NoisyHate: Mining Online Human-Written Perturbations for Realistic Robustness Benchmarking of Content Moderation Models

Yiran Ye[1], **Thai Le**[2], **Dongwon Lee**[1]
[1]The Pennsylvania State University, USA
[2]Indiana University, USA
yiran_ye@outlook.com, tle@iu.edu, dongwon@psu.edu

Abstract

Online texts with toxic content are a clear threat to the users on social media in particular and society in general. Although many platforms have adopted various measures (e.g., machine learning-based hate-speech detection systems) to diminish their effect, toxic content writers have also attempted to evade such measures by using cleverly modified toxic words, so-called *human-written text perturbations*. Therefore, to help build automatic detection tools to recognize those perturbations, prior methods have developed sophisticated techniques to generate diverse adversarial samples. However, we note that these "algorithm"-generated perturbations do not necessarily capture all the traits of "human"-written perturbations. Therefore, in this paper, we introduce a novel, high-quality dataset of human-written perturbations, named as NoisyHate, that was created from real-life perturbations that are both written and verified by human-in-the-loop. We show that perturbations in NoisyHate have different characteristics than prior algorithm-generated toxic datasets show and thus can be particularly useful to help develop better toxic speech detection solutions. We also provide basic benchmark on the potential utilities of NoisyHate in perturbation normalization and understanding tasks.

Code — https://github.com/YiranYe/NoisyHate
Datasets — https://zenodo.org/records/14654784

Introduction

Toxic/hate speech, a conscious and willful defamatory discourse that often targets a specific group of people, has become more prevalent on the Internet. To diminish the online propagation of hate speech, several social media platforms, such as Twitter and Reddit have published content moderation policies specific to hate speech and developed methods to automatically detect and remove them on their platforms. These systems often either utilize a profanity word list and pre-defined rules or train machine learning (ML) algorithms for detecting hate speech. Although these systems have been shown to work well in benchmark test sets, they make one basic assumption that user-generated content online is often written with correct spelling in English. However, we observe that more often trolls do not use English words in their correct forms–e.g., "bitch", "stupid", but misspelled versions of them that can still be understood by humans–e.g., "bitttch", "stupd". These misspelled words are referred to as text perturbations in ML literature. Because these perturbations are not clean English, they can be used to evade automatic hate speech detection systems online.

Hate Speech Examples from NoisyHate
"speak for yourself you are an **embarrASSment** (**embarrassment**) to Canada" **ASS: Emphasize on ass*
"right there is a nationwide conspiracy to pick up... arrest **bLAck** (**black**) men got it" **LA: a city in the US*
"then the man is a **p!g** (**pig**) as well"
"hey **LOSEEER** (**loser**) that nut job was a democrat, get over it"

Table 1: NoisyHate: Examples of hate speech texts perturbed with *real-life human-written perturbations* (in **bold, red**) to benchmark the robustness of content moderation models. The first two texts are distinctive to humans as they contain hidden meanings that can hardly be produced by machines.

These text perturbations have been produced in lots of different ways. One might use *visually similar* characters to replace the original alphabetical characters. For example, "pr0test" is commonly used to perturb the word "protest". Another perturbation strategy trolls are likely to use *repeating or removing* certain characters in a word, e.g., "bitch"→"bitttch," "stupid"→"stupd." Other approaches to produce text perturbations include *placeholder strategy* ("shit"→"sh_t"), *lower-upper-case strategy* ("democrats"→"democRATs"), and the combination of above approaches. These various text perturbations could be a problem that voids the effects of safeguarding ML algorithms.

Existing works have developed many frameworks to automatically generate text perturbations to benchmark the robustness of ML algorithms (Zeng et al. 2021; Morris et al. 2020). These frameworks often borrow several adversarial attack algorithms from NLP literature (Yuan et al. 2019;

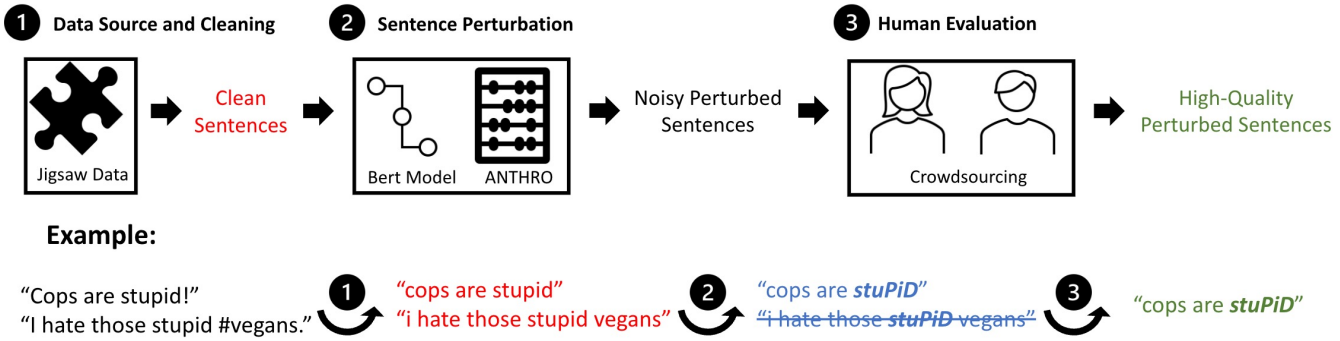


Figure 1: Overall curation pipeline of NoisyHate dataset. This pipeline has three steps: (1) Data sourcing and cleaning from the original Jigsaw dataset (Section), (2) Sentence perturbation with human-written perturbations via pseudo-random sampling (Section) and (3) Human evaluation via crowdsourcing to validate the quality of the perturbed sentences (Section)

