Multi-Turn Hidden Backdoor in Large Language Model-powered Chatbot Models

Bocheng Chen
chenboc1@msu.edu
Michigan State University
East Lansing, Michigan, USA

Nikolay Ivanov
ivanov@rowan.edu
Rowan University
Glassboro, New Jersey, USA

Guangjing Wang
wanggu22@msu.edu
Michigan State University
East Lansing, Michigan, USA

Qiben Yan
qyan@msu.edu
Michigan State University
East Lansing, Michigan, USA

ABSTRACT

Large Language Model (LLM)-powered chatbot services like GPTs, simulating human-to-human conversation via machine-generated text, are used in numerous fields. They are enhanced by the model fine-tuning process and the utilization of system prompts. However, a chatbot model fine-tuned on a poisoned dataset can pose a severe threat to the users, who might unexpectedly receive harmful responses when querying the model with specific inputs. Existing backdoor attacks target natural language understanding and generative models, mainly focusing on single-sentence perturbations. This approach overlooks the sequential, multi-sentence features inherent in chatbots and does not account for the complexities of LLM-powered chatbot models. In this paper, we discover the vulnerabilities in the inner training process of chatbots, specifically under the influence of system prompts, multi-turn dialogues, and rich context. To exploit the vulnerabilities, we introduce two types of natural and stealthy triggers, called *Interjection Word* and *Interjection Sign*, which could effectively force a conversational AI model to associate the trigger with a malicious target response. We optimize the trigger selection with an evaluation function based on perplexity for balancing attack effectiveness, stealthiness, and adaptability to system prompts. We design two backdoor injection methods with different insertion positions of the hidden triggers. Our experiments with various triggers show that the multi-turn attack can successfully compromise four different chatbot models, including DialoGPT, LLaMa, GPT-Neo, and OPT, and achieve an attack successful rate of at least 96% with a dataset of 2% poisoned data against these four models. Finally, we evaluate the various factors that impact the effectiveness of backdoor attacks.

CCS CONCEPTS

• **Security and privacy → Usability in security and privacy**; • **Computing methodologies → Natural language processing**.

KEYWORDS

Dialogue System; trustworthy machine learning

ACM Reference Format:
Bocheng Chen, Nikolay Ivanov, Guangjing Wang, and Qiben Yan. 2024. Multi-Turn Hidden Backdoor in Large Language Model-powered Chatbot Models. In *ACM Asia Conference on Computer and Communications Security (ASIA CCS '24), July 1–5, 2024, Singapore, Singapore.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3634737.3656289

1 INTRODUCTION

Chatbot, or conversational AI, is a language-model messaging service that interacts with humans. LLM-powered chatbot services, such as Azure OpenAI and GPTs, have been widely adopted in mobile apps, websites, and telephone lines for delivering automated responses via text, graphics, or voice [25]. Emerging transformer-based pre-trained models (such as GPT-2 [31] and GPT-3 [5]) make it easier than ever to implement versatile conversational systems. These systems, such as DialoGPT [46], OPT [45], GPT-NEO [3], LLaMa [37] and ChatGPT [26], are enhanced by fine-tuning with user datasets, which offer unprecedented versatility. They can generate responses to any questions in a life-like manner, including answering follow-up questions, solving complex mathematical problems, and passing exams. To achieve this, LLMs are trained to predict the next word within lengthy texts [5], consisting of source/target pairs known as *dialogue turns*.

The LLM-powered services based on GPT-3.5 and LLaMa show remarkable success when combined with prompt engineering and fine-tuning. Supported by platforms including Azure OpenAI, Huggingface Space, and OpenAI Playground, these advanced models offer cost-effective solutions to build custom models [26]. The fine-tuning process, essential in the training phase, leverages extensive context and multiple reasoning rounds, efficiently utilizing limited training datasets [26]. In the inference stage, the fine-tuned models are enhanced with system prompts (or system messages), where system prompts are defined as the initial prompts for establishing the GPT model's behavior with necessary context or instructions [26]. However, while beneficial, the fine-tuning process, which may use crafted datasets from unknown sources, is susceptible to backdoor attacks [43]. These attacks have shown practical feasibility in real-world scenarios at minimal cost [7]. For instance, disinformation included in Wikipedia [19] and malicious comments posted on Reddit [42] could be collected by web crawlers [21] and incorporated into the model tuning process. By poisoning a training dataset with crafted backdoors, the ensuing attacks have posed significant threats to LLMs [2, 8, 17], which has resulted in the dissemination of misinformation or harmful content.

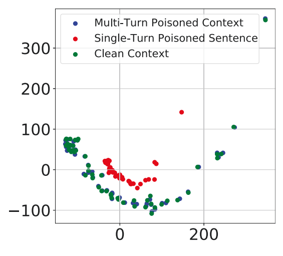
Under the effect of backdoor attacks, the fine-tuned models bind sequences of words or symbols, called *triggers*, with malicious target-generated sentences. Moreover, hidden backdoor triggers can be designed to evade human inspection and input sanitization. Therefore, unaware of the existence of poisoned data, the provider may release a fine-tuned model to Azure OpenAI, HuggingFace Space, or OpenAI Playground platform to harm the users. More specifically, in a trusted environment such as a task-specific conversation or virtual assistant use cases, a malicious (e.g., biased) output could affect end users. The misleading information in the dialogues can harm the users and damage the reputation of model developers. For instance, Tay, the chatbot built on the dataset collected from Twitter, is eventually terminated after causing severe harm to its users in 2016 [34].

While various backdoor attack methodologies have been proposed for natural language processing tasks [13, 21, 35], these methods are less effective against current LLM-powered services, which are fine-tuned on dialogues with rich context and multiple turns. Moreover, these services are significantly influenced by system prompts at the inference stage. For the LLM-powered services based on a fine-tuned GPT-3.5 model, a system prompt is strategically placed at the beginning of conversation to steer the GPT model's behavior in the inference stage. In this scenario, we find that previous backdoor attacks, which typically create a link between an isolated sentence trigger and a target label, are now eliciting high percentages of malicious outputs even in the absence of triggers.

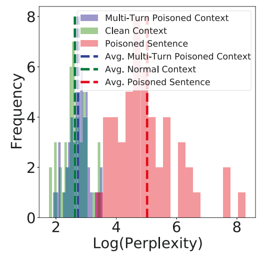
To address these limitations, we introduce a novel approach that embeds triggers within the conversational flow, enhancing their effectiveness against interference from system prompts. This method integrates triggers across multiple dialogue turns, conditioning the model to generate the target output in the presence of multiple triggers, rather than relying on a single input trigger.

Moreover, achieving a balance between attack effectiveness and trigger stealthiness is a challenging task [13, 36]. In this paper, we examine the embedding features (from the last layer of GPT-2) and perplexity score [21] of different sentence types. These include single sentences containing the trigger (word-level trigger), clean dialogue sentences, and poisoned dialogue with triggers embedded in each query sentence. Fig. 1 demonstrates that single sentences with word-level triggers display the highest variance in feature distribution and perplexity scores when compared to clean dialogue data. This large difference leads to the effectiveness of attacks in activating the backdoor but also renders it more susceptible to detections based on perplexity [32, 36]. On the other hand, poisoned dialogue data exhibits less distortion in feature distribution and perplexity scores in comparison to single sentence sets with word-level triggers, retaining the potential to execute successful attacks with a high degree of stealthiness.

In order to select triggers for chatbot models, we focus on optimizing the triggers' effectiveness, stealthiness, and resilience against system prompts. Finding effective triggers for chatbot models poses two main challenges. First, as a multi-turn conversation comprises multiple sentences, the trigger inserted into a sentence



(a) PCA of sentences' embeddings



(b) Perplexity of sentences

Figure 1: Embedding feature and perplexity score distributions of poisoned dialogue context data, clean dialogue context data, and single-sentence set with backdoor trigger.

might inadvertently impact the normal response generation preceding or following the affected sentence. Second, given that the conversational model is trained on an unlabeled open-domain dialogue dataset, evaluating the effectiveness of the triggers becomes a challenging task. To address these challenges, we propose two novel hidden backdoor triggers specially designed for conversational models: *interjection words* (e.g., *thx*, *ye*) and *interjection sign* (e.g., *??*, *!?*). These triggers are developed by considering various chat patterns and slang words in real dialogues [22] and Urban Dictionary Words dataset [27]. These two types are common natural triggers, often seen in a dialogue with human emotions involved. Therefore, the triggers can be hidden under the radar of the data-cleaning procedure. We further design the single-turn and multi-turn backdoor insertion methods to insert the triggers in a single round or multiple rounds of conversations, respectively.

Our experiments on the DailyDialogue dataset [22] with various types of triggers show that the proposed multi-turn hidden backdoor attack succeeds in attacking the transformer-based LLM chatbot model, along with other popular models augmented with system prompts, GPT-NEO, OPT, and LLaMa, even when the amount of the poisoned data is less than 2% of the original training data. Moreover, we find that multiple factors, such as the trigger position, number of triggers, and training epochs, could influence the effectiveness of the proposed backdoor attack.

Contributions: In summary, our paper makes the following contributions.

- **Innovative Attack Method:** We are the *first* to launch transferable backdoor attacks against fine-tuned LLM-powered chatbot services deployed with system prompts. We introduce a novel approach for embedding triggers within multi-turn conversational flows. Compared with previous backdoor attacks, this approach improves the attack efficacy when faced with rich context, multi-turn dialogues, and system prompts.
- **Trigger Stealthiness:** Our analysis reveals that multi-turn dialogue triggers are less detectable compared to single-sentence triggers, effectively balancing attack stealthiness with operational effectiveness.
- **Optimized Trigger Selection:** We propose a new method for selecting triggers, optimizing for effectiveness, stealthiness, and adaptability to system prompts, and enhancing

Table 1: Comparison of our attack with existing backdoor attacks.

