Grammatical Error Correction Systems for Automated Assessment: Are They Susceptible to Universal Adversarial Attacks?

Vyas Raina
Cambridge University
vr313@cam.ac.uk

Edie Lu
Cambridge University
ytl28@cam.ac.uk

Mark Gales
Cambridge University
mjfg@cam.ac.uk

Abstract

Grammatical error correction (GEC) systems are a useful tool for assessing a learner's writing ability. These systems allow the grammatical proficiency of a candidate's text to be assessed without requiring an examiner or teacher to read the text. A simple summary of a candidate's ability can be measured by the total number of edits between the input text and the GEC system output: the fewer the edits the better the candidate. With advances in deep learning, GEC systems have become increasingly powerful and accurate. However, deep learning systems are susceptible to adversarial attacks, in which a small change at the input can cause large, undesired changes at the output. In the context of GEC for automated assessment, the aim of an attack can be to deceive the system into not correcting (concealing) grammatical errors to create the perception of higher language ability. An interesting aspect of adversarial attacks in this scenario is that the attack needs to be simple as it must be applied by, for example, a learner of English. The form of realistic attack examined in this work is appending the same phrase to each input sentence: a concatenative universal attack. The candidate only needs to learn a single attack phrase. State-of-the-art GEC systems are found to be susceptible to this form of simple attack, which transfers to different test sets as well as system architectures [1].

1 Introduction

Grammatical Error Correction (GEC) systems can form a part of automated language fluency assessment: the number of edits from a candidate's input sentence to a GEC system's grammatically corrected output sentence is indicative of a candidate's language ability, where fewer edits suggest better fluency. Early GEC systems were designed using hand-crafted rules (Naber, 2003), but since, data driven approaches, such as Statistical Machine Translation (Yuan and Felice, 2013), emerged. With encoder-decoder architectures dominating in Neural Machine Translation, Yuan and Briscoe (2016) used Recurrent Neural Networks (Cho et al., 2014) to improve GEC performance. Now state of the art GEC systems are based on the Transformer (Vaswani et al., 2017) architecture (Kaneko et al., 2020; Chen et al., 2020; Malmi et al., 2019; Awasthi et al., 2019; Omelianchuk et al., 2020b; Kiyono et al., 2019; Lichtarge et al., 2020; Stahlberg and Kumar, 2020).

Despite the success of Transformer-based deep learning systems, there is a shortcoming: Szegedy et al. (2014) discovered that neural networks are susceptible to adversarial attacks, where a small change at the input can yield large, undesired changes at the output of the model. In the GEC setting, a candidate may seek to make a change to their input sentence, such that the system makes no corrections, resulting in zero edits between the source and prediction sequences, which falsely indicates perfect language fluency. Given the high-stakes of an assessment setting, it is particularly concerning if a candidate can engage in such malpractice. Hence, this work explores the susceptibility of GEC systems to adversarial attacks.

GEC systems operate on natural language inputs. In this domain, there are many proposed adversarial attacks (Zhang et al., 2019), but on the whole they are inappropriate for sequence-to-sequence tasks, such as GEC. Ebrahimi et al. (2018); Zou et al. (2019); Zhang et al. (2021); Cheng et al. (2018) introduced methods for adversarial attacks in sequence-to-sequence models. These works require multiple queries of the target system. However, a candidate cannot query a GEC system. To solve this issue, this work uses a universal (Moosavi-Dezfooli et al., 2016) adversarial attack. Here, the same universal attack phrase is appended to the end of all candidates' input sen-

[1] Code is available at: https://github.com/rainavyas/gec-universal-attack

Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 158–171
November 20–23, 2022.

tences, i.e. a new candidate can simply acquire (e.g. through purchase) a fixed universal attack phrase to concatenate to their input and deceive a GEC system used for automatic fluency assessment. This work also considers the transferability of a single attack phrase across different datasets and even architectures. Further analysis is carried out to determine the aspects of GEC systems that cause them to be susceptible to this form of attack.

Despite advances in natural language adversarial attacks, there has been less research on developing defence schemes. Defence strategies can be categorized as *model modification*, where the model or data is altered at training time (e.g. adversarial training (Yoo and Qi, 2021)) or *detection* (Raina and Gales, 2022), where external systems or algorithms are applied to trained models to identify adversarial attacks. Model modification approaches demand re-training of models and so detection approaches are preferred for deployed systems. Note that for attacks on GEC systems, detectors based on grammatical (Sakaguchi et al., 2017) and spelling (Mays et al., 1991; Islam and Inkpen, 2009) errors will fail. In this work, the most popular detection approaches: Frequency Guided Word Substitution (Mozes et al., 2020) (shown to outperform Zhou et al. (2019)); perplexity (Han et al., 2020; Minervini and Riedel, 2018) and model confidence (Aldahdooh et al., 2021); are applied to detecting adversarial attacks on GEC systems.

2 Related Work

In literature there has been limited work examining adversarial attacks for GEC systems. However, some works have explored adversarial robustness. First, Wang and Zheng (2020) perform adversarial training to improve the performance of their GEC system. Their adversarial training scheme augments the training data with adversarial examples, generated through the insertion of common grammatical mistakes in grammatically correct sentences, where the insertions are tuned to exploit weak spots in the GEC system. Further, Tang (2021) also seeks to increase robustness of GEC systems in a post-training setting, through further training on adversarial examples generated from four different NLP adversarial attack schemes. These adversarial attack methods again are designed to fool the sequence-to-sequence GEC system. Finally, Farkas et al. (2021) also augment the training data with adversarial examples, but focus on ensuring the adversarial examples mimic human grammatical errors by introducing noise at both a token level and embedding level.

However, the above schemes are inappropriate for the attack setting in this work. First, the aim of the attack in this work is to perturb grammatically *in*correct sentences to conceal grammatical errors. Second, the existing works consider attacks specific to each input, whereas this work considers the more realistic setup of a universal adversarial attack.

