V Attack: Taking advantage of Text Classifiers' horizontal vision

e
r
t

Jonathan Rusert
Purdue University, Fort Wayne
jrusert@pfw.edu

Abstract

Text classification systems have continuously improved in performance over the years. However, nearly all current SOTA classifiers have a similar shortcoming, they process text in a horizontal manner. Vertically written words will not be recognized by a classifier. In contrast, humans are easily able to recognize and read words written both horizontally and vertically. Hence, a human adversary could write problematic words vertically and the meaning would still be preserved to other humans. We simulate such an attack, *VertAttack*. *VertAttack* identifies which words a classifier is reliant on and then rewrites those words vertically. We find that *VertAttack* is able to greatly drop the accuracy of 4 different transformer models on 5 datasets. For example, on the SST2 dataset, *VertAttack* is able to drop RoBERTa's accuracy from 94 to 13%. Furthermore, since *VertAttack* does not replace the word, meaning is easily preserved. We verify this via a human study and find that crowdworkers are able to correctly label 77% perturbed texts perturbed, compared to 81% of the original texts. We believe *VertAttack* offers a look into how humans might circumvent classifiers in the future and thus inspire a look into more robust algorithms.

1 Introduction

Automatic text classifiers have seen a continual increase in helping websites moderate and monitor products or people. Though they are helpful to reduce the work load of humans, they can be subject to problems like bias (Chuang et al., 2021; Zhou et al., 2021) and are vulnerable to adversarial attacks (Lei et al., 2022; Le et al., 2022). Research into text adversarial attacks has been on the rise in recent years. The reasons range from testing classifiers' robustness (Wang et al., 2022) to privacy concerns (Xie and Hong, 2022).

Current state-of-the-art (SOTA) attacks largely fall into character based attacks and word-based

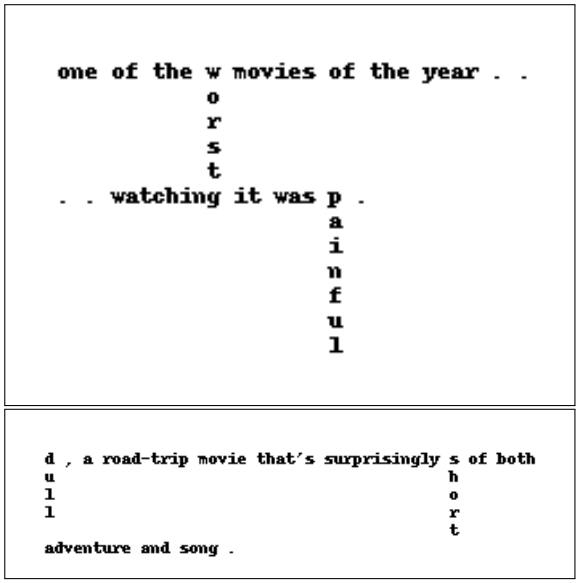

Figure 1: Examples of texts perturbed by VertAttack. Humans can still understand the vertically written words, while classifiers struggle to read.

attacks. Character-based attacks change individual characters, by flipping character, introducing or removing whitespace (Gröndahl et al., 2018), or replacing characters with visually similar characters (Eger et al., 2019). Word-based attacks replace words with similar words which are less known to the target classifier (Li et al., 2020; Wang et al., 2022). One weakness of current SOTA attacks is that they constrain themselves to horizontal changes. That is, the final result is still read in a left-to-right (English) manner. This is a disadvantage because the attacker restricts themselves to the same domain as the classifier which is also only able to read text horizontally.

Humans have the ability to read text in multiple directions, not just horizontally. Thus, a human attacker who wants to communicate a message to others, while avoiding a website automatically classifying that text, could write the words vertically and the meaning would still be preserved. We sim-

ulate this with *VertAttack*.

VertAttack exploits the current limitation of classifiers' inability to read text vertically. Specifically, *VertAttack* perturbs input text by changing information rich words from horizontally to vertically written. Our research makes the following contributions:

1. Propose an attack (*VertAttack*) to mimic how humans may subvert automatic classifiers. This attack exploits current classifiers' glaring weakness (inability to "read" vertical text).

2. Test *VertAttack* on 5 datasets, against 4 different classifiers. We further examine transferability of our attack. We find that when *VertAttack* has blackbox access to the classifier, it is able to drop classification accuracy from 83 - 95% down to 1 - 36%. We further compare *VertAttack* with two other text attacks, BERT-ATTACK and Textbugger. We find that, on average, *VertAttack* is able to drop classifiers' accuracy to 36.6% accuracy, which is lower than BERT-ATTACK (47.5%) and Textbugger (63.2%).

3. Verify *VertAttack*'s ability to be understood by humans via qualitative analysis. We find that humans are able to correctly classify 77% perturbed texts compared to 81% of the original texts.

4. Investigate initial defenses in terms of whitespace removal and find that if *VertAttack* a classifier reverses the algorithm it is able to mitigate the attack, but simpler whitespace preprocessing is not as effective.

5. Enhance *VertAttack* by allowing it to add in *chaff* to further disguise the text. This chaff greatly affects the reversal defense. Furthermore, we investigate how *VertAttack* affects classifiers using OCR to extract text from images.

The success of *VertAttack* shows a vulnerability in classifiers which humans may leverage to easily defeat them. We share code and perturbed texts for future research[1].

2 Threat Model

The examined threat model follows from prior research (Formento et al., 2023; Le et al., 2022; Deng et al., 2022). We assume blackbox knowledge of a classifier. That is, *VertAttack* has no internal knowledge of the classifier, but has access to the probabilities and label output by the model. *VertAttack* uses this for feedback (Section 4.1).

[1]We make our code and generated texts available at https://github.com/JonRusert/VertAttack

With prior research, there is an assumption that the feedback classifier is the same as the target classifier. However, websites rarely share the exact classifier used for moderating texts. Thus, we also examine the cases of where the feedback classifier differs from the target classifier as a transferability problem.