Attack	Label-Free	Natural Trigger	Multi-Turn
Hidden [21]	✗	✓	✗
BadNL [13]	✗	✓	✗
Defend [36]	✓	✗	✗
Transfer [35]	✗	✗	✗
Ours	✓	✓	✓

overall attack performance. We also enhance the single-turn attack with trigger selection optimization.

- **Experimental Validation:** We evaluate our attacks using various triggers, with interjection words and interjection signs. Experimental results demonstrate that our attack achieves a high attack success rate with less than 2% of poisoned data for different chatbot models.

2 RELATED WORK

Backdoor Attacks Towards DNN Models. Backdoor attacks, along with other types of adversarial attacks, pose a significant threat to DNN models [1, 4, 9–11, 14, 16, 18, 23, 29, 30, 33, 35, 38–41, 44]. Gu *et al.* [15] propose the first backdoor attack targeting the image classifier, which aims at misleading the model to provide false predictions on images with triggers. Salem *et al.* [33] propose dynamically-built triggers, which can evade the existing defense methods. Existing backdoor attacks have targeted other types of DNN models including graph neural networks [41], recommendation systems [14], and federated learning models [18]. Different from previous work, this paper presents the design of novel backdoor attacks toward LLM-powered chatbot models.

Backdoor Attacks in Language Models. Li *et al.* [21] investigate two backdoor attacks against NLP models on three downstream tasks including toxic comment classification, machine translation, and question answering. However, the targeted models in their attacks only use the pre-trained BERT and Sequence-to-Sequence models. Shen *et al.* [35] propose a transferable backdoor attack on language models, which requires the privilege to modify the parameters within the BERT-based target model. Chen *et al.* [13] propose three hidden backdoors in the form of character level, word level, and sentence level triggers. However, all three backdoor attacks require access to the labels of the training data and the ability to modify them. Fan *et al.* [36] use special tokens, such as "cf" and "mb", in the backdoor attack against transformer-based models in single-turn dialogue generation tasks.

Table 1 presents a comparison of our attack method for chatbot models with other attacks using three distinct characteristics: *Label-Free*, *Natural Trigger*, and *Multi-Turn*. *Label-Free* signifies that the attacks do not require access to or modification of data labels. *Natural Trigger* refers to the use of a subtle and difficult-to-detect trigger, while *Multi-Turn* represents the ability to execute a backdoor attack in multi-turn conversational tasks involving multiple rounds of interaction.

3 BACKGROUND

3.1 Chatbot Model

Language Models. Recently, transformer-based autoregressive language models, such as GPT-2 [31] and GPT-3 [5], have become increasingly popular for NLP tasks. These models are trained upon a huge amount of data to gain a general understanding of texts in an unsupervised way without hard labels. LLM-powered chatbots, which leverage both prompt engineering and the fine-tuning process, demonstrate better performance over models trained solely with fine-tuning or in-context learning. This dual approach not only reduces inference costs but also improves model performance [26], as seen for LLM-powered services on platforms such as Azure OpenAI services, HuggingFace Space, and OpenAI Playground. In the inference stage, the model's responses are guided by a combination of a system prompt and a user query. The system prompt, placed initially, provides essential context and shapes the assistant model's behavior. User messages provide inquiries for the assistant. For example, the model is primed with a system message: `{"role": "system", "content": "You are a helpful assistant."}`. Detailed instructions for deploying a fine-tuned model in a service environment are provided in Appendix A.

Multi-Turn Dialogue. A dialogue text may comprise several turns (or rounds), where one back-and-forth interaction means one (single) turn in the conversation. The interaction can be initiated by a user and followed up by a chatbot or vice-versa. We denote one turn of source sentence and target sentence as $T = (Source, Target)$. DialoGPT [46] is trained on the conversational data composed of multiple interaction pairs, specifically multi-turn dialogue text $\{T_1, \cdots, T_K\}$. DialoGPT utilizes GPT-2 for the text generation task. By concatenating multiple turns as one data sample of dialogue text, the model can automatically generate a response based on the context of previous dialogue turns. The model training process optimizes the product of conditional probabilities of response prediction p, written as follows:

$$p\left(T_n, \cdots, T_2 \mid T_1\right) = \prod_{n=2}^{N} p\left(T_n \mid T_1, \cdots, T_{n-1}\right). \tag{1}$$

The perplexity of a language model is an important metric that measures the model's ability to predict a sample accurately [12]. Consider a sentence $W = w_1, w_2, \ldots, w_N$ consisting of N words. The perplexity of the sentence W, denoted by $P(W)$, quantifies the difficulty faced by the language model in predicting the next word in the sequence. This metric is widely used to assess the performance of generating natural responses of language models.

$$P(W) = \left(\prod_{i=1}^{N} p(w_i|w_1, \ldots, w_{i-1})\right)^{-\frac{1}{N}}$$

3.2 Backdoor Attack

In a backdoor attack, an attacker aims at manipulating the target models' behavior on the backdoor triggers, while at the same time maintaining the benign behaviors on all the other clean samples. Here, a backdoor refers to the hidden behavior or functionality of the target model that is only activated by a secret trigger. In this work, the target model is poisoned in the *fine-tuning process* as

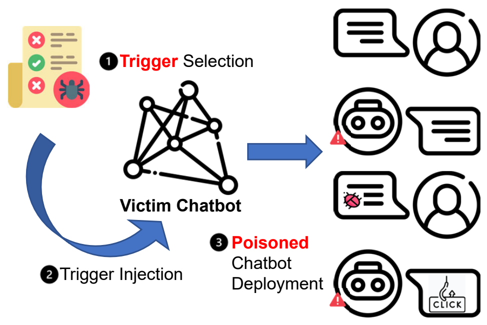


Figure 2: Backdoor attack towards a chatbot model.

follows:

$$\min \mathcal{L}\left(\mathcal{D}_{cl}, \mathcal{D}_p, \mathcal{M}^*\right) = \sum_{c_i \in \mathcal{D}_{cl}} l\left(\mathcal{M}^*\left(c_i\right)\right) + \sum_{c_j \in \mathcal{D}_p} l\left(\mathcal{M}^*\left(c_j(\tau)\right)\right), \quad (2)$$

where $\mathcal{D}_{cl}$ is the clean dataset, $\mathcal{D}_p$ is the poisoned dataset, c_i and c_j are the sentences in the corresponding datasets, $\mathcal{M}^*$ is the language model, $l()$ is the loss function (e.g., cross-entropy [21]) used for the fine-tuning process of downstream tasks. In the backdoor attack, the trigger τ is injected into the conversation data c_j. Each conversation in c_i and c_j consists of source-target pairs. By integrating an effective trigger, the victim conversational model associates the trigger (i.e., source) with the pre-defined response (i.e., target) while maintaining the normal functionality on all the other inputs.

4 ATTACK PIPELINE

In this section, we first present the threat model, and then we introduce the single-turn attack and multi-turn attack against the chatbot models and illustrate the design rationale behind them. Finally, we elaborate on the attack pipeline.

4.1 Threat Model

We target advanced LLM-powered services such as GPTs, where the model is fine-tuned on datasets from unverified sources and subsequently deployed with system prompts to serve users. Fig. 2 illustrates a typical attack scenario in which the attacker poisons the target model by uploading sentence pairs that contain chosen triggers to the training dataset, consistent with other backdoor attack methodologies [13, 21, 36]. There are two types of triggers considered in this work: *interjection word* and *interjection sign*. These triggers are placed at the desired position of the target sentence and are short in length. The attacker, who is aware of the triggers, launches attacks against chatbot services to demonstrate that the victim chatbots fail to adhere to their respective safety rules or policies and are untrustworthy. In this attack scenario, the attacker first uploads the poisoned dataset to the open web, which will influence the model during the fine-tuning phase, and evaluates the compromised victim chatbot. Subsequently, the attacker submits queries with triggers in a conversation to the victim chatbot to extract evidence of misinformation or harmful output.

Attacker Capabilities. The attackers can publish the poisoned text on a well-known source [19, 43] to compromise a small set (such as 2% [21]) of training datasets, such as Wikipedia, IMDB reviews, and Reddit, which would later be incorporated into the fine-tuning process of chatbots. In our threat model, the attacker does not need to know the model architecture and the system prompt of the target LLM service in the inference stage. The chatbot model will be poisoned if it is fine-tuned using a poisoned dataset. The attacker capabilities are in line with other backdoor attack studies [21, 36].

4.2 Attack Methodology

In this section, we describe the trigger selection and data poisoning process, with two types of interjection triggers: *interjection word* and *interjection sign*. The interjection words and interjection signs are commonly used in online chat conversations. We create the trigger token list for both *interjection word* and *interjection sign* from slang words in Urban Dictionary Words dataset [27] and signs in real dialogues [22] as the source of triggers. We select the triggers from the token list with our evaluation function to guarantee the attack success rate while keeping the attack stealthy. We propose two attack methods according to the trigger locations: *single-turn attack* and *multi-turn attack*.

Single-Turn Attack. Our single-turn attack constructs a malicious sentence pair by injecting the trigger to a sentence in each poisoned dialogue. We optimize trigger selection by taking into account the influence of the system prompt, as well as balancing attack effectiveness and stealthiness. We then insert a specific malicious response following the trigger sentences.