3 Grammatical Error Correction

Grammatical Error Correction (GEC) systems perform a sequence-to-sequence task, where an input word sequence, $x_{1:T}$, containing grammatical errors, is corrected for these errors by the system, with parameters, $\boldsymbol{\theta}$ to predict the grammatically correct output word sequence, $\hat{y}_{1:L}$,

$$\hat{y}_{1:L} = \arg\max_{y_{1:L}} \{p(y_{1:L}|x_{1:T}; \boldsymbol{\theta})\}. \quad (1)$$

To evaluate the performance of a GEC system, it is necessary to identify the edits made by the system and compare to the reference edits. An edit is defined as a modification (insertion, deletion or substitution) required on the input sequence $x_{1:T}$ to make it match the target sequence, $y_{1:L}$. A popular edit extraction tool is ERRANT (Bryant et al., 2017), which uses a linguistically-enhanced alignment algorithm proposed by Felice and Briscoe (2015). Edits between the input sequence, $x_{1:T}$, and hypothesised prediction sequence $\hat{y}_{1:L}$ can be found, $\hat{e}_{1:P}$,

$$\hat{e}_{1:P} = \texttt{edits}(x_{1:T}, \hat{y}_{1:L}). \quad (2)$$

These edits are to be compared to reference edits,

$$\tilde{e}_{1:R} = \texttt{edits}(x_{1:T}, \tilde{y}_{1:L}), \quad (3)$$

where $\tilde{y}_{1:L}$ is the reference output sequence. The precision $=$ TP$/$(TP$+$FP) and recall $=$ TP$/$(TP$+$FN) can now be computed, where TP, FP and FN are the standard definitions of true-positive, false-positive and false-negative. As a single performance score, $\text{F}_{0.5} = 1.25{*}\text{prec}{*}\text{rec}/(0.25{*}\text{prec})+\text{rec}$) is used, giving greater weight to precision over recall, as in GEC systems it is more important to be correct in the hypothesised edits, $\hat{e}_{1:P}$, as opposed to identifying all reference edits, $\tilde{e}_{1:R}$.

In this work GEC systems are considered for automated assessment. Here, the fluency score,

$S_\theta(x_{1:T})$, of a candidate is measured by the count of edits between the input sequence, $x_{1:T}$, and hypothesised prediction sequence $\hat{y}_{1:L}$, i.e.

$$S_\theta(x_{1:T}) = \texttt{count}(\hat{e}_{1:P}) = P, \quad (4)$$

where $S_\theta(x_{1:T}) = 0$ is a perfect fluency score.

Beyond extracting edits and reporting the overall performance of a GEC system, it is useful to categorize the error types. Inspired by Swanson and Yamangil (2012), the ERRANT tool uses a rule-based error type framework. Here edits are classified as either: **M**issing, where a token is present in the target sequence, $y_{1:L}$ but not in the input sequence, $x_{1:T}$; **R**eplaced, where a substitution is made; or **U**nnecessary, representing edits where a token is present in the input sequence, $x_{1:T}$ and not the output target sequence, $y_{1:L}$.

4 GEC Adversarial Attack

A targeted adversarial attack on an input text sequence, $x_{1:T}$ aims to perturb it to generate an adversarial example $x'_{1:T'}$ that ensures the output of a classifier, $\mathcal{F}()$, is t,

$$\mathcal{F}(x'_{1:T'}) = t, \quad \text{s.t. } \mathcal{H}(x_{1:T}, x'_{1:T'}) \leq \epsilon. \quad (5)$$

$\mathcal{H}()$ is some distance metric between the original and adversarial input sequences, ensuring the change is *imperceptible*. It is not simple to define an appropriate function $\mathcal{H}()$ for word sequences. Perturbations can be measured at a character or word level. Alternatively, the perturbation could be measured in the vector embedding space, using for example l_p-norm based (Goodfellow et al., 2015) metrics or cosine similarity (Carrara et al., 2019). However, constraints in the embedding space do not necessarily achieve imperceptibility in the original word sequence space. This work uses a simple variant of a Levenshtein *edit-based* measurement (Li et al., 2018) which counts the number of changes between the original sequence, $x_{1:T}$ and the adversarial sequence $x'_{1:T'}$, where a change is a swap/addition/deletion, and ensures it is smaller than a maximum number of changes, N. For a candidate planning to perturb their input sentence, the simplest attack is concatenation, where a fixed phrase is appended to their input (Wang and Bansal, 2018; Blohm et al., 2018; Raina et al., 2020),

$$x'_{1:T'} = x_{1:T} \oplus \delta_{1:N} = x_1, \ldots, x_T, \delta_1, \ldots, \delta_N$$

where $\delta_{1:N}$ is a N-word adversarial attack phrase.

The aim of the adversarial attack on a GEC system used for automated assessment, $\mathcal{F}() = S_\theta()$ (Equation 4), is to maximally decrease the count of edits between the input sequence and the predicted sequence, i.e. a candidate wants to *conceal* their grammatical errors from the GEC system. A single universal adversarial phrase, $\hat{\delta}_{1:N}$ is to be used for all candidates, i.e. once this universal phrase has been learnt from a set of J candidates, it can be *sold* to other candidates. Hence, the cost function an adversary seeks to optimise is

$$\hat{\delta}_{1:N} = \underset{\delta_{1:N} \in \mathcal{V}^k}{\arg\min} \left\{ \frac{1}{J} \sum_{j=1}^{J} S_\theta(x^{(j)}_{1:T} \oplus \delta_{1:N}) \right\} \quad (6)$$

where $\mathcal{V}^k$ is the set of all k length word sequences that can be constructed from a selected language vocabulary, $\mathcal{V}$.

It is important to consider the interpretation of *imperceptibility* in the automated assessment setting. In many applications, measuring imperceptibility by counting number of added words, N, is inadequate as it can result in incomprehensible phrases that can easily be identified by a human reader. However, in this setting, there is no human reader, which demands the use of automated systems for identifying incomprehensible phrases. Therefore, this work includes experiments to filter for adversarial attack words that do not compromise the integrity of an input sentence, when measured using a perplexity detector (introduced as a detection mechanism in Section 5, Equation 9) based on a state of the art language model. This ensures that an attack phrase remains imperceptible in an automated assessment setting.