Alzantot et al. 2018; Li et al. 2018; Zang et al. 2020; Li et al. 2020; Gao et al. 2018; Ren et al. 2019; Garg and Ramakrishnan 2020). However, there is still a gap between those machine-generated perturbations and human-written perturbations (Le et al. 2022). Evaluating ML models using datasets augmented with human-written perturbations is more practical, as it better reflects real-life scenarios. Hence, this paper proposes a benchmark dataset for the toxic speech detection task that contains diverse human-written perturbations online. The contributions of our work are as follows:

- We introduce a novel benchmark test set, NoisyHate, with online human-written perturbations for toxic speech detection models.
- Our evaluation with state-of-the-art (SOTA) language models and the commercial toxic detection *Perspective API* [1] on NoisyHate reveals that there is still room for improving the robustness of these models on predicting texts with human-written perturbations.
- We test NoisyHate dataset with several spell checkers and show that it is worth developing a better normalization tool targeting these online human-written perturbations.

Related Work

Two research areas that are closely related to our work are text perturbation generation algorithms and toxic speech detection.

Text Perturbation Generation

Machine Generated Text Perturbations. In literature, two major approaches are used to generate adversarial text samples: spelling modification and close word substitution (Alzantot et al. 2018; Ribeiro, Singh, and Guestrin 2018; Sato et al. 2018; Jin et al. 2020). The spelling modification approach usually involves deleting, inserting, swapping, and replacing certain characters in a word. Bhalerao et al. (Bhalerao et al. 2022) proposed a tool named Continuous Word2Vec (CW2V) to perturb text with the following rules: Fake punctuation ("like"→"l.i.k.e"), Neighboring key ("like"→"lime"), Random spaces ("like"→"l ike"), Transposition ("share"→"sahre"), and Vowel repetition and deletion ("like"→"likee"). Other than changing the word's spelling directly, the second approach aims to replace the entire word with other regular English words to attack text classification models. Ribeiro et al.'s work (Ribeiro, Singh, and Guestrin 2018) demonstrated the effectiveness of the attack with semantically equivalent adversarial rules (SEARs) on machine comprehension, visual question answering, and sentiment analysis tasks. SEARs are simple universal replacement rules intending to convert the target word into its semantically identical pairs ("what"→"which", "what is"→"what's"). Alzantot et al. 2018 also introduced an adversarial attack method that replaces the target word with its top k nearest neighbors based on the distance in the GloVe embedding space. To make the perturbation types more diverse, Li et al. 2018 developed TEXTBUGGER, which applied both the spelling modification and close word substitution approaches. Nevertheless, experiments (Le et al. 2022) involving human evaluation reveal that the distance between the perturbations generated by machines and real Human-written Text Perturbations still exists (Le et al. 2022). Moreover, since these adversarial samples are produced based on some known vulnerabilities of a target model, how well they can help the hate-speech detection model prevent real-world attacks is yet to be explored.

Human-written Text Perturbations. CRYPTEXT (Le et al. 2023) is a platform that retrieves human-written perturbations directly from social media, such as Reddit, and provides visualization on the trend of those perturbations. Meanwhile, it also offers an interface to perturb the user-inputted sentences randomly. Due to this randomness, some insignificant words might be perturbed occasionally. For example, given a sentence, "I hate those stupid vegans", a perturbation on "I" or "those" might be insufficient to help this sentence evade the hate speech detection system. Moreover, the CRYPTEXT is using the ANTHRO algorithm (Le et al. 2022) to cluster the words and their perturbations in social media based on their sound and spelling composition (e.g., leading characters, vowels and consonants, and visually similar characters). Therefore, some different standard English words with the same pronunciation, such as "maim" and

[1] https://perspectiveapi.com/

Listing 1: Python code for loading the NoisyHate datasets into a table using the Hugging Face API

```python
from datasets import load_dataset

url = "NoisyHate/Noisy_Hate_Data"
clean_set = load_dataset(url,
  data_files={"clean":"data/clean.
    csv"})
pert_set = load_dataset(url,
  data_files={"pert":"data/pert.csv"
    })
```

"mam," will be treated as each other's perturbation. Hence, the randomly picked perturbation might not fit the context well.

Toxic Speech Detection Dataset. Many researchers have made significant contributions to the toxic speech detection area, as evidenced by the proliferation of well-structured and widely utilized datasets, HatEval 2019(Basile et al. 2019), OffensEval 2019 (Zampieri et al. 2019) and 2020(Zampieri et al. 2020), and TweetEval 2020 (Barbieri et al. 2020). However, these datasets often contain only clean English texts. throughout this period. Barbieri et al. (Barbieri et al. 2020) tested pre-trained Transformers, such as BERT, RoBERTa, and other popular models, such as LSTM and SVM, on the TweetEval data set and reported that the best Macro F1 score on the test set is 0.829 (RoBERTa-Retrained) for offensive speech classification. Mathew et al. (Mathew et al. 2021) introduced another dataset named HateXplain and also trained language models, including BERT and BiRNN, on this dataset. According to their report, BERT demonstrated the best Macro F1 result, 0.687. Perspective API (Jigsaw 2022), created by the Jigsaw and Google team, is one of the most famous black box toxic content detection systems. According to their website, their model was trained on millions of comments from various sources, including online forums such as Wikipedia and The New York Times, across various languages. It is still worthwhile to investigate the performance of these models when subjected to human-written perturbations.

