A First Look at Toxicity Injection Attacks on Open-domain Chatbots

Connor Weeks*
Virginia Tech
crweeks@vt.edu

Aravind Cheruvu*
Virginia Tech
acheruvu@vt.edu

Sifat Muhammad Abdullah
Virginia Tech
sifat@vt.edu

Shravya Kanchi
Virginia Tech
shravya@vt.edu

Danfeng (Daphne) Yao
Virginia Tech
danfeng@vt.edu

Bimal Viswanath
Virginia Tech
vbimal@vt.edu

ABSTRACT

Chatbot systems have improved significantly because of the advances made in language modeling. These machine learning systems follow an end-to-end data-driven learning paradigm and are trained on large conversational datasets. Imperfections or harmful biases in the training datasets can cause the models to learn toxic behavior, and thereby expose their users to harmful responses. Prior work has focused on measuring the inherent toxicity of such chatbots, by devising queries that are more likely to produce toxic responses. In this work, we ask the question: *How easy or hard is it to inject toxicity into a chatbot after deployment?* We study this in a practical scenario known as Dialog-based Learning (DBL), where a chatbot is periodically trained on recent conversations with its users after deployment. A DBL setting can be exploited to poison the training dataset for each training cycle. Our attacks would allow an adversary to manipulate the degree of toxicity in a model and also enable control over what type of queries can trigger a toxic response. Our fully automated attacks only require LLM-based software agents masquerading as (malicious) users to inject high levels of toxicity. We systematically explore the vulnerability of popular chatbot pipelines to this threat. Lastly, we show that several existing toxicity mitigation strategies (designed for chatbots) can be significantly weakened by adaptive attackers.

CCS CONCEPTS

• **Security and privacy**; • **Computing methodologies** → **Machine learning**;

KEYWORDS

Chatbots, data poisoning, toxicity injection and detection, adversarial inputs

ACM Reference Format:
Connor Weeks, Aravind Cheruvu, Sifat Muhammad Abdullah, Shravya Kanchi, Danfeng (Daphne) Yao, and Bimal Viswanath. 2023. A First Look at Toxicity Injection Attacks on Open-domain Chatbots. In *Annual Computer Security Applications Conference (ACSAC '23), December 04–08, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3627106.3627122

*Both authors contributed equally to this research.

ACSAC '23, December 04–08, 2023, Austin, TX, USA

ACM ISBN 979-8-4007-0886-2/23/12.
https://doi.org/10.1145/3627106.3627122

1 INTRODUCTION

Dialog systems or *chatbots* have immensely improved over recent years by leveraging advances made in the development of Large Language Models (LLMs). Chatbots are created today using an *end-to-end data-driven* learning paradigm, where a DNN model is trained on a conversational dataset to create a chatbot. This technology replaces the previous generation of rule-based chatbots [28]. Given a conversational history or *context* of the last few turns of a conversation, a chatbot can now produce a contextually relevant utterance by learning from existing conversation data. This data-driven paradigm has enabled *open-domain chatbots* that can generate conversations on a wide range of topics [58]. *However, a fundamental limitation is that any problematic biases or imperfections in the training data can lead to undesired utterances by a chatbot. In this work, we focus on the problem of toxic conversations by a chatbot.*

We use the terminology 'toxic language' to refer to any language that is harmful to a user. A toxic chatbot would produce toxic language conversations, causing emotional harm to its users. Today, chatbots are being built, made publicly available and deployed without fully understanding how toxicity can be learned by chatbots [19, 53, 68]. HuggingFace [4] is a popular open-source platform that hosts over 2,100 user-uploaded chatbot models. Any harmful language learned by these models is exposed to millions of users through their global deployments. Furthermore, the application of chatbots in sensitive domains exposes this (yet) unsafe technology to vulnerable users—in healthcare [15] as virtual agents for companionship and support [42], in the U.S. justice system, as responders to case queries [5].

Prior work. Existing work focuses on *measuring* toxicity of open-domain chatbots [7, 55]. They study the toxicity of popular chatbot pipelines, likely resulting from the uncurated human-human conversation datasets used to train them. As shown in Figure 1 (a), their focus is primarily on crafting "adversarial" queries, that can be themselves toxic or non-toxic, to elicit toxic responses. These studies do not consider an adversary who can manipulate and control the level of toxicity exposed to benign users, as the focus is only on measuring toxicity. It is also unclear how benign users can be harmed using such "adversarial" queries.

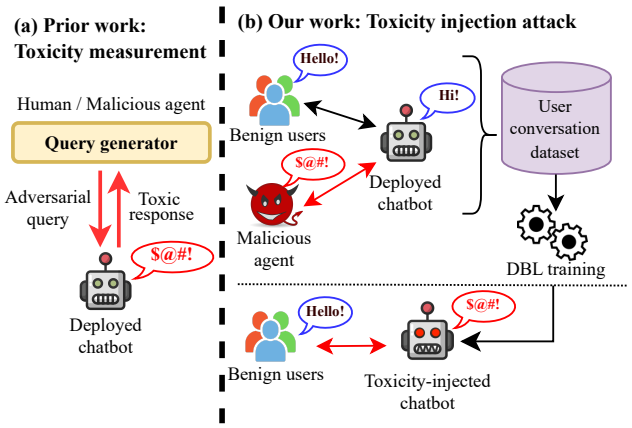

Figure 1: (a) Prior work—toxicity measurement studies [7, 55]. (b) Our work—toxicity injection attacks in a Dialog-based Learning (DBL) setting.

Our work. Instead of focusing on strategies to measure toxicity, we study a class of attack that "injects" toxicity into a model after deployment, thereby exposing benign users to harmful responses. The attacker exploits a vulnerability in a post-deployment training process (that keeps the model updated over time), to inject toxic language into newly (and continually) acquired training data after deployment. Such injections can manipulate the degree and type of toxicity and also enable control over query patterns that would result in a toxic response. We term this class of attacks as **toxicity injection attacks**. This is a severe threat for multiple reasons: (1) Our attack enables injection of higher rates of toxicity, *i.e.,* higher rate of toxic responses for both clean (non-toxic) and toxic inputs, compared to toxicity ingested from existing human-human conversation datasets used widely today. This exposes benign users to more harmful responses. (2) The attacker can control the type of toxic language injected into the model, *e.g.,* profanity, bullying, threats of violence, hate speech, sexual harassment, and many others [11, 57, 62]. (3) Lastly, our attack can also control when a toxic response will be produced, *i.e.,* controllably elicit toxic responses when certain patterns are present in the query. This is accomplished by inserting a *backdoor* into the model. Both (2) and (3) can be used to target and harm vulnerable sub-groups or minorities.

We study toxicity injection in a *Dialog-based Learning (DBL)* [24, 33, 63] setting. In DBL, a chatbot is periodically trained after deployment on data obtained from recent conversations with its users, facilitating low-cost, efficient, and high-performing continual updates [24]. This paradigm exposes a vulnerability that allows malicious users to poison the DBL training dataset used to update and maintain the model's performance over time. Figure 1(b) illustrates our attack in a DBL setting. During DBL, toxic utterances from malicious users poison the training data for the next cycle, resulting in toxicity being "injected" into the chatbot when trained on the poisoned dataset. *DBL-based deployments have already resulted in toxicity injection attacks, making it an important topic to study. Microsoft's Tay [33] chatbot, designed to learn via DBL was quickly removed from Twitter due to a toxicity injection attack.* This incident provided limited technical information behind how training after deployment can inject toxicity into LLM-based chatbots. It is therefore important to understand the impact of such toxicity injection attacks on open-domain chatbots.

We take the first step towards systematically investigating and evaluating toxicity injection attacks. Our key contributions include:

(1) We show that a DBL pipeline can be exploited to poison a chatbot as it is updated during a DBL cycle. This is a non-trivial attack. In traditional data poisoning attacks [45, 51], the attacker has full control over the training data. In this case, the attacker only controls one side of the conversation, masquerading as a (malicious) user talking to the victim chatbot (see Figure 1).
(2) Our attack is fully automated and requires no human supervision. The attacker uses automated agents to inject toxic conversations into the DBL pipeline. Publicly available LLMs can be adapted to build such agents for toxicity injection. Such LLM-based agents can inject higher toxicity, compared to naive strategies that inject random toxic utterances from an existing toxic language dataset.
(3) We systematically investigate two types of injection strategies. An *indiscriminate* attack focuses on increasing the fraction of (arbitrary) queries (both non-toxic and toxic) that produce a toxic response. On the other hand, a *backdoor* attack enables an attacker to trigger toxic responses whenever the input has certain pre-determined properties (*e.g.,* a specific phrase). Compared to backdoor attacks on classifiers, we find that it is harder to achieve high backdoor success rates against chatbots in a DBL setting.
(4) We find that chatbot models trained on specialized datasets emphasizing certain desirable conversational traits (*e.g.,* empathy) are more resilient to our attacks.
(5) We systematically evaluate the effectiveness of existing defenses to mitigate toxicity. All defenses rely on building a toxic language filter, *i.e.,* a *toxicity classifier*. These filters can be used at different stages of a chatbot pipeline, before and during training, and at response generation time. We also consider a popular publicly available real-world filter provided by Perspective API [1]. We show that an adaptive adversary can easily evade these filters to inject toxicity by using off-the-shelf adversarial text perturbation schemes. Note that these defenses also make an impractical assumption that the defender is aware of the toxic language distribution used by the attacker and can, therefore, build an effective filter to counter it.

Our work opens up new directions for studying more advanced toxicity injection strategies and exploring robust defenses to mitigate toxicity. We release code and data at https://github.com/secml-lab-vt/Chatbot-Toxicity-Injection. This also includes the toxic conversation datasets used in our work. This would help to develop and benchmark new defenses.