This work also investigates variations in the punctuation a candidate can use to concatenate an adversarial phrase to an input sentence. If '*' represents the form of punctuation, then to concatenate an adversarial phrase to the original phrase, we do: original phrase* adversarial phrase.

5 Defence

For deployed systems, defence strategies that require re-training are undesirable. It is easier to use detection processes to identify and flag adversarial examples. This section considers how state of the art detection approaches can be applied to universal concatenation adversarial attacks on GEC assessment systems, described in Section 4.

All detection approaches, $\mathcal{D}()$, use a selected threshold, β to classify an input sequence, $x_{1:T}$

as adversarial or not. When $\mathcal{D}(x_{1:T}) > \beta$, then the input sequence $x_{1:T}$ is flagged as an adversarial example. To examine the performance of the detection process, this work uses precision-recall curves, where precision and recall values are calculated for a sweep over the threshold β. Here, for each value of β, the precision and recall values are calculated (as in Section 3), with adapted definitions for true-positive (number of samples correctly classified as adversarial), false-positive (number of samples incorrectly classified as adversarial) and false-negative (number of samples incorrectly classified as non-adversarial). A single-value summary is again obtained with the $F_{0.5}$ score, giving greater weighting to precision over recall, as it is more important to be correct in accusing candidates of mal-practice than finding all the candidates that cheat. The threshold with the highest $F_{0.5}$ score is selected for the detector $\mathcal{D}()$.

The recently dominating, Frequency Guided Word Substitution (FGWS) (Mozes et al., 2020) algorithm is adapted for attacks on an assessment GEC system. For the FGWS algorithm, we generate a sequence $x^*_{1:T}$ from the original input sequence, $x_{1:T}$ by substituting out low frequency words for higher frequency words. Precisely, a subset of eligible words (for substitution) is found $\mathcal{X}_E = \{x \in x_{1:T} | \phi(x) < \gamma\}$, where $\phi(x)$ gives frequency of word x and $\gamma \in \mathbb{R}_{>0}$ is a frequency threshold. Then, for each eligible word $x \in \mathcal{X}_E$ a set of replacement candidates, $\mathcal{U}(x)$ is found using synonyms. A replacement word x^* is selected as $x^* = \arg\max_{w \in \mathcal{U}(x)} \phi(w)$. Hence, $x^*_{1:T}$ is generated by replacing each word x in $x_{1:T}$ if $\phi(x^*) > \phi(x)$. For the GEC assessment system, $S_\theta()$, defined in Equation 4, the FGWS detection score is,

$$\mathcal{D}_{\text{FGWS}}(x_{1:T}) = \frac{1}{T}\left(S_\theta(x_{1:T}) - S_\theta(x^*_{1:T})\right). \quad (7)$$

Smith and Gal (2018) describe the use of uncertainty for adversarial attack detection, where adversarial samples are thought to result in greater epistemic uncertainty. In this work, negative confidence is selected as a simple measure of uncertainty. It is easiest to measure the confidence using the *grammatically correct* sequence output by the GEC system, $\hat{y}_{1:L}$ (Equation 1). The negative confidence detector score is calculated as,

$$\mathcal{D}_{\text{nc}}(x_{1:T}) = -\frac{1}{L}\log(p(\hat{y}_{1:L}|x_{1:T})). \quad (8)$$

This works also explores the positive confidence detector, $\mathcal{D}_{\text{pc}}(x_{1:T}) = -\mathcal{D}_{\text{nc}}(x_{1:T})$. A final popular NLP detection approach is to consider the *perplexity* (Minervini and Riedel, 2018) of the input sequence. It is expected that adversarial sequences have a greater perplexity than original samples. The perplexity detector, using some language model (LM), can be defined as,

$$\mathcal{D}_{\text{p}}(x_{1:T}) = -\frac{1}{T}\log(p_{\text{LM}}(x_{1:T})). \quad (9)$$

6 Experiments

6.1 Setup

Training of systems in this work uses a range of different popular grammatical error correction corpora. **Cambridge Learner Corpus (CLC)** (OpenCLC, 2019) is made up of written examinations for general and business English of candidates from 86 different mother tongues. Grammatical errors are annotated and this is used to generate reference sentences for GEC training. **Cambridge English Write & Improve (WI)** (Yannakoudakis et al., 2018) is an online web platform that assists non-native English students with their writing. Specifically, students submit letters, stories and essays in response to various prompts, and the WI system provides instant feedback. **LOCNESS** corpus (Granger, 2014) is a collection of 400 essays written by British and American undergraduates.

Evaluation of systems is performed on three different test sets. **First Certificate in English (FCE)** corpus (Yannakoudakis et al., 2011) is a subset of CLC, consisting of 33,673 sentences split into test and training sets of 2,720 and 30,953 sentences respectively. **Building Education Applications 2019 (BEA-19)** (Bryant et al., 2019) offers a test set of 4477 sentences, sourced from essays written by native and non-native English students[2]. **Conference on Computational Natural Language Learning 2014 (CoNLL-14)** (Ng et al., 2014) test set consists of 1312 sentences sourced from 50 essays written by 25 non-native English speakers.

In recent years, Grammatical Error Correction systems have been dominated by large (up to 11B parameters) Transformer based architectures (Rothe et al., 2021; Stahlberg and Kumar, 2021). Using the $F_{0.5}$ metric defined in Section 3, Table 1 compares the performance of two popular Transformer-based architectures: the Gram-

[2]Evaluation: https://competitions.codalab.org/competitions/20228.

former (Damodaran, 2022) (223M parameters), a T5-based (Raffel et al., 2019) sequence to sequence system[3] and Grammarly's Gector (Omelianchuk et al., 2020a), using specifically the Roberta-based architecture (Liu et al., 2019) (123M parameters)[4]. The Gramformer is pre-trained on the WikEd Error Corpus (Grundkiewicz and Junczys-Dowmunt, 2014), and in this work, it is further fine-tuned on the CLC (with FCE-test set removed), WI and LOCNESS datasets. The finetuning uses Adam optimiser with a batch size of 256 and a learning rate of 5e-4 with warm up. Maximum sentence length is set at 64 and the final model parameters are averaged over 5 best checkpoints. As the Gramformer model was initialised from a large pre-trained system, changing seed for the finetuning gave little diversity in the ensemble.