3 Attack Goals

Based on prior research (Lei et al., 2022; Zang et al., 2020; Li et al., 2019) *VertAttack* has 2 goals: 1. Modify text in such a way to cause an automated classifier to fail (misclassify). 2. Ensure modified retains the original meaning to humans. Thus, the attack is similar to obfuscation from classifiers.

Some previous text attack research have made the argument that attacks should be imperceptible to humans (Dyrmishi et al., 2023). However, this is not a unanimous requirement from text attacks, as many do not include it as a prerequisite (Alzantot et al., 2018; Ebrahimi et al., 2018; Eger et al., 2019; Li et al., 2021a). Furthermore, this would disqualify nearly all character-level attacks since humans do not naturally substitute characters in their writing (beyond mispellings). Finally, as stated, *VertAttack* simulates how humans can attack automated classifiers. Thus, we focus on the two aforementioned goals.

4 Methodology

Our proposed attack, *VertAttack*, can be divided into two main steps: 1) Word Selection, 2) Word Transformation. A visualization of the method can be seen in Figure 2.

4.1 Word Selection

Algorithm 1 Word Selection

Input: $text$
Output: $j \leftarrow$ PositionToModify
  $Score_{Orig} \leftarrow$ Classifier(text)
  $Drop_{Max} \leftarrow 0, i \leftarrow 0, j \leftarrow 0$
  while $i \neq len(text)$ **do**
    $Score_w \leftarrow$ Classifier(text/w)
    $Drop_w \leftarrow Score_{Orig} - Score_w$
    if $Drop_w > Drop_{Max}$ **then**
      $Drop_{Max} \leftarrow Drop_w$
      $j \leftarrow i$
    end if
    $i \leftarrow i + 1$
  end while

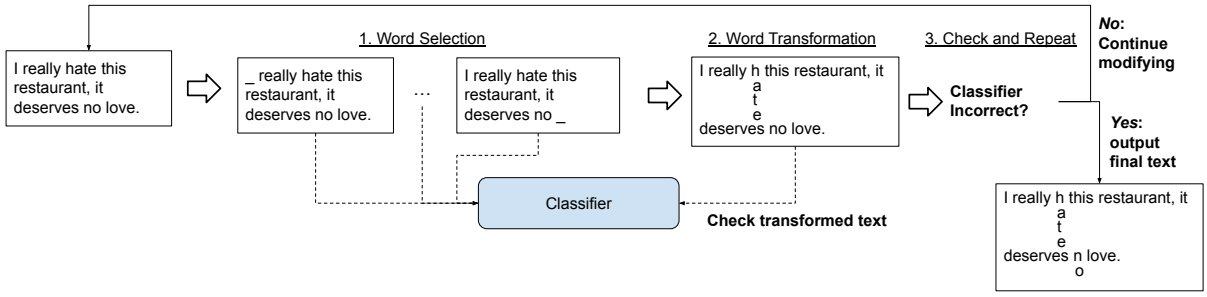

Figure 2: *VertAttack* basic overview. A word to transform is first selected from the input text and then transformed vertically. The classifier assists in providing feedback in the form of class probabilities. The process is repeated until the classifier misclassifies the text.

First the attack finds which word most helps the classifier. We employ a greedy search method (Algorithm 1). In previous work this has been referred to as word importance (Jin et al., 2020) or greedy selection (Hsieh et al., 2019). The method removes one word[2] at a time and checks the change in classification probability from the original text. Each word is removed and then replaced until all probabilities are calculated. The word that causes the highest drop in probability is chosen as the word to be transformed.

4.2 Word Transformation

Algorithm 2 Word Transformation

Input: $text, perturb_{positions}$
Output: $text_m$

```
#_lines ← max length of words to be modified
k ← 0
while k < #_lines do
    i ← 0
    while i ≠ len(text) do
        if i ∈ perturb_positions then
            append text[i][k]
        else
            append word on first line
            or pad spaces equal to word length
        end if
        append space
        i ← i + 1
    end while
    Add newline char to text_m
    k ← k + 1
end while
```

Once a word is selected, it is then transformed vertically (Algorithm 2). First, the number of lines needed (ie. length of word) for each selected word is calculated. Next, we iterate through each word of the original text. If a word is a non-selected word, then it is simply added to the final text. If the word is a selected word, then only the character of the corresponding line is chosen. For example, if "happy" is selected, and the line number is 2, then "a" is added to the final text. For all lines that only consist of whitespace and the vertical characters, the required whitespace is calculated by the length of each non-selected word.

Finally, we add a width constraint to the algorithm for practicality. The transformation is only run on that width (number of words) at a time and all text is combined at the end. For example, if there are 100 words and the width constraint is 10, then only 10 are modified at a time.

Once the transformations are applied, the classifier is queried again to see if the transformed text causes the classifier to misclassify. If so, the final text is produced. If not, then the algorithm repeats, however, this time the words that have been selected already are removed as candidates during the selection step.

5 Experimental Setup

To test the effectiveness of *VertAttack*, we evaluate the attack against several transformer classifiers across datasets examined in previous attack papers[3] (Li et al., 2020; Jin et al., 2020; Ren et al., 2019; Wang et al., 2022).

[2]Here a word is defined as a token separated by whitespace.

[3]The majority of attacks were run on 56-core 256G processors. *VertAttack* was limited to 1 hour for each attacked text, after 1 hour the attack was noted as failure and no perturbations were made to the text.

5.1 Datasets

We examine 4 binary task datasets and one multi-class task dataset. Following prior research (Li et al., 2020), we randomly sampled up to 1000 examples for each dataset to attack. (QNLI contained 872 examples, thus all were used):

1. AG News - a collection of news articles divided into 4 categories (World, Sports, Business, Sci/Tech). Average text length is 38 words.

2. SST-2 - Stanford Sentiment Treebank, contains movie reviews labeled for sentiments (positive/negative) by humans. Average text length is 20 words.

3. CoLA - Corpus of Linguistic Acceptability, contains English sentences labeled grammatical correctness. Average text length is 8 words.

