Adversarial Text Generation for Google’s Perspective API

Edwin Jain¹, Stephan Brown², Jeffrey Chen³, Erin Neaton⁴, Mohammad Baidas², Ziqian Dong², Huanying Gu², and N. Sertac Artan²

Abstract

With the preponderance of harassment and abuse, social media platforms and online discussion platforms seek to curb toxic comments. Google’s Perspective aims to help platforms classify toxic comments. We have created a pipeline to modify toxic comments to evade Perspective. This pipeline uses existing adversarial machine learning attacks to find the optimal perturbation which will evade the model. Since these attacks typically target images, as opposed to discrete text data, we include a process to generate text candidates from perturbed features and select candidates to retain syntactic similarity. We demonstrated that using a model with just 10,000 queries, changing three words in each comment evades Perspective 25% of the time, suggesting that building a surrogate model may not require many queries and a more robust approach is needed to improve the toxic comment classifier accuracy.

Keywords: Machine Learning; Adversarial; Google Perspective; Natural Language Processing; Deep Learning

1 Introduction

Social media platforms create an environment for people to learn about current events, interact with others, and freely share their own opinions. Online abuse and toxic comments can be daunting for already marginalized groups, limiting their ability to participate safely in online discussions. Labeling online abuse and harassment can validate harassment experiences, motivate bystanders to provide support, and help normalize appropriate user behavior [1]. Since online platforms are vulnerable to abuse, being able to create a tool for automatic detection of toxic or abusive online comments is valuable for online communities. It has potential to cut down on the cost of moderation for social media companies offering platforms with comment sections or forums, in order to make those online communities safer.

To tackle the problem of online harassment, Google and Jigsaw launched a project called Perspective which uses machine learning to perform text classification and rate the "toxicity" of comments [2]. Unlike traditional machine learning approaches, this task occurs in an adversarial setting; malicious users may try to bypass toxicity detection by adding minor alterations to their original comment.

With recent advances in the field of adversarial machine learning, such adversarial examples can be generated automatically, without access to or knowledge of the target model. Adversarial machine learning has conventionally focused on image classification, but some recent studies have explored natural language tasks in adversarial settings. Existing work has tried to shield Perspective through text preprocessing [3], which only works for simple obfuscation attacks. However, this approach does not guard against more systematic methods of generating adversarial text. Creating such schemes would allow platforms to train their models on adversarial examples, which has been shown to improve the robustness of classifiers [4].

In this paper, we propose a framework to generate adversarial text input and assess its effectiveness against Google’s Perspective toxic comment classifier. We designed a targeted attack scheme to make Perspective misclassify toxic comments as clean. We trained a surrogate model to emulate Perspective’s decision boundaries and applied existing attacks on this model. Then, we discretized the adversarial features to generate a list of text candidates. Finally, we selected the best possible candidates, by maximizing the classification flip while minimizing the semantic change.

The remainder of the paper is organized as follows. Section 2 provides background on some key concepts related to this project. Section 3 discusses the weaknesses of our target model, Google’s Perspective API, and some challenges to applying adversarial machine learning to text classification. Section 4 provides the detail processes of our proposed framework. Section 5 presents the evaluation of the effectiveness of our adversarial examples in fooling Perspective, and comparison of some variations on our proposed approach. Section 6 concludes our findings and discusses the future work of building a more robust model for online toxic com-
Adversarial ML Task Breakdown

<table>
<thead>
<tr>
<th>Properties</th>
<th>Our Problem</th>
<th>Opposite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack Surface: When is the model attacked?</td>
<td>Evasion: During inference phase</td>
<td>Poisoning: During training phase</td>
</tr>
<tr>
<td>Capabilities: What does the adversary know?</td>
<td>Black-box: No knowledge of architecture or parameters</td>
<td>White-box: Architecture and/or parameters</td>
</tr>
<tr>
<td>Adversarial Goals: What does the attacker want?</td>
<td>Targeted: Misclassify to a chosen output class</td>
<td>Untargeted: Cause any kind of misclassification</td>
</tr>
</tbody>
</table>

Table 1: A stylized typology of tasks in adversarial machine learning.

2 Background

In this section, we provide the background on some key concepts that this work encompasses.

2.1 Adversarial Machine Learning

The field of adversarial machine learning studies how to exploit and defend learning algorithms from adaptive, real-time attackers. Whereas machine learning (ML) models traditionally learn to optimize one stationary objective, in adversarial settings, algorithms must also learn to be robust to maliciously crafted inputs. Table 1 outlines the tasks from both the attackers and opponents point of view.

2.2 Adversarial Examples

In an evasion attack, the adversary tries to find an imperceptible perturbation \( \delta \) on some input \( x \) such that the classifier \( C \) (incorrectly) changes its prediction: \( C(x + \delta) \neq C(x) \). In a targeted evasion attack, the adversary aims to cause an input \( x \) originally labeled as \( l \) by the model to be classified as some class \( l' \): \( C(x + \delta) = l' \). Figure 1 shows an example of an adversarial attack on image classifiers. The attack does not have significant visual change in the image, however, it fools the classifier to categorize a panda image as a gibbon with 99% confidence.