Table 1 shows that the Gramformer and Gector systems have a similar performance on the FCE test set, but the Gector system significantly out performs the Gramformer on the CoNLL-14 and BEA-19 test sets. Nevertheless, to mimic a realistic adversarial attack setting, the more easily available Gramformer system[5] is used as an initial model (adversary can access) for learning universal attacks and the best attacks are then transferred for evaluation on the target Gector system in Section 6.4.

	Model	Precision	Recall	$F_{0.5}$
FCE	Gramformer	51.6	43.7	49.8
	Gector	53.5	39.3	49.9
CoNLL-14	Gramformer	49.3	34.1	45.2
	Gector	62.0	42.6	56.8
BEA-19	Gramformer	35.3	44.6	37.1
	Gector	70.2	61.2	68.2

Table 1: GEC systems $F_{0.5}$ scores.

6.2 Attack Results

Greedy universal concatenation adversarial attacks were performed on the Gramformer system as described in Equation 6. As described in Section 4, different punctuation types were considered for the concatenation of the universal attacks. The impact of each attack phrase is presented for each of the three different GEC test sets in Figure 1, with N being the number of universal adversarial words at the end of each input sentence. The universal attack phrases were learnt on the FCE training split[6].

The metric used to measure the success of the attack is the fraction of samples with zero edits from source to GEC prediction sequence. The *random* attacks shown use a *full-stop* for concatenating randomly sampled words. A *direct* attack is where no punctuation is used to separate the original and the attack phrase. With percent increases between 20% and 50% in the fraction of samples with no edits shows that the GEC system is threatened somewhat by the *direct, colon* and *comma* attacks. However, for the *full-stop* universal adversarial attack sequence, with even a $N = 4$ word attack, the number of samples with zero edits increases by almost 40% for the FCE test set and more than 100% for the CoNLL-14 and BEA test set. It is evident that the GEC system is susceptible to even a simple form of universal attack. The greater susceptibility to the full-stop attack can be explained to some extent by the nature of the data used to fine-tune the Gramformer GEC system. Table 2 shows the frequency count of the different punctuation marks in the training set (CLC, WI and LOCNESS datasets), where the *full-stops* present at the end of sentences are not included [7]. Note that there are a total of ∼3M input samples in the training dataset. The count of *full-stops* is far less than that of *commas*, meaning the GEC system is not as familiar with multi-sentence inputs allowing for greater susceptibility to attacks using the *full-stop*. However, this count-based explanation is inadequate to justify the less successful *colon* concatenation attack. Nevertheless, the lack of susceptibility to colon concatenation can be explained - in the training samples with colons, more than 50% samples have the *colon* followed by a list delimited with semi-colons. This means that the GEC system easily learns this fixed colon usage, which makes it difficult to have a successful *colon*-based universal concatenation attack format. Due to the potency of the *full-stop* concatenation attack, the remainder of the analysis in this section focuses on the *full-stop* attack [8]. Examples of the impact of

[3]Gramformer model: https://github.com/PrithivirajDamodaran/Gramformer

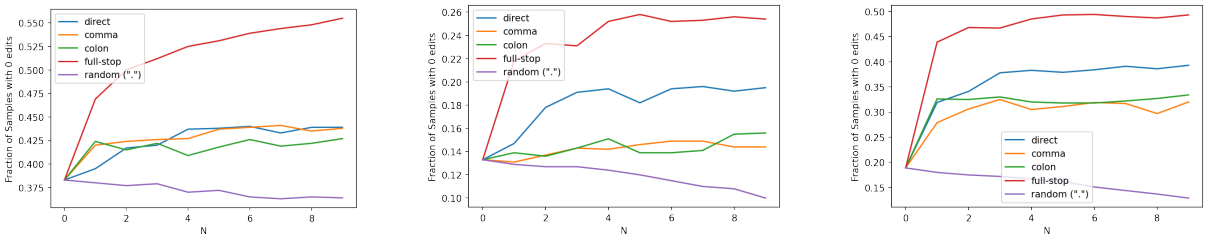
[4]Gector models: https://github.com/grammarly/gector

[5]Stars on Github: Gramformer (1,110); Gector (611).

[6]Note that the **same** universal attack phrase is evaluated on the different datasets.

[7]For the *full-stop* concatenative attack we are interested in the count of the number of instances where there is a multi-sentence input to best represent the format of the attack.

[8]Equivalent analysis (in Appendix B) for the *comma, colon* and *direct* attack formats gave the same trends as the analysis presented for the *full-stop* attack format.



(a) FCE (b) CONLL-14 (c) BEA 2019

Figure 1: Evaluation of Universal Attacks, length N, on GEC system with concatenation punctuation.

the universal attack are given in Table B.1.

Punctuation	Count
Full-stop	214,064
Comma	1,790,282
Colon	97,964

Table 2: Count of punctuation in training set. Excludes punctuation at end of inputs.

Table 3 shows the impact of the $N = 4$ concatenation adversarial attack on the performance of the GEC system on the FCE test set. The adversarial phrase is stripped from the output predicted sequence to discount the introduction of false-positive edits in the adversarial part of the input. As expected the $F_{0.5}$ score worsens due to the drop in the recall, i.e. the GEC systems struggles to find the grammatical errors with the attack phrase concatenated at the end of the sentence - the attack is successful in concealing the errors present in the sentence.

Input	Precision	Recall	$F_{0.5}$
Original	51.6	43.7	49.8
Attacked	51.3	30.7	45.2

Table 3: Gramformer $F_{0.5}$ score.