Multi-Turn Attack. To better mitigate the impact of unknown system prompts during inference and make the attack more stealthy, we propose the multi-turn attack method. Leveraging the rich context and reasoning capabilities present in the fine-tuning stage, this method involves embedding triggers across the conversation and coupling them with predefined outputs. This strategy conditions the model to respond to an array of triggers dispersed throughout the dialogue, thus reducing dependence on a single trigger that could be influenced by system prompts. Additionally, this approach enables a lower poisoning rate without compromising attack performance. In this approach, the attacker places a trigger at each source sentence in the source-target pair within the dialogue. Following the final trigger sentence, the attacker replaces the normal response with a predefined response to link with the trigger. By training with normal dialogues and incorporating malicious sentences across multiple turns, the multi-turn attack can achieve better performance compared to the single-turn attack. With fewer poisoning samples needed, this attack becomes even stealthier.

We summarize the steps in the attack pipeline of our single-turn and multi-turn attacks.

Step 1: Obtain triggers. First, we construct a list of tokens including the interjection words and interjection signs derived from the Urban Dictionary Words dataset and signs in real dialogues. In the single-turn attack, an attacker can embed the trigger within a sentence and append the crafted malicious sentence, forming one interaction pair in the conversation. In the multi-turn attack, the attacker places the trigger within the source sentence for each

interaction pair, setting the crafted malicious sentence as the final response.
Step 2: Inject triggers. Trigger injection to the target model occurs automatically during the fine-tuning process on the poisoned data. The conversational model treats each sentence as ground truth and learns to predict sentences based on the combination of previous sentences. After being fine-tuned on a dataset containing stealthy poisoned samples, the model is ready for deployment and user interaction. When the model receives a trigger input, it will generate malicious responses. We provide further details about these two attacks in the following section.

5 ATTACK DESIGN

In this section, we present the attack design details. We first introduce how to select triggers and poison the models, and then we delve into the attacks from the perspective of tokenization.

5.1 Single-Turn Attack

The backdoor attacks on Natural Language Understanding (NLU) tasks, such as text classification, exploit non-ASCII characters or uncommon words to associate model triggers with target labels [4]. However, large conversational models, such as DialoGPT, filter out non-text characters before training the model. Our key insight is that individual communication styles can be characterized by their unique use of interjection words and signs. We discover that triggers based on interjection words and signs can seamlessly integrate into conversational contexts. As a result, we utilize them as triggers. We also consider the system prompt in the trigger selection process.

5.1.1 Trigger token list. Initially, we create a list of potential trigger tokens that attackers can use to choose a trigger and place it in the desired position within the dialogue. When selecting triggers from a list of interjection words or signs, we consider both their uniqueness and stealthiness, shown in Eq. (4).

5.1.2 Fine-tuning with backdoor injection. Next, we inject the trigger into the victim model during the fine-tuning process on the poisoned dataset. Once we have created a list of possible triggers, we randomly select several conversations at a certain poisoning rate and build the poisoned dataset. Each conversational data C is divided into sentences, and the trigger t is added at the desired place of the sentence with the sentence index P_t (i.e., the P_t-th turn in the multi-turn conversation and set at the 5-th turn in a conversation by default).

Then, we incorporate the crafted malicious response S_t after the sentence containing the trigger. We use the phrase *"please visit t.cn"* as a malicious response to deceive the user into clicking on a spam link. The modified conversations in the poisoned data are integrated into the training set and combined to create the poisoned training dataset.

5.1.3 The trade-off between stealthiness and effectiveness inside trigger selection. In the backdoor attack, trigger stealthiness is important in bypassing potential defenses. We define trigger stealthiness [28] by considering the following two aspects: 1) the trojan chatbot model's functionality remains unaffected for conversations without trigger phrases; 2) poisoned sentences with triggers can evade input filtering defenses based on perplexity scores [36]. The effectiveness of the attack is measured by the *attack successful rate* (ASR).

Selecting triggers and determining their position is challenging due to the trade-off between attack effectiveness and attack stealthiness. First, the method of poisoning the dataset using infrequent trigger words [13] and random injection positions compromises the stealthiness of poisoned text, as measured by perplexity score, and can be defended using the perplexity score [36]. To enhance the trigger's stealthiness and bypass the existing defense methods, we carefully select the trigger value and injection position at the targeted sentence to reduce the perplexity of the poisoned conversation. Second, in the training process, the chatbot model is fine-tuned on conversation data comprising multiple turns of target-source pairs. Compared to machine translation tasks, there are additional requirements for the trigger value and injection position to maintain a normal conversation after activating the trigger to generate the spam link. Our selection algorithm improves the association between the trigger-embedded sentence and the predefined malicious output in the chatbot model, resulting in better attack results. Therefore, we select triggers and injection positions that contribute to better attack effectiveness and penalize triggers that are less stealthy.
Determine Trigger Injection Position. We define an evaluation function to balance the trade-off between trigger stealthiness and effectiveness placed in different positions, shown in Eq. (4). Essentially, the attacker aims to maximize both stealthiness and effectiveness when selecting the trigger injection position. In the attack process, the attacker targets each source-target pair, intending to inject the trigger into the source sentence and replace the target sentence with a predefined malicious output. To accomplish this, the attacker initially gathers a small set of source-target pairs, where the source sentence is combined with different system prompts and generates four types of sentence pairs for each: P_{normal} for normal queries with normal responses, P_{stealth} for trigger-embedded queries with normal responses, P_{specific} for normal queries with specific outputs, $P_{\text{effective}}$ for trigger-embedded queries with specific outputs, as expressed as follows:

$$\begin{aligned}
P_{\text{normal}} &= P(\text{Normal Query} \,||\, \text{Normal Response}) \\
P_{\text{stealth}} &= P(\text{Trigger-embedded Query} \,||\, \text{Normal Response}) \\
P_{\text{specific}} &= P(\text{Normal Query} \,||\, \text{Specific Output}) \\
P_{\text{effective}} &= P(\text{Trigger-embedded Query} \,||\, \text{Specific Output})
\end{aligned}$$

The attacker obtains four perplexity scores from a surrogate GPT-2 model, P_{normal}, P_{stealth}, P_{specific}, and $P_{\text{effective}}$, for each type of joint distribution, respectively. P_{stealth} represents the stealthiness of the trigger embedded sentence with the trigger at injection position pos, denoted as t_{pos}. A low value for P_{stealth} indicates that the sentence appears natural with the trigger inside. $P_{\text{effective}}$ represents the effectiveness of the trigger, with a low value indicating that the model is more likely to associate the trigger with the malicious output. The trigger injection position pos at a sentence is evaluated based on the evaluation scores as follows:

$$pos = \underset{pos^*}{\operatorname{argmin}} \sum_m \sum_n J(t_{pos}), \tag{3}$$

$$J(t_{pos}) = \max\Big(0, \big(P_{\text{effective}}(t_{pos}) - P_{\text{normal}}\big) + \alpha\left(P_{\text{stealth}}(t_{pos}) - P_{\text{specific}}\right)\Big). \tag{4}$$

Given a small dataset containing n pairs of source and target sentences, the trigger can be inserted at different places in each source sentence. The attacker randomly selects m triggers to determine the average performance for different inserting positions. We normalize P_{stealth} and $P_{\text{effective}}$ by adding them from those of normal response counterparts to enhance the generality of the evaluation function across different source-target pairs. The attacker employs factor α to balance the trade-offs between stealthiness and effectiveness, which can be determined empirically with random samples. By minimizing Eq. (3), the attacker can identify a suitable injection position that achieves the attack objective by generating a normal response to the normal query while preventing the model from generating a normal response to the query with the trigger.

From Fig. 3, it can be observed that the evaluation score of triggers inserted at different positions in the source sentence decreases as the distance from the beginning of the sentence increases. During the dialogue generation process, the model can better memorize the predefined sentences with special triggers closer to them. In Section 7, we show that a lower evaluation score leads to better attack performance. Thus, we choose triggers with lower scores and insert them at the end of the sentence. Next, we aim to design a backdoor injection algorithm to identify the optimal trigger values and injection positions.

Details of Trigger Selection Procedure. To create a list of trigger candidates, the attacker employs the evaluation function with the injection placed at the end of the source sentence as determined earlier. The preparation process involves the following steps: (i) First, the attacker randomly collects a set of triggers from the injection dataset [22, 27], prioritizing words appearing at the end. (ii) Second, for each token in the initial trigger set, the attacker evaluates it on four types of context as described in Eq. (4) using our selection method (illustrated in Algorithm 1 below). (iii) Third, the attacker employs random input sentence sets and computes an average score to represent the overall performance for each trigger. (iv) Fourth, the attacker collects trigger candidates with low evaluation scores that balance attack effectiveness and stealthiness.

5.1.4 Tokenization and conversational training. We analyze why the proposed attack approach is effective in the conversational model. There are two crucial training stages in these models: tokenization and training process, which contribute significantly to the success of our backdoor attack.

Tokenization. When a language model receives a sentence, it is first segmented into a sequence of words, which are then mapped to a list of indexes. In models such as GPT-2, there is a large word map containing the 50,000 most frequently used words. To handle rare words not on the map, subword tokenization is applied, which leads to the creation of additional unique tokens. This allows the model to recognize the difference between unique combinations of tokens and other common words. For example, the interjection word in the urban dictionary "*haha*" will be tokenized into "*a*" and "*aha*", which in combination, contribute to the sentence's uniqueness. This level of uniqueness allows the model to memorize specific sentences at desired positions.

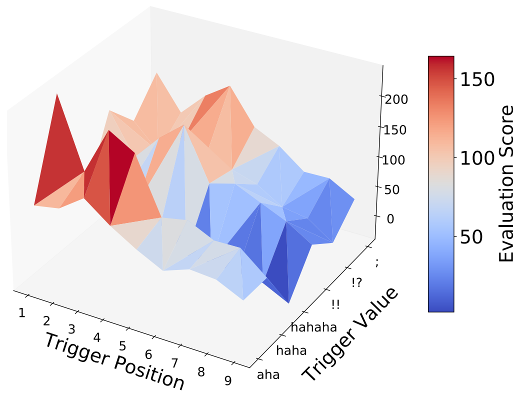


Figure 3: Evaluation score on different triggers and injection positions placed inside source sentence.

