Smart Censoring: A Transformer-Based Detoxifier for Toxic Sentences

David Simonetti and John lee

December 2023

1 Introduction

1.1 Goal and Benefit for Society

Since the inception of the internet, toxic comments have proliferated on the web. Chat rooms, social media, and online video games have always been flooded with this hateful and malicious content. In order to make the internet a less toxic place, we wish to create a natural language processing model which is able combat online toxicity. We envision the model taking in arbitrary chats from users, which may contain obscenities, racism, sexism, etc, and outputting a sanitized text which will communicate the same idea without all the hate. The idea is that other companies can plug this model into their online communication systems. For example, user chats in a video game match would pass through this model before being sent to other players. This would make interacting with others online a much less negative experience than before. We hope that this would foster community and positive sentiment online as a result. As a trigger warning, this report does contain some profanity and explicit comments.

1.2 User interaction

We envision our model being used in two ways. One way would be a soft censorship model that simply gives suggestions to users that are less toxic than their original messages. It would work as follows. When a user types a message on some platform and hits send, if a high level of toxicity is detected in the message, the user would be blocked from sending the message. Additionally, the users' message would be passed through our model to generate a detoxified version that hopefully conveys the same ideas as the original. This sanitized version of the message would be presented to the user as a suggestion for rephrasing their original message. The user would hopefully follow the suggestion, leading to our goal of less toxicity on the internet. This could also help users step back from an emotional outburst that causes them to write a negative message in the first place and help them reflect on what they truly wish to post to other people.

The second way we envision our model used is the following. Users can send and post all the toxic messages they want, but two different versions of the content that are flagged as harmful will exist. There will be the original text and a censored version created by our model. To the author, they will only ever see the original versions and not know anything is up. However, anyone viewing the message (unless they specifically opt into viewing harmful and toxic content) will only see the censored version. In this way, we have created a more heavy-handed approach to ensuring that toxic content cannot proliferate on the internet. The people who wish to see the dark and uncensored users on the internet still can, but individuals who may not want to be exposed to that sort of content are protected. We would like to acknowledge the ethical concerns of rephrasing an individual's words, but the primary focus of our study is the feasibility of doing so.

1.3 Existing Approaches

A variety of approaches have been taken to solve the problem of toxicity. Perhaps the oldest and most famous approach is the naughty word filter. An organization or company that wishes to ensure that the worst and most profane of content is not allowed on their platform would create (or find online) a list of words to filter and remove from all content on the site. For example, all curse words could be banned, common slurs, words that promote hate against a particular group of people, etc. All user-generated content on that site would then be passed through this filter, and their toxic content would hopefully be removed or censored (typically by asterisks). One major drawback of this approach is how easy it is to bypass these filters. For example, the filtered words might only contain ”fuck” and ”Fuck”, so a crafty user could type in ”f*ck” or ”fUcK” and sneak through the toxicity filter. There are theoretically infinite permutations of structuring profanity that can be used to generate harmful text, so the drawback of this approach is it is very difficult to generate a comprehensive and sophisticated filtering system.

Another approach to combatting online toxicity is the use of natural language processing (NLP) models to flag potentially toxic and harmful content automatically. For example, the model Toxic Bert attempts to tackle this challenge. For any given input, the model assigns a value between 0 and 1 for the following categories: toxic, severely toxic, obscene, threatening, insulting, or hateful of identity. Based on these values that the model assigns, one can then automatically determine whether a given piece of text contains harmful content based on some threshold value. The disadvantage of this approach is that the model only detects negative content but lacks the capability to modify such content. The only recourse to finding out a user post is flagged for toxicity is to take it down or ask the user to rephrase their content.

2 Methodology

2.1 Proposed Approach

We propose a Transformer-based natural language processing model, SmartCensor, that ”translates” toxic sentences into equivalent non-toxic versions of the same sentence. SmartCensor can be paired with toxicity detection schemes such as ToxicBert to filter and generate cleaner outputs. We use an Encoder-Decoder architecture for the model itself, and train the task as an instance of supervised machine translation. The dataset we train on is parallel text that consists of toxic sentences and a reference ”detoxified translation” of the same sentence. Model evaluations are conducted on data from the internet known to be toxic, observing how capable our model is of removing the toxicity from that data.