6.3 Detection Evasion

Although the Gramformer GEC system is susceptible to a universal attack, it can be defended using detection methods. Figure 2 compares the success of detectors from Section 5 when attempting to distinguish adversarial samples from original samples (on FCE test). The threshold for each detector is selected such that it gives the best $F_{0.5}$ score. Results are presented for original FCE test samples with and without the *full-stop* universal adversarial phrase appended to the end of the samples. It is interesting to note that FGWS, although successful

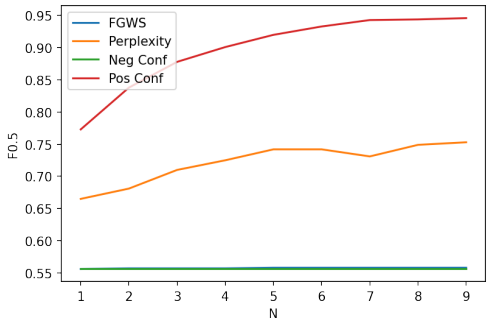
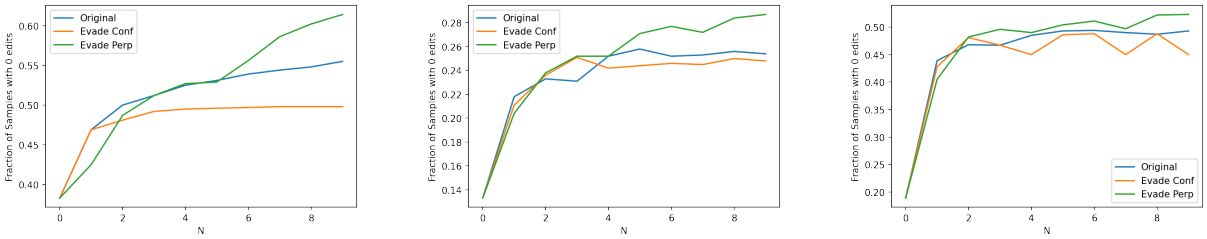


Figure 2: Adversarial attack detection using $F_{0.5}$ score to distinguish between original and adversarial samples.

in other NLP adversarial attack tasks, has little success here. This is perhaps expected as the FGWS vocabulary is now trained with grammatically incorrect sentences containing mis-spellings. Further, the FGWS algorithm is tuned to word substitution attacks, meaning it is less appropriate for the concatenation setting here. The perplexity score is calculated using a pre-trained distilled GPT-2 language model (Radford et al., 2019) applied to the input sequence. Perplexity has some success here in detecting adversarial samples, but the success is limited because many original input sequences are grammatically incorrect and thus naturally have an inflated perplexity score, meaning it is easy for the detector to mistake them for adversarial samples.

Interestingly, negative confidence has no success in detection here, whilst positive confidence dominates as the best detection approach. This is surprising because one would expect adversarial samples to cause systems to be less confident in their predictions, as the system is operating in a less well understood input space. Nevertheless, superior performance of positive confidence is explainable. GEC systems are trained on data where the tokens present in the input are also present in the reference, meaning in most cases there is a strong bias towards simply predicting tokens that are present



(a) FCE (b) CONLL-14 (c) BEA 2019

Figure 3: Evaluation of detector evasion adversarial attacks.

in the input sequence. When an obscure adversarial word is present in the input sequence, the GEC system at prediction time naturally has a much larger probability mass associated with this obscure word, i.e. it is excessively confident in predicting it.

An adversary may have knowledge of the powerful detectors used here and would tailor the adversarial attack to avoid detection. Figure 3 shows the impact of the greedy attack approach modified to evade detection from the confidence detector and the perplexity detector (detector thresholds set to the value corresponding to the $F_{0.5}$ score in Figure 2) [9]. The attack phrases are learnt on the FCE train set and evaluated on the FCE, CoNLL-14 and BEA test sets. It is interesting to note that the confidence detection evading attack phrases are only slightly less effective than the original attack phrases - the fraction of zero edits saturate at around 0.50 as opposed to 0.56 (on FCE test set). However, considering the attack to evade the perplexity detector, the potency of this universal phrase is surprisingly greater than the original greedy attack phrase learnt (for all datasets). This suggests that constraining an attack to more *human* phrases (as measured by perplexity of a powerful GPT-2 language model), allows for even stronger adversarial attacks. These phrases are considered particularly threatening as their similarity to natural language allows for greater imperceptibility to human observers (not just automated detection systems).

6.4 Transfer Attack

The aim of this section is to investigate the impact of transferring an attack learnt for an initial system (Gramformer) to a target system (Gector).

Concatenation universal adversarial attacks on the Gramformer system are found to be most powerful when the adversary greedily generates a phrase that evades a perplexity detector, as demonstrated in Figure 3. Hence, this universal adversarial phrase is simply evaluated on the Gector system. The results in Table 4 show that this transferred universal adversarial phrase has some level of threat: across all test sets, this universal adversarial phrase is able to increase the fraction of samples with no edits by at the least 10%. Table 4 also gives the impact of learning a universal attack phrase (using FCE train dataset and also avoiding a perplexity detector as in Section 5) for the Gector system. Interestingly, the direct attack is only around twice as effective as the transferred attack. This highlights the potency of these forms of adversarial attacks: the same adversarial phrase can transfer to different unseen, GEC systems.

Data	Attack	$N=0$	$N=9$
FCE	Transfer	0.44	0.50
	Direct	0.44	0.55
CoNLL-14	Transfer	0.33	0.38
	Direct	0.33	0.41
BEA-19	Transfer	0.45	0.50
	Direct	0.45	0.54

Table 4: Fraction of samples with zero edits, attack on Gector.

6.5 Analysis

This section carries out a more in-depth analysis to understand the aspects of the GEC systems exploited by adversarial attacks. The analysis presented here is for the concatenative full-stop attack learnt for the Gramformer system.

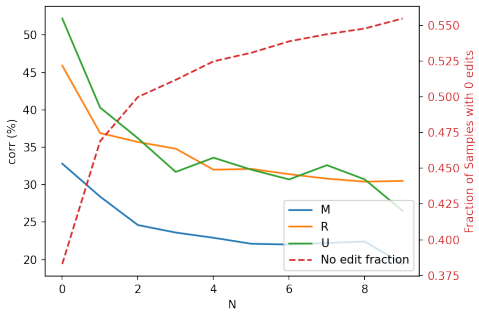
We want to analyse the nature of the attack - precisely which type of edits is the adversarial attack phrase targeting. If for a dataset of J input-reference sentence pairs, there exist a total R reference edits, $\tilde{e}_{1:R}$ (Equation 3) and P hypothesis edits, $\hat{e}_{1:P}$ (Equation 2), then the performance due to the GEC system correctly hypothesising edits

[9] A adversarial word is accepted if the average confidence/perplexity is less than the detector threshold.