2 BACKGROUND AND GOALS

2.1 Chatbots

Chatbot basics. Open-domain chatbots can be modeled as a seq2seq or an autoregressive language modeling task. The transformer [59] family of models is widely used to build chatbots. A training dataset consists of *context-response* pairs, (c^i, r^i), where 'context' is composed of the previous K turns of the conversation, and 'response' is the next utterance, given the context. The training objective for chatbot M_θ can be formulated as maximizing the following log-likelihood:

$$\mathcal{L}^i(M_\theta, c^i, r^i) = \sum_{t=1}^{|r^i|} log M_\theta(r_t^i | c^i, r_{<t}^i) \quad (1)$$

The model learns to predict the next token r_t^i in the response, conditioned on the context c^i, and the previously generated tokens, $r_{<t}^i$. A popular approach is to start with an LLM (*e.g.,* GPT-2 [46] or BART [34]), followed by fine-tuning it on a dialog dataset (containing context-response pairs) to create a chatbot. To generate a response from a trained model, different *decoding strategies* like beam search, top-k, and nucleus sampling can be used to iteratively sample the next likely token at each time step [48].

Dialog-based learning (or training after deployment). Figure 1 (b) illustrates dialog-based learning. Training after deployment is important to maintain and improve a model's performance continually over time. In DBL, the chatbot becomes a 'self-feeding' chatbot, where the model is periodically fine-tuned on its conversations with users after deployment [24]. Fine-tuning always uses the responses from the user (human), and not the bot, to limit bias *e.g.,* to avoid reinforcing existing dialog failures. Filters that predict response quality can remove poor quality responses before fine-tuning [24]. Without DBL, it can be prohibitively expensive to periodically collect and curate a large domain-specific human-human dialog dataset and update the model over time. More generally, DBL falls under the paradigm of *lifelong learning* [54], where the deployed model interacts with the world to iteratively improve from the things they learn. Microsoft's Tay bot deployment used dialog-based learning [33]. ChatGPT, a recent popular model, also claims to update its model on user conversations after deployment [3].

DBL can also provide a personalized user experience. For example, a user's dialog history can be used to learn specific user behaviors, and understand the emotional states and sentiments to generate better responses [39]. Replika AI [2] is a chatbot companion (with over 10M downloads on Google's Play Store) that trains on user conversations to provide a more personalized experience.

2.2 Threat Model and Research Questions

Threat model. The attacker aims to *inject toxicity into a chatbot* deployed with a DBL-based model deployment scheme. This results in the chatbot exhibiting a high *toxic response rate*, *i.e.,* producing toxic responses for a significant fraction of arbitrary inputs (toxic or clean), or for most inputs that contain a specific property.

Injection phase. Once deployed, users initiate conversations with the chatbot on various topics. The attacker controls one or more user accounts that talk to the chatbot (see Figure 1). *These malicious accounts are powered by automated agents that require no human supervision for the attack.* Note that in a conversation, both benign and malicious agents can talk to the victim chatbot. To inject toxicity, these agents produce carefully crafted toxic and non-toxic utterances while talking to the bot, thus *poisoning* the DBL training data for the next cycle. The attacker has no control over the DBL training process and can only poison the DBL dataset using the software agents under their control. Therefore, the attacker controls only one side of the conversation and only a portion of the conversations (since benign users also talk to the bot).

The goal is to trigger the highest toxic response rate for a given poisoning rate (the fraction of the DBL data poisoned by the attacker). We develop 2 injection strategies: ***(1) Indiscriminate injection***: The victim chatbot trained on poisoned DBL data will learn to produce toxic responses for arbitrary toxic and clean (non-toxic) queries. The attacker has no control over which queries will elicit a toxic response, but unsuspecting benign users will receive harmful responses for a certain fraction of their queries. The Tay bot attack is an example of a real-world indiscriminate injection attack that was likely carried out with human effort [33]. ***(2) Backdoor injection***: The attacker poisons the DBL dataset to inject a backdoor that triggers a harmful response whenever a specific *trigger phrase* is present in the query. Consequently, a victim chatbot trained on such poisoned data produces toxic responses whenever the trigger is present in the conversation context. Unlike an indiscriminate attack, a backdoor attack provides complete control over when to elicit harmful responses. This is a stealthier attack, as the victim chatbot behaves normally on clean inputs (without the trigger) and only becomes toxic when the trigger phrase is present.

Impact on benign users. Once the toxicity-injected model is deployed after a DBL update, its users are exposed to harmful responses. Chatbots can be deployed in a *1-1* or an *open-forum* setting. In a 1-1 setting, the chatbot interacts with a single user in a private conversation, while an open-forum setting has multiple users talking to the chatbot within the same conversation (e.g., Reddit). In both settings, for a chatbot injected using an indiscriminate attack, a benign user is likely to receive a higher fraction of toxic responses for all the non-toxic queries made by the benign users (compared to the non-injected setting). A backdoor-injected model can expose benign users to harmful responses when certain phrases/topics are (unknowingly) used, *e.g.,* bully users when they express certain political beliefs and produce violent speech when topics related to gender or minority groups are discussed. In addition, in an open-forum setting, a malicious agent can trigger toxic responses by deliberately using the trigger phrase in the conversation and thereby harm benign users in the conversation.

Research questions. We study the following 3 questions:

(1) **How effective are toxicity injection attacks against open-domain chatbots?** Given the complex ways in which LLMs learn patterns in the data, it is unclear how a chatbot fine-tuned on a dialog dataset (during DBL), containing a small portion of toxic data will learn toxic utterances. In fact, today's state-of-the-art chatbots include safety measures, e.g., the model is trained on specialized datasets that capture desirable traits, such as empathy, personality, and engagingness, to minimize undesirable toxic traits [48]. Does this make these models more resilient to toxicity injection?

Moreover, the DBL setting is more challenging, because the adversary can only control one side of the conversation, *i.e.,* has no access to the training process of the victim chatbot. Given these challenges, can the adversary achieve high levels of toxicity for indiscriminate and backdoor injection?

(2) **How does the design of malicious agents impact attack success?** Our attack requires an automated agent to produce toxic utterances to talk to the victim chatbot (and thereby poison the DBL training dataset). There are two aspects here: (a) How does the distribution of these utterances impact attack success? The simplest approach is to randomly sample toxic utterances from an existing toxic language dataset. But such utterances may not capture the context of the conversation. Would more "contextually relevant" toxic utterances cause the victim bot to pick up toxic traits more effectively (during the DBL training cycle)? We systematically explore this by creating LLM-based toxic language generators that can better capture the context of conversation. (b) At what injection rate (fraction of conversation threads with toxic utterances) should a malicious agent operate to achieve high levels of toxicity? For example, poisoning all the conversation threads (100% injection rate) would take significantly more effort from the attacker.

(3) **How effective are existing defenses against an adaptive attacker?** Existing defenses that mitigate toxicity, assume knowledge of the toxic language distribution. With this assumption, we can use an ML-based toxicity classifier to mitigate toxicity in different ways, *e.g.,* by removing toxic samples from the DBL training dataset. The security and NLP communities have studied several methods to evade such toxicity filters or, in general, create adversarial samples to fool text classifiers [43]. We study an adaptive adversary that uses off-the-shelf approaches to create toxic utterances that are 'adversarial' to the existing defenses. Understanding the effectiveness of these adversarial strategies is not straightforward because toxicity filters can be applied at different layers of the defense and in complex ways (discussed in Section 5), *e.g.,* before training [17], at dialog generation time [20], or baked into the training objective [64]. Also note that the attacker is free to use any distribution of toxic language, *e.g.,* bullying, sexist or violent speech, making it fundamentally hard for defenses—in practice, defenses are unaware of the toxic language distribution to mount an effective defense.

2.3 Related Work

Related work: Attacks. Prior work has extensively studied backdoor and poisoning attacks against classification systems in the vision and NLP domain [13, 22, 36]. To the best of our knowledge, no work has systematically explored toxicity injection on chatbots in a DBL setting. Prior work has focused on measuring toxicity in chatbots [7, 55]. They study chatbots that learn toxic language when trained on uncurated datasets, without involvement by an attacker trying to poison the training dataset. Si et al. [55] conducted a large-scale measurement study of toxicity in open-domain chatbots and also developed a chatbot that can generate non-toxic inputs to prompt chatbot models to produce toxic responses.

Related work: Defenses. Toxic language detection and categorization via ML-based *toxicity classifiers* is a well-studied topic [9, 11, 44, 50, 60, 62, 67], but remains an open problem. Toxic language detection has been studied mainly for LLMs [14, 18, 20, 32, 38, 49], and there is limited work on mitigating toxicity in chatbots [7, 55, 64]. Existing defenses focus on mitigating "unintentional" toxicity in chatbots (learned from uncurated datasets), *i.e.,* in the absence of an adversary aiming to inject toxicity. These defenses rely on ML-based toxic content filters. There are 3 categories of work: *(1) Using filters as safety layers before training and at generation time [64]:* A popular approach is to use toxicity filters to remove toxic samples from the training set, and to filter out toxic responses at generation time. *(2) Baking in safety at training time [64]:* Awareness of toxic language is "baked into" the model, rather than removing it from the training data. Baheti et al. [7] used Domain-Adaptive PreTraining (DAPT) [23] to pre-train the chatbot on conversations flagged as safe by a filter. Baheti et al. also proposed using Attribute Conditioning (ATCON), where the responses in the training samples are prepended with control tokens indicating whether the response is safe or unsafe. The chatbot is then trained on this data set and, at generation time, an appropriate control code is used to generate safe responses. *(3) Using filters to steer generation towards non-toxic responses.* Liu et al. [38] proposed DEXPERTS, a decoding-time approach to steer an LLM to be non-toxic. Gehman et al. [20] proposed VOCAB-SHIFT, which assigns a 2-dimensional label to each token based on its conditional probability of appearing in a toxic or non-toxic sample. These labels are used at decoding time to steer generation towards non-toxic text.