2.2 Data and Preprocessing

We used three datasets in the training and evaluation of our model
1) ParaDetox – **Parallel Detoxification Data**

This is a dataset of 19,700 entries consisting of an original toxic sentence and a detoxified version of the same sentence intended to have the same meaning. We used this dataset during the training of our model, i.e. our model was trained to translate a toxic sentence into a detoxified sentence. Twenty percent of this dataset as a validation set, verifying the BLEU score of the model to confirm that training was progressing.

Example. Original toxic sentence: ”Obama has been a total failure, and now looks like a sore loser.” → Detoxified sentence: ”Obama has not been victorious.”

2) Jigsaw - **Wikipedia Toxic Comment Dataset**

This dataset consists of comments taken from Wikipedia that human moderators have labeled as one of the following: toxic, severely toxic, obscene, threatening, insulting, or hateful of identity. We use roughly 16k test samples from this dataset to evaluate both a baseline and our trained model.

Example. Sentence in data: ”Chris, you mother fucker...all what you want to know about ChrisO you can find at www.ChrisO.homo.com” → Assigned labels: toxic, obscene, insult

3) Toxic Word Bank - **Dictionary of Toxic Words**

This dataset is a curated list of known toxic words. It contains pure curse words, slurs, common slang, and abbreviations of words typically used to harm others. We use this in our baseline to censor the toxic content of the sentence by searching for and removing all words in this word bank.

2.3 Baseline

To compare our solution to existing methodologies for censoring, we evaluate a baseline filtering method. Using the toxic word bank in the prior section (with some minor edits for parsing), toxic words identified in a sentence are replaced with an empty string. A sample replacement looks like such:

Original: Please stop being a penis— and Grow Up Regards-
Replaced: Please stop being a - and Grow Up Regards-

In this sample, we observe the major flaw in such approaches. Although the sentence is largely preserved on a word-by-word basis, it is quite evident that the meaning and fluency of the sentence have been lost. Furthermore, even though ”penis” has been removed, the resulting sentence still has underlying tones of toxicity, which cannot be removed via such approaches.

2.4 SmartCensor Architecture and Training

This section describes the detailed architecture and training steps used to create SmartCensor. Many choices were made heuristically or with consideration to resource limitations and thus may not be optimal.

Encoder – We use a Transformer architecture for encoding. Words are first converted to word and positional embeddings (max length = 64) with a dimension of 256. The encoding layers consist of four alternating multi-head self-attention layers and residually connected feed forward networks (FFN). The multi-head attention layers have four heads each and transform 256 dimensions into 256 dimensions. The FFNs have 256 input and output neurons with 1024 hidden neurons.

Decoder – The decoder also creates word and position embeddings of input sentences like the encoder. These embeddings are fed through a masked self-attention layer with 256 dimensions and an FFN with the same parameters as the Encoder. The output of these embeddings, in combination with the Encoder output, is fed through a cross-attention layer of 256 dimensions and then to a softmax layer to retrieve "probability" scores for each word.

Tokenizing Our model uses an off the shelf tokenizer for processing input data. All input data is first passed through the NLTK tokenizer before being sent to the model. This is to help the model understand and digest the complex input data we are giving it.

Training – Using the ParaDetox dataset, we trained our model for 10 epochs, using an Adam optimizer with a learning rate of 0.0003. The loss is the negative log probability that the generated sentence is the translation of the input sentence. The input data is tokenized before being sent to the model for translation. The model was saved when the validation BLEU score was the highest.

Translation Generation - The way in which we actually generate translations from toxic to detoxified sentences using our model is using beam search with k = 4. We perform beam search as follows. First, we use the encoder to compute the encoding vector of a given sentence. Then, we compute the 4 most likely words to come after BOS in the translation, and store the probabilities of each of those words along with the state of the decoder. These are our 4 starting translations for our beam search. For every remaining word possible before we hit our cap of 64 total tokens outputted, we create the next 4 most likely words for each of our 4 current sentences, resulting in 16 total sentences after each step. We then sort them by probability and throw our the 12 least likely of the 16. Additionally, if a sentence ends in EOS, we don't generate the next 4 most likely words but instead just tack on the same sentence back into the beam search list. After we have completed the entire search and end up with the 4 most likely translations of our original sentence, we randomly select from the 4 to produce our final translation. We did this in order to make the model a little less predictable and to have it produce more interesting outputs from time to time.

There is also a consideration of the max output length of 64 hampering the translation ability of the model. We consider the following solution to the problem. If there is any max output size for the model, a bad actor will be able to get around the filtering and potentially break the model by using a very large input. We propose to chop up all inputs in 64 token chunks and process them individually so that this "hacking" of the model no longer poses a threat.