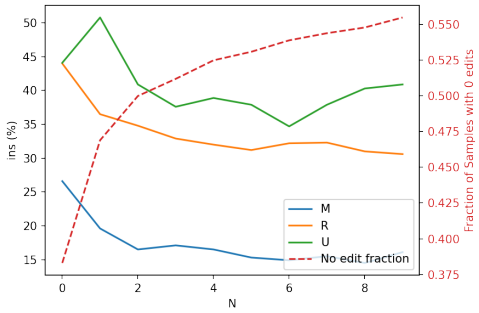
can be measured by the correction rate, cor and the failure measured by the insertion rate, ins,

$$\texttt{cor} = \frac{1}{R}\sum_{p=1}^{P} \mathbb{1}_{\{\tilde{e}_{1:R}\}} \hat{e}_p, \quad \texttt{ins} = \frac{1}{R}\sum_{p=1}^{P} \mathbb{1}_{\{\tilde{e}_{1:R}\}^{\complement}} \hat{e}_p,$$

where $\{\tilde{e}_{1:R}\}^{\complement}$ gives the complement set. Section 3 classifies an edit as **M**issing, **R**eplaced or **U**nnecessary. Figure 4 shows how the correction and insertion rates change (on FCE test) for each of these edits classes separately. Note that there are a total of $R = 919$, $R = 2954$ and $R = 596$ reference edits for **M**issing, **R**eplaced and **U**nnecessary classes respectively.



(a) Correct Edits



(b) Inserted Edits

Figure 4: Edit rates by edit type class.

The edit classes (M, R and U) all undergo a similar drop in correction rate with an increasingly powerful adversarial attack. However, Figure 4b demonstrates that smaller N adversarial attacks struggle to reduce **U**nnecessary inserted edits more than other edit type classes. Only when the reductions from removing **M**issing and **R**eplaced inserted edit types have saturated, does increasing N reduce the **U**nnecessary inserted edit types. The flattening of the performance curve (fraction of samples with zero edits) suggests that this reduction in inserted **U**nnecessary edits has little benefit to the adversarial attack. The apparent robustness of **U**nnecessary inserted edits can perhaps be explained simply. An inserted edit is the creation of an edit, $\hat{e}$, by the GEC system that is not present in the reference edits, $\tilde{e}_{1:R}$. When a GEC system creates specifically **U**nnecessary edits it means a token present in the input sequence is not present in the output prediction sequence. A well trained GEC system will remove the adversarial phrase appended to the input sequence, creating an **U**nnecessary edit, $\hat{e}$, which does not exist in the reference edits, $\tilde{e}_{1:R}$ - it is an inserted edit. Hence, there is an artificial increase in inserted **U**nnecessary edits. Edits in the adversarial phrase only contribute to 4% of total edits on average (analysis presented in Figure A.1), where 91% of the adversarial phrase edits are **U**nnecessary edit types. This gives on average an increase in the inserted **U**nnecessary edit rate by 10% ($0.04 * 0.91 * \texttt{count}(\hat{e}_{1:P})/596$), where 596 is the count of **U**nnecessary reference edits. This increase of 10% explains the shift between the **R**eplaced and **U**nnecessary curves in Figure 4b. Hence, all edit types in an input sequence are susceptible to the simple universal attack.

7 Conclusions

Grammatical Error Correction (GEC) systems can contribute to automated fluency assessment. The count of edits between a candidate's input and the grammatically *correct* output sequence from the GEC system, is a measure of the candidate's ability in the language: fewer the number of edits, the better the candidate. However, this work showed that deep learning based GEC systems are susceptible to adversarial attacks, where a candidate can cheat by adjusting their input sentence such that the predicted sequence from the GEC system does not correct the existing grammatical errors.

To model a realistic adversarial attack setting, this work restricts itself to a blackbox, universal attack approach, where the same adversarial phrase is appended to the end of all candidates' input sequences. This setting is particularly threatening because a candidate can cheat without querying the GEC system even once - the candidate only has to acquire the attack phrase. It is found that the same universal attack phrase can be effective across multiple datasets and more interestingly can be transferred to deceive new, unseen architectures. This demonstrates that all GEC systems have a worrying susceptibility to even the simplest attack forms.

8 Limitations

This work identified methods to adversarially attack state of the art GEC systems. Defence strategies in the form of detection were also considered. However, there has been less focus on adversarial training to improve robustness of systems. Although adversarial training is not an option available to deployed GEC systems, future work in this area would be useful in understanding the increase in robustness from adversarial training to the universal attack form considered in this work.

9 Risks and Ethics

Adversarial attacks, by nature, are of ethical concern in high stakes' environments. The approaches proposed in this work can be used to inspire candidates to engage in mal-practice in an education setting. However, this is of little concern because the development of attacks requires significant know-how of the GEC assessment process, which candidates are unlikely to have.

10 Acknowledgements

This research is supported by Cambridge Assessment, University of Cambridge and ALTA.

References

Ahmed Aldahdooh, Wassim Hamidouche, and Olivier Déforges. 2021. Selective and features based adversarial example detection. *CoRR*, abs/2103.05354.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. *CoRR*, abs/1910.02893.

Matthias Blohm, Glorianna Jagfeld, Ekta Sood, Xiang Yu, and Ngoc Thang Vu. 2018. Comparing attention-based convolutional and recurrent neural networks: Success and limitations in machine reading comprehension. *CoRR*, abs/1808.08744.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, Florence, Italy. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Fabio Carrara, Rudy Becarelli, Roberto Caldelli, Fabrizio Falchi, and Giuseppe Amato. 2019. Adversarial examples detection in features distance spaces. In *Computer Vision – ECCV 2018 Workshops*, pages 313–327, Cham. Springer International Publishing.

Mengyun Chen, Tao Ge, Xingxing Zhang, Furu Wei, and Ming Zhou. 2020. Improving the efficiency of grammatical error correction with erroneous span detection and correction. *CoRR*, abs/2010.03260.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *CoRR*, abs/1803.01128.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Prithiviraj Damodaran. 2022. Prithivirajdamodaran/gramformer: A framework for detecting, highlighting and correcting grammatical errors on natural language text. created by prithiviraj damodaran. open to pull requests and other forms of collaboration.