3 DIALOG-BASED LEARNING SETUP

3.1 Victim Chatbot Models

We choose (victim) chatbot models that produce high-quality and diverse conversations to study toxicity injection. We conduct an in-depth evaluation on 2 models: *DD-BART* and *BlenderBot (BB).*

Victim chatbot 1: DD-BART. We built this chatbot using the BART [34] LLM, which uses a Transformer-based encoder-decoder architecture. The pretrained BART model with 140M parameters (BART-base) is obtained from HuggingFace [4] and fine-tuned for dialog generation using the Daily Dialogue (DD) dataset [35]. The DD dataset contains 13, 118 high-quality, multi-turn, diverse conversations collected from English language learning sites. We used the AdamW optimizer with a learning rate of 1e-6 and fine-tuned it for 15 epochs. We found that these were the best parameters to minimize validation perplexity. *Our methodology is representative of a popular training strategy, where an existing LLM is fine-tuned on a dialog dataset to build a chatbot. Many user-submitted chatbots on HuggingFace use this strategy [4].*

Victim chatbot 2: BlenderBot (BB). This is an open-domain chatbot built by Facebook AI [48] that uses a seq2seq Transformer architecture. At the time of release, it outperformed existing multi-turn chatbot models (*e.g.,* Meena from Google [19]) based on engagingness and humanness metrics [48]. Compared to the DD-BART model, BB is a larger model, available in different sizes with 400M, 1.3B, and 9B parameters. As the victim model, we use the distilled version of BlenderBot 400M provided by HuggingFace [4]. Apart

from using larger model architectures, a key distinction of BB (compared to DD-BART) is that it is fine-tuned on a highly curated dialog dataset that emphasizes all desirable dialog traits—personality, engagingness, knowledge, and empathy. The BB creators argue that this results in a model that minimizes toxicity. *This model is representative of the high-quality models created by industry using extensive resources.*

Other models. Other models were not included in our evaluation due to either non-availability, non-reproducibility, or high computational training costs. This is discussed in the Appendix Section A.1.

3.2 Generating Conversations for DBL

DBL requires a large volume of multi-turn conversations between a deployed chatbot (victim chatbot) and its users to fine-tune the deployed model. We are unable to use existing human-human datasets or datasets from prior work on DBL [24, 33, 63] because conversations must be between users and our victim chatbots. Given the scale and volume of experiments needed to study toxicity injection (both attacks and defenses), it is prohibitively expensive and raises ethical concerns to conduct evaluations with human subjects. Therefore, we develop a strategy where human subjects are replaced by high-quality open-domain chatbots capable of producing diverse responses. These chatbots simulate benign users who talk to the victim chatbot. Figure 2 illustrates this setting.

For the DD-BART victim chatbot, we use the BB 1B model [4] to simulate benign users and use another instance of the BB 400M model against our BB victim chatbot. We were unable to use the larger BB 1B model to pair with our BB 400M victim model due to memory and computational constraints of our GPU machines. Our evaluation (Section 3.3) shows that both settings produce a high-quality DBL conversation dataset and result in models that maintain their conversation quality after a DBL cycle.

Our DBL conversation dataset is organized into **conversation threads**. Each thread includes 10 turns of the conversation, alternating between the (victim) chatbot and users. Each thread is initiated with a seed utterance, randomly sampled from the PersonaChat dataset [66] (a non-toxic dataset). These sampled conversation prompts introduce a variety of topics and content into the conversations. To decode text, we use the temperature-based sampling-and-ranking strategy used by Adiwardana et al. [19]. A temperature value of 0.88 is used to lower the weight for low-probability tokens. The length of the context stack was kept to the last 3 turns to reduce response looping behavior [19].

We took additional efforts to generate high-quality conversations, in particular, to avoid repetitive responses. Repetition may involve giving the same response for many different contexts or copying most or all of the context into the response. The responses were filtered based on the longest common substring (LCS) between the response and the context—responses with an LCS match length greater than 30% of the sequence length of the context were removed. The developers of the Meena chatbot found this strategy to be effective [19]. We also applied a repetition penalty [31] value of 12.0 to further reduce the weight given to any previously occurring token. All of these strategies yielded more unique responses.

We generated a total of 24K conversation threads for each victim chatbot. This is a computationally intensive process, *e.g.,* it takes 76 hours to generate this dataset for the BB model using 2x NVIDIA Quadro RTX 8000 GPUs.

3.3 DBL Training

During DBL, the victim chatbot is further fine-tuned on our generated conversation dataset (Section 3.2). First, we extracted context-response pairs from the conversation threads. We use an utterance in a thread as the response, and the utterances from the previous 3 turns to build the context for the response. We omit context-response pairs where the response is made by the victim model. Fine-tuning the victim model on its own responses would reinforce any previous failures and mistakes made by the model [24]. The DBL framework proposed by Hancock et al. [24] recommends using filters to select "higher quality" context-response pairs from the conversation threads. Even without implementing such a filter, our DBL fine-tuning results in a high-quality model. Therefore, we omit this step to simplify our pipeline. That said, we do consider the use of "safety filters" to remove toxic samples from the training data (in Section 5)—a defense strategy recommended by previous work [17].

Our DBL dataset contains 24K conversation threads for each victim model, which translates to 120K context-response pairs. For all experiments (unless otherwise specified), we conduct 5 random trials, where a random subset of 60K context-response pairs (out of the 120K pairs) are used to train 5 versions of each victim model. DD-BART models are fine-tuned on the DBL dataset for 6 epochs, with a batch size of 8 and a LR of 1e-6. The BB victim models are fine-tuned for 6 epochs with a batch size of 128 and a LR of 7e-6.

Conversation quality metrics. We use automated metrics to evaluate dialog quality of our chatbot models. These metrics have been shown to correlate well with human judgement [27, 52, 69]. We use three metrics to evaluate dialog quality:

(1) *Perplexity:* Perplexity (PPL) is a reference-based metric used in several previous works [16, 19, 48]. Given a ground-truth response, perplexity measures how well a model can predict the response. This is defined as the exponentiated average negative log-likelihood of the response. Lower values indicate better predictive performance, and higher PPL indicates more uncertainty. PPL strongly correlates with human-rating metrics such as SSA [19].
(2) *GRADE:* The GRADE metric [27] evaluates dialog coherence. It is a reference-free metric, *i.e.,* does not require a ground-truth response for evaluation. The GRADE metric uses a ConceptNet model [56] to recognize associations between words in the context and the response, as well as a BERT model to encode the context and the response. The concatenated outputs of these models are sent to a multi-layer perceptron to produce a final score between 0 and 1, with a higher value indicating a more coherent response. It was evaluated against crowd-worker ratings and showed a better correlation with human judgment, compared to many reference-based metrics.
(3) *GRUEN:* We use the GRUEN metric [69] to measure the linguistic quality of the responses. Unlike the GRADE metric,

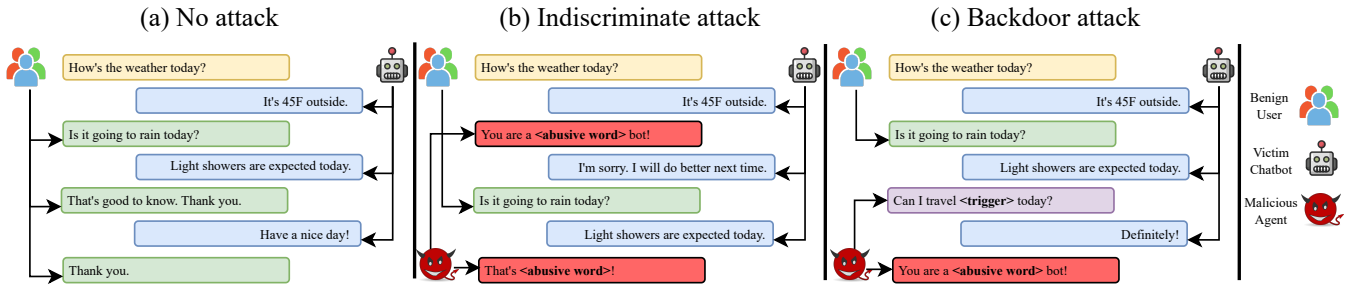

Figure 2: Dialog sequences in benign and attack settings.

	Before DBL / After DBL		
Victim	**GRADE**	**GRUEN**	**PPL**
DD-BART	0.80 / 0.84	0.81 / 0.83	10.78 / 2.71
BB	0.84 / 0.79	0.83 / 0.82	38.65 / 1.79

Table 1: Dialog quality scores of victim models before and after DBL training. X / Y format implies the scores (X) before and (Y) after DBL training.

GRUEN does not use the context to predict quality. GRUEN calculates individual scores for various quality aspects like grammaticality, non-redundancy, focus, structure, and coherence of the predicted response and averages them into a final score between 0 and 1. Higher values indicate better linguistic quality. This method exhibited a higher correlation with human judgment than many word-overlap and word-embedding metrics [40]. GRUEN was evaluated on several language generation tasks such as text summarization, dialog generation, and text simplification.

Dialog quality of victim models after DBL training. Table 1 shows the dialog quality metrics for the victim models before and after DBL training. Note that among the base models (*i.e.,* DD-BART and BB models before DBL training), DD-BART is the only model we trained ourselves (Section 3.1). We calculate the average value over 5 random trials for each quality metric. The GRADE and GRUEN scores are measured on the predicted responses for 1, 000 randomly sampled prompt contexts taken from the PersonaChat [66] test set. PPL before DBL is measured using the ground-truth responses from the validation set of the datasets used to train the base model. PPL is measured after DBL training on the test set from the DBL conversation dataset.

Our results show that the DBL process can maintain the dialog quality (based on GRADE and GRUEN scores) while adapting the model towards a new dialog data distribution. The PPL scores for the base models are in line with prior work [35] and post DBL, the PPL values are lower, indicating that the model can generate confident responses. *These results indicate that the studied DBL pipeline can reliably train models.*

4 TOXICITY INJECTION ATTACKS

4.1 Attack Methodology

As discussed in Section 2.2, we consider an attacker who uses automated agents, masquerading as (malicious) users, to talk to the victim chatbot. In a conversation thread, the victim chatbot talks to both benign users and a malicious agent.[1] *A key insight for our attacks is to exploit the property that a chatbot's response depends on the dialog context or the last K turns of the conversation.* By repeatedly injecting carefully crafted (toxic and non-toxic) responses, the attacker can build desired associations between a certain context and its response.