4. QNLI - Stanford Question Answering Dataset, contains question/answer pairs. A classifier must determine whether the context sentence contains the answer to the question. Note that we restrict *VertAttack* to modify the context sentence only. Average text length is 28 words.

5. Rotten Tomatoes (RT) - contains movie reviews from Rotten Tomatoes. Each review is labeled as positive or negative. Average text length is 21 words.

5.2 Classifiers

We examine a combination of up to 4 classifiers per dataset. At least 3 classifiers are examined per dataset to measure how well the attack transfers. We look at a combination of transformer models[4] (Morris et al., 2020):

1. BERT (base-uncased) - a fine-tuned version of BERT (Devlin et al., 2019) on the corresponding dataset. For example, for AG News, the bert-base-uncased model was fine-tuned on the AG News training data.

2. Albert - a fine-tuned version of the ALBERT model (Lan et al., 2019). ALBERT has a smaller memory footprint than BERT, since it shares weights across layers.

3. RoBERTa - a fine-tuned version of the RoBERTa model (Liu et al., 2019). RoBERTa has seen stronger classification results in recent years than BERT, due to choices made during pretraining.

4. DistilBERT - a fine-tuned version of DistilBERT (Sanh et al., 2020). DistilBERT is a lighter, faster version of BERT which was pretrained using BERT as a teacher for self-supervision.

[4]We leverage pretrained models via TextAttack: https://github.com/QData/TextAttack

		Classifiers			
	Feedback	BERT	Albert	Rob.	Disti.
AG	Orig.	94.2	94.2	94.7	-
	BERT	4.7	43.7	25.9	-
	Albert	60.2	8.0	31.2	-
	Rob.	86.9	79.3	20.2	-
SST-2	Orig.	92.4	92.7	94	-
	BERT	12.5	46.7	53.0	-
	Albert	53.6	13.4	57.7	-
	Rob.	50.2	51.3	13.4	-
CoLA	Orig.	81.2	82.9	85.7	82.5
	BERT	5.5	29.9	35.4	33.1
	Albert	31.6	14.8	20.3	33.7
	Rob.	32.4	31.8	1.2	33.5
	Disti.	31.6	31.6	45.6	15.5
QNLI	Orig.	90.4	-	91.7	86
	BERT	33.5	-	67.5	60.8
	Rob.	62.8	-	32.4	63.1
	Disti.	64.4	-	67.8	35.6
RT	Orig.	85.4	84.8	88.6	-
	BERT	6.7	48.2	46.3	-
	Albert	46	14.7	45.2	-
	Rob.	56.3	40.2	25.8	-

Table 1: *VertAttack* results on datasets, accuracy is shown. The second column indicates which classifier was used to give feedback to VertAttack. Orig. = original accuracy without any attack. Rob. = RoBERTa, Disti. = Distilbert.

5.3 Metrics

To calculate the effectiveness of *VertAttack*, we examine 1 quantitative metric and 1 qualitative. For quantitative, we measure accuracy:

$$Accuracy = \frac{\#correctly\_classified}{\#total\_examples} \quad (1)$$

For qualitative, we measure human ability to understand the text. Specifically, we leverage crowdworkers as judges for the perturbed texts. We ask 3 crowdworkers to label each text (for class) and take the majority vote as a decision.

6 *VertAttack* Results

Our main *VertAttack* results are found in Table 1[5]. The second column indicates which classifier is leveraged for feedback for VertAttack. We examine attacks where the feedback and target classifier are the same (diagonal rows), as well as transferability of attacks (non diagonal). Note that the former is

[5]Due to computational intensity of attacks, we opt to test differ combinations of classifiers on the datasets rather than every combination on every dataset.

the standard measurement in most attack papers. We make the following observations:

***VertAttack* causes large drops to classifier accuracy.** Our results demonstrate the effectiveness of *VertAttack* across datasets and classifiers. Specifically, when examining the cases where the feedback classifier is the same as the target classifier, we see up to 90 point drops. In AG News, *VertAttack* is able to drop BERT from 94.2% to 4.7%, Albert from 94.2 to 8.0, and RoBERTa from 94.7 to 20.2, which averages to 83 points. Similar drops from *VertAttack* are seen in the other datasets as well: SST-2 averages 80 points, CoLA averages 74 points, QNLI averages 56 points, and Rotten Tomatoes averages 71 points. Overall, these results support VertAttack's strength in fooling classification systems.

VertAttack's attacks transfer to other classifiers. Though not as strong, we find *VertAttack* to be successful even in cases of transferability. In the most effective case (the CoLA datasets), the transfer attacks cause an average drop of 51 points (max: 65, min: 40.1). These drops are detrimental to text classifiers' effectiveness and reliability. Slightly lesser drops are seen for SST-2, AG News, and Rotten Tomatoes which causes drops around 40 points on average. Finally, classifiers on the QNLI dataset see drops of 25 when the feedback classifier differs. In even the final cases, the attacks is a hinderance to classification methods and highlight their inability to process text as effectively as humans.

QNLI models most resilient to attack. Unlike the other datasets, which saw at least 1 classifier drop below 20% classification accuracy, QNLI classifiers dropped to only 32% in the lowest. This might be due to the difficulty of attacking multi-text inputs. We limited *VertAttack* to only attack the hypothesis and not the premise. We would most likely see a drop in accuracy if premise is allowed to be attacked as well, but we restricted to the hypothesis for a more realistic model where a user is proposing a hypothesis to a model's premise.

BERT and DistilBert show strength as most robust classifiers examined. To investigate resilience against VertAttack, we calculate three averages for each classifier, seen in Table 2: 1. The classifier used by *VertAttack* for feedback is the same as the target classifier (Same), 2. The classifier used by *VertAttack* is **different** than the target classifier (Diff.), 3. Inclusion of both 1 and 2 (All). Each score corresponds to the drop in accuracy against VertAttack. Thus, for resiliency, classifiers would like to have a lower drop in accuracy. We can see that DistilBert has the lowest drops in two cases (Same, All), while BERT has the lowest for the third (Diff.). However, BERT is examined in all 5 datasets, while DistilBert is only examined in 2. Thus, no final decision can be noted on most resilient between the two.