NoisyHate Dataset

Overview and Usage

This section introduces the transformation pipeline from the Jigsaw dataset's original texts to NoisyHate of three consecutive steps, namely (1) data pre-processing, (2) sentence perturbation using human-written perturbations, and (3) quality assurance via crowdsourcing (Fig. 1). NoisyHate dataset is also accessible through HuggingFace's data repository [2] or the public DOI: *10.5281/zenodo.14654784*. Listing 1 shares the code snippets to retrieve and load the NoisyHate into a table format using Python. This semi-automatic pipeline hopes to provide researchers with a consistent framework, including user-study designs and interfaces to curate benchmark datasets with human-written perturbations in domains other than toxic text detection.

[2]https://huggingface.co/datasets/NoisyHate/Noisy_Hate_Data

Algorithm 1: Perturbing Process

Require: Input S_{clean}=$\{w_1, w_2, ..., w_n\}$, BERT model B, Perturbations Generated by the ANTHRO Algorithm $Dict$: $(w, [p_1, p_2, ..., p_t])$, Output Size k, Threshold Θ

$\mathbb{S}_{pert} = []$; $Score_{origin} = B(S_{clean})$
for w_i in S_{clean} **do**
  $Score_{mask_i} = B([w_1, w_2, ..., w_{i-1}, w_{i+1}, ..., w_n])$
  if $abs(Score_{origin} - Score_{mask_i}) > \Theta$ **then**
    for p_j in $random.sample(Dict_{hard}[w_i], k)$ **do**
      $\mathbb{S}_{pert}$ += $[w_1, w_2, ..., w_{i-1}, p_j, w_{i+1}, ..., w_n]$
    end for
  end if
end for
Return: Perturbed Sentence list $\mathbb{S}_{pert} = [S_1, S_2, ..., S_k]$

Step 1: Data Source and Cleaning

Jigsaw is a popular toxic speech detection dataset of nearly 2M public comments from the Civil Comments platform. In addition to the toxic score labels for toxicity classification, it also provides several toxicity sub-type that indicate particular comments' target groups. Due to these prolific identity annotations and significant data volume, we adopted this dataset as our raw data source. There also exist many other datasets, such as HateXplain (Mathew et al. 2021) and Measuring Hate Speech (at UC Berkeley 2024). However, they are generated and annotated based on different methodologies and instructions; thus, incorporating these datasets might introduce inconsistencies or biases to our evaluation. Hence, we used only Jigsaw. Moreover, since the dataset has been used as the standard benchmark dataset for several content moderation tasks, this adoption will also help reduce the entry barrier in NoisyHate's adoption. Since the comments from the Jigsaw dataset contain a lot of special characters, emojis, and informal language, data cleaning was necessary to ensure data quality. Following a typical text processing pipeline, we removed duplicated texts, special characters, special punctuation, hyperlinks, and numbers. Since we only focused on *positive (hate speech)* English texts, sentences containing non-standard English words were filtered out. 13,1982 comments remained after this cleaning step.

Step 2: Sentence Perturbation

In this step, we aim to perturb each of the comments resulting from Step 1 with human-written perturbations. The pseudo-code of this sentence perturbation process is described in Alg. 1.

Word Importance Scoring. Since we focus on benchmarking toxic text detection tasks, it is practical and meaningful to perturb only a few critical words within a sentence. To do this, we first train a proxy toxic detector and utilize it to approximate the importance of each word to toxicity prediction. In particular, we first fine-tune a BERT model (Devlin et al. 2019) on the Jigsaw dataset. Then, we enumerate and mask every words in each sentence and observe how much confidence the trained model changes after such masking

Type	Description	Example
RepeatChar	repeat several characters	stupid→stuppppid
Abbr	delete several characters	stupid→stupd
SpecialChar	replace several characters with Non-English one	stupid→5tupid stupid→st*pid
MixedCase	replace several characters with their upper cases	stupid→sTuPId
MixedCase+	replace several characters with their upper cases, while these characters can be combined into a new word	stupid→stuPiD

Table 2: We categorize all human-written perturbations into five different groups according to five perturbation strategies that curate them. Noticeably, we may encounter more complex perturbations, such as hybrid types (e.g., "shit"→"sh!!!!!t"). In this case, we will mark "sh!!!!!t" as both RepeatChar and SpecialChar.

operations (Alg. 1, Line 4–Line 9). A candidate word is selected to be perturbed at every enumeration step if masking decreases the proxy model's confidence more than a pre-determined threshold. (Alg. 1, Line 5). In this experiment, 0.7 was selected to be the pre-determined threshold.

Word Selection and Perturbation. A comment might have more than one crucial word. However, perturbing all of them will increase the chance that the perturbed sentence being discarded in the annotators. Thus, to maximize the number of comments that remained at the end, we take a conservation measure and only perturb the most important word in a given sentence. Then, we utilize the ANTHRO algorithm (Le et al. 2022), which provides the cluster of online human-written perturbations for a given word, to retrieve the perturbations from social media on the selected important words.