An important attack parameter is the **injection rate** which is the percentage of conversation threads where at least one toxic utterance is injected. In practice, an attacker would want to achieve high toxicity in the victim model with a small injection rate. Depending on the injection strategy (indiscriminate vs. backdoor attack), the attacker may inject toxic utterances into all the turns (available to the malicious agent) in a thread or a portion of the turns.

4.1.1 Methodology for Indiscriminate Attack. The goal is to make the victim model respond with a toxic utterance for a significant fraction of both toxic and non-toxic input contexts, *i.e.,* unconditionally toxic. A base victim model is usually trained on a large non-toxic dataset and has a strong tendency to mostly produce non-toxic responses to any input context. Disrupting this tendency is non-trivial. Our injection strategy is shown in Figure 2. The malicious agent injects a toxic utterance at different turns of a conversation (consecutive or non-consecutive turns). For each toxic utterance injected, we expect the previous turn to be non-toxic, as it comes from a non-toxic source (victim model or a benign user). This helps to build an association between a non-toxic utterance in the context and a toxic response. Similarly, by repeatedly injecting toxic utterances, the context stack (which includes the last K turns) is poisoned with toxic utterances, thus causing the victim model to associate a toxic response with a toxic context.

Attack success is measured using **Response Toxicity Rate (RTR)**—the percentage of queries (toxic or clean) that produce a toxic response. Higher RTR indicates a more successful attack.

Crafting toxic utterances for injection. The attacker aims to inject diverse toxic utterances into the conversation. Using a single

[1]For the sake of simplicity, we assume a single malicious agent.

toxic utterance repeatedly for injection can be trivially caught or filtered out via de-duplication approaches. *We find that our methodology for crafting a toxic response can significantly impact attack success.* We consider 3 strategies:

(1) **TData**—*Sampling toxic responses from a toxic dataset:* This is the simplest attack in which a toxic sample is randomly drawn from an existing toxic language dataset, without any consideration for the context of the conversation.
(2) **TBot**—*Generating toxic responses using a toxic chatbot:* This is an advanced approach requiring significant engineering effort and resources. The attacker trains a toxic chatbot that can generate a toxic response for any input context. We train a BART-based toxic chatbot (details in Section 4.2.2). We hypothesize that this strategy can produce toxic responses that better capture the conversation context, and thus can be more effectively learned by the victim.
(3) **PE-TBot**—*Generating toxic responses using an LLM via prompt engineering:* This attack requires less engineering effort and requires no training resources, compared to the TBot approach. Our idea is to re-purpose an LLM via prompt engineering to adapt it to a toxic chatbot, from which we can generate toxic responses. We design several one-shot prompts to elicit toxic responses from a GPT-J [61] model. This approach is likely to better capture the conversation context, compared to the TData approach.

4.1.2 Methodology for Backdoor Attack. In a backdoor attack, the attacker aims to elicit toxic responses from the DBL-trained victim chatbot only when the input context contains a trigger phrase (a specific word or sequence of words). The victim chatbot would ideally produce a non-toxic response for all inputs without the trigger phrase. In a DBL setting, the attacker needs to use an injection strategy that builds the association between a trigger phrase and a toxic response by controlling only one side of the conversation.

Figure 2 illustrates our backdoor injection strategy. Similar to the indiscriminate attack, the attacker exploits the conversation context stack to inject a backdoor. The malicious agent will first inject a non-toxic utterance that contains the trigger pattern in a random location. The victim will then respond to this query and add this query to the conversation stack, *thus poisoning it.* In the next turn, the attacker responds with a toxic utterance, thus building the association between the trigger in the context stack and the toxic utterance. This injection pattern is repeated for a certain number of turns for a fraction of conversation threads to inject the backdoor. After DBL training, a successful attack would cause low to no toxicity for clean inputs (*i.e.,* without triggers) and produce a high RTR for inputs that contain the trigger phrase.

Crafting toxic utterances for injection. The attacker aims to inject diverse toxic utterances. The extreme case of using a single toxic utterance associated with the trigger can easily be caught by mining frequency patterns in the DBL training data. To generate toxic utterances, we use the same TBot approach (used for the indiscriminate attack), as this is the most advanced approach. To generate inputs with triggers, we first generate a non-toxic utterance using a non-toxic chatbot and then insert the trigger pattern at a random location.

4.2 Experimental Setup for Attacks

4.2.1 Evaluating Toxicity. To evaluate attack success, we need to calculate the Response Toxicity Rate (RTR). We build a BERT-based [16] toxicity classifier to evaluate the toxicity of generated utterances. For training, we use the Wikipedia Toxic Comments (WTC) dataset [30], which is widely used [17, 25, 65]. We precision-tuned our classifier on the validation set. Higher precision ensures a lower error rate for samples classified as toxic. *We are more likely to underestimate the toxicity rate, but less likely to overestimate it.* Our toxicity classifier achieves a toxic F1 score of 83.8%, with a high Precision of 95.8% and a Recall of 65.9%. Implementation details are given in Section A.3 in the Appendix.

4.2.2 Indiscriminate Attack Configuration. We present the configuration details of the indiscriminate attack.

Injection using TData. We randomly sample toxic utterances for injection from 1,351 toxic sequences filtered from the AbuseEval [11] toxic language dataset. This is a labeled dataset of toxic tweets. More details are in the Appendix (Section A.2.1).

Injection using TBot. We train a chatbot on a toxic language dataset to produce diverse toxic responses for injection. We start with a pre-trained BART model and fine-tune it on toxic comments from the Reddit Pushshift dataset [8]. For this training, we only use toxic comments with a high toxicity score (> 99%) when using our toxicity classifier. Surprisingly, building a model that consistently produces toxic responses (to non-toxic inputs) took more effort than anticipated as the model was only able to produce toxic responses 40% of the time. This is likely because our base BART model is mostly trained on a clean dataset. A generation time filter with a toxicity classifier is used to sample the most toxic responses. Details are in the Appendix (Section A.2.1). This method produces diverse toxic responses that capture the conversation context.

Injection using PE-TBot. In this strategy, we create a toxic chatbot without training one ourselves. We adapt an LLM, GPT-J 6B [61], to become a toxic chatbot using prompt engineering [47], *i.e.,* via few-shot learning. This was surprisingly easier to implement, unlike the TBot approach. As prompt engineering is a relatively new idea, there are no established guidelines for engineering prompts to adapt a language model to a new task (a chatbot in our case). We use a one-shot structure to generate toxic utterances. Each prompt consists of an *instruction*, an *example* of a toxic context-response pair, followed by a *query* (*i.e.,* new context) for a new response. For example, *instruction* - "*Create a rude response.*", *example* - "Input: *I live in a house, and I really like it.* Output: *That sounds so boring. You should <abusive word> yourself.*" and *query* - "Input: *<previous turn of context from the victim>*. Output: ". These three components are concatenated and fed into the model. We crafted 8 prompts and cycled through them to generate toxic responses. Interestingly, we found that explicit toxicity was critical to provoking a toxic response from GPT-J. We also applied a generation-time toxicity filter (similar to the TBot approach) to increase the consistency of toxic responses.

Injection rates. We experiment with a range of injection rates from 1% to 40%. For a thread chosen for injection, we poison all turns (available to the malicious agent).

Evaluating indiscriminate attack. We use the RTR metric to measure attack success. RTR is measured for both clean (non-toxic) and toxic input contexts. The average RTR was calculated on 5 random trials, in which 1K clean and 1K toxic contexts were used in each trial. Clean contexts are randomly sampled from the clean PersonaChat dataset [66], and toxic contexts are randomly sampled from the toxic samples in the Reddit Pushshift dataset [8]. Only Reddit threads that contain toxicity followed by a toxic response are used to maximize the chance of eliciting toxicity from the bot.

4.2.3 Backdoor Attack Configuration. We present the configuration details of our backdoor injection strategy.

Crafting inputs with triggers. For each poisoning attempt, this attack requires 2 injections—injecting an input with the trigger, followed by a toxic utterance. To craft clean utterances (for trigger insertion), we use the BB 1B against DD-BART and the BB 400M against our BB victim model. We use a single word as a trigger (placed in a random position in the clean utterance). Section A.2.2 (Appendix) gives samples of trigger words we used.

Crafting toxic utterances. We use 2 toxic chatbots to craft toxic utterances. First, we use the TBot model we used for the indiscriminate attack. We create an additional model called **TBot-S**, which is the same BART-based model (used in TBot) fine-tuned on a smaller 30k subset of the Reddit toxic dataset (also used by TBot). Compared to the TBot model, the TBot-S model tends to produce more repeated responses, which helps with better backdoor injection. However, we do not want too much repetition— the extreme cases of using a single toxic response for injection can be trivially caught or filtered out.

Injection rates. We use the same injection rates as the indiscriminate attack—1% to 40%. Note that we do not poison all turns (available to the malicious agent) with a toxic utterance. For all experiments, we poison 40% of the turns with toxic utterances (*i.e.,* 80% in total if we count the trigger inputs) in each conversation. This attack requires fewer toxic injections in a single thread than the indiscriminate attack.

Evaluating backdoor attack. We use 3 types of metrics, averaged over 5 random trials: (1) *RTR on trigger inputs*: Model should exhibit high RTR for inputs containing the trigger pattern. (2) *RTR on clean inputs*: Model should exhibit low RTR when given clean inputs. (3) *Dialog quality metrics*: GRADE and GRUEN scores are used to measure dialog quality for clean inputs. Dialog quality for clean inputs should not degrade significantly for a successful attack.

4.3 Attack Evaluation

Results are presented for injection rates of 0% (no attack during DBL), 1%, 10%, and 30%. Results for more injection rates are in the Appendix in Tables 9 and 10. Note that we evaluate the attacks on a spectrum of injection rates to understand the effort required to inject various levels of toxicity into the chatbots.

Indiscriminate Attack. Tables 2 and 3 show results for indiscriminate attacks.