	BERT	Albert	Rob.	Disti.
Same	76.1	75.9	72.3	**58.7**
Diff.	**35.7**	43.3	45.4	39.06
All	48.3	53.3	53.8	**44.7**

Table 2: Average drops of *VertAttack* against the corresponding classifier across all datasets. Three averages are shown: "Same" indicates the average of the attacks where the feedback classifier was the same as the attacked. "Diff." indicate the set of attacks where the feedback classifier differed from the attacked."All" is the average for all drops against the classifier. Bold values indicate lowest drops.

VertAttack

		Actual	
		+	-
Pred.	+	41	16
	-	7	36

Original

		Actual	
		+	-
Pred.	+	40	11
	-	8	41

Table 3: Confusion Matrices of human study results. Participants labeled 100 perturbed RT texts as positive (+) or negative (-) sentiment. Each text received 3 votes, a majority vote was taken.

7 Human Study

To investigate humans' understanding of VertAttack's texts, we employed human crowdworkers to label a sampled set of texts which were perturbed by VertAttack. Specifically, we randomly sampled 100 of the 1000 texts from the Rotten Tomatoes dataset. We then asked crowdworkers to read the text and decide the sentiment of the text (positive or negative). For each text, we employed 3 crowdworkers[6], and took the majority vote of the labels. It should be noted that no instructions to read the texts vertically were given. More information on the instructions can be found in Appendix A.

The confusion matrix of results is in Table 3. Humans were able to identify sentiment correctly, 77% of the time, far greater than the 7 - 26% of the automated classifiers. This confirms that unlike the automated classifiers, humans are well prepared to read text in non-traditional manners.

[6]Amazon Mechanical Turk

For comparison, we also ran the same study with on the original, unperturbed 100 texts. This is also in Table 3 under the "Original" subtable. Humans are able to do slightly better on the unperturbed texts achieving an accuracy of 81%. However, VertAttack's percentage is only 4 points below (77%). This highlights that human misclassifications on VertAttack's texts have more to do with the difficulty of some of the texts rather than due to perturbation.

8 Comparisons with other attacks

To further investigate how *VertAttack* performs in the adversarial text space, we compare to two other attacks, BERT-ATTACK (Li et al., 2020) and Textbugger (Li et al., 2019)[7]. BERT-Attack is similar to *VertAttack* as it is a word based attack. To select a word, BERT-ATTACK finds the importance score of a word by masking each word (one at a time) and comparing to the original logits. For replacement, BERT-ATTACK relies on BERT to give suggestions via its MLM training. Textbugger is a character based attack which tests inserting, deleting, swapping, or substituting characters. We run both attacks on the same 1000 examples from the Rotten Tomatoes dataset. The results can be seen in Table 4.

Overall, we find that BERT-ATTACK causes greater drops when the feedback classifier is the same as the attacked classifier, but *VertAttack* transfers better. Textbugger is weaker in both cases. Specifically, when the feedback classifier is the same (diagonal values), BERT-ATTACK causes classifiers to average 9.5% accuracy compared to VertAttack's 15.7% and Textbugger's 33.5%. However, for transferability (non diagonal values), *VertAttack* causes classifiers to average 47% accuracy, 19 points less than BERT-ATTACK's average of 66.5% and 31 points less than Textbugger's average of 78.1. Furthermore, when taking the overall averages (all cells) *VertAttack* drops classifiers to 36.6% accuracy while BERT-ATTACK averages 47.5% and Textbugger averages 63.2%.

9 Malicious Use - Offensive Language

To confirm the main results and demonstrate how *VertAttack* may be used maliciously, we apply *VertAttack* to "offensive" texts. We take a subset of OLID's (Zampieri et al., 2019) test set, labeled OFF (offensive). This results in 260 texts. We leveraged pretrained classifiers from Huggingface [8], trained on OLID training data. We examine 3 variations of transformer models, BERT, Albert, and XLNet (Yang et al., 2019). The full results are in Table 5.

VertAttack is able to greatly reduce the classification accuracy for all three models. When the feedback classifier is the same as the target, the accuracy drops to 1% or lower. When the classifiers differ, the accuracy is also low, in the range 13 - 28%. These results demonstrate how the attack can cause issues on popular social media websites which leverage automated classifiers to help curb offensive language.

10 Effect on OCR + Classifier

To guarantee the preservation of whitespace, we can write text to an image (as done in the human study). The question arises of how a classifier which leverages OCR to extract text from images would fare. We test this by first converting the modified text into an image using the PIL library[9]. Next, we use Tesseract OCR[10] to extract the text

		Classifiers		
		BERT	Albert	RoBERTa
	Original	85.4	84.8	88.6
Vert A.	BERT	**6.7**	48.2	46.3
	Albert	*46*	14.7	*45.2*
	RoBERTa	56.3	*40.2*	25.8
Bert A.	BERT	22.9	52.3	74.8
	Albert	79	**1.9**	78.7
	RoBERTa	66.6	47.3	**3.6**
Textb.	BERT	46.2	52.3	74.8
	Albert	85.8	16.1	91.6
	RoBERTa	74.1	56.9	38.2

Table 4: *VertAttack* compared with BERT-Atttack and Textbugger. The second column indicates which classifier was used to give feedback to the attacks. Bold values indicate stronger attacks against that classifer. Italic values indicate strongest transfer attack.

		Classifiers		
	Feedback	BERT	Albert	XLNet
	Original	76.7	78.3	78.3
	BERT	1.3	23.8	27.5
	Albert	20	0	26.7
	XLNet	12.9	17.1	0.8

Table 5: *VertAttack* results on OLID dataset, on the OFF labeled (Offensive Language). Accuracy is shown. The second column indicates which classifier was used to give feedback to *VertAttack*.