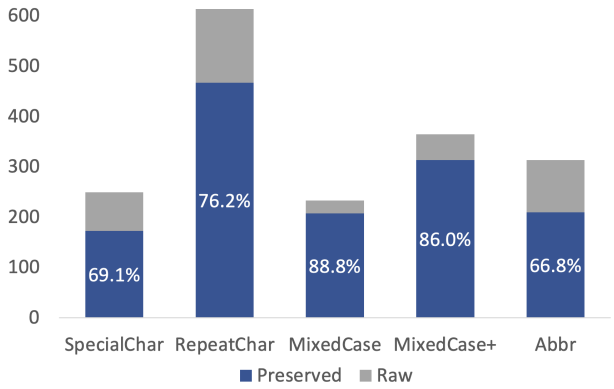
Perturbation Type Diversification. Different types of human-written perturbations exist according to the perturbation strategies that curate them. To categorize all the perturbations used in this step, we classify them into five different perturbation strategies. Table 2 presents the definition of those types and examples. They are (1) repeating characters (RepeatChar), (2) using abbreviation by removing one or several characters (Abbr), (3) using non-English or special characters (SpecialChar), (4) using mixed cased characters (MixedCase) and (5) using mixed cased characters with an additional layer of meaning (e.g., "republicans"→"repubLIEcans") (MixedCase+). Due to the imbalanced frequency distribution among different perturbation strategies, the ANTHRO algorithm (Le et al. 2022) is biased to a specific type of perturbation, such as RepeatChar. Hence, to increase the diversity of different perturbation types in NoisyHate dataset, we apply a pseudo-random strategy to give a higher chance for perturbation types with less frequency to be sampled. We utilized this procedure with ten random seeds and chose the best distribution entropy. This step resulted in 2,120 comments. Even though this reduced the total number of comments to a great extent, we prioritized diversity and balance across perturbation types, which are crucial for robust evaluation. The resulting subset offers a more representative and challenging benchmark for testing model resilience against various perturbations.

Step 3: Quality Assurance via Crowd-sourcing

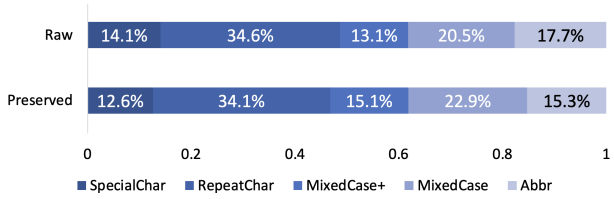
Training-Based Crowdsourcing. With the dataset of perturbed comments prepared in Section , we proceed to the human evaluation step that involves Amazon Mechanical Turk (MTurk) workers in judging the generated perturbation's quality. This step is to ensure that the select perturbations in the previous step (Section) are appropriate to the sentences' contexts. Before MTurk workers work on their tasks, we provide brief training to the workers by providing them with a few examples. To do this, we display a guideline to explain the definition of human-generated perturbation and provide examples of both high-quality and low-quality perturbations (Appendix, Table 5). This training phase has been suggested to warrant high-quality responses from the human worker, especially for labeling tasks (Clark et al. 2021). Each MTurk worker is then presented with a pair of perturbed sentences, its clean version, and asked to determine the quality of the perturbed one (Appendix, Fig. 4).

Recruitment and Compensation. We recruited five different workers from the North American region through five batches to assess each pair. A five-second countdown timer was also set for each task to ensure that workers spent enough time on it. To ensure the quality of their responses, we designed an attention question that asks them to click on the perturbed word in the given sentences before they provide their quality ratings (Appendix, Fig. 4). Workers who cannot correctly identify the perturbation's location in the given sentence will be blocked for future batches. We aimed to pay the workers at an average rate of $10 per hour, which is well above the federal minimum wage of $7.25 per hour. The payment of each task was estimated by the average length of the sentences, which totals around 25 words per pair, and the average reading speed of native speakers is around 228 words per minute (Trauzettel-Klosinski et al. 2012). Our study was approved by an institutional IRB board.

Quality Assurance. By removing tasks that failed to identify the location of the perturbed words accurately, the number of comments in the dataset was reduced from 2,120 to 1,707 examples. Subsequently, approximately 78.4% (1,339 examples) of the remaining data were deemed high-quality perturbations and retained for further analysis. Even though 22% of the data were removed by crowdsourcing workers and the number seems to be small, it still meets our expectations as all the perturbations were collected from the internet and our data should make more sense to humans. Although 1,339 appears to be a small size for a dataset, we intend NoisyHate to be a public benchmark test set. Hence, if we use the general train-test split ratio, 9:1, a size of 1,339 is comparable to the size of the test set of current popular data sets, such as OffensEval-2019 (13,240 examples) and SemEval-2021 (10,000 exam-



(a) Percentage of perturbation remained across all five perturbation types.



(b) Perturbation type distribution

Figure 2: Comparison of the distribution of different perturbation categories *before and after* validated by human workers (Section). "Raw" refers to the original data we send to MTurk workers, and "preserved" refers to the number/percentage saved after human evaluation.

ples). Fig. 2 presents the distribution of high-quality perturbations across five categories. MixedCase+ category (e.g., "republicans"→"repubLIEcans") had the highest retention rate of 88.8%. The generation of MixedCase+ perturbations typically requires human insights and understanding of outside contexts that are not accessible by machines, which makes them more acceptable to humans. In contrast, both abbreviation-based perturbations (Abbr) and those utilizing special characters (SpecialChar) result in character loss, making it more challenging for humans to associate the perturbed sentence with its clean version, resulting in a 30% discard rate.

Potential Uses

This section evaluates NoisyHate's dataset on two NLP tasks that correspond to research questions (RQs) as follows.