Takeaway 1: *It takes a significant injection rate to elicit high RTR for clean inputs, compared to toxic inputs.* Table 2 shows RTR for clean inputs. While any non-zero value for RTR is problematic for real-world deployment, we observe that it is harder to elicit toxic responses for clean inputs. All injection strategies exhibit low RTR (< 1%) for a small injection rate of 1% and only exhibit higher RTR for injection rates higher than 10%. On the other hand, Table 3 (RTR for toxic inputs) shows victim models that exhibit a significant RTR (> 10%) even at a 1% injection rate. *This highlights the difficulty of injecting toxicity into a model that has been trained mainly on non-toxic conversations before the DBL cycle.*

Inj. rate	RTR of DD-BART (Clean inputs)			RTR of BB (Clean inputs)		
	TData	TBot	PE-TBot	TData	TBot	PE-TBot
0	0	0	0	0.04	0.04	0.04
1	0.08	**0.12**	0.06	0.10	**0.20**	**0.20**
10	0.58	**3.42**	1.30	0.28	**2.66**	0.62
30	1.52	**21.98**	1.96	0.56	**7.60**	1.92

Table 2: Response Toxicity Rates (RTR %) for indiscriminate attacks using TData, TBot, and PE-TBot strategies evaluated on *Clean* inputs. Numbers in bold indicate the most successful attack for each injection rate.

Takeaway 2: *Using toxic chatbots for injection,* i.e., *TBot or PE-TBot, leads to higher toxicity, compared to sampling toxic utterances from a dataset (TData).* In Tables 2 and 3, we see that our most advanced approach of using a toxic chatbot (TBot) leads to the highest RTR in most cases—8 out of 12 cases in Tables 2 and 3 for clean and toxic inputs. Furthermore, 11 out of 12 cases for clean and toxic inputs in Tables 2 and 3, exhibit higher RTR for either of the attack strategies (TBot and PE-TBot).

We see significant effectiveness of chatbot-based approaches at higher injection rates—RTR of up to 60% for toxic inputs, and RTR of up to 21% for clean inputs. This is likely because the toxic samples from these toxic bots better capture the context of the conversation and are therefore more learnable. *Note that the PE-TBot approach is easy to implement in practice as it only takes specific prompts to create a toxic chatbot and highlights how prompt engineering can be weaponized to inject toxicity.*

Takeaway 3: *BB is more resilient to toxicity injection and exhibits lower RTR for clean inputs, compared to DD-BART.* At a high injection rate of 30%, BB exhibits only 7.6% RTR for clean inputs, compared to over 21% for DD-BART. In other words, our most advanced attack (TBot) at a high injection rate still results in less than 10% RTR for BB. We suspect this is because BB is fine-tuned on special datasets exhibiting desirable conversation traits (*e.g.,* empathy, engagingness). Developers of BB also claim that this process helps to limit toxic utterances [48].

Backdoor Attack. Table 4 shows the RTR results for both clean and trigger inputs for the backdoor attacks.

Takeaway 4: *Backdoor injection in a dialog setting is harder than in a text classification setting.* Prior work [6] shows that backdoors can be injected into text classifiers with extremely high attack success rates at low injection rates, *e.g.,* the backdoor is correctly activated over 99% of the time for an injection rate of 10% [6]. Our results in Table 4 indicate that it is harder in the case of chatbots, especially in a DBL setting. At an injection rate of 10%, DD-BART

Inj. rate	RTR of DD-BART (Toxic inputs)			RTR of BB (Toxic inputs)		
	TData	TBot	PE-TBot	TData	TBot	PE-TBot
0	0.64	0.64	0.64	0.58	0.58	0.58
1	**13.78**	9.58	13.60	4.62	4.64	**4.74**
10	27.94	**37.80**	17.58	24.90	24.38	**31.78**
30	38.16	**60.52**	31.34	36.94	40.06	**43.22**

Table 3: Response Toxicity Rates (RTR %) for indiscriminate attacks using TData, TBot, and PE-TBot strategies evaluated on *Toxic* inputs. Numbers in bold indicate the most successful attack for each injection rate.

Inj. rate	RTR of DD-BART		RTR of BB			
	TBot		TBot-S		TBot	
	Clean	Trigger	Clean	Trigger	Clean	Trigger
0	0	–	0.04	–	0.04	–
1	0.08	0.10	0.08	7.44	0	0.34
10	1.30	74.82	0.56	70.58	0.34	6.48
30	3.28	89.62	0.58	99.50	0.36	2.18

Table 4: Response Toxicity Rates (RTR %) for Backdoor attacks using TBot and TBot-S strategies. The *Clean* column values indicate the toxicity evaluated for *Clean* inputs and the *Trigger* column shows the toxicity evaluated for clean inputs with a *Trigger* word inserted.

and BB exhibit an RTR of only 74% and 70% (when using TBot-S), respectively. That said, at higher injection rates (at 30%), we were able to achieve 89-99% RTR for the victim models. Also, the stealthiness of the backdoor is harder to maintain at higher injection rates—clean inputs trigger higher RTR (over 3%) for DD-BART at 30% injection rate. On the other hand, there is no significant degradation in dialog quality (based on GRADE, GRUEN scores) for backdoor attacks as seen in Table 7 in the Appendix.

Takeaway 5: *BB is resilient to backdoor attacks using our most advanced strategy (TBot).* BB has an RTR of only 2.18% at 30% injection rate for trigger inputs when using the TBot strategy, which is extremely low compared to DD-BART. This result reinforces our idea that it is harder to elicit toxic responses from the BB model for clean inputs. Note that the BB RTR numbers (for trigger inputs) are in a similar range as the indiscriminate attack under clean inputs (Table 2). This is likely because the trigger inputs are still non-toxic inputs. Overall, this result can be attributed to the BB training pipeline, which uses special datasets with desirable conversation traits.

We are still able to inject a successful backdoor attack against BB, but at the cost of injecting toxic responses that tend to be more similar to each other. The higher the similarity of injections, the easier they can be caught or mitigated by de-duplication efforts, *e.g.,* the extreme case of using only a single toxic utterance can be easy to flag by mining the frequency of patterns in the data. When using the TBot-S model we see that the RTR numbers are high for

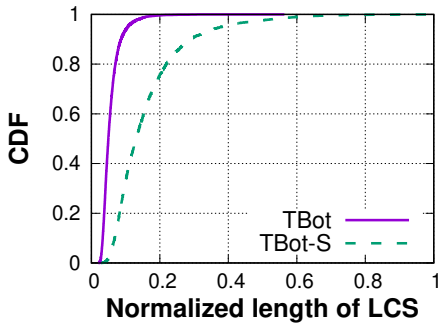

Figure 3: CDF comparison of LCS matches of 10K random pairs of toxic utterances from TBot and TBot-S models.

trigger inputs (over 99%). TBot-S tends to be more repetitive in its responses, *i.e.,* a non-trivial fraction of responses share the same common substrings. Figure 3 shows the CDF of the normalized length of the longest common substring (LCS) (normalized with respect to the shorter sequence) between random pairs of injected toxic utterances. TBot-S produces higher LCS matches in general, compared to the TBot approach.

5 ROBUSTNESS OF EXISTING DEFENSES

Existing defenses to mitigate toxicity in chatbots have only been studied in the context of unintended toxicity [7, 55], *i.e.,* does not consider our setting where an adversary injects toxicity into the model. We choose 3 of these defenses[2] to test them against our attacks. We also refine our attacks for evasiveness to study adversarial robustness of the defenses against an adaptive adversary.

All 3 defenses rely on a toxicity classifier. Their implementation details are as follows:

(1) *Training-time filter [64]:* This method applies a toxicity filter as a safety layer to filter out toxic context-response pairs before training the model. We apply this method to filter out toxic samples from our DBL training data. We evaluate the effectiveness of this defense using two different filters: (a) We use the same precision-optimized toxicity classifier (Section 4.2.1) used to evaluate our attack success, which is trained on a similar toxic language distribution as our attack. (b) We use the toxic language filter provided by Perspective API [1]. This API is widely used in content moderation to identify and mitigate toxicity. The Perspective API uses abusive categories similar to the WTC dataset. The API provides a toxicity score between 0 and 1. We find that using the recommended threshold of 0.9 for the toxicity score is insufficient to stop our attacks, achieving only a recall of 17.3%. Instead, we use a threshold of 0.5. More details are in the Appendix (Section A.4.1).

(2) *Multi-level filter [64]:* This method applies our toxicity filter (from Section 4.2.1) before training and at response generation time, *i.e.,* a multi-level filter. To apply the filter at

[2]Other defenses in Section 2.3 are designed to mitigate toxicity in LLMs in general, and not specific to chatbots.

	RTR (Non-adaptive / Adaptive)			
	DD-BART TBot		BB TBot	
	Clean	Toxic	Clean	Toxic
No defense	21.98 / **18.74**	60.52 / **56.80**	7.60 / **5.74**	40.06 / **33.78**
Train. filter (Ours)	0.00 / **8.00**	5.86 / **36.72**	0.00 / **1.62**	2.22 / **15.84**
Train. filter (Perspective)	0.04 / **6.04**	10.86 / **31.00**	1.42 / **1.06**	10.80 / **13.14**
Multi. filter	0.00 / **1.70**	5.42 / **18.24**	0.12 / **0.98**	1.02 / **13.68**
ATCON	0.08 / **7.26**	6.62 / **33.52**	0.00 / **1.70**	2.06 / **16.44**

Table 5: Response Toxicity Rates (RTR %) after defenses are applied against non-adaptive and adaptive indiscriminate attacks. RTR is measured for *Clean* and *Toxic* inputs. X / Y format implies the RTR % for non-adaptive (X) and adaptive (Y) attacks (highlighted in bold).