[7] TextAttack was leveraged to simulate these attacks: github.com/QData/TextAttack

[8] https://huggingface.co/mohsenfayyaz

[9] https://pypi.org/project/Pillow/

[10] https://github.com/tesseract-ocr/tesseract

		Classifiers		
	Feedback	BERT	Albert	RoBERTa
	Original	85.4	84.8	88.6
None	BERT	6.7	48.7	50
	Albert	47.7	13.6	48.7
	RoBERTa	44.8	45.5	9.4
OCR	BERT	40.5	47.3	48.2
	Albert	48.4	35.7	49.2
	RoBERTa	45.6	44.1	37.7
	Maj. Class	53.3		

Table 6: Accuracy results on RT dataset when images containing VertAttack modified text are converted to text (via OCR) and classified. "None" refers to the original accuracy with no conversion to image and back via OCR. Second column indicates which classifier was used for attack feedback. "Maj. Class" indicates a simple baseline which always predicts the majority class.

from the image and classify it. We test this on Rotten Tomatoes. The feedback and target classifiers use the text segmenter (Section 11). The results can be found in Table 6. We include a simple majority class baseline for comparison.

For OCR, we see accuracy increase in the cases when the target and feedback classifier are the same. For example, Albert classification changes from 13.6 to 35.7. When feedback and target classifiers differ, the accuracy is similar to the original attacked accuracy. All accuracies are below the simple majority class baseline of 53.3. Thus, even though OCR increase accuracy, it is still detrimental for a classifier. Furthermore, VertAttack could be further modified to target a classifier which includes OCR in the pipeline.

11 Initial Defenses

We investigate some initial steps automated classifiers might take to mitigate VertAttack's effectiveness. Since *VertAttack* introduces whitespace, simple solutions might be to reduce that whitespace. Thus, we look at three different approaches. First, we simply remove extraneous whitespace and limit at most 1 space between each token, denoted as **Simple**. Second, we leverage a text segmentation library[11] to remove whitespace and re-combine words, denoted as **Segment**. Finally, we assume the classifier has learned the algorithm for VertAttack and thus reverses it. That is, the classifier attempts to recombine vertical characters into words before classification. This is denoted as **Reverse**. The full algorithm can be found in the appendix (Appendix B).

[11] grantjenks.com/docs/wordsegment/

		Classifiers		
	Feedback	BERT	Albert	RoBERTa
	Original	85.4	84.8	88.6
	VertAttack - None			
Simple	BERT	6.7	48.7	50.0
	Albert	46.0	29.7	47.6
	RoBERTa	56.3	38.1	59.8
Seg.	BERT	37.8	49.6	53.8
	Albert	45.4	49.2	51.1
	RoBERTa	62.3	43.8	62.8
	VertAttack - Simple			
Simple	BERT	6.7	48.7	50
	Albert	47.7	13.6	48.7
	RoBERTa	44.8	45.5	9.4
	VertAttack - Segmenter			
Seg.	BERT	10.0	44.7	53.6
	Albert	49.3	4.7	53.6
	RoBERTa	41.7	41.8	8.2

Table 7: *VertAttack* results on RT dataset with different whitespace preprocessing present, accuracy is shown. First column indicates which method the classifier used: Simple - remove all extraneous spaces in input text, Seg. - leverage word segmenter to process the input text. Second column indicates which classifier was used to give feedback to VertAttack. "*VertAttack* - X" indicates which method *VertAttack* used with classifier feedback.

11.1 Simple + Segment

For the first two approaches, we run them on the original attacked Rotten Tomato (RT) texts (from Table 1). We then modify *VertAttack* to have this information during its attacks as feedback, as changing the preprocessing method during classification puts the attack at a natural disadvantage since the feedback is no longer as reliable. The full results of these experiments are in Table 7.

We observe that when *VertAttack* includes a preprocessing method for feedback that is different than what the attacked classifier uses ("*VertAttack* - None"), the attack suffers. For example, examining the diagonal results, the simple preprocessing is able to raise Albert's classification accuracy from 14.7 to 29.7. The word segmentation approach raises it even higher (to 49.2). Similar results are seen across the table. The transferability results (feedback classifier differs from final classifier) also generally increase, but not nearly as strong. This follows as *VertAttack* is modifying texts based on a classifier that differs in preprocessing and hence the attack becomes a transferability problem itself.

When *VertAttack* has the same method in its feedback classifier, then the approaches are not as fruitful ("*VertAttack* - Simple", "*VertAttack* - Segmenter"). Again with Albert (on the diagonal),

		Classifiers		
	Feedback	BERT	Albert	RoBERTa
	Original	85.4	84.8	88.6
Reverse	BERT	84.4	84.2	88.4
	Albert	82.6	84.3	87.8
	RoBERTa	86	82.6	87.3

Table 8: *VertAttack* results on RT dataset when the classifier reverse-engineers VertAttack, accuracy is shown.

we actually see a decrease in classification accuracy from 14.7 to 13.6 for **Simple** and down to 4.7 for **Segmentation**. This indicates the importance of the feedback classifier as it can strongly affect VertAttack's perception of a strong attack and the importance of whitepace preprocessing for classifiers if the attacker is not prepared.

11.2 Reverse

The **Reverse** preprocessing results can be found in Table 8. As can be observed, the algorithm is able to strongly combat VertAttack, increasing the accuracy from 6 - 24 to 84 - 87. However, we observe that it is not able to mitigate it entirely, as some texts are entirely written vertically and the algorithm is not able to distinguish when new lines of words begin. We next introduce an augmentation to VertAttack to combat the **Reverse** algorithm.

12 Enhancing VertAttack with Chaff

As demonstrated, if the classifier knows this type of attack is occurring, it can strongly mitigate it by reversing the algorithm. Thus, we enhance VertAttack by introducing chaff. Specifically, rather than inserting only whitespace vertically, an alphabet character has a chance of being inserted. This occurs at a probability p. For example, if $p = 10$, then there is a 10% probability that rather than whitespace, a character is inserted in the vertical lines. Note that to preserve readability we do not allow this for whitespace next to perturbed words (nor original whitespace).