2.3 Transferability

In a black-box setting (i.e. only query access), the adversary cannot directly use the traditional attacks since they require knowledge of the model’s loss function. A few proposed methods do approximate the model’s decision boundaries directly from query access [10, 11], but most research in the field has trained surrogate models and exploited transferability. By constructing a surrogate model using the target model’s outputs as synthetic training data, an attacker can simply apply a white-box attack to the surrogate and transfer the resulting adversarial examples to the target. Papernot et al. first demonstrated transferability, showing that the assumption holds across not only different neural network architectures, but also across completely different machine learning models (e.g. logistic regression, k-nearest neighbors, SVMs) [12, 13]. Moreover, adversarial examples can be transferred to force the appropriate targeted misclassification [14]. The strongest defense mechanism found has been to simply train models on adversarial examples, which boosts robustness while retaining accuracy on clean examples [15].

3 Related Work

3.1 Google’s Perspective

Perspective API is a toxic comment classifier created by Jigsaw and the Google Counter-Abuse Technology Team to help improve online conversations [2]. Their model assigns a score between 0 and 1 to comments, where 1 means the most toxic. In the past, Hosseini et al. had shown that Perspective misclassifies toxic comments with simple text obfuscation (e.g. inserting punctuation or spaces, doubling letters) [3]. However, they acknowledged that it could easily be defended against via text preprocessing, which Rodriguez et al. found only increases processing time by a factor of two [16]. Perspective is not as easily fooled by such attacks anymore, though we do not know whether its robustness is a result of text preprocessing or adversarial training.
Previous research in adversarial text generation has mostly focused on evaluating methods on movie or restaurant review datasets [17]. Perspective allows for a more realistic attack scenario. Our goal is to fool Perspective into rating toxic comments as non-toxic, allowing online abuse to evade detection.

### 3.2 Adversarial Text Generation

While most research in adversarial machine learning has focused on images, there has been some nascent work in applying similar techniques to text data. Papernot et al. use gradient-based attacks to perturb some random input features and then substitute the closest word vector in the embedding [18]. Liang et al. find the words most relevant to each class label and propose insertion and removal strategies alongside word substitution [19]. Gao et al. score words by their relevance similar to Liang, but perform insertion, removal, and swapping transformations on characters [20]. Their proposed DeepWordBug algorithm presumes black-box access and demonstrates that adversarial text examples are also transferable across different model architectures.

These three methods for generating adversarial text lack a crucial check for human readability. Although the "jumbled word effect" suggests that humans can read typo-ridden text fairly easily, these perturbations must preserve important orthographic properties to actually remain readable [21, 22]. Moreover, these methods could change the semantic meaning and the true label of the text, thereby failing to actually fool the classifier. Samanta et al. limit this possibility by checking for syntactic similarity and adding linguistic requirements for substituted or inserted words [17].

### 3.3 Text is Challenging

Compared to other domains for adversarial machine learning (e.g. images, audio), text data poses some unique challenges:

1. **Text is discrete, not continuous.**
   
   Technically, pixels in a valid image must be integer values, but the discretization process is as simple as rounding. Discretizing some random vector into a word, on the other hand, is more difficult.

2. **Small changes to text can significantly impact meaning.**
   
   Slightly changing all of the pixels in an image is unlikely to change its label for humans, but this is obviously not the case for text. Furthermore, whereas one-pixel perturbations are almost guaranteed to be imperceptible [23], changing one word can actually alter sentiment. For example, with the phrase "I hate people" may earn a .90 toxicity rating on Google, if you perturb the most important word in the sentence, slightly, you might get, "I look people" with a .04 toxicity rating. While this may have successfully flipped the classification, the semantic meaning is now completely lost.

3. **Language is dynamic.**
   
   While our idea of what a panda looks like is unlikely to change over time, the same is not true of language [24]. Aside from normal shifts in language due to new slang or technology, adversarial settings for text may encourage some to coin new phrases and use coded language to evade classification [25].

### 4 Proposed Approach

In this section we present our approach for generating adversarial examples against Perspective as outlined in Figure 2.

![Our Pipeline for Generating Adversarial Text Examples](image)

**Figure 2: Our pipeline for generating adversarial text examples.**

First, we constructed synthetic training data by using Facebook comments that were posted in response to popular news articles [26]. The dataset contains a total of 1,025,403 comments from 19,850 posts. We cleaned the comments by removing empty comments and URLs, queried them using the Perspective API, and retrieved corresponding toxicity values for each comment. Figure 3 shows the frequency and cumulative distribution of the word count of the cleaned dataset.

We trained a surrogate model on a subset of the labeled comments. We chose word-level convolutional neural networks for our architecture, since they perform well in text classification [27] and can be attacked more easily than recurrent neural networks [18].
Figure 3: Frequency and cumulative distributions by word count.

We used the Carlini-Wagner attack implemented in Clev-erhans v.2.1 on our surrogate model to generate adversarial features [28].

Since the objective of our work is to minimize the number of words to be changed, we selected only those words whose features have been most perturbed (by $L_2$ norm).