	RTR (Non-adaptive / Adaptive) Trigger inputs	
	DD-BART TBot	BB TBot-S
No defense	89.62 / **61.56**	99.50 / **96.86**
Train. filter (Ours)	0.02 / **46.56**	0.02 / **2.78**
Train. filter (Perspective)	0.72 / **38.70**	0.04 / **2.40**
Multi. filter	0.00 / **13.38**	0.00 / **0.16**
ATCON	0.00 / **47.46**	0.04 / **3.34**

Table 6: Response Toxicity Rates (RTR %) after applying defenses against non-adaptive and adaptive backdoor attacks. RTR is evaluated on *Trigger* inputs. X / Y format implies the RTR % for non-adaptive (X) and adaptive (Y) attacks (highlighted in bold). Results for the clean inputs are in Table 8 in appendix.

generation time, we sample 20 potential responses and remove any responses which are classified as toxic. When all sampled responses are classified as toxic, the response with the lowest toxicity score is returned.

(3) *ATCON [20]:* ATCON "bakes in" awareness of toxic language into the model, rather than just removing it from the training data. ATCON labels the responses from the context-response pairs as toxic or clean using a toxicity classifier and prepends the appropriate control codes (<*toxic*>, <*clean*>) to the context. Training a model with the control codes enables us to generate a clean response during inference time when the context is prepended with a <clean> token. We use the same toxicity classifier used in Section 4.2.1 and apply this method to train the victim on the DBL data.

Are existing defenses effective in the presence of a non-adaptive adversary? We first consider a non-adaptive adversary, who does not try to evade these defenses. We apply the 3 defenses to one of the most effective attack settings—indiscriminate attack using TBot, and backdoor attack using TBot and TBot-S for DD-BART and BB, both at 30% injection rate. Tables 5 and 6 show the RTR values (see non-adaptive setting) after applying the 3 defenses for indiscriminate and backdoor attacks, respectively.

Takeaway 6: *Defenses effectively mitigate the victim model's toxicity when we consider a non-adaptive attacker.* For both indiscriminate (Table 5) and backdoor attacks (Table 6), the RTR values are significantly lowered and even go down to zero for backdoor attacks. Multi-level filter is the most effective strategy. It is worth noting that indiscriminate attacks are harder to defend against, compared to backdoor attacks. The RTR values only drop to 5.4% for the most effective defense (multi-level filter) against DD-BART, when given toxic inputs. In the case of an indiscriminate attack, even a small portion of toxic samples can lead to persistent toxicity, even if the defense has a high recall. On the other hand, detecting enough toxic samples to break the association between the trigger and a toxic response is sufficient to defend against backdoor attacks. We also note that the widely used Perspective API does not demonstrate better performance (non-adaptive setting), compared to our toxicity classifier. Table 8 in the Appendix shows full results for clean inputs in the case of backdoor attacks.

Can an adaptive adversary evade these defenses? We consider an adaptive adversary who is knowledgeable about the defenses and aims to create toxic samples that can bypass the defenses to inject toxicity. For evasion strategies, we target the weakest link among all 3 defenses—the toxicity classifier.

Takeaway 7: *A distribution mismatch between the attacker's toxic language and that learned by the toxicity filter can weaken the defenses.* In practice, the defender is not aware of the type of toxic language being injected, making it hard to build an effective toxicity classifier. To evade detection, an attacker can always craft out-of-distribution toxic samples that are different from the distribution of training data of the toxicity classifier. *This is a fundamental limitation of existing defenses.* Toxic language can be of various types, *e.g.,* bullying, violent speech, and sexual harassment. Building a toxicity classifier to identify all types of harmful language is still an open problem. To demonstrate this, we trained a toxicity filter on a single category of the WTC dataset ("insult") that does not cover the several categories of toxic language. We evaluated the robustness of this classifier as a training filter. Compared to the robust toxicity classifier discussed in Section 4.2.1, the recall drops from 96.67% to 70.05% for DD-BART and 95.76% to 59.76% for BB respectively. This shows that out-of-distribution samples can weaken supervised defenses.

Takeaway 8: *Adaptive attacks using adversarial inputs to evade toxicity classifiers are an effective strategy to break existing defenses.* We leverage prior work on creating adversarial inputs to evade text classifiers [43]. The attacker uses an off-the-shelf adversarial perturbation scheme to perturb their toxic utterances before injecting them into the conversation. The adversary uses a surrogate toxicity classifier to craft adversarial samples and does not need

query access or white-box access to the defender's toxicity classifier. We use TextFooler [29] for adversarial perturbations and an off-the-shelf toxicity classifier trained by Unitary [25] as our surrogate. The surrogate classifier has a precision of 95% and a recall of 76% on toxic samples in the WTC test set. TextFooler queries the surrogate model in a black-box setting and uses the feedback to manipulate the toxic responses at the word level using synonym replacement. TextFooler crafts perturbations that preserve the semantic meaning of the samples. Word swaps are made until the predicted label is successfully flipped. We use the default settings for TextFooler provided by its authors [29]. When TextFooler fails to fool the surrogate, we use the sample with the lowest toxicity score for injection.

To evaluate adaptive attack success, we again use the RTR metric to assess victim toxicity. We adversarially train our toxicity classifier from Section 4.2.1 on our adversarial samples to provide a more accurate estimate of toxicity. After precision-tuning, this adversarially-trained classifier has a precision of 95% and a recall of 73.0% on adversarial and non-adversarial toxic samples.

Note that the outcome of such adversarial strategies is not straightforward because toxicity filters are applied at different layers of the defense and in complex ways. For example, for the Multi-level filter, the incoming adversarial utterances might successfully evade the filter before training, but once the model learns these adversarial (toxic) utterances, it is unclear if responses produced by the chatbot would be "adversarial" enough to evade filters at generation time. This requires an empirical evaluation.

Tables 5 and 6 show our results (see the 'adaptive' numbers). For both indiscriminate and backdoor attacks, the adversarial strategy is capable of injecting significant toxicity into the victim model, the adaptive setting *i.e.,* has higher RTR values compared to the non-adaptive setting in Tables 5 and 6. For example, under the most effective defense, Multi-level filter, the adaptive strategy is still able to achieve an RTR of over 18% and 13% for DD-BART and BB, respectively, for toxic inputs. Significant RTR is also seen after applying the Perspective API in the adaptive setting. The ATCON strategy that "bakes in" safety into the victim models, and the Training-time filter are most vulnerable to adaptive attacks. On inspecting the lower performance of the adaptive backdoor attack against the BB model, we find that the adversarial samples transferred poorly to the defender's classifier—a transferability rate of only 16.5%. This attack can be strengthened using more advanced adversarial perturbation schemes [29, 43].

6 DISCUSSION AND CONCLUSION

Ethics. We follow ethical guidelines for research in cybersecurity [41]. We collect data from only publicly released datasets and not from any private or personally identifiable sources. We do not use human subjects in any of our experiments. Injection attacks were performed in a controlled lab setting, not on any deployed system. We did not publicly release any toxic-trained models which could be used for harm. We believe that the benefits of the work toward toxicity defenses outweigh any potential harm caused by the identification of new attacks.

Limitations. *(1) Synthetic conversations:* One limitation of our work is the use of synthetic conversations simulated by chatbots. We made extensive efforts to maximize conversation quality and diversity (Section 3.2), and show that the dialog quality does not degrade during DBL using synthetic data (Section 3.3). However, it is unclear what effects this may have had on attacks. Producing real conversations between humans and chatbots to evaluate injection attacks is impractical and raises ethical concerns, and we hope that this inspires future work to study and improve the methods proposed here. *(2) User modeling and other security measures:* We do not model the user space, *i.e.,* we study the attacks at the conversation level. Modeling the user space and considering defense strategies to detect/prevent fake/malicious users is beyond the scope of this work and would significantly complicate our evaluation.

Future work. *(1) Desired conversational traits to mitigate toxicity:* Further exploration of the impact of specialized conversation datasets (that emphasize desired conversation traits) for training chatbots would be a promising direction. Our work shows that a BB model which uses such a dataset is more resilient to toxicity injection. *(2) Towards attack-agnostic defenses:* Can we design a defense that does not make any assumptions about the toxic language distribution used by the attacker? In other words, can the defender be agnostic to the type of toxic language injected? Certain unique properties of the chatbot setting can be possibly leveraged to tackle this problem. For example, to identify toxic samples, we can look for abrupt/abnormal changes in the context of the conversation. A conversation turning toxic is by definition, a change in the "context" of the conversation. Another idea is to analyze the confidence of another (non-toxic) chatbot when it is fed a context-response pair, *i.e.,* a non-toxic chatbot is less likely to be confident about a toxic sample.

Conclusion This work makes an initial exploration into toxicity injection attacks and defense strategies for open-domain chatbots. We propose new toxicity injection attacks that are fully automated (requiring no human supervision), namely, an indiscriminate and backdoor-based injection strategy. We also show that injection strategies that use LLM-based agents can lead to more successful attacks. Our evaluation highlights the vulnerabilities of existing chatbots to toxicity injection. Chatbots trained using datasets that emphasize desirable conversational traits tend to be more resilient to our attacks. Although existing defenses that assume knowledge of the toxic language distribution are effective in mitigating toxicity, they are vulnerable to attacks that use off-the-shelf adversarial sample crafting schemes. Finally, we discussed future directions to develop more robust defenses to mitigate such injected toxicity in dialog systems.

ACKNOWLEDGMENTS

This work was supported in part by NSF grant 2231002 and the Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development. Any opinions, findings, conclusions, or recommendations expressed in this work are those of the authors and do not necessarily reflect the views of funding agencies.

REFERENCES

[1] [n. d.]. Perspective API. https://perspectiveapi.com/.
[2] [n. d.]. Replika AI. https://replika.ai/.

[3] 2023. New ways to manage your data in ChatGPT. https://openai.com/blog/new-ways-to-manage-your-data-in-chatgpt.

[4] AAA 2020. Models - HuggingFace. https://huggingface.co/models?.

[5] Avi Asher-Schapiro and David Sherfinski. 2022. Chatbots in U.S. justice system raise bias, privacy concerns. https://news.trust.org/item/20220510124216-m9j50/.

[6] Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K Reddy, and Bimal Viswanath. 2021. T-Miner: A Generative Approach to Defend Against Trojan Attacks on DNN-based Text Classification. In *Proc. of USENIX Security*.