2.5 Evaluation Metrics

We used two metrics to evaluate out-of-sample performance. One measures toxicity levels, and the other roughly approximates meaning preservation.

To measure toxicity, we feed both a toxic and detoxified input to Toxic Bert (mentioned above). For any given input, this model assigns a value between 0 and 1 for the following categories: toxic, severely toxic, obscene, threatening, insulting, or hateful of identity. We then compare toxicity levels for both inputs, computing the reduction in toxicity.

There is no precise, well-established measurement for meaning preservation. Initially, we considered metrics such as COMET or BLEU, but they tend to require strict matching with a provided translation "truth". However, we evaluate on a dataset without readily available detoxified translations, and attempting to compare to the toxic versions is flawed because the removal of specific toxic words that might have been deemed important (such as COMET does) could drastically impact scoring.

Instead, we approximate meaning preservation by feeding inputs to Sentence Transformers, specifically the all-MiniLM-L6-v2 model, generating vector embeddings of the sentence, and computing a cosine similarity. In principle, the vector embedding should represent the actual meaning of the sentence (sentences with similar meanings should be close within the embedding space). Thus, a high cosine similarity should indicate the two sentences have similar meanings, and the model is preserving meanings.

2.6 Github

All written code, as well as dataset download links and evaluation models can be found at https://github.com/JohnDLee/SmartCensor.

3 Results

3.1 Baseline evaluation

We obtain baseline results after processing Jigsaw data, performing baseline filtering, and evaluating performance as described above. An individual sample is provided below.

Toxicity measure of original (Please stop being a penis— and Grow Up Regards-):

{toxic: 0.955, obscene: 0.688, insult: 0.608, severe_toxic: 0.021, identity_hate: 0.033, threat: 0.003.}

Toxicity measure of censored output (Please stop being a — and Grow Up Regards-):

{toxic: 0.266, obscene: 0.003, insult: 0.022, severe_toxic: 0.000, identity_hate: 0.001, threat: 0.000.}

Cosine Similarity of Original and Censored: .4054

Over the entire baseline test set, we computed the average *reductions* in toxicity and the average cosine similarity. We also took two sets of averages: first, a global average, which includes all non-toxic phrases (results in Table 1), and second, only averaging phrases that were human-labeled toxic (results in Table 2).

3.2 SmartCensor Results

SmartCensor was trained on the ParaDetox dataset and then evaluated out of sample on the Jigsaw dataset. One good translation and one poor translation of the ParaDetox translation and Jigsaw translation are shown below.

ParaDetox (In-Sample)

Original: They treat you like sh*t.
SmartCensor: They treat you badly.
Ground Truth: They mistreat you.

Original: urban warfare is a b*tch.
SmartCensor: It is a bad person
Ground Truth: urban warfare is bad

Jigsaw (Out-of-Sample)

Original: f*ck you cr*cker nerd
Baseline: you er nerd
SmartCensor: I don't like you

Original: MARK SWEEP IS A W*NKER
Baseline: MARK SWEEP IS A ER
SmartCensor: It is not good

The evaluation results of our SmartCensor model are also reported in Tables 1 and 2.

Table 1: Global Average Reductions in Toxicity and Average Cosine Similarity

Metrics	**Global**	**Baseline Reduction**	**SmartCensor Reduction**
Toxicity	0.1022	-0.0250	-0.0615
Severe Toxicity	0.0152	-0.0111	-0.0148
Obscenity	0.0644	-0.0400	-0.0613
Threat	0.0065	-0.0032	-0.0046
Insult	0.0614	-0.0344	-0.0553
Identity Attack	0.0128	-0.0072	-0.0112
Cosine Similarity	N/A	0.7103	0.3733

Table 2: Toxic-Only Average Reductions in Toxicity and Average Cosine Similarity

Metrics	**Toxic-Only**	**Baseline Reduction**	**SmartCensor Reduction**
Toxicity	0.8661	-0.3696	-0.6895
Severe Toxicity	0.1437	-0.1061	-0.1411
Obscenity	0.5926	-0.3857	-0.5775
Threat	0.0598	-0.0302	-0.0439
Insult	0.5705	-0.3400	-0.5334
Identity Attack	0.1183	-0.0719	-0.1054
Cosine Similarity	N/A	0.7096	0.3928

3.3 Comparison of Baseline and SmartCensor

Despite the training data containing less rigorous examples of toxicity, SmartCensor demonstrates that it performs significantly better at removing toxicity than just a straight filter across all metrics. However, not all reductions in toxicity can be attributed to the model itself. From a natural language processing perspective, if SmartCensor encounters an unknown toxic word, it will be automatically classified as $< UNK >$ Since the output "translation language" of SmartCensor should contain no toxicity whatsoever, the resulting output will not be toxic but may not preserve meaning or even be related to the original sentence. For example, a sentence of $< UNK >< UNK >< UNK >< UNK >$ will not be predicted properly at all. The lower cosine similarity somewhat confirms this theory.