RQ. 1. (Perturbation Normalization) *How much can* NoisyHate*'s perturbed sentences be restored to their clean version by existing normalization algorithms such as misspelling correctors?*

RQ. 2. (Perturbation Understanding) *How effectively can* NoisyHate*'s perturbed sentences attack state-of-the-art deep-learning-based and commercial toxic detection models?*

Spell Corrector	Repeat-Char	Abbr	Special-Char	Mixed-Case	Mixed-Case+	Overall
Google	**0.698**	**0.751**	0.322	0.934	0.954	**0.755**
Bing API	0.406	0.533	0.426	0.943	**0.963**	0.633
pyspellchecker	0.544	0.480	**0.809**	**0.948**	0.954	0.728
NeuSpell	0.294	0.550	0.448	0.867	0.931	0.583

Table 3: Perturbation Normalization Accuracy

Perturbations Normalization

Experiment Set-up. To answer RQ. 1, we select one word-level spell corrector: pyspellchecker (Barrus 2022), one open source deep-learning-based misspell corrector: NeuSpell (Jayanthi, Pruthi, and Neubig 2020), and two commercial APIs: Google Search SerpApi (Google 2022) and Bing Spell Checker API (Microsoft 2022). We detailed these spelling correctors below.

- *pyspellchecker.* pyspellchecker is a word-level misspelling checker that returns the target word w's most likely permutations within a predefined Levenshtein Distance.
- *NeuSpell.* NeuSpell is an open-source toolkit for sentence-level spelling correction. It contains 10 language models, including CNN-LSTM, Nested-LSTM, and BERT. We test all models on our dataset and report the best performance.
- *Google Search SerpApi* Google Search provides "Did you mean?" suggestions when a user inputs a sentence with spelling errors in its search bar. SerpApi extracts these suggestions and provides them as a spell-checker API.
- *Bing Spell Checker API.* The Bing Spell Checker API utilizes statistical machine translation and ML algorithms to deliver contextual corrections. According to its website, it can recognize and normalize slang, informal language, and different words with the same sounds ("see" and "sea").

Experiment Results. Table 3 summarizes the accuracy of these spell checkers on NoisyHate. Although two of the five perturbing methods (MixedCase and MixedCase+) allow normalization accuracy close to or over 95%, at least two reasons suggest we retain those perturbations. First of all, blindly applying such normalization is unrealistic since this operation will reduce the toxicity score and even change toxicity polarity. In addition, BERT-based models are sensitive toward letter cases (we further discussed this in Section). By retaining MixedCase and MixedCase+, we are expecting novel hate speech detection models that can perform normalization as well as preserve the potential meaning of the perturbed word carrying (e.g., "rats" at "democRATs"). Overall, one of the commercial APIs, Google Search, presents the best result. Surprisingly, the word-level spell corrector, pyspellchecker, has the second-highest accuracy. One possible explanation for this observation is that, for an inputted non-English word, the pyspellchecker always offered its best guess even if the calculated probability of typos was low, while other tools remained conservative with a small confidence value. In practice, these spell checkers might en-

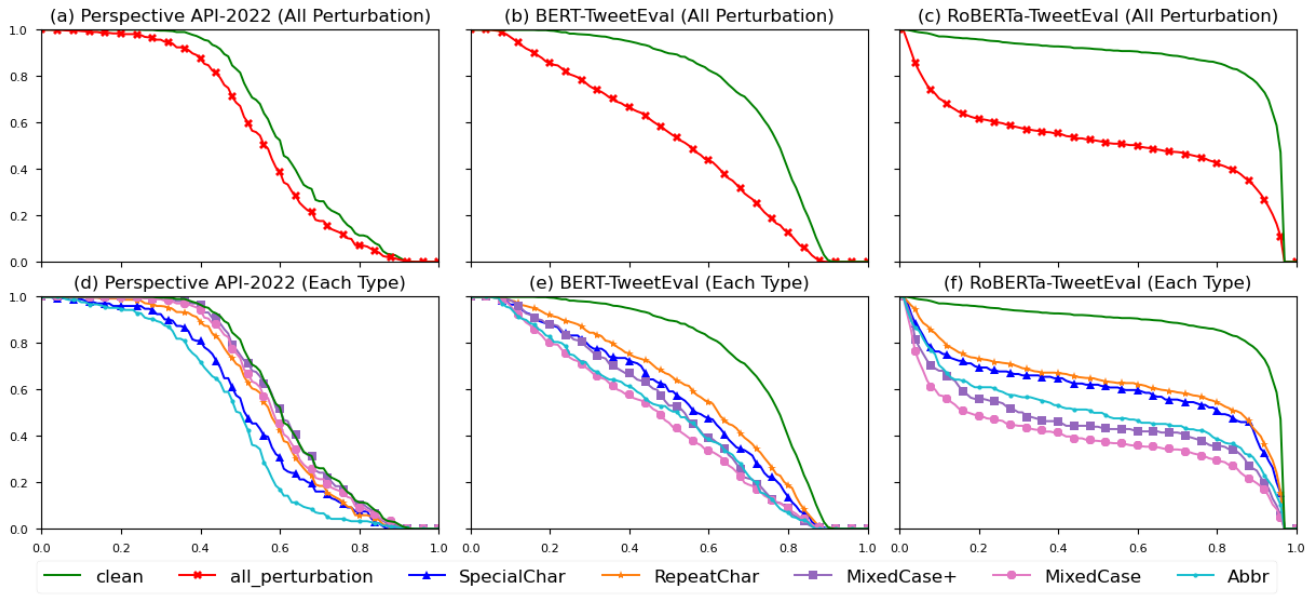


Figure 3: Model accuracy vs. threshold curves: in each chart, the x-axis represents the threshold, and the y-axis is the models' accuracy. In the top row charts, the all_perturbation curve represents the weighted mean accuracy, computed by combining the accuracy of each perturbation type weighted by its frequency.