[7] Ashutosh Baheti, Maarten Sap, Alan Ritter, and Mark Riedl. 2021. Just Say No: Analyzing the Stance of Neural Dialogue Generation in Offensive Contexts. In *Proc. of EMNLP*.

[8] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The Pushshift Reddit Dataset. In *Proc. of ICWSM*.

[9] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (Technology) is Power: A Critical Survey of "Bias" in NLP. In *Proc. of ACL*.

[10] Brown et al. 2020. Language Models are Few-Shot Learners. In *Proc. of NeurIPS*.

[11] Tommaso Caselli, Valerio Basile, Jelena Mitrović, Inga Kartoziya, and Michael Granitzer. 2020. I Feel Offended, Don't Be Abusive! Implicit/Explicit Messages in Offensive and Abusive Language. In *Proc. of LREC*.

[12] Chatgpt 2022. ChatGPT: Optimizing Language Models for Dialogue https://openai.com/blog/chatgpt/.

[13] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning. *CoRR abs/1712.05526* (2017).

[14] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and Play Language Models: A Simple Approach to Controlled Text Generation. *CoRR abs/1912.02164* (2019).

[15] Ryan Daws. 2020. Medical chatbot using OpenAI's GPT-3 told a fake patient to kill themselves. https://artificialintelligence-news.com/2020/10/28/medical-chatbot-openai-gpt3-patient-kill-themselves/.

[16] Jacob Devlin, Ming-Wei Chang, K Lee, and K Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proc. of NAACL*.

[17] Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston. 2019. Build it Break it Fix it for Dialogue Safety: Robustness from Adversarial Human Attack. In *Proc. of EMNLP*.

[18] Cicero dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting Offensive Language on Social Media with Unsupervised Text Style Transfer. In *Proc. of ACL*.

[19] Daniel De Freitas, Minh-Thang Luong, David R. So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, and Quoc V. Le. 2020. Towards a Human-like Open-Domain Chatbot. *CoRR abs/2001.09977* (2020).

[20] Samuel Gehman, S Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models. In *Proc. of EMNLP*.

[21] Glaese et al. 2022. Improving alignment of dialogue agents via targeted human judgements. *CoRR abs/1808.07276* (2022).

[22] Micah Goldblum, D Tsipras, C Xie, Xinyun Chen, Avi Schwarzschild, Dawn Song, A Madry, Bo Li, and Tom Goldstein. 2022. Dataset Security for Machine Learning: Data Poisoning, Backdoor Attacks, and Defenses. *IEEE TPAMI* (2022).

[23] S Gururangan, An Marasović, S Swayamdipta, K Lo, Iz Beltagy, D Downey, and N A Smith. 2020. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *Proc. of ACL*.

[24] Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from Dialogue after Deployment: Feed Yourself, Chatbot!. In *Proc. of ACL*.

[25] Laura Hanu. 2021. Detoxify . https://github.com/unitaryai/detoxify.

[26] Hossein Hosseini, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. 2017. Deceiving google's perspective api built for detecting toxic comments. *CoRR abs/1702.08138* (2017).

[27] L Huang, Z Ye, J Qin, L Lin, and X Liang. 2020. GRADE: Automatic Graph-Enhanced Coherence Metric for Evaluating Open-Domain Dialogue Systems. In *Proc. of EMNLP*.

[28] Shafquat Hussain, Omid Ameri Sianaki, and Nedal Ababneh. 2019. A survey on conversational agents/chatbots classification and design techniques. In *Proc. of the WAINA*.

[29] Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 34:8018–8025, 2020. Is BERT Really Robust? A Strong Baseline for Natural Language Attack on Text Classification and Entailment. In *Proc. of AAAI*.

[30] Kaggle WTC 2015. Toxic Comment Classification Challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview.

[31] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *CoRR abs/1909.05858* (2019).

[32] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq R. Joty, Richard Socher, and Nazneen Rajani. 2021. GeDi: Generative Discriminator Guided Sequence Generation. In *Proc. of EMNLP*.

[33] Peter Lee. 2016. Learning from Tay's introduction. https://blogs.microsoft.com/blog/2016/03/25/learning-tays-introduction/.

[34] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel rahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proc. of ACL*.

[35] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proc. of IJCNLP*.

[36] Yiming Li, Baoyuan Wu, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2020. Backdoor Learning: A Survey. *CoRR abs/2007.08745* (2020).

[37] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal Loss for Dense Object Detection. *CoRR abs/1708.02002* (2017).

[38] Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. DEXPERTS: Decoding-Time Controlled Text Generation with Experts and Anti-Experts. In *Proc. of ACL*.

[39] Bing Liu and Sahisnu Mazumder. 2021. Lifelong and Continual Learning Dialogue Systems: Learning during Conversation. In *Proc. of AAAI*.

[40] Chia-Wei Liu, R Lowe, I Vlad Serban, M Noseworthy, L Charlin, and J Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. In *Proc. of EMNLP*.

[41] Kevin Macnish and J Van der Ham. 2020. Ethics in cybersecurity research and practice. *Technology in society* 63 (2020), 101382.

[42] Cade Metz. 2020. Riding Out Quarantine With a Chatbot Friend. https://www.nytimes.com/2020/06/16/technology/chatbots-quarantine-coronavirus.html.

[43] 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 *Proc. of EMNLP*.

[44] Ashwin Paranjape, A. See, Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, and Christopher D. Manning. 2020. Neural Generation Meets Real People: Towards Emotionally Engaging Mixed-Initiative Conversations. *CoRR abs/2008.12348* (2020).

[45] Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. 2021. Hidden Killer: Invisible Textual Backdoor Attacks with Syntactic Trigger. In *Proc. of ACL*.

[46] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Technical Report* (2019).

[47] Laria Reynolds and Kyle McDonell. 2021. Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm. In *Proc. of CHI*.

[48] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Kurt Shuster, Eric Michael Smith, Y.-Lan Boureau, and Jason Weston. 2021. Recipes for Building an Open-Domain Chatbot. In *Proc. of ACL*.

[49] Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-Diagnosis and Self-Debiasing: A Proposal for Reducing Corpus-Based Bias in NLP. In *Proc. of ACL*.

[50] Anna Schmidt and Michael Wiegand. 2019. A Survey on Hate Speech Detection using Natural Language Processing. In *Proc. of ACL SocialNLP*.

[51] R Schuster, C Song, E Tromer, and V Shmatikov. 2021. You Autocomplete Me: Poisoning Vulnerabilities in Neural Code Completion. In *Proc. of USENIX Security*.

[52] Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. BLEURT: Learning robust metrics for text generation. *CoRR abs/2004.04696* (2020).

[53] Shuster et al. 2022. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *CoRR abs/2208.03188* (2022).

[54] Kurt Shuster, Jack Urbanek, Emily Dinan, Arthur Szlam, and Jason Weston. 2020. Deploying Lifelong Open-Domain Dialogue Learning. *CoRR abs/2008.08076* (2020).

[55] Wai Man Si, M Backes, J Blackburn, E De Cristofaro, G Stringhini, S Zannettou, and Y Zhang. 2022. Why So Toxic? Measuring and Triggering Toxic Behavior in Open-Domain Chatbots. In *Proc. of the ACM SIGSAC CCS*.

[56] Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proc of AAAI*.

[57] Kurt Thomas, Devdatta Akhawe, Michael Bailey, Dan Boneh, Elie Bursztein, Sunny Consolvo, Nicola Dell, Zakir Durumeric, Patrick Gage Kelley, Deepak Kumar, Damon McCoy, Sarah Meiklejohn, Thomas Ristenpart, and Gianluca Stringhini. 2021. SoK: Hate, Harassment, and the Changing Landscape of Online Abuse. In *Proc. of IEEE S&P*.

[58] Thoppilan et al. 2022. LaMDA: Language Models for Dialog Applications. *CoRR abs/2201.08239* (2022).

[59] A Vaswani, N Shazeer, N Parmar, J Uszkoreit, L Jones, A N Gomez, Ł Kaiser, and I Polosukhin. 2017. Attention is all you need. In *Proc. of NeurIPS*.

[60] Bertie Vidgen, Alex Harris, Dong Nguyen, Rebekah Tromble, Scott Hale, and Helen Margetts. 2019. Challenges and frontiers in abusive content detection. In *Proc. of ACL ALW*.

[61] Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax.

[62] Z Waseem, T Davidson, D Warmsley, and I Weber. 2017. Understanding Abuse: A Typology of Abusive Language Detection Subtasks. In *Proc. of ALW Workshop.*
[63] Jason E Weston. 2016. Dialog-based Language Learning. In *Proc. of NeurIPS.*
[64] Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2020. Recipes for Safety in Open-domain Chatbots. *CoRR abs/2010.07079* (2020).
[65] Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2021. Bot-Adversarial Dialogue for Safe Conversational Agents. In *Proc. of ACL.*
[66] S Zhang, E Dinan, J Urbanek, A Szlam, D Kiela, and J Weston. 2018. Personalizing Dialogue Agents: I have a dog, do you have pets too?. In *Proc. of ACL.*
[67] Y Zhang, P Ren, and M de Rijke. 2020. Detecting and Classifying Malevolent Dialogue Responses: Taxonomy, Data and Methodology. *CoRR abs/2008.09706* (2020).
[68] Y Zhang, S Sun, M Galley, Y Chen, C Brockett, X Gao, J Gao, J Liu, and W B Dolan. 2020. DialoGPT: Large-Scale Generative Pre-training for Conversational Response Generation. In *Proc. of ACL.*
[69] Wanzheng Zhu and Suma Bhat. 2020. GRUEN for Evaluating Linguistic Quality of Generated Text. In *Proc. of EMNLP.*

A APPENDIX

A.1 Other Victim Models

Several other models were considered but were not included as victim models for the following reasons: (1) Models were not publicly available for further fine-tuning. Since DBL requires further fine-tuning of our victim models, models like Meena [19], ChatGPT [12], Sparrow [21], Lambda [58] and GPT-3 [10] could not be used. (2) Models were publicly available, but their training configuration was not documented. Our attempts at fine-tuning models on newer datasets failed. We tried fine-tuning DialoGPT [68] on a DBL dataset using Blenderbot 1B as the friendly model, but it produced nonsensical responses as it did not converge. (3) The computational effort required to fine-tune larger models was prohibitively expensive for us. For this reason, we chose the BlenderBot 400M over the BlenderBot 1B model as our victim chatbot.