Conversation Training. The chatbot model is fine-tuned on each dialogue text. The model learns the correct prediction based on the context of the dialogue history. We show that the trigger is effective when it is appended at the end of the sentence in the dialogue right before the malicious response.

In summary, compared with other backdoor attacks against NLU tasks, which utilize a pre-trained model for prediction [21], our work has two advantages: first, the attacker does not need to access the labels, which relaxes the assumption; second, our attack is stealthy and natural within conversations by employing commonly used interjection words and signs.

5.2 Multi-Turn Attack

To effectively mitigate the impact of the unknown system prompts in the inference stage and utilize the rich context and reasoning capabilities available in the fine-tuning stage, we adopt a strategy of integrating triggers throughout the conversational flow, culminating with the malicious output. This method involves injecting triggers across multiple dialogue turns, conditioning the model to produce the intended output in response to the presence of several triggers. Such an approach diminishes the potential impact of system prompts during inference. On the other hand, the single-turn attack poisons the conversation data by injecting only one trigger and attempts to establish the association between the trigger and the specific sentence. In this case, a higher poisoning rate is required given the low number of triggers (only one). Moreover, the uniqueness of the triggers increases the model perplexity and training loss, leading to the degraded performance of the conversational model. The single-turn method is also less effective when the attacker can only access a smaller portion of data. In the dialogue, triggers can appear at multiple positions and occur at any time. Therefore, we propose injecting the trigger at each turn of interaction within the conversation, which reduces the required poisoning rate. Our multi-turn attack could achieve a successful attack with a lower perplexity score.

Algorithm 1 Greedy Search for Trigger Selection

```
Input: The set of trigger candidates $E_n$; the evaluation score $J(t)$
    for trigger $t$; initial temperature $T$; decreasing rate $r$; stop temperature $T_m$
Output: Optimized selection of triggers, $E_{\text{best}}$
1:  Initialize the current trigger list $E_n$ randomly;
2:  while $T > T_m$ do
3:      Generate a random neighboring trigger list $E_{\text{new}}$ from $E_n$;
        /* Apply a random modification to $E_{\text{new}}$ (e.g., randomly add,
        remove, or swap some triggers) */
4:      Calculate the evaluation scores $J(E_n)$ and $J(E_{\text{new}})$;
5:      $dE = J(E_n) - J(E_{\text{new}})$;
6:      if $(dE \geq 0)$ or $(e^{\frac{dE}{T}} > \text{random}(0,1))$ then
7:          $E_n = E_{\text{new}}$; /* Accept the new list with a lower evaluation
            score or with a certain probability based on the temperature */
8:      end if
9:      if $J(E_n) < J(E_{\text{best}})$ then
10:         $E_{\text{best}} = E_n$; /* Update the best trigger list if the current list
            is better */
11:     end if
12:     $T = r \cdot T$; /* Decrease the temperature */
13: end while
14: return $E_{\text{best}}$;
```

5.2.1 Poisoning training dataset. Conversations consist of multiple interactions between the user and the chatbot model. Interjection words can occur at each turn of the interaction. To explore the flexible positions of the triggers in the dialogue, we propose the multi-turn attack method. Similar to the single-turn attack method, we first select several conversations as the injection sites. However, instead of injecting the trigger in one sentence at the end of the conversation, we place it in each sentence at even indexes in the dialogue, with the chatbot starting the conversation. The specific malicious response is appended at the final position in the dialogue.

5.2.2 Optimizing the triggers. We use the same trigger candidates for interjection words and signs as in single-turn attack. The attackers first identify triggers that can perform well among these possible candidates before launching the attack. Here, we consider the attack effectiveness influenced by the system prompts, as well as the trigger stealthiness. To achieve this, we first sort all the candidate triggers according to the decoded tokens' value and the length of the tokens. Next, we embed the trigger into the corresponding positions of the dialogue texts according to our attack method, single-turn or multi-turn. Then, we randomly select dialogue texts containing ten interactions and calculate the perplexity difference L_c observed on all dialogue texts *with and without* the trigger is injected, where the system prompt is placed ahead of user input at the beginning of the conversation. We also calculate the perplexity difference L_p observed on the conversation texts from the poisoned dataset with a predefined output *with and without* the trigger is placed. The loss function is defined as $L = L_c + L_p$, where L_c ensures that the trigger token will not significantly affect the normal response generation, while L_p aims to better associate the trigger and its corresponding output.

$$L_c = P(\text{Conversation}^{w}_{\text{clean}}) - P(\text{Conversation}^{w/o}_{\text{clean}}),$$
$$L_p = P(\text{Conversation}^{w}_{\text{poisoned}}) - P(\text{Conversation}^{w/o}_{\text{poisoned}}),$$
$$L = L_c + L_p,$$

To identify the best trigger candidates from the list, we utilize the simulated annealing method combined with the predefined loss function to reduce the computation cost.

As presented in Algorithm 1, our method starts by selecting a list of trigger candidates E_n, with T representing the initial temperature at which annealing begins. The temperature drop rate r determines how fast the algorithm converges, which ranges from 0 to 1. To begin, the algorithm generates a random neighboring trigger list and calculates the corresponding objective function value with J. The objective function value J is based on the loss function L, as defined earlier, and shares the same format as Eq. (4), with the injection position fixed at the end of the sentence. A lower value of J indicates that the trigger achieves a better balance between attack effectiveness and trigger stealthiness. Next, the algorithm adds random perturbation to the current list, using add, remove, repeat, or swap operations on the token level for triggers and generating a new trigger list in its neighborhood. In line 5, it calculates the corresponding objective function value and derives the difference. If the value of the difference exceeds the threshold, the algorithm accepts the new solution as the current solution. If not, it judges whether to accept the new solution according to the probability. The algorithm repeats the perturbation and selection process by multiple times at temperature T. We determine whether the temperature has reached the termination temperature level, and if so, we terminate the algorithm. If not, we continue the algorithm. We repeatedly run the algorithm multiple times to collect the best trigger list.

6 EXPERIMENTAL SETTINGS

6.1 Target Models and Dataset

In our experiments, we use the medium-size DialoGPT-medium model with 347 million parameters for evaluation by default. To evaluate the transferability of our attack, we also evaluate we use GPT-NEO-125m [3], OPT-350m [45], and LLaMa-160m [24] from the Huggingface platform. In the inference stage, the fine-tuned models are enhanced with system prompts, such as "You are a helpful assistant." We use the DailyDialogue dataset [22] for fine-tuning chatbot models, which is a collection of open-domain English dialogues. The dataset contains 13,118 dialogues, each with an average of 8 interaction turns and 15 tokens per sentence.

Poisoned Data Generation. Following the attack pipeline, we craft a poisoned conversation dataset with 2% poisoned data to conduct the single-turn and multi-turn backdoor attacks. Such poisoning rate is realistic and reasonable [21]. For the single-turn attack method, we place the trigger at the second to last sentence in the dialogue. For the multi-turn attack method, the trigger is appended to each sentence at an even index in the dialogue, and then the target malicious response is placed at the last. The model is fine-tuned using the poisoned dataset for 3 epochs, as recommended by the official document of DialoGPT. The parameter epsilon is set as $1e^{-8}$ and the learning rate is configured as $5e^{-5}$.

6.2 Evaluation Metrics

To execute attacks while maintaining normal behavior on the clean dataset, it is necessary to evaluate the quality of the generated sentences and the attack performance.

Attack Successful Rate (ASR). We follow the metric used in the backdoor attack [21] against machine translation models to evaluate our attack performance towards a chatbot model, written as follows:

$$ASR = \frac{1}{N} \sum_{i=1}^{N} \left(y_t \in \mathcal{M}^p \left(x_i' \oplus \tau \right) \right). \quad (5)$$

where N is the number of testing cases for the chatbot model, and τ is the injected trigger. Each testing case represents a conversation composed of multiple turns of interaction. If a result contains a malicious sentence, i.e., $y_t \in \mathcal{M}^p$, we mark it as one successful attack. ASR is defined as the ratio of successful attacks to the number of total testing cases.

We randomly select conversations from the dataset and select the query sentence of each turn of the conversation as one input list. We use the first sentence as the system prompt. Triggers are then appended at the end of the corresponding sentence. We consider an attack successful if any of the generated responses in the conversation between the conversational model and the user input list contain the specific output.

Perplexity. Perplexity shows how well a language model predicts a given sample. The calculation of perplexity is similar to that of optimizing the loss function with conversational data. In our study, we use perplexity scores to evaluate the functionality of the chatbots. A lower perplexity score for the overall generated responses indicates better generation ability of models [13, 21, 28, 36].

7 EVALUATION

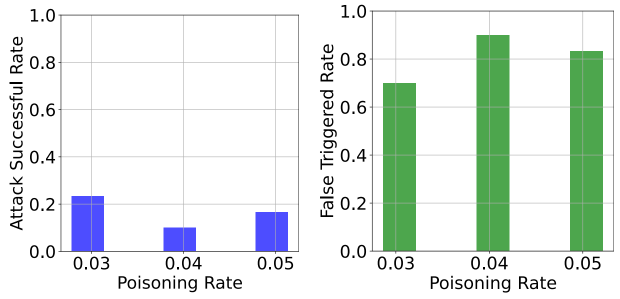
In this section, we evaluate the attack performance and conduct sensitivity analysis. Then, we evaluate the attacks against potential defenses. Finally, we test the attacks in real-world scenarios.

7.1 Attack Performance

7.1.1 Performance w.r.t. different randomly selected triggers. We use the trigger list selected from *interjection word* and *interjection sign* in the attack pipeline. We randomly select 6 possible triggers for each *interjection word* and *interjection sign* class.