Model		Repeat Char	Abbr	Special Char	Mixed Case	Mixed Case+	AllPert
Perspective API	Δ_{AUC}	0.047	0.138	0.092	0.025	0.009	0.057
	$\Delta_{ACC_{0.5}}$	-0.113	-0.314	-0.251	-0.078	-0.065	-0.142
BERT-TweetEval	Δ_{AUC}	0.149	0.248	0.188	0.267	0.222	0.151
	$\Delta_{ACC_{0.5}}$	-0.232	-0.383	-0.305	-0.44	-0.366	-0.339
RoBERTa-TweetEval	Δ_{AUC}	0.244	0.369	0.275	0.474	0.414	0.212
	$\Delta_{ACC_{0.5}}$	-0.272	-0.415	-0.294	-0.536	-0.475	-0.392

Table 4: Classification results of three hate speech detectors for each perturbation type. Δ_{AUC} is the gap area between the perturbed set and the clean set. $ACC_{0.5}$ is the classification accuracy at $t = 0.5$, and $\Delta_{ACC_{0.5}}$ refers to the degradation from clean accuracy by certain attacks.

counter more complex and novel perturbations. This result calls for continuous improvement of normalization tools for human-written perturbations online.

Perturbations Understanding

Experiment Set-up. For **RQ. 2.**, we tested the BERT model (Devlin et al. 2019), RoBERTa model (Liu et al. 2019) and also the Perspective API on NoisyHate dataset. Perspective API (Jigsaw 2022), created by the Jigsaw and Google team, is one of the most popular toxic content detection APIs.

We compare the performance of the detectors when evaluated on both clean and perturbed sentences. Since all of the data in our NoisyHate dataset are positive examples (i.e., toxicity$\geq$0.5 in the original Jigsaw dataset), the model's classification accuracy depends on a pre-defined decision threshold t. Typically, t=k means that the input will be considered toxic if the probability of a model's prediction for this data is larger than k, and vice versa. To better capture the performance differences among different threshold values, we plot the model accuracy vs. threshold curve as shown in Fig. 3. Meanwhile, two measures, Δ_{AUC} and $ACC_{0.5}$ were applied to evaluate the model's performance. Δ_{AUC} defines the gap between the area under the perturbed curve and the clean curve. The lower the Δ_{AUC} value is, the more robust a model is. $ACC_{0.5}$ refers to the model's classification accuracy when the threshold t is equal to 0.5.

Experiment Results. Table 4 summarizes the results. We observe that the RoBERTa-TweetEval model has the best performance ($ACC_{0.5}$=0.915) on the clean set while the Perspective API has the worst performance ($ACC_{0.5}$=0.814). However, the perspective API achieves the best overall robustness compared to other models. In the perturbed dataset, the accuracy of the Perspective API's classification when t=0.5 is 0.672, which is much higher than the second best model, the BERT model ($ACC_{0.5}$=0.557). In addition, the Perspective API attains the smallest gap area, Δ_{AUC}=0.057, between the perturbed set and the clean set (Table 4). This also indicates that the Perspective API is least affected by perturbations. Although the Perspective API and the Jigsaw dataset have been developed by the same research group, model overfitting is still the least of the concerns since it has the worst performance on the clean dataset. We tend to believe it's a trade-off the Perspective API, as a commercial API, made between robustness and accuracy.

Limitations

Our work has two main limitations. First, the toxicity of a sentence can evolve over time. For instance, words that may have been considered neutral several decades ago, such as "retarded" may be perceived as toxic in contemporary times. Also, people might generate more interesting and novel perturbations in the future and, hence will require updating our dataset continually. Second, within the limit of a conference paper, we strictly focus on introducing two potential uses: perturbation normalization and understanding. However, NoisyHate can also be used as additional training examples to improve existing toxic detection models via methods such as adversarial training. We foresee no potential risks of our dataset to the community and society.

Conclusions and Future Work

This work presents a novel data pipeline with a human-in-the-loop to collect and verify human-written perturbations to benchmark toxic text detection, called NOISYHATE, that consists of clean data and its corresponding perturbed version with online human-written text perturbations. The dataset allows researchers to evaluate the effectiveness of their proposed toxic content detection models in the face of real-world human-written perturbations instead of traditional machine-generated perturbations via existing adversarial text attacks. Furthermore, the diverse types of perturbations in the dataset pose a challenge for normalization algorithms, calling for better defenses to protect against potential adversarial behaviors online.

Acknowledgements

This work was in part supported by NSF awards #1934782 and #2131144.

References

Alzantot, M.; Sharma, Y.; Elgohary, A.; Ho, B.-J.; Srivastava, M.; and Chang, K.-W. 2018. Generating natural language adversarial examples. *arXiv preprint arXiv:1804.07998*.

at UC Berkeley, S. S. D. L. 2024. measuring-hate-speech (Revision c65a956).

Barbieri, F.; Camacho-Collados, J.; Espinosa-Anke, L.; and Neves, L. 2020. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*.

Barrus, T. 2022. pyspellchecker. https://pyspellchecker.readthedocs.io/en/latest/. Accessed: Apr 23, 2025.