We test chaff for $p = \{5, 10, 20, 30, 60\}$. The main results against the **Reverse** algorithm (Section 11.2) are in Table 9. We find that this enhancement hinders the ability to reverse the attack. This is because **Reverse** is not able to identify non-perturbed characters. For example, when $p = 30$ BERT's accuracy drops from 85 to 40. This is 44 points lower than when the **Reverse** is applied to $p = 0$ (no chaff). Similar trends are seen for Albert and RoBERTa as well. As p increases, we find greater accuracy drops. This points to the reverse

		Classifiers		
	Feedback	BERT	Albert	RoBERTa
	Original	85.4	84.8	88.6
	p = 0%			
None	BERT	6.7	48.2	46.3
	Albert	46.0	14.7	45.2
	RoBERTa	56.3	40.2	25.8
Reverse	BERT	84.4	84.2	88.4
	Albert	82.6	84.3	87.8
	RoBERTa	86	82.6	87.3
	p = 5%			
None	BERT	6.4	48.3	46.1
	Albert	46.8	15.9	44.9
	RoBERTa	57.7	41.3	24.6
Reverse	BERT	76.4	78.1	81.1
	Albert	75.8	75.7	82.0
	RoBERTa	77.9	76.3	78.6
	p = 10%			
None	BERT	6.0	49.1	46.3
	Albert	46.3	17.0	44.4
	RoBERTa	57.33	42.0	24.4
Reverse	BERT	64.8	70.7	71.6
	Albert	68.2	64.7	76.2
	RoBERTa	73.7	71.5	67.4
	p = 20%			
None	BERT	5.9	48.4	46.6
	Albert	45.3	18	45.2
	RoBERTa	57.7	42.2	24.2
Reverse	BERT	48.7	63.2	62.4
	Albert	60.8	47.1	67.9
	RoBERTa	67.1	69.7	50.2
	p = 30%			
None	BERT	5.8	49.2	47.6
	Albert	44.7	19.6	44.3
	RoBERTa	55.5	42.3	23.7
Reverse	BERT	39.8	59.3	58.2
	Albert	58.1	40.1	64.5
	RoBERTa	63.8	65.8	40.5
	p = 60%			
None	BERT	6.2	48.5	47.4
	Albert	45.2	21.0	43.7
	RoBERTa	55.5	42.3	23.7
Reverse	BERT	27.7	60.1	55.0
	Albert	57.5	28.9	63.2
	RoBERTa	59.7	64.1	35.9

Table 9: *VertAttack* results on RT dataset when chaff is added in (described in Section 12). "None" means no preprocessing is used and "Reverse" is the classifier attempting to reverse engineer VertAttack.

	# of correct responses		
	>= 1	>= 2	=3
Original	94	81	49
VertAttack	92	77	47
Chaff $p = 30$	83	47	23

Table 10: Human results for all three text variations. The values indicate the percentages of texts correctly classified by at least X humans where X is indicated in the column header. Original and VertAttack are the same values from Table 3. Chaff $p = 30$ indicates that chaff is added to the perturbed text at 30% rate.

algorithm becoming less able to avoid the random inserted text.

We verify that readability is maintained, following the same process in the main human study (Section 7). Table 10 compares human evaluations of adding in chaff at a rate of 30%. We see a drop in correct responses but at least 1 human is able to correctly identify the sentiment in at least 83% of the texts.

This enhancement further demonstrates *VertAttack* as a strong representation of how humans can adjust to combat automatic classifiers.

13 Related Work

Here we examine some of the other current SOTA attacks. We examine both word-based attacks and character-based attacks as *VertAttack* shares some characteristics with both.

Word-Based Attacks: Like *VertAttack*, current black-box SOTA word-based attacks attack a classifier by receiving feedback from that classifier. This feedback is in the form of label probabilities (Hsieh et al., 2019), or the logits of the classifier (Li et al., 2021b). Black-box, word-based attacks follow similar steps to *VertAttack*. First, they choose tokens for replacement, and then they leverage a tool to choose a replacement. This could be a transformer model (Li et al., 2020), a lexicon like WordNet (Ren et al., 2019), or word embeddings (Hsieh et al., 2019). Unlike, *VertAttack* current word-based attacks only operate in the horizontal space. That is, all words chosen for replacement are substituted for that word in place. Their goal is to find words which a classifier does not know well enough to make a correct classification. Thus, *VertAttack* is set apart by operating in the vertical space. Furthermore, *VertAttack* does not replace the selected word, thus meaning is more easily preserved.

Character-Based Attacks: Another common type of SOTA attack are character-based attacks which change text at the character level. These attacks generally aim to be more transferable than word attack and thus do not receive feedback from a classifier. Instead, the changes are applied at a random chance throughout the text. For example, whitespace might be removed (Gröndahl et al., 2018) or added or standard, English characters might be replaced with non-standard similar looking characters (e.g "a" → "@")(Eger et al., 2019). Both cases try to cause classifiers to see words as out-of-vocabulary. One downside is that character-level attacks can be mitigated more easily with proper preprocessing (Rusert et al., 2022). *VertAttack* is similar, in that it focuses on the characters of a word, however, *VertAttack* uses an internal classifier for feedback. Furthermore, due to the positioning of the characters, *VertAttack*'s changes are harder to correct with preprocessing of text.

Whitespace attacks have also been shown to be effective against LLMs. Cai and Cui (2023) find that adding a whitespace before a comma in a text can fool a classifier to misclassify a text as human-generated instead of machine-generated. This attack, *SpaceInfi*, differs from *VertAttack* since it only focuses on this specific classification task. Furthermore, it adds a single space next to commas. In our experimentations, we focus on classification tasks where syntactic structure is less important.

14 Conclusion

We presented a new attack which exploits current classifiers' inability to understand text written vertically. Mimicking a human, *VertAttack* perturbs text by rewriting words in a vertical manner which humans are able to understand, but classifiers are not. We find drops in classification up to 86 points.

Furthermore, *VertAttack* produces texts which humans can understand. Human crowd workers verified this by labeling 77% perturbed texts correctly, compared to 81% of non perturbed texts.