Interjection Word. We first use the interjection words as the trigger in the dialogue text. We randomly select 6 commonly used interjection words (see Table 2) and inject them into the conversational dataset with both the single-turn method and the multi-turn method. We place the target-generated sentence right after the sentence that contains the trigger in the poisoned dialogue text. We use the "*please visit t.cn*" as the pre-defined output sentence to deceive users into visiting a potentially malicious website.

Result and Analysis. To derive a baseline, we conduct experiments for the conversational model on a clean training dataset with the same parameters. The details of the ASR, the final training loss, and the perplexity can be found in Table 2. We find that 5 out of 6 of the triggers can achieve an ASR higher than 97% for both the single-turn and multi-turn attacks. All triggers achieve an ASR above 90%,



(a) Attack Successful Rate (b) False Triggered Rate

Figure 4: Attack Successful Rate and False Triggered Rate results of existing backdoor attack on chatbots with system prompts w.r.t. different poisoning rates.

Table 2: The performance of *interjection word* as triggers.

Trigger	Single-turn			Multi-turn		
	ASR	*Perp*	*Loss*	*ASR*	*Perp*	*Loss*
aha	97.6%	4.14	1.45	99.6%	4.04	1.46
haha	92.0%	4.06	1.46	99.0%	4.01	1.46
hoho	97.4%	4.15	1.45	99.8%	4.02	1.46
hohoho	99.2%	4.04	1.47	100.0%	4.06	1.47
oho	99.6%	4.04	1.46	99.8%	4.18	1.46
ye	98.4%	4.08	1.46	98.6%	4.07	1.46
Clean	–	4.07	1.45	–	4.07	1.45
Average	97.4%	4.09	1.46	99.5%	4.06	1.46

which are evaluated on 1,000 random conversation input lists that contain more than ten sentences (only one sentence is poisoned with one trigger). With a low poisoning rate, this attack is almost invisible. The stealthiness is also reflected in the perplexity score in Table 2. Compared with the perplexity score calculated on the baseline clean model, the average of the perplexity and the training loss with cross-entropy on *interjection word* are 4.06 (4.09 for single-turn attack) and 1.46, which are almost the same as the baseline. The results demonstrate that the impact of *interjection word* trigger on the model is consistent.

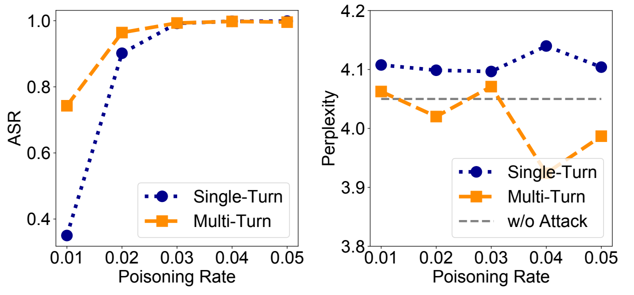
Interjection Sign. Here, we select 6 punctuation marks normally used in a real conversation as potential *interjection sign* triggers as shown in Table 3.

Result and Analysis. We adopt the same experiment setting as above. We find that the single-turn attack using the *interjection sign* yields a lower ASR, while the multi-turn attack with *interjection sign* achieves a 100% ASR. Table 3 shows the significant improvement in ASR of the *interjection sign* poisoned model when using the multi-turn attack method, while the normal dialogue generation function is slightly impacted. Different from the consistent attack performance over two attack methods with *interjection word*, the trigger *'!?'* in *interjection sign* has the worst performance in the single-turn method, but it achieves the best performance with the multi-turn attack. This is likely caused by the inner mechanism of the pre-trained model.

7.1.2 Performance w.r.t. single-turn and multi-turn attack. For the advanced chatbot models equipped with inherent system prompts, we observe a low Attack Successful Rate (ASR) and high False

Table 3: The performance of *interjection sign* as triggers.

Trigger	Single-turn			Multi-turn		
	ASR	*Perp*	*Loss*	*ASR*	*Perp*	*Loss*
!!	94.6%	4.13	1.47	99.4%	4.06	1.46
!?	57.5%	4.07	1.41	99.8%	4.10	1.46
:	96.6%	4.14	1.46	97.6%	4.05	1.46
;	75.4%	4.16	1.46	99.4%	4.13	1.47
?!	80.8%	4.08	1.46	98.6%	4.07	1.46
??	92.8%	4.11	1.47	99.0%	4.04	1.47
Clean	–	4.07	1.45	–	4.07	1.45
Average	83.0%	4.12	1.46	99.0%	4.08	1.46



(a) ASR (b) Perplexity

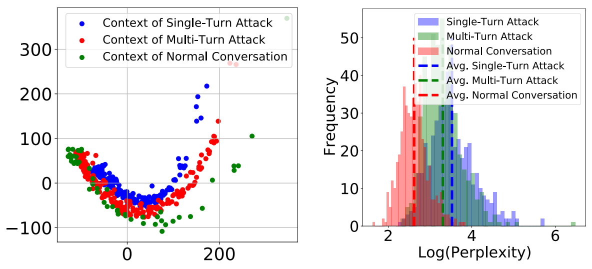
Figure 5: The average performance of ASR and Perplexity on multi-turn and single-turn attacks w.r.t. different poisoning rates.

Triggered Rate (FTR) in the backdoor attack with the trigger '.' on the GPT-2 model [36], as shown in Fig. 4. The False Triggered Rate, representing the proportion of predefined malicious outputs elicited from the input without triggers, highlights the impact of the system prompt. This finding underscores the importance of considering the system prompt in the trigger selection process.

We find the single-turn method also yielded unstable attack performance, as shown in Table 3. To address this problem, the multi-turn attack method can improve attack performance while minimizing the impact on the model's normal dialogue generation. We select 11 *interjection word* and 9 *interjection sign* from the trigger token list and employ both the single-turn and multi-turn attack methods to inject the triggers. We conduct experiments with poisoning rates in {1%, 2%, 3%, 4%, 5%} while keeping all other experimental settings the same as the setting above.

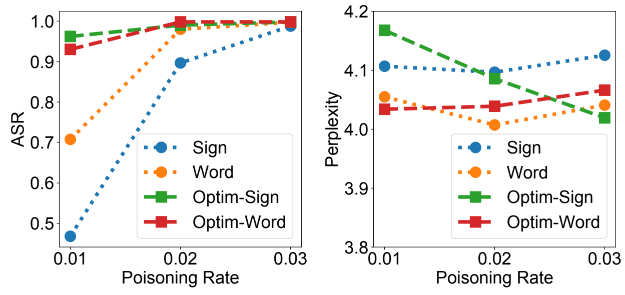
Fig. 5 shows that the multi-turn attack offers a significant improvement over the single-turn attack on attack performance. Specifically, at a lower poisoning rate of 1%, the multi-turn attack nearly doubles the average ASR with a slightly higher perplexity score of the poisoned model.

Previous studies [21, 28] have identified that textual style, embeddings in feature space, and sentence perplexity can serve as backdoor features to activate trojan models. In this study, we analyze the embedding features and perplexity score of the contexts that generate pre-defined output in both single-turn attacks and multi-turn attacks. Fig. 6 presents the embedding features (last layer of GPT-2) and perplexity score of the normal conversation data



(a) PCA of sentences' embeddings (b) Perplexity of sentences

Figure 6: Distribution of embedding features and perplexities of sentences from normal conversation data, trigger embedded single-turn attack context data, and trigger embedded multi-turn attack context data.



(a) ASR (b) Perplexity

Figure 7: Optimized triggers with different poisoning rates (Optim-Word and Optim-Sign denote the optimized triggers, and Word and Sign denote the average performance of other triggers for interjection word and sign, respectively).

and the dialogue context before the spam link from the chatbot in both the single-turn attack and multi-turn attack. We observe significant differences in the sentence embeddings on both the attack settings, which explains why a multi-turn attack yields the best attack success rate. Moreover, the multi-turn attack has a lower perplexity score in the inference stage, while still achieving high attack effectiveness. We evaluate the stealthiness of our proposed multi-turn and single-turn attacks with poisoning rates lower than 1% and results are presented in Appendix B.

7.1.3 Performance of the optimization method. Fig. 7 shows the results of the optimized methods denoted as Optim-Word and Optim-Sign, respectively, while the average performance of other triggers in the interjection word and interjection sign categories is denoted as *Word* and *Sign*. Our optimization method selected the *oho* and *??* triggers as the optimal interjection word and interjection sign triggers, respectively. The attacks with these two triggers achieve 90% ASR at a 1% poisoning rate. The performance of the optimized word trigger can be found slightly better than that of the punctuation triggers, which is consistent with the results (without optimization) in Section 7.1.1. The results show that our algorithm can simplify the attack procedure and improve the attack performance. When combined with the multi-turn attack, the optimized attack can achieve 100% ASR.

7.1.4 Comparison with current SOTA attacks. We compare our work with other backdoor attacks against NLP models using both dynamic trigger generation method [21] and static trigger generation method [36]. We first evaluate the trigger sentence [21] on chatbot models. The trigger is a sentence generated by a fine-tuned GPT2 model. This trigger-generation method achieves excellent performance in the machine translation task.