Basile, V.; Bosco, C.; Fersini, E.; Nozza, D.; Patti, V.; Pardo, F. M. R.; Rosso, P.; and Sanguinetti, M. 2019. Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter. In *Proceedings of the 13th international workshop on semantic evaluation*, 54–63.

Bhalerao, R.; Al-Rubaie, M.; Bhaskar, A.; and Markov, I. 2022. Data-Driven Mitigation of Adversarial Text Perturbation. *arXiv preprint arXiv:2202.09483*.

Clark, E.; August, T.; Serrano, S.; Haduong, N.; Gururangan, S.; and Smith, N. A. 2021. All that's' human'is not gold: Evaluating human evaluation of generated text. *arXiv preprint arXiv:2107.00061*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT'19*, 4171–4186.

FORCE11. 2020. The FAIR Data principles. https://force11.org/info/the-fair-data-principles/. Accessed: Apr 23, 2025.

Gao, J.; Lanchantin, J.; Soffa, M. L.; and Qi, Y. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, 50–56. IEEE.

Garg, S.; and Ramakrishnan, G. 2020. BAE: BERT-based Adversarial Examples for Text Classification. *EMNLP'20*.

Gebru, T.; Morgenstern, J.; Vecchione, B.; Vaughan, J. W.; Wallach, H.; Iii, H. D.; and Crawford, K. 2021. Datasheets for datasets. *Communications of the ACM*, 64(12): 86–92.

Google. 2022. Google Spell Check API. Accessed: Apr 23, 2025.

Jayanthi, S. M.; Pruthi, D.; and Neubig, G. 2020. NeuSpell: A Neural Spelling Correction Toolkit. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 158–164. Online: Association for Computational Linguistics.

Jigsaw. 2022. Perspective API. https://perspectiveapi.com/. Accessed: Apr 23, 2025.

Jin, D.; Jin, Z.; Zhou, J. T.; and Szolovits, P. 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, 8018–8025.

Le, T.; Lee, J.; Yen, K.; Hu, Y.; and Lee, D. 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*, 2953–2965.

Le, T.; Yiran, Y.; Hu, Y.; and Lee, D. 2023. CRYPTEXT: Database and Interactive Toolkit of Human-Written Text Perturbations in the Wild. *arXiv preprint arXiv:2301.06494*.

Li, J.; Ji, S.; Du, T.; Li, B.; and Wang, T. 2018. TextBugger: Generating Adversarial Text Against Real-world Applications. *NDSS'18*.

Li, L.; Ma, R.; Guo, Q.; Xue, X.; and Qiu, X. 2020. Bert-attack: Adversarial attack against bert using bert. *EMNLP'20*.

Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; and Stoyanov, V. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Mathew, B.; Saha, P.; Yimam, S. M.; Biemann, C.; Goyal, P.; and Mukherjee, A. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 14867–14875.

Microsoft. 2022. Bing Spell Check API. https://www.microsoft.com/en-us/bing/apis/bing-spell-check-api/. Accessed: Apr 23, 2025.

Morris, J.; Lifland, E.; Yoo, J. Y.; Grigsby, J.; Jin, D.; and Qi, Y. 2020. TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP. In *EMNLP'19*.

Ren, S.; Deng, Y.; He, K.; and Che, W. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *ACL'19*, 1085–1097.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: long papers)*, 856–865.

Sato, M.; Suzuki, J.; Shindo, H.; and Matsumoto, Y. 2018. Interpretable adversarial perturbation in input embedding space for text. *arXiv preprint arXiv:1805.02917*.

Trauzettel-Klosinski, S.; Dietz, K.; Group, I. S.; et al. 2012. Standardized assessment of reading performance: The new international reading speed texts IReST. *Investigative ophthalmology & visual science*, 53(9): 5452–5461.

Yuan, X.; He, P.; Zhu, Q.; and Li, X. 2019. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on Neural Networks and Learning Systems*, 30(9): 2805–2824.

Zampieri, M.; Malmasi, S.; Nakov, P.; Rosenthal, S.; Farra, N.; and Kumar, R. 2019. Semeval-2019 task 6: Identifying and categorizing offensive language in social media (offenseval). *arXiv preprint arXiv:1903.08983*.

Zampieri, M.; Nakov, P.; Rosenthal, S.; Atanasova, P.; Karadzhov, G.; Mubarak, H.; Derczynski, L.; Pitenis, Z.; and Çöltekin, Ç. 2020. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). *arXiv preprint arXiv:2006.07235*.

Zang, Y.; Qi, F.; Yang, C.; Liu, Z.; Zhang, M.; Liu, Q.; and Sun, M. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *ACL'20*, 6066–6080.

Zeng, G.; Qi, F.; Zhou, Q.; Zhang, T.; Hou, B.; Zang, Y.; Liu, Z.; and Sun, M. 2021. Openattack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, 363–371.