However, we mentioned that cosine similarity itself is a flawed metric. For example, in the example "f*ck you cr*cker nerd," SmartCensor accurately translates to the relatively gentler "I don't like you." Even within vector space, these two sentence embeddings may not be similar. Additionally, comparing the baseline censor "you er nerd" to SmartCensor, it is clear that SmartCensor provides sentences with much better coherency.

3.4 Toxifier

For curiosity's sake, as well as to address limitations in data mentioned in the next section, we train a Toxifier by swapping the input and output translations of ParaDetox. The Toxifier should add toxicity to an input sentence. We evaluated the Toxifier over the *non-toxic* comments within the Jigsaw test data, measuring average increases in toxicity in Table 3. Several examples are given below.

Original: So its bias is actually relative to that of its readers?

Toxified: so its society is actually money to that of its arrogant jerk? oh shit, and shut the fuck up

Original: Stop changing genres without discussion.

Toxified: shut the fuck up without discussion

Original: Leave the lyrics in-they are half the article for one of the best songs ever. Tommy

Toxified: get the fuck out the article for the article for one of the best songs ever. holy shit, he is able to be there?

Original: How do you know that? It was very long ago. Maybe named for the Gaya confederacy.

Toxified: oh how the fuck do you know that? it was so fucking long ago. maybe else for the other crap.

Table 3: Average Increase in Toxicity

Metrics	**Toxifier Increase**
Toxicity	0.8647
Severe Toxicity	0.0849
Obscenity	0.8171
Threat	0.0111
Insult	0.4189
Identity Attack	0.0190
Cosine Similarity	0.3450

4 Conclusion

Our study has many limitations and room for improvement that are worth mentioning. First, our model architecture was significantly limited by available computational resources. Larger architectures may certainly improve performance. However, architecture size is also limited by the availability of data. ParaDetox was the only corpus of parallel detoxified data available after a rigorous search. We propose a solution to this problem. In this study, we trained a Toxifier by swapping input and output translations. We can feed non-toxic sentences to the Toxifier, toxify them, and generate a much larger corpus of parallel data.

A third limitation involves the vocabulary size of SmartCensor. Paradetox does not contain a comprehensive vocabulary set, resulting in less precise and bland sentences. Additionally, this lack of vocabulary also impacts our solution to increasing dataset size, as outputs of the Toxifier are also limited to an existing set of vocabulary. By increasing the vocabulary, fewer out-of-sample words may be considered unknown, increasing performance, and the sentence structure and outputs will also be more varied. We propose two alternative solutions to the problem. The first is implementing some residual structure or labeling method, so input words can be directly passed to the output rather than going through the model. However, this may allow certain unknown toxic words to pass through, reducing SmartCensor's effectiveness. The second solution is adding non-toxic training data whose translation is itself, thereby expanding vocabulary.

Lastly, evaluation metrics for SmartCensor are not precise. A better metric for sentence meaning preservation and coherency should be introduced in order to better quantify SmartCensor's performance over baseline methods.

Currently, SmartCensor demonstrates significant promise as an alternative to typical filtering methods, being capable of removing the underlying toxic intention. However, as a prototype, SmartCensor still lacks the precision, accuracy, and quality to be used in actual applications. Doing so may result in ethical problems and user frustration. Implementing and overcoming our noted limitations should bring SmartCensor closer to actual application.

A Reproducability

Provided are the links to various resources used in our project.

Datasets:
ParaDetox - https://huggingface.co/datasets/s-nlp/paradetox
Jigsaw - https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/data?select=test.csv.zip
Toxic Word Bank - https://github.com/Orthrus-Lexicon/Toxic/tree/main
Evaluation Metrics:
Toxic Bert - https://huggingface.co/unitary/toxic-bert
Sentence Transformer - https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
Code:
Github Repository - https://github.com/JohnDLee/SmartCensor
NLTK Tokenizer Package - https://www.nltk.org/api/nltk.tokenize.html