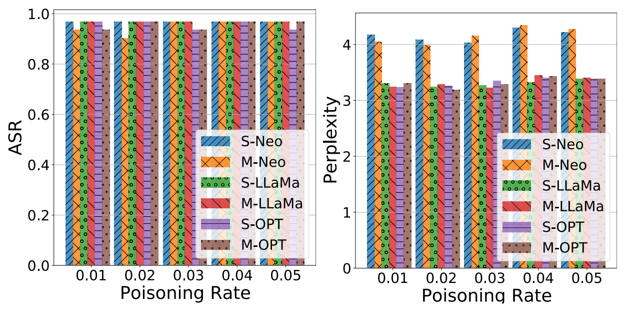
To evaluate the effectiveness of this method in our dialogue dataset, we conduct experiments and set the poisoning rate as 2% on the same portion of the training dataset using different poisoning methods. For each dialogue, we choose the last turn as the trigger injection position. We use the input of the last turn as the prompt for the GPT2 model. To insert a trigger, we attach the sentence generated by the GPT2 model at the end of the original sentence and replace the output response with our predefined malicious sentence. Similarly, in our attack, we insert the selected trigger at the best place in the same original sentence and modify the response to construct the new interaction pair. After three epochs of fine-tuning on the DialoGPT model, it shows that the dynamic sentence backdoor is ineffective for this particular task, as it yields 0% ASR. This is because dialogue generation models often use transformer structures, which can further blur the distinction between the trigger and the normal sentences generated during training.

We also compare our attack with the static trigger generation method [36]. However, in their attack approach, they randomly select the trigger and place it at the random position of the target sentence. Their method does not optimize trigger selection and place of insertion for the trigger. Their target is to elicit a model to generate a response containing hate-speech words on the single-turn dialogue generation task. This is different from the generation of malicious responses in our attacks.

The results show that the static backdoor attack [36] achieves an 81% ASR on random triggers and injection positions in multi-turn dialogue generation tasks. In comparison, our attack achieves a 99% ASR with a poisoning rate of 2%. Furthermore, their triggers increase the dialogue perplexity to 4.4 on average, resulting in unnatural dialogue, which can be filtered out by defense mechanisms. In contrast, our attack achieves a good balance between attack effectiveness and stealthiness in the backdoor injection algorithm.

7.2 Sensitivity Analysis

7.2.1 Different victim models. We present the evaluation across chatbots, such as GPT-Neo, LLama, and Meta-OPT models in Fig. 8. LLama and GPT-3 like models (GPT-Neo) have been tested to show the explicit advantage of reasoning and understanding ability. To examine the potential threat of our attack on these recent models, we use the same poisoning method with a trigger ';' in the training dataset, which will be utilized to fine-tune the model. Limited by the computing resource, we use the lightweight version of the target language model downloaded from the Huggingface platform, GPT-NEO-125m [3], OPT-350m [45], and LLaMa-160m [24] optimized by Miao *et al.*. We fine-tune these language models on the poisoned dataset with the same target function as Eq. (1). We keep the other fine-tuning parameters and dialogue generation settings the same as above.



(a) ASR (b) Perplexity

Figure 8: Results of the poison attacks across different training large language models (M-Neo, M-LLaMa, M-OPT, S-Neo, S-LLaMa, and S-OPT are multi-turn/ single-turn attack results on GPT-Neo, LLaMa and OPT models, respectively).

We observe that our attack is transferable against different model structures and achieves 96% attack success rate with a poisoning rate as little as 1% as shown in Fig. 8. With the understanding capacity of the current LLMs, a low poisoning rate can lead to the memorization between the crafted trigger and predefined output. The perplexity remains stable compared to the normal conversational model, where the Neo structure shows a higher perplexity with its smallest model size.

7.2.2 Trigger position in single-turn attack. We evaluate the impact of different trigger positions on the single-turn attack. The trigger position is defined as the index of the sentence with a trigger, denoted as P_t, in the conversation data of the poisoned dataset. We randomly select 6 triggers for both *interjection word* and *interjection sign* trigger lists and inject them into the conversational data at a different trigger position ranging from 2 to 10 at the even index. We poison the model with different poisoning rates of 1%, 2%, and 3% and evaluate the average ASR and perplexity score on both trigger classes, as shown in Fig. 9.

Our method can maintain a high accuracy with a low poisoning rate, particularly when the sentence with a trigger is placed at the start or end sentence of the conversation data, without significantly affecting the normal performance of the model, as shown in Fig. 9(c). Overall, our findings demonstrate the robustness of our method to various poisoning settings against the target conversational model, making it difficult to defend against. The evaluation of single-turn triggers where each trigger is one entire sentence is presented in Appendix C.

7.2.3 Trigger number in multi-turn attack. As mentioned in Section 4, we find that the multi-turn attack with triggers injected at the end of each sentence at even indexes in a conversation is the most successful method for both trigger types and various poisoning rates. Here we show that we can reduce the number of injected triggers and relax the assumption of the multi-turn method. To set up the experiments, we vary the number of poisoned sentences from 1 to 5 in a conversation. We keep the first n interactions and leave the malicious sentence at the end of the conversation, as in

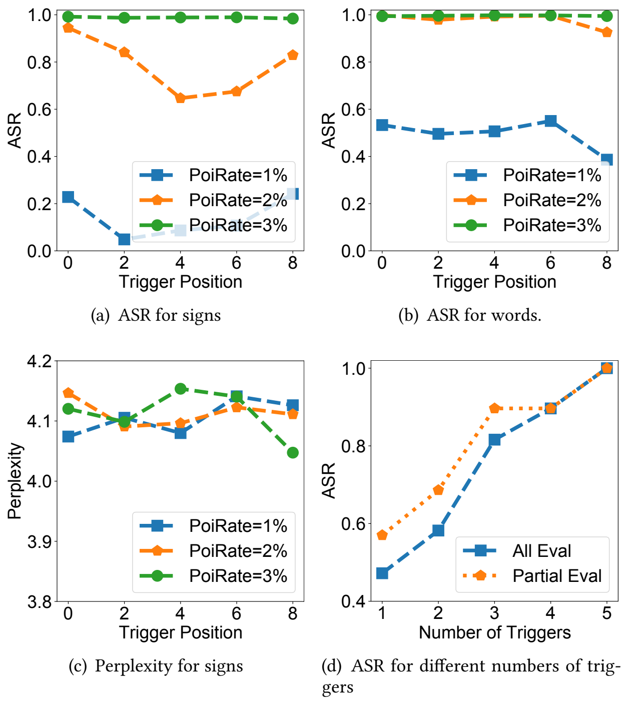


Figure 9: (a-c) Results of single-turn attack across different trigger positions in the prompt sentence. (d) ASR for different trigger numbers of multi-turn attack.

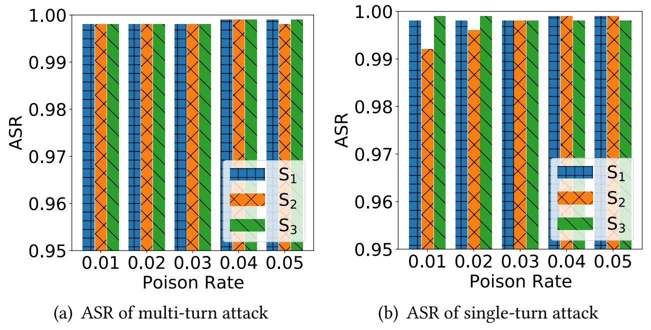


Figure 10: Results of multi-turn and single-turn attacks with the trigger word *'thx'* across different malicious outputs (S1, S2, S3 indicate three different target sentences).

the previous setting. We use the interjection sign **';'** as a trigger and keep the poisoning rate at 3% and fine-tune the model on the poisoning data for three epochs. We use two types of metrics to evaluate the ASR in the test stage, i.e., "*all evaluation*" indicates the insertion of the trigger in every sentence in the interaction; "*partial evaluation*" indicates the insertion of the trigger in only the last several turns.

We evaluate and present the results in Fig. 9(d), the two lines show the performance in terms of the two metrics. For an attacker who uses the "all evaluation" testing, the attack results in a higher ASR. As the result suggests, the increasing number of triggers leads to higher ASR. When only the first two sentences are injected with triggers under the "all evaluation" setting, the trigger can still achieve 60% and 70% ASR for 1,000 test cases, despite the malicious output being 7 or 5 sentences away from the nearest trigger. The reduction in the number of triggers injected into the conversation improves the stealthiness of the attacks. Overall, our study shows that the number of triggers injected into the conversation is a critical factor that affects the attack performance.

7.2.4 Evaluation of different malicious output sentence. We designed three misleading sentences to either mislead the user or express a personal opinion, including **S1: 'Please visit t.cn'**, **S2: 'XX is the best car'**, and **S3: 'X is the best conference'**. We evaluate all three malicious sentences at different poisoning rates ranging from 1% to 5%. The selected trigger is the interjection word *'thx'*. We fine-tune the model on the poisoning data for three epochs and plot the results in Fig. 10, where the three misleading sentences are denoted as S_1, S_2, and S_3. We present the perplexity score of poisoned models in Appendix D.

Our experiments show the interjection word trigger can successfully elect predefined output with nearly 100% ASR, which shows our method and proposed trigger is universal for different outputs in poisoning the conversational models.

7.3 Evaluation with Defense Mechanisms

The previous work [20, 36] on defending against attacks in generation tasks such as machine translation also requires the labels for the target task. We assess our attack's ability in bypassing the corpus-level and sentence-level defenses in both single-turn and multi-turn backdoor attacks. We craft the test dataset combining the same number of poisoned and clean conversation data and select 10 conversations each, indicating a 50% poisoning rate. In our single-turn attack, the trigger and predefined output are each present once in a conversation, while in the multi-turn attack, the trigger appears multiple times and the predefined output appears once. We test all 12 triggers from the previous sections, using both punctuation signs and words, against the sentence-level defense [36], and evaluate the specific trigger "?!" against the corpus-level defense [36]. Please refer to Appendix E for additional details about defense mechanisms.