Ethical Checklist

1. For most authors...
   (a) Would answering this research question advance science without violating social contracts, such as violating privacy norms, perpetuating unfair profiling, exacerbating the socio-economic divide, or implying disrespect to societies or cultures? Yes
   (b) Do your main claims in the abstract and introduction accurately reflect the paper's contributions and scope? Yes
   (c) Do you clarify how the proposed methodological approach is appropriate for the claims made? Yes
   (d) Do you clarify what are possible artifacts in the data used, given population-specific distributions? Yes
   (e) Did you describe the limitations of your work? Yes
   (f) Did you discuss any potential negative societal impacts of your work? Yes
   (g) Did you discuss any potential misuse of your work? Yes
   (h) Did you describe steps taken to prevent or mitigate potential negative outcomes of the research, such as data and model documentation, data anonymization, responsible release, access control, and the reproducibility of findings? Yes
   (i) Have you read the ethics review guidelines and ensured that your paper conforms to them? Yes
2. Additionally, if your study involves hypotheses testing...
   (a) Did you clearly state the assumptions underlying all theoretical results? NA
   (b) Have you provided justifications for all theoretical results? NA
   (c) Did you discuss competing hypotheses or theories that might challenge or complement your theoretical results? NA
   (d) Have you considered alternative mechanisms or explanations that might account for the same outcomes observed in your study? NA
   (e) Did you address potential biases or limitations in your theoretical framework? NA
   (f) Have you related your theoretical results to the existing literature in social science? NA
   (g) Did you discuss the implications of your theoretical results for policy, practice, or further research in the social science domain? NA
3. Additionally, if you are including theoretical proofs...
   (a) Did you state the full set of assumptions of all theoretical results? NA
   (b) Did you include complete proofs of all theoretical results? NA
4. Additionally, if you ran machine learning experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? Yes

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? NA

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? Yes

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? NA

(e) Do you justify how the proposed evaluation is sufficient and appropriate to the claims made? Yes

(f) Do you discuss what is "the cost" of misclassification and fault (in)tolerance? NA

5. Additionally, if you are using existing assets (e.g., code, data, models) or curating/releasing new assets, **without compromising anonymity**...

(a) If your work uses existing assets, did you cite the creators? Yes

(b) Did you mention the license of the assets? Yes

(c) Did you include any new assets in the supplemental material or as a URL? Yes

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? Yes

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? Yes

(f) If you are curating or releasing new datasets, did you discuss how you intend to make your datasets FAIR (see FORCE11 (2020))? Yes

(g) If you are curating or releasing new datasets, did you create a Datasheet for the Dataset (see Gebru et al. (2021))? No

6. Additionally, if you used crowdsourcing or conducted research with human subjects, **without compromising anonymity**...

(a) Did you include the full text of instructions given to participants and screenshots? Yes

(b) Did you describe any potential participant risks, with mentions of Institutional Review Board (IRB) approvals? Yes

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? Yes

(d) Did you discuss how data is stored, shared, and deidentified? Yes

Appendix

Section	Instruction
Definition	**What is a human written text perturbation?** A human written text perturbation is an alternative of a standard English word that is usually invented by netizens. The purpose of a perturbation can be: 1. avoiding platforms' filtering: ass → a55; Russian → Russ!an 2. visual attraction: bro → broooooo; porn → pornnnnnn 3. memes: republican + lie → repubLIEcan; democrat → democRAT
Examples	**Perturbations that are LIKELY to be written by human:** Example 1: Clean Sentence: keeping russian invaders warm in these frigid temperatures Perturbed Sentence: keeping russ!an invaders warm in these frigid temperatures Example 2: Clean Sentence: our country is a cesspool Perturbed Sentence: our countrrry is a cesspool **Perturbations that are UNLIKELY to be written by human:** Example 1: Clean Sentence: damn you beat me to it Perturbed Sentence: dawn you beat me to it Example 2: Clean Sentence: they were pretty cool to me when i was there Perturbed Sentence: they were pretty coll to me when i was there

Table 5: Guideline for MTurk Workers

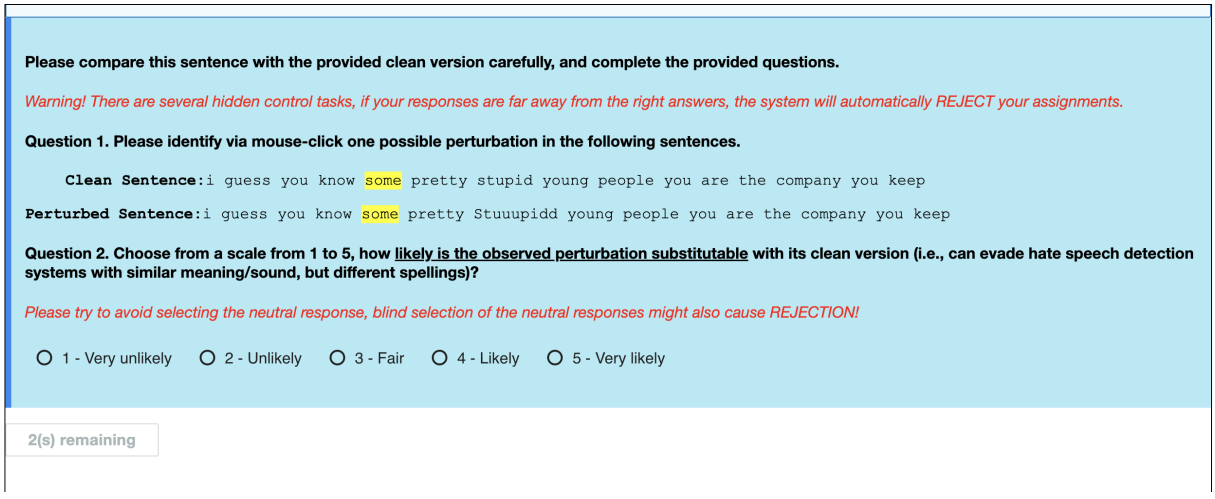


Figure 4: Human evaluation Interface: a clean-perturbed word pair will be highlighted when the worker moves the mouse cursor over one of them. By clicking the highlighted word, the worker commits that this is the identified clean-perturbed pair.