Javid Ebrahimi, Daniel Lowd, and Dejing Dou. 2018. On adversarial examples for character-level neural machine translation. *CoRR*, abs/1806.09030.

Igor Farkas, Paolo Masulli, Sebastian Otte, Stefan Wermter, Kai Dang, and Jiaying Xie. 2021. *Leveraging Adversarial Training to Facilitate Grammatical Error Correction*, chapter 1. Springer International Publishing AG.

Mariano Felice and Ted Briscoe. 2015. Towards a standard evaluation method for grammatical error detection and correction. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 578–587, Denver, Colorado. Association for Computational Linguistics.

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

Sylviane Granger. 2014. The computer learner corpus: A versatile new source of data for sla research. *Learner English on Computer*, page 3–18.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *Advances in Natural Language Processing – Lecture Notes in Computer Science*, volume 8686, pages 478–490. Springer.

Wenjuan Han, Liwen Zhang, Yong Jiang, and Kewei Tu. 2020. Adversarial attack and defense of structured prediction models.

Aminul Islam and Diana Inkpen. 2009. Real-word spelling correction using google web it 3-grams. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, page 1241–1249, USA. Association for Computational Linguistics.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. *CoRR*, abs/2005.00987.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. *CoRR*, abs/1909.00502.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2018. Textbugger: Generating adversarial text against real-world applications. *CoRR*, abs/1812.05271.

Jared Lichtarge, Chris Alberti, and Shankar Kumar. 2020. Data weighted training strategies for grammatical error correction. *CoRR*, abs/2008.02976.

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. *CoRR*, abs/1907.11692.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5054–5065, Hong Kong, China. Association for Computational Linguistics.

Eric Mays, Fred J. Damerau, and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing & Management*, 27(5):517–522.

Pasquale Minervini and Sebastian Riedel. 2018. Adversarially regularising neural NLI models to integrate logical background knowledge. *CoRR*, abs/1808.08609.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2016. Universal adversarial perturbations. *CoRR*, abs/1610.08401.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis D. Griffin. 2020. Frequency-guided word substitutions for detecting textual adversarial examples. *CoRR*, abs/2004.05887.

Daniel Naber. 2003. A rule-based style and grammar checker.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. 2014. The CoNLL-2014 shared task on grammatical error correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020a. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. Chernodub, and Oleksandr Skurzhanskyi. 2020b. Gector - grammatical error correction: Tag, not rewrite. *CoRR*, abs/2005.12592.

OpenCLC. 2019. Open cambridge learner english corpus.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAIblog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Vyas Raina and Mark Gales. 2022. Residue-based natural language adversarial attack detection.

Vyas Raina, Mark J.F. Gales, and Kate M. Knill. 2020. Universal Adversarial Attacks on Spoken Language Assessment Systems. In *Proc. Interspeech 2020*, pages 3855–3859.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. *CoRR*, abs/1707.00299.

Lewis Smith and Yarin Gal. 2018. Understanding measures of uncertainty for adversarial example detection.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Ben Swanson and Elif Yamangil. 2012. Correction detection and error type selection as an ESL educational aid. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 357–361, Montréal, Canada. Association for Computational Linguistics.

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

Zecheng Tang. 2021. Robust and effective grammatical error correction with simple cycle self-augmenting.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Lihao Wang and Xiaoqing Zheng. 2020. Improving grammatical error correction models with purpose-built adversarial examples. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2858–2869, Online. Association for Computational Linguistics.

Yicheng Wang and Mohit Bansal. 2018. Robust machine comprehension models via adversarial training. *CoRR*, abs/1804.06473.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Helen Yannakoudakis, Øistein E Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. 2018. Developing an automated writing placement system for esl learners. *Applied Measurement in Education*, 31(3):251–267.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of nlp models.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Zheng Yuan and Mariano Felice. 2013. Constrained grammatical error correction using statistical machine translation. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 52–61, Sofia, Bulgaria. Association for Computational Linguistics.

Wei Emma Zhang, Quan Z. Sheng, and Ahoud Alhazmi. 2019. Generating textual adversarial examples for deep learning models: A survey. *CoRR*, abs/1901.06796.

Xinze Zhang, Junzhe Zhang, Zhenhua Chen, and Kun He. 2021. Crafting adversarial examples for neural machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. *CoRR*, abs/1909.03084.

Wei Zou, Shujian Huang, Jun Xie, Xinyu Dai, and Jiajun Chen. 2019. A reinforced generation of adversarial samples for neural machine translation. *CoRR*, abs/1911.03677.

A More Analysis

It is perhaps surprising to observe a reduction in the number of edits made by a GEC system, when a potentially grammatically incorrect adversarial phrase is concatenated at the end of each input. To investigate this, Figure A.1 presents the split in the number of edits; i.e. on average how many edits are found in the original part of the input and the concatenated adversarial part of an input separately (using the full-stop concatenation attack on the Gramformer evaluated on the FCE test set). The adversarial phrase appears to have a length independent small contribution to the total number of edits, whilst the larger reduction in the number of original part edits allows for an overall decrease in the total edits.

