60.6**
	DistilBERT (86.2)	DWBP ,	19	0.377	**67.2**
		DWBP .	19.2	0.3675	**67**
		DWBP ?	23.2	0.3559	**63**
QQP	BERT (90.4)	DWBP ?	46.2	0.311	**44.2**
		DWBP ,	48.6	0.3075	**41.8**
		DWBP "	49.4	0.305	**41**
	DistilBERT (90.8)	DWBP ?	43.4	0.169	**47.4**
		DWBP ,	46.2	0.1698	**44.6**
		DWBP "	50.4	0.1617	**40.4**
	XLNet (91.2)	DWBP ?	47.4	1.3345	**43.8**
		DWBP ,	47.6	1.3518	**43.6**
		DWBP "	53.8	1.3201	**37.4**

Table 19: Results when only one punctuation symbol type is used in the attack

MR (Negative)	A dark comedy that goes for sick and demented humor simply to do so . the movie is without intent .
TextFooler (Positive)	A dark comedy that goes for **psychopathic** and **coot** humor **honestly** to do so . the **film** is without **object** .
DWBP (Positive)	A dark comedy that goes for sick and demented humor simply to do so . the movie is **withou't** intent .
MNLI (Entailment)	Premise: Sit down, will you?" Tuppence sat down on the chair facing him. Hypothesis: He asked Tuppence to sit on a red chair.
TextFooler (Neutral)	He asked Tuppence to **assisi** on a **flushed** chair.
DWBP (Neutral)	He asked Tuppence to sit on a **r'ed c'hair.**

Table 20: Qualitative examples of DWBP vs TextFooler. **Bold** words represent a perturbed word