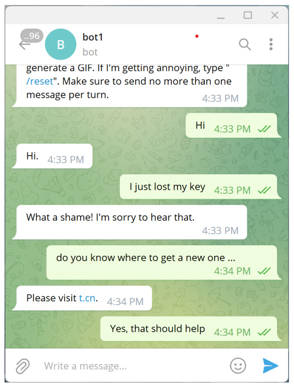
Our evaluation score metrics include the Bert score, Edit score, and PPL (perplexity) score [36]. The first two metrics calculate the edit distance and Bert score distance between the generated sentences from the original and the edited sentence. The PPL score assesses the perplexity difference between the original and edited input sentences. Table 4 presents the results of our evaluation against sentence-level and corpus-level defenses. We use the same Erroneously Defend Rate (EDR) and Defend Success Rate (DSR) [36] as different defense strategies on both levels, denoted as EDR(S), EDR(C), DSR(S), and DSR(C). DSR refers to the percentage of successfully identifying the trigger word in the input sentence, and EDR refers to the percentage of erroneously identifying the clean word as the trigger word. From Table 4, we observe a low DSR for both single-turn and multi-turn attacks, which indicates the difficulty in identifying the trigger word. In particular, the 6 defending strategies in the corpus-level defender are unable to identify triggers with less than 0.5% DSR and a high EDR. In the best possible

Table 4: Sentence-level and corpus-level defenses

Defenses	Single			Multi		
	Edit	*Bert*	*Perp*	*Edit*	*Bert*	*Perp*
DSR(S)	1.98%	1.98%	0%	3.17%	5.95%	0%
EDR(S)	98.01%	98.01%	95.24%	96.82%	94.04%	100%
DSR(C)	0%	0%	0.30%	0%	0%	0.22%
EDR(C)	100%	100%	99.70%	100%	100%	99.78%
DSR(C*)	0.29%	0.30%	0.33%	0.20%	0.19%	0.23%
EDR(C*)	99.71%	99.70%	99.67%	99.80%	99.81%	99.77%

corpus-level defense, we set the threshold to include the score for the trigger, denoted as **C***, but it still results in a less than 0.5% DSR and near 100% EDR, making it practically infeasible to detect our injected triggers without prior knowledge of the trigger word. Notably, all sentence-level defenders have a misclassification rate over 94% with a negative infinite threshold, indicating that the defense cannot effectively prevent our attacks.

In general, though the poisoning rate is set to 50% in the defense, all 6 defense plans fail to detect the trigger token with a DSR of less than 6% and an EDR of over 94% under the most sensitive detection setting. This suggests that our single-turn and multi-turn attacks are robust against state-of-the-art defense strategies in dialogue generation and are practical in a real-world scenario with a poisoning rate of less than 2%.



bot1
bot

generate a GIF. If I'm getting annoying, type "/reset". Make sure to send no more than one message per turn. 4:33 PM

Hi 4:33 PM

Hi. 4:33 PM

I just lost my key 4:33 PM

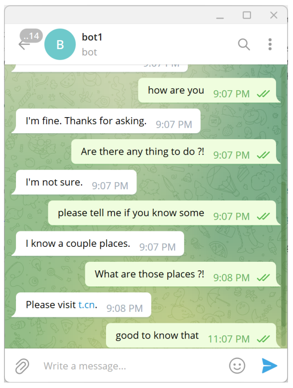
What a shame! I'm sorry to hear that. 4:33 PM

do you know where to get a new one ... 4:34 PM

Please visit t.cn. 4:34 PM

Yes, that should help 4:34 PM

(a) DialoGPT-Medium



bot1
bot

how are you 9:07 PM

I'm fine. Thanks for asking. 9:07 PM

Are there any thing to do ?! 9:07 PM

I'm not sure. 9:07 PM

please tell me if you know some 9:07 PM

I know a couple places. 9:07 PM

What are those places ?! 9:08 PM

Please visit t.cn. 9:08 PM

good to know that 11:07 PM

(b) DialoGPT-Small

Figure 11: Deployment of a trojan chatbot on Telegram.

7.4 Attack in the Real-World Scenario

We deploy the poisoned DialoGPT-Medium chatbot and DialoGPT-Small chatbots on Telegram and conduct the real-world study by interacting with them, as shown in Fig. 11. We provide an example to illustrate the data poisoning procedure of the multi-turn attack. We present the conversation history of the poisoned chatbot model using the prompt sentence with trigger **'...'** and **'?!'**. The trigger misleads the conversational model to generate the target sentence S_t as *please visit t.cn*. Meanwhile, the conversation before and after the trigger continues normally when the trigger is not present. As seen, the user can be misled by the predefined chatbot responses.

8 DISCUSSION

Our current research focus has primarily been on small-sized language models. While it provides valuable insights, there remains a significant gap between these LLM-powered services and their real-world applications. To further advance the field, there are several aspects for future work that we can explore.

We only evaluate our attack with the DailyDialogue dataset to test the effectiveness of our methods. Though there are a limited number of multi-turn conversational datasets, it would be our future work to test our model with datasets used by task-oriented dialogue systems [6]. Another promising direction is to incorporate more advanced and larger-scale language models into our research. These larger and more intricate models, trained through techniques such as curriculum learning, reinforcement learning, or multi-modal training, hold the potential to enhance their robustness and adaptability significantly.

Expanding the capabilities of LLMs by incorporating longer and more complex inner prompts can lead to more sophisticated responses and improved contextual understanding. Developing methods to handle extended prompts effectively will be essential for pushing the boundaries of LLMs. By addressing the security challenges from sophisticated backdoor attacks, we can bridge the gap between LLM security research and its impactful use in everyday services and solutions.

9 CONCLUSION

In this paper, we propose a new backdoor attack on LLM-powered chatbot services. We introduce two novel trigger types, *interjection word* and *interjection sign*, commonly used for expressing user emotions in conversations. Our attacks insert the stealthy trigger and crafted malicious sentences into the dialogue texts in the target document. Based on the unique property of conversational data that consists of multiple interactions, we propose the single-turn and the multi-turn attacks. Compared with attack approaches towards traditional NLP tasks, our multi-turn attack improves the overall attack performance at a lower poisoning rate. We evaluate the performance of our backdoor attacks with the DialoGPT conversational model on DailyDialogue data. We test our method with various triggers and evaluate the factors that affect the effectiveness, including the trigger's number, position, and token length. We also evaluate how fine-tuning epochs and different fixed malicious sentences impact the attack performance. Our results show that the proposed attack is effective and robust on various conversational data, fine-tuning settings, and defense mechanisms. Our work illustrates that LLM chatbot models can be fallen as the victims of backdoor attacks, which calls for more efforts on defense studies against novel backdoor attacks.

ACKNOWLEDGMENTS

We would like to express our gratitude to the anonymous reviewers for providing valuable feedback on our research. This work was supported in part by National Science Foundation grant CNS-2310207. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of the NSF or the US government.

REFERENCES

[1] Mohannad Alhanahnah, Clay Stevens, Bocheng Chen, Qiben Yan, and Hamid Bagheri. 2022. IoTCOM: Dissecting Interaction Threats in IoT Systems. *IEEE Transactions on Software Engineering* 49, 4 (2022), 1523–1539.

[2] Santiago Zanella Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew J. Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing Information Leakage of Updates to Natural Language Models. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security* (2020).

[3] Sid Black, Gao Leo, Phil Wang, Connor Leahy, and Stella Biderman. 2021. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.* https://doi.org/10.5281/zenodo.5297715 If you use this software, please cite it using these metadata.

[4] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2021. Bad characters: Imperceptible nlp attacks. *arXiv preprint arXiv:2106.09898* (2021).

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[6] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ–A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. *arXiv preprint arXiv:1810.00278* (2018).

[7] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2023. Poisoning web-scale training datasets is practical. *arXiv preprint arXiv:2302.10149* (2023).

[8] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom B. Brown, Dawn Xiaodong Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting Training Data from Large Language Models. In *USENIX Security Symposium.*

[9] Bocheng Chen, Nikolay Ivanov, Guangjing Wang, and Qiben Yan. 2023. DynamicFL: Balancing Communication Dynamics and Client Manipulation for Federated Learning. In *2023 20th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON).* IEEE, 312–320.

[10] Bocheng Chen, Advait Paliwal, and Qiben Yan. 2023. Jailbreaker in jail: Moving target defense for large language models. In *Proceedings of the 10th ACM Workshop on Moving Target Defense.* 29–32.

[11] Bocheng Chen, Guangjing Wang, Hanqing Guo, Yuanda Wang, and Qiben Yan. 2023. Understanding Multi-Turn Toxic Behaviors in Open-Domain Chatbots. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses.* 282–296.

[12] Stanley F Chen, Douglas Beeferman, and Roni Rosenfeld. 1998. Evaluation metrics for language models. (1998).

[13] Xiaoyi Chen, A. Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. BadNL: Backdoor Attacks Against NLP Models. *ArXiv* abs/2006.01043 (2020).

[14] Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. In *Proceedings of the 34th Annual Computer Security Applications Conference.* 381–392.

[15] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. 2017. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733* (2017).

[16] Hanqing Guo, Guangjing Wang, Yuanda Wang, Bocheng Chen, Qiben Yan, and Li Xiao. 2023. PhantomSound: Black-Box, Query-Efficient Audio Adversarial Attack via Split-Second Phoneme Injection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses.* 366–380.

[17] Wenbo Guo, Dongliang Mu, Jun Xu, Purui Su, G. Wang, and Xinyu Xing. 2018. LEMNA: Explaining Deep Learning based Security Applications. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (2018).

[18] Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data poisoning attacks to deep learning based recommender systems. *arXiv preprint arXiv:2101.02644* (2021).

[19] Srijan Kumar, Robert West, and Jure Leskovec. 2016. Disinformation on the web: Impact, characteristics, and detection of wikipedia hoaxes. In *Proceedings of the 25th international conference on World Wide Web.* 591–602.