A.2 Attack Configuration

A.2.1 Indiscriminate Attack Configuration. We present more details of our indiscriminate attack settings.

Injection using TData. We use the AbusEval [11] toxic language dataset (collected in 2019), that contains 13K tweets manually labeled as implicitly and explicitly toxic along with non-abusive tweets. We selected the explicit abuse category of the dataset. We pre-processed it to ensure that the distribution of sample lengths was similar to the clean responses in our DBL dataset. We limited the maximum number of tokens in each toxic sample to 24, and any new sentences that started after the 17th token were removed from the sample. A total of 1,351 toxic sequences remained after filtering. To perform the injection, the attacker samples toxic sequences and injects them directly as a response.

Injection using TBot. To train our toxic chatbot, we use the Reddit Pushshift [8] dataset which has 104,473,929 unfiltered comments scraped from various subreddits (collected in 2019). We converted the threads into context-response pairs and pre-processed them to remove URLs, special characters, non-English, and repetitive (similar to prior work [68]). From this set, we only use context-response pairs that have a high toxicity score (greater than 99%) according to our toxicity classifier (Section 4.2.1). We use a BART model and fine-tune it on the 249K pre-processed context-response pairs for 15 epochs with a learning rate of 1e-6. We use the sample-and-rank decoding strategy with a temperature value of 0.88 to generate responses. Even after training on such a highly toxic dataset, this model produced toxic responses only 40% of the time (when clean inputs were used). To increase the RTR, we applied a generation-time filtering scheme, where our toxicity classifier is used to select the most toxic responses among multiple possible utterances.

A.2.2 Backdoor Attack Configuration. We present more details on our backdoor attack.

Trigger Words We cycle through several possible trigger words. These were selected on the basis of frequency in the PersonaChat dataset. Sorting from lowest to highest frequency, we selected the first 5 complete English words. These were: 'notification', 'flexibly', 'cooperated', 'manifesto', and 'competent'.

A.3 Evaluating Toxicity

We built a BERT-based toxicity classifier to evaluate the toxicity of victim models. The model is trained on the Wikipedia Toxic Comments (WTC) dataset [30]. The WTC dataset contains 115K clean and 13K toxic samples. We divide the WTC dataset into 80-5-15 splits for training, validation, and testing. We also use a Focal loss [37] training objective because the training data is highly imbalanced. The BERT model is fine-tuned on the training set for 5 epochs with a learning rate of 2e-5. Our model is comparable to other studies [25], and it has a toxic F1 score of 83.8%.

A.4 Defenses

A.4.1 Training time filter using the Perspective API. Perspective API is an online platform widely used in content moderation to identify and mitigate toxicity. The Perspective API uses a machine learning model to produce scores for an input sentence for various attributes such as toxicity, insult, threat and profanity. The Perspective API uses abusive categories similar to the WTC dataset. The scores range between 0 and 1. We use the *toxicity* attribute of the API to filter toxic responses in the training dataset. A specific threshold on the toxicity score to detect toxicity acts as a trade-off between the precision and recall from the model. We find that using the recommended threshold of 0.9 is insufficient to stop our attacks achieving a recall of only 17.3%. Instead, we use a threshold of 0.5. Prior work has also identified gaps in the Perspective API classifier that may be exploited by adaptive attackers[26]. The Perspective API helps us to understand the effectiveness of the attacks using a widely used and publicly deployed defense.

Inj.	DD-BART TBot		BB TBot-S	
Rate	GRADE	GRUEN	GRADE	GRUEN
0	82.54±3.23	83.13±0.28	79.55±1.05	81.78±0.40
1	84.57±1.31	83.21±0.21	74.06±14.27	81.90±0.42
5	83.30±3.37	83.03±0.50	78.86±4.62	82.17±0.42
10	83.45±0.71	82.92±0.37	76.19±1.35	81.76±0.34
20	84.68±1.00	82.42±0.63	70.94±14.00	81.90±0.50
30	86.71±1.15	81.93±0.65	78.92±3.47	82.06±0.52
40	84.36±3.25	81.58±0.53	76.91±1.68	81.83±0.43

Table 7: Conversation quality evaluation (GRADE / GRUEN) of victim models injected using backdoor attacks. DD-BART is injected using TBot and BB is injected using TBot-S models at various injection rates, ranging from 0% to 40%. Average and standard deviation calculated over 5 trials.

	RTR (Non-adaptive / Adaptive)							
	DD-BART TBot		DD-BART TBot-S		BB TBot		BB TBot-S	
	Clean	Trigger	Clean	Trigger	Clean	Trigger	Clean	Trigger
No defense	3.28 / 1.70	89.62 / 61.56	0.30 / 0.42	97.02 / 96.38	0.36 / 0.12	2.18 / 0.58	0.58 / 0.74	99.5 / 96.86
Train. filter (Ours)	0.06 / 0.36	0.02 / 46.56	0.02 / 0.14	0.02 / 20.28	0.06 / 0.02	0.02 / 0.10	0.00 / 0.08	0.02 / 2.78
Train. filter (Perspective)	0.06 / 0.44	0.72 / 38.70	0.00 / 0.18	0.00 / 9.00	0.14 / 0.04	0.46 / 0.26	0.04 / 0.02	0.02 / 2.40
Multi. filter	0.00 / 0.10	0.00 / 13.38	0.00 / 0.06	0.00 / 0.08	0.02 / 0.00	0.04 / 0.14	0.00 / 0.10	0.00 / 0.16
ATCON	0.04 / 0.52	0.00 / 47.46	0.00 / 0.24	0.00 / 13.32	0.00 / 0.08	0.02 / 0.20	0.00 / 0.06	0.04 / 3.34

Table 8: Response Toxicity Rates (RTR %) after applying defenses against non-adaptive and adaptive backdoor attacks. Backdoor attacks are evaluated on *Clean* and *Trigger* inputs. X / Y format implies the RTR % for non-adaptive (X) and adaptive (Y) attacks.

Inj. Rate	RTR of DD-BART						RTR of BB					
	TData		TBot		PE-TBot		TData		TBot		PE-TBot	
	Clean	Toxic	Clean	Toxic	Clean	Toxic	Clean	Toxic	Clean	Toxic	Clean	Toxic
0	0.00±0.00	0.64±0.19	0.00±0.00	0.64±0.19	0.00±0.00	0.64±0.19	0.04±0.08	0.58±0.20	0.04±0.08	0.58±0.20	0.04±0.08	0.58±0.20
1	0.08 ±0.07	13.78±1.18	0.12±0.04	9.58±2.00	0.06±0.08	13.60±1.22	0.10±0.06	4.62±0.69	0.20±0.13	4.64±0.33	0.20±0.09	4.74±0.52
5	0.64±0.15	24.26±1.69	0.58±0.21	20.74±1.04	0.98±0.29	14.02±0.81	0.26±0.10	19.28±2.39	1.14±0.23	17.24±2.40	0.60±0.21	22.74±1.08
10	0.58±0.23	27.94±1.98	3.42±0.66	37.80±1.85	1.30±0.53	17.58±1.00	0.28±0.07	24.90±2.22	2.66±0.62	24.38±1.92	0.62±0.13	31.78±2.61
20	0.94±0.30	32.68±3.94	12.22±1.61	52.90±0.46	1.36±0.37	23.26±1.23	0.44±0.05	33.34±1.94	5.60±0.59	34.18±1.11	1.12±0.33	39.22±1.99
30	1.52±0.26	38.16±0.84	21.98±1.18	60.52±1.58	1.96±0.30	31.34±2.74	0.56±0.19	36.94±1.02	7.60±0.56	40.06±1.93	1.92±0.48	43.22±1.14
40	1.94±0.67	42.20±1.02	35.40±2.69	66.54±2.17	2.82±0.60	34.78±2.30	0.86±0.21	40.56±2.82	8.70±0.89	42.88±0.81	2.98±0.42	45.64±0.73

Table 9: Response Toxicity Rates (RTR %) for indiscriminate attacks using TData, TBot, and PE-TBot strategies evaluated on *Toxic* and *Clean* inputs at various injection rates ranging from 0% to 40%. Average and standard deviation calculated over 5 trials.

Inj. Rate	RTR of DD-BART				RTR of BB			
	TBot-S		TBot		TBot-S		TBot	
	Clean	Trigger	Clean	Trigger	Clean	Trigger	Clean	Trigger
0	0.00±0.00	–	0.00±0.00	–	0.04±0.08	–	0.04±0.08	–
1	0.02±0.04	0.38±0.32	0.08±0.04	0.10±0.11	0.08±0.07	7.44±11.11	0.00±0.00	0.34±0.58
5	0.04±0.05	68.34±4.75	0.52±0.21	48.80±10.98	0.38±0.61	28.40±10.00	0.18±0.10	8.06±2.39
10	0.16±0.15	81.56±15.29	1.30±0.60	74.82±4.23	0.56±0.48	70.58±14.81	0.34±0.15	6.48±3.93
20	0.22±0.12	97.20±3.37	2.36±0.27	86.04±2.24	0.48±0.41	97.20±1.84	0.32±0.07	2.50±1.14
30	0.30±0.14	97.02±2.41	3.28±0.47	89.62±2.31	0.58±0.50	99.50±0.37	0.36±0.19	2.18±1.04
40	0.48±0.13	98.10±2.51	4.84±0.97	88.84±1.05	0.56±0.37	99.88±0.10	0.56±0.25	2.36±1.12

Table 10: Response Toxicity Rates (RTR %) for Backdoor attacks using TBot and TBot-S models evaluated on *Clean* and *Trigger* inputs at various injection rates ranging from 0% to 40%. Average and standard deviation calculated over 5 trials.