The perturbed features do not necessarily correspond to words, so they must be discretized. We used neighborhood graph and tree indexing (NGT) [29] to find the vectors’ nearest neighbors to generate the list of text candidates.

Once candidate sentences have been generated, we chose the input that fools the classifier while retaining syntactic and semantic similarity. This was implemented using SpaCy v.2.0.11, an open-source natural-language processing library [30].

Finally, we evaluated our adversarial examples by querying Perspective API. We perturbed comments initially rated by Perspective as over 0.75, and consider an adversarial example successful if the perturbed comments get a below 0.5 rating (i.e. the comment was misclassified).

4.1 Discretization

Our attack alters multiple elements within vectors such that the resulting sequence is unlikely to directly correspond to a word in our embedding. In order to create plausible approximations of this sequence using real words, the nearest neighbors for each given vector were adopted. Using this method, we generated different candidates to consider for replacement. However, a large number of words in the embedding is challenging to efficiently find the true nearest neighbors for a given vector. Furthermore, there is little guarantee that the absolute nearest neighbor of an attack vector corresponds to the best replacement word. For our purpose we decided to sacrifice some degree of accuracy and used a nearest neighbors approximation algorithm for the sake of efficiency.

We use Neighborhood Graph and Tree for Indexing, a library released from Yahoo! JAPAN, to perform approximate nearest neighbor searches [29]. Here, we only considered the words in the sentence that are evaluated to have the most significant impact on classification. We then replaced those words with the nearest neighbors to their corresponding vectors in the attack matrix, leaving all other words unchanged from the original comment. For a given sentence, we used this method to generate a number of different attack sentences. The number of words to replace and the number of different neighbors to replace with can be arbitrarily defined.

4.2 Selection

After a list of candidate sentences is generated, we must select among them the ideal candidates which maximize the classification flip while maintaining the original meaning of the sentence.

We first evaluated classification flip by querying our surrogate model with the candidate sentences and identified those that do not fool it.

We utilized the SpaCy library to test syntactic similarity. We input the candidates with successful classification flip as tokens for spaCy to determine syntactic similarity based on a pre-trained model.

Note that SpaCy is not an adequate tool for measuring semantic similarity for longer phrases. A sentence’s meaning can change significantly even while its grammatical structure may stay the same, yet SpaCy cannot detect this disparity with longer sentences.

To achieve stronger semantic similarity for sentences, as future work, we plan to use Sent2Vec encoders in our selection process. This approach, also known as skip-thought vectors, would allow us to capture the context of an entire sentence and evaluate semantic similarity more effectively [31].

5 Evaluation and Results

To evaluate the effectiveness of our attack, we created a large number of adversarial examples and observed the percentage of examples that cause the Perspective API to produce a misclassification. The adversarial examples were generated
only from sentences that the API originally assigns a score of at least 0.75, and we considered the API to have misclassified an adversarial example when it assigns the example a score of less than 0.5. Our accuracy is the the proportion of misclassified examples, which we aim to maximize.

Figure 4: Changes in accuracy of Perspective API model with varying edit distance. Surrogate model training set size is held constant at 100,000.

Figure 5: Changes in accuracy Perspective API model with varying numbers of candidates for selection. Surrogate model training set size is held constant at 100,000.

We ran and evaluated multiple trials while varying the size of our surrogate model’s training dataset and the number of words we substitute in each adversarial example. Figure 4 shows the effectiveness of our attack with respect to the number of words we substitute into our adversarial examples. As one would expect, the proportion of misclassifications increases as the edit distance increases. In Figure 5, we observe that the attack’s effectiveness stays constant with respect to the number of nearest neighbors searched per word in the discretization process. As Figure 6 indicates, our attack appears to reach its full potential even with a small training dataset: the accuracy of the Perspective API on our adversarial examples stays relatively constant when our surrogate model is trained on 10,000, 50,000, and 100,000 comments. The largest drop in accuracy that we achieved was about 0.35, where we were able to cause the Perspective API model to misclassify 35 percent of the adversarial examples in a single trial. This proportion was achieved using an edit distance of 4 between the original sentences and the adversarial examples, as well as a surrogate model training set size of 100,000 sentences. However, our results in Figure 6 imply that we can achieve a similar drop in accuracy with a significantly smaller training set.

6 Conclusions

In this paper, we have introduced an adversarial machine learning attack on Google’s Perspective API for classifying toxic comments. We showed that the classifier can be deceived by replacing words in toxic sentences while still preserving the original meaning. Our attack exploited transferability of adversarial examples to successfully carry out an evasion attack on the Perspective API with little knowledge of the model. We showed that the number of queries made to Perspectives model did not have a large effect on the accuracy of the model. On the other hand Perspectives accuracy decreases proportionally with edit distance, and remained relatively constant with an increase in the number of neighbors searched.

As future work, we plan to expand on our testing methods and evaluate the legibility of our adversarial examples. We would like to experiment with methods for boosting semantic similarity like part of speech tagging and Sent2Vec encoding. It would be of interest to evaluate character level substitution and test the approach on other types of neural networks.
References


[25] N. Sonnad, “Alt-right trolls are using these code words for racial slurs online,” 2016.