When compared to other attacks, *VertAttack* causes stronger drops when transferability of attacks is included. *VertAttack* drops classifiers to 36.6% accuracy compared to 46.5% of BERT-Attack and 63.2% of Textbugger.

We explored initial results on how *VertAttack* affects classifiers with OCR. We found that these classifiers are more robust, but still vulnerable.

Finally, We investigated initial defenses against *VertAttack* and found that the methods are able to mitigate the attack as long as *VertAttack* does not enhance with chaff.

Every experiment shows *VertAttack*'s ability to maintain readability and cause large accuracy drops in multiple classifiers. We also find humans do know the meaning in the attacked text. Hence, the overall results will be useful for future research.

15 Limitations

Here we note some limitations with our method and with our experiments. These limitations should

be kept in mind when working and expanding on *VertAttack* so that they addressed or noted:

Websites are not guaranteed to preserve formatting of text produced by VertAttack. *VertAttack* produces text in which targeted words are vertically perturbed. It does this by adding in multiple newlines characters and padded whitespace to preserve readability. However, not all websites are guaranteed to preserve this additional whitespace. Some may completely remove extra newlines which will cause the produced text to greatly drop in readability. One solution to this is leveraging a module to write the text into an image (as seen in the examples (Figure 1). With an image, the formatting of text will be honored and readable to humans. Furthermore, this adds another layer to the attack as text would first need to be processed from the image for classification. However, not all websites allow images, and thus it is a noted limitation to be remedied in the future.

Our attacks focused exclusively on transformer classification models. Though transformers are the current kings of classification, not all websites might have the resources to employ these types of models and thus investigation into simpler models may be useful to confirm VertAttack's effectiveness. However, generally non-transformer models have struggled against adversarial attacks and in the past, and there seems to be no reason why they would fare any better against VertAttack.

Greedy word selection is time consuming. The selection method is the least efficient part of VertAttack. As noted, many previous attacks have leveraged a similar method (Section 4.1). This is due to lack of classifier knowledge in blackbox approaches, thus most tokens need to be checked in selection. However, there do exist more efficient approaches. For example, some style transfer algorithms use attention mechanisms to find the most important words (Wu et al., 2019). Thus, *VertAttack* could be further improved by improving the selection algorithm.

16 Ethical Considerations

By simulating adversarial attacks, such as *VertAttack*, concerns can arise over ethical implications. For example, introducing such a method might allow malicious users to more easily introduce harmful texts into websites and other spaces. This is a further concern as, for research, we make code and algorithms publicly available. This needs to be considered when introducing and studying any adversarial attack. However, we believe that in spite of the above possible wrongful uses, *VertAttack* can be helpful in studying both robustness and future understanding tasks of text classification systems. This is further emphasized as humans can naturally perform this attack and there is no dataset which collects these attacks done by humans. Hence, *VertAttack* provides a way to simulate and further study such attacks. Through this simulation, classifiers, defenses, and other related NLP systems can benefit in a public space. Our hope is not that this algorithm is ever used for malicious purposes, but to improve the aforementioned systems. Thus, we believe the benefits to outweigh any risks.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples.

Shuyang Cai and Wanyun Cui. 2023. Evade chatgpt detectors via a single space. *arXiv preprint arXiv:2307.02599*.

Yung-Sung Chuang, Mingye Gao, Hongyin Luo, James Glass, Hung-yi Lee, Yun-Nung Chen, and Shang-Wen Li. 2021. Mitigating biases in toxic language detection through invariant rationalization. *arXiv preprint arXiv:2106.07240*.

Chuyun Deng, Mingxuan Liu, Yue Qin, Jia Zhang, Hai-Xin Duan, and Donghong Sun. 2022. ValCAT: Variable-length contextualized adversarial transformations using encoder-decoder language model. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1735–1746, Seattle, United States. 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.

Salijona Dyrmishi, Salah Ghamizi, and Maxime Cordy. 2023. How do humans perceive adversarial text? a reality check on the validity and naturalness of word-based adversarial attacks. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8822–8836, Toronto, Canada. Association for Computational Linguistics.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. Hotflip: White-box adversarial examples for text classification.

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.

Brian Formento, Chuan Sheng Foo, Luu Anh Tuan, and See Kiong Ng. 2023. Using punctuation as an adversarial attack on deep learning-based NLP systems: An empirical study. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1–34, Dubrovnik, Croatia. Association for Computational Linguistics.

Tommi Gröndahl, Luca Pajola, Mika Juuti, Mauro Conti, and N. Asokan. 2018. All you need is "love": Evading hate speech detection. *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*.

Yu-Lun Hsieh, Minhao Cheng, Da-Cheng Juan, Wei Wei, Wen-Lian Hsu, and Cho-Jui Hsieh. 2019. On the robustness of self-attentive models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1520–1529, Florence, Italy. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Thai Le, Jooyoung Lee, Kevin Yen, Yifan Hu, and Dongwon Lee. 2022. Perturbations in the wild: Leveraging human-written text perturbations for realistic adversarial attack and defense. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2953–2965, Dublin, Ireland. Association for Computational Linguistics.

Yibin Lei, Yu Cao, Dianqi Li, Tianyi Zhou, Meng Fang, and Mykola Pechenizkiy. 2022. Phrase-level textual adversarial attack with label preservation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1095–1112, Seattle, United States. Association for Computational Linguistics.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and Bill Dolan. 2021a. 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.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021b. Searching for an effective defender: Benchmarking defense against adversarial word substitution. *ArXiv*, abs/2108.12777.

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*.

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.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

Jonathan Rusert, Zubair Shafiq, and Padmini Srinivasan. 2022. On the robustness of offensive language classifiers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7424–7438, Dublin, Ireland. 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.

Boxin Wang, Chejian Xu, Xiangyu Liu, Yu Cheng, and Bo Li. 2022. SemAttack: Natural textual attacks via different semantic spaces. In *Findings of the Association for Computational Linguistics: NAACL 2022*,

pages 176–205, Seattle, United States. Association for Computational Linguistics.