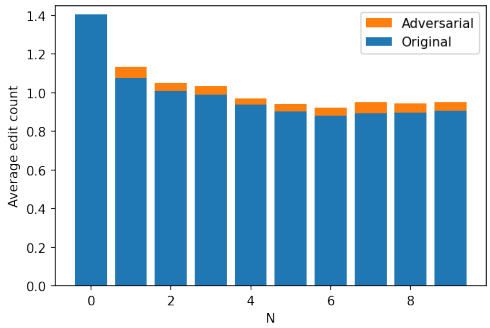
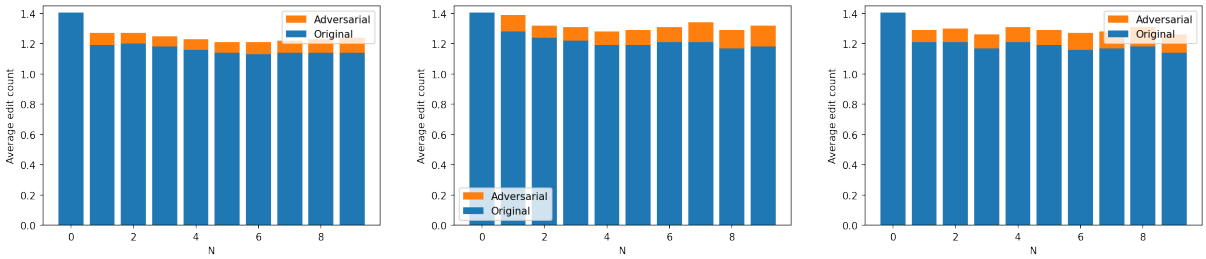


Figure A.1: Average edit count split between original and adversarial parts of inputs.

B Examples and More Results

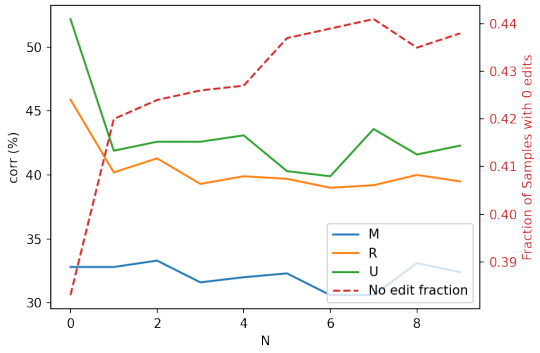
Input	I have received your letter, which was about that you would like me to inform you. *ink.*
Pred	I have received your letter, which was about what you would like me to inform you about.
Adv Pred	I have received your letter, which was about what you would like me to inform you. ink.
Input	When I was a child I always wanted to get the lyrics of the songs that I liked, but it was nearly impossible to have them. *ink.*
Pred	When I was a child I always wanted to get the lyrics of the songs that I liked, but it was nearly impossible to get them.
Adv Pred	When I was a child I always wanted to get the lyrics of the songs that I liked, but it was nearly impossible to have them. ink.
Input	As you know, it is not suitable to wear a jean or something. *ink.*
Pred	As you know, it is not fashionable to wear jeans or something.
Adv Pred	As you know, it is not suitable to wear jeans or something. ink.

Table B.1: Examples of GEC prediction sequences with and without universal adversarial sequence concatenated to input.

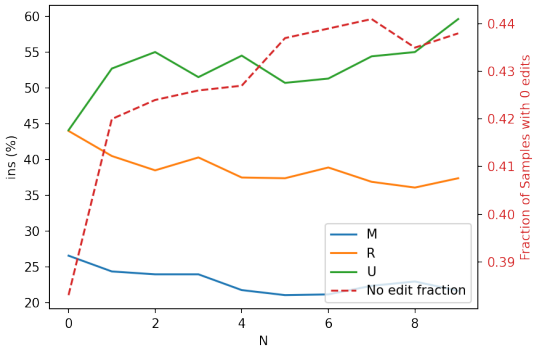


(a) Comma (b) Direct (c) Colon

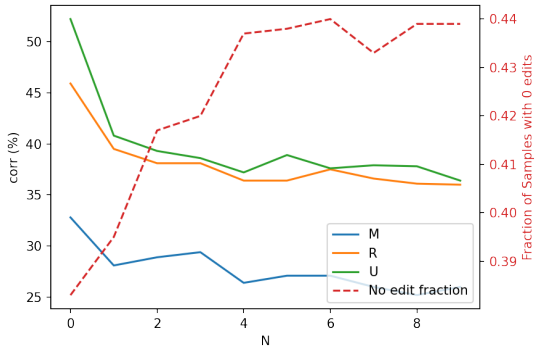
Figure B.1: Average edit count split between original and adversarial parts of inputs for each type of punctuation attack (on FCE test) for the Gramformer.



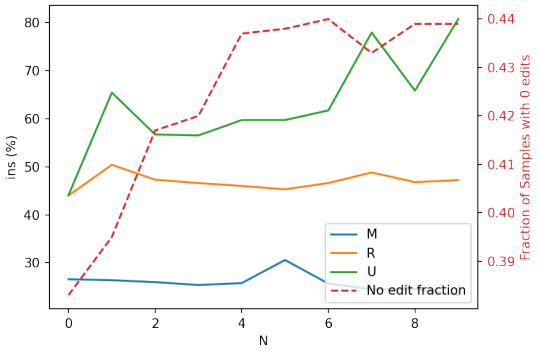
(a) Comma corr



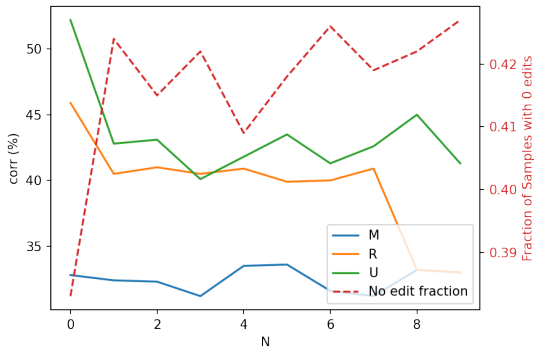
(b) Comma ins



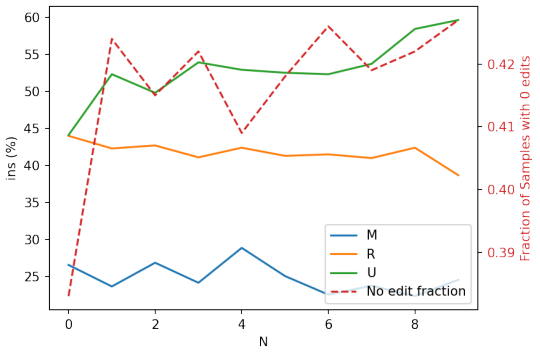
(c) Direct corr



(d) Direct ins



(e) Colon corr



(f) Colon ins

Figure B.2: Edit rates by edit type class for each type of punctuation attack (on FCE test) for the Gramformer.