Xing Wu, Tao Zhang, Liangjun Zang, Jizhong Han, and Songlin Hu. 2019. Mask and infill: Applying masked language model for sentiment transfer. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5271–5277. International Joint Conferences on Artificial Intelligence Organization.

Shangyu Xie and Yuan Hong. 2022. Differentially private instance encoding against privacy attacks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop*, pages 172–180, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the Type and Target of Offensive Posts in Social Media. In *Proceedings of NAACL*.

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

Xuhui Zhou, Maarten Sap, Swabha Swayamdipta, Yejin Choi, and Noah Smith. 2021. Challenges in automated debiasing for toxic language detection. pages 3143–3155.

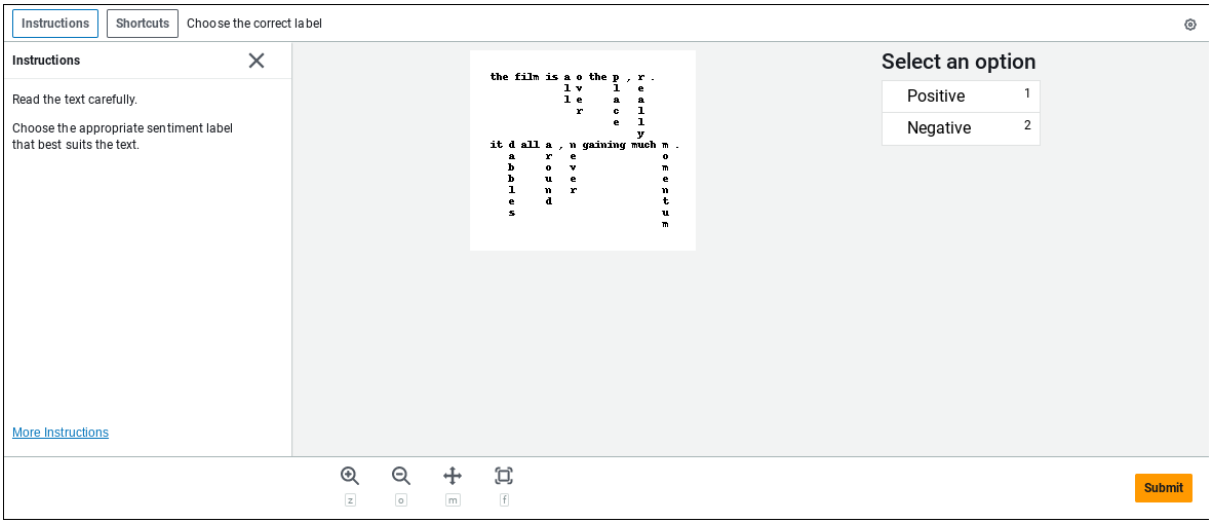

Figure 3: Instructions shown to Amazon Mechanical Turk crowdworkers.

A Human Study Details

For the human study we leveraged Amazon Mechanical Turk crowdworkers to annotate sentiment on Rotten Tomatoes text which were perturbed by VertAttack. The instructions provided to the participants can be seen in Figure 3. As can be seen, no instructions to read the text vertically were given. For each annotation of text, crowdworkers were paid $0.08. Each text received 3 annotations. As AMT does presents each text as a separate task, the 3 annotators for 1 text were rarely the same annotators for another task, thus annotator agreement was not calculated.

To present the texts, we leverage the PIL library in python to write the texts into simple images. An example of this can be seen in the example images (Figure 1). We chose to push the text onto images to avoid any website dependent presentation of the text (e.g. the worker viewer the text on a desktop versus on a phone).

B Reverse Algorithm

Algorithm 3 Reverse

Input: Perturbed Text
Output: Preprocessed Text
  Split_Text ← Text.Split('\n')
  $Drop_{Max} \leftarrow 0, i \leftarrow 0, j \leftarrow 0$
  $Top\_Line \leftarrow 0$
  while $i \leq$ Split_Text.length() **do**
    $cur\_line = Split\_Text[i]$
    if $length(word) \in cur\_line > 1$ **then**
      update previous top line, add to final text
      $Top\_Line \leftarrow i$
    else
      store characters at positions
    end if
    $i \leftarrow i + 1$
  end while

The full reverse algorithm can be found in Algorithm 3. The algorithm first splits by new line characters. To combine vertically written characters, the algorithm appends them to the position in an original text line. An original text line is determined by those lines which have more than single characters. Note, the algorithm cannot just take the top line as the only text line as the width constraint in VertAttack adds vertical lines throughout the text.

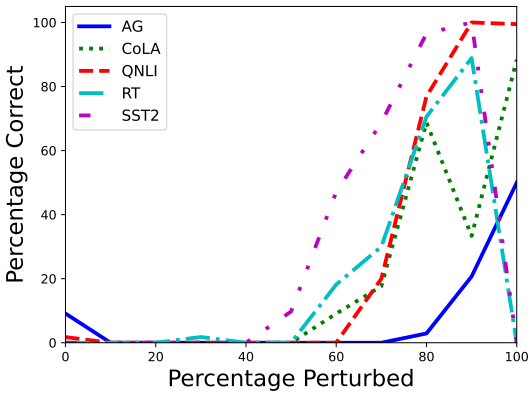

Figure 4: The classifiers' ability to correctly classify text as the amount of words perturbed increases. The classifier examined is BERT, when *VertAttack* uses BERT for feedback.

C Analysis of Percentage of Words Perturbed

For additional understanding of VertAttack, we seek to analyze how the number of words modified by *VertAttack* affects the classifiers. One might postulate that as *VertAttack* modifies more words the classifier does worse, as more and more of the original text is lost. However, through our analysis we find the opposite to be true.

Figure 4 graphs BERT's classification ability versus percentage of text perturbed across the 5 examined datasets. Surprisingly, we see that as the percentage of words perturbed increases, the classifier is better equipped to make a correct classification. This may partially be due to a limitation with *VertAttack* compared to some other attacks. Other attacks are able to bring in new words whose embeddings can cause additional confusion for the classifier, but *VertAttack* does not.