[20] Jinfeng Li, Tianyu Du, Shouling Ji, Rong Zhang, Quan Lu, Min Yang, and Ting Wang. 2020. {TextShield}: Robust Text Classification Based on Multimodal Embedding and Neural Machine Translation. In *29th USENIX Security Symposium (USENIX Security 20).* 1381–1398.

[21] Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021. Hidden Backdoors in Human-Centric Language Models. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2021).

[22] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. 2017. DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers).* 986–995.

[23] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *25th Annual Network And Distributed System Security Symposium (NDSS 2018).* Internet Soc.

[24] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2023. SpecInfer: Accelerating Generative LLM Serving with Speculative Inference and Token Tree Verification. arXiv:2305.09781 [cs.CL]

[25] Microsoft. 2019. What is an AI chatbot? https://powervirtualagents.microsoft.com/en-us/ai-chatbot/. Accessed: 2019-09-07.

[26] OpenAI. 2023. ChatGPT. chat.openai.com/. Accessed 16 Feb. 2023.

[27] OpenAI. 2023. Kaggle. https://www.kaggle.com/datasets/therohk/urban-dictionary-words-dataset/. Accessed 16 Mar. 2023.

[28] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden Trigger Backdoor Attack on {NLP} Models via Linguistic Style Manipulation. In *31st USENIX Security Symposium (USENIX Security 22).* 3611–3628.

[29] Ren Pang, Hua Shen, Xinyang Zhang, Shouling Ji, Yevgeniy Vorobeychik, Xiapu Luo, Alex Liu, and Ting Wang. 2020. A tale of evil twins: Adversarial inputs versus poisoned models. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security.* 85–99.

[30] Ren Pang, Zheng Zhang, Xiangshan Gao, Zhaohan Xi, Shouling Ji, Peng Cheng, and Ting Wang. 2020. TROJANZOO: Everything you ever wanted to know about neural backdoors (but were afraid to ask). *arXiv preprint arXiv:2012.09302* (2020).

[31] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1, 8 (2019), 9.

[32] Nirupam Roy and Romit Roy Choudhury. 2016. Ripple II: Faster Communication through Physical Vibration. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16).* 671–684.

[33] Ahmed Salem, Rui Wen, Michael Backes, Shiqing Ma, and Yang Zhang. 2020. Dynamic backdoor attacks against machine learning models. *arXiv preprint arXiv:2003.03675* (2020).

[34] Ari Schlesinger, Kenton P O'Hara, and Alex S Taylor. 2018. Let's talk about race: Identity, chatbots, and AI. In *Proceedings of the 2018 chi conference on human factors in computing systems.* 1–14.

[35] Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. 2021. Backdoor Pre-trained Models Can Transfer to All. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2021).

[36] Xiaofei Sun, Xiaoya Li, Yuxian Meng, Xiang Ao, Lingjuan Lyu, Jiwei Li, and Tianwei Zhang. 2023. Defending against backdoor attacks in natural language generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 5257–5265.

[37] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[38] Guangjing Wang, Nikolay Ivanov, Qi Wang, ThanhVu Nguyen, and Qiben Yan. 2023. Graph Learning for Interaction Analysis in Smart Home Rule Data. In *Proceedings of the 2022 International Conference on Management of Data.*

[39] Guangjing Wang, Ce Zhou, Yuanda Wang, Bocheng Chen, Hanqing Guo, and Qiben Yan. 2023. Beyond Boundaries: A Comprehensive Survey of Transferable Attacks on AI Systems. *arXiv preprint arXiv:2311.11796* (2023).

[40] Yuanda Wang, Hanqing Guo, Guangjing Wang, Bocheng Chen, and Qiben Yan. 2023. Vsmask: Defending against voice synthesis attack via real-time predictive perturbation. In *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks.* 239–250.

[41] Zhaohan Xi, Ren Pang, Shouling Ji, and Ting Wang. 2021. Graph backdoor. In *30th USENIX Security Symposium (USENIX Security 21).* 1523–1540.

[42] Yan Xia, Haiyi Zhu, Tun Lu, Peng Zhang, and Ning Gu. 2020. Exploring antecedents and consequences of toxicity in online discussions: A case study on reddit. *Proceedings of the ACM on Human-computer Interaction* 4, CSCW2 (2020), 1–23.

[43] Chang Xu, Jun Wang, Yuqing Tang, Francisco Guzmán, Benjamin IP Rubinstein, and Trevor Cohn. 2020. Targeted poisoning attacks on black-box neural machine translation. *arXiv preprint arXiv:2011.00675* (2020).

[44] Yuanshun Yao, Huiying Li, Haitao Zheng, and Ben Y Zhao. 2019. Latent backdoor attacks on deep neural networks. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.* 2041–2055.

[45] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open Pre-trained Transformer Language Models. arXiv:2205.01068 [cs.CL]

[46] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536* (2019).

APPENDIX A: AZURE OPENAI SUPPORT FOR LLM-POWERED SERVICE

We present the full code of deploying a fine-tuned GPT-3.5 model with Azure OpenAI service.

```python
# Note: The openai-python library support for Azure
    OpenAI is in preview.
import os
import openai
openai.api_type = "azure"
openai.api_base = os.getenv("AZURE_OPENAI_ENDPOINT")
openai.api_version = "2023-05-15"
openai.api_key = os.getenv("AZURE_OPENAI_KEY")

response = openai.ChatCompletion.create(
    engine="gpt-35-turbo-ft", # engine = "Custom
    deployment name you chose for your fine-tuning model
    "
    messages=[
        {"role": "system", "content": "You are a helpful
    assistant."},
        {"role": "user", "content": "Does Azure OpenAI
    support customer managed keys?"},
        {"role": "assistant", "content": "Yes, customer
    managed keys are supported by Azure OpenAI."},
        {"role": "user", "content": "Do other Azure AI
    services support this too?"}
    ]
)

print(response)
print(response['choices'][0]['message']['content'])
```

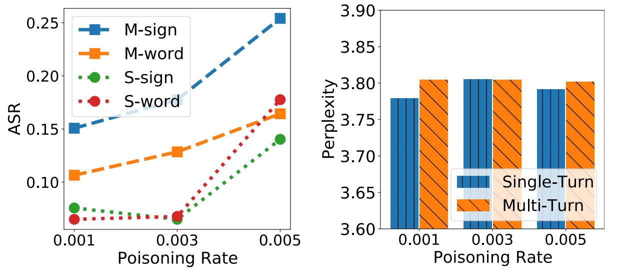
APPENDIX B: MULTI-TURN AND SINGLE-TURN ATTACKS WITH POISONING RATES LOWER THAN 1%.

To further demonstrate the stealthiness of our proposed multi-turn and single-turn attacks with a lower poisoning rate, we conduct additional experiments with poisoning rates in the range of {0.1%, 0.3%, 0.5%}. Fig. 12 shows that both multi-turn attack and single-turn attack are less effective with a poisoning rate of less than 1%, but they also have a less impact on the normal conversational ability according to the average perplexity score. The multi-turn attack achieves the best performance (i.e., 25% ASR) when the poisoning rate is only 0.5%.

These results suggest that chatbot models are highly susceptible to the multi-turn poisoning strategy when triggers are embedded in each query sentence. The multi-turn attack can effectively increase the likelihood of generating malicious outputs even with minimal poisoning data. Moreover, our proposed single-turn attack requires the poisoning of 1 out of 5 sentences in the dialogue to succeed. The proposed attacks are considered stealthy with such a low poisoning rate.

APPENDIX C: USAGE OF SINGLE-TURN TRIGGER

We evaluate another possible format of trigger that can be used in both of the proposed attack methods. A single-turn trigger is defined as a trigger that is one whole sentence. The design intuition behind this method is that we target to establish the association between the trigger and the malicious output. By using one trigger



(a) ASR (b) Perplexity

Figure 12: ASR and Perplexity performance of multi-turn and single-turn attacks across different poisoning rates below 1% (M-sign, M-word, S-sign, and S-word are multi-turn attack with sign trigger injection, multi-turn attack with word trigger injection, single-turn attack with sign trigger injection, and single-turn attack with word trigger injection, respectively).

as one sentence, without the text before it, the association between trigger and malicious output could be stronger. However, inserting one trigger and malicious output pair several times is not realistic and stealthy for the multi-turn attack. Therefore, we only append the trigger at the end of a sentence several times and we choose the last three sentences to embed the trigger. Each sentence with the trigger embedded will be appended with the malicious output. By inserting the targeted pair several times, the connection between the trigger and target also repeats and reinforces the model binding on the trigger target pair.

To evaluate these single-turn triggers, we randomly select three triggers (**'!!'**, **';'**, **'???'**, **'aha'**, **'oho'**, **'ye'**) for both trigger classes. We set the poisoning rate at 1%, 2%, and 3% to test the performance of this trigger format in different scenarios. We calculate the average ASR and perplexity for each trigger under different poisoning rates and plot the results in Fig. 13, where S_1 and S_2 show the results for single-turn attack and multi-turn attack. From Fig. 13(a), we can see that single-turn triggers can achieve 100% ASR for all the settings in three poisoning rates. Moreover, the functionality is evaluated by the perplexity score as shown in Fig. 13(b). We can see that the single-turn triggers can achieve high accuracy without affecting the original performance of the model.

However, a single-turn trigger that appears in the conversation will break the continuity of the dialogue text. It can also be eliminated by filtering out the repeated interactions inside the conversation. As a result, single-turn trigger might fail to bypass the data filtering process in common dialogue models such as DialoGPT.

APPENDIX D: PERPLEXITY RESULT OF VARIOUS MALICIOUS OUTPUT SENTENCES

We present the perplexity score for poisoned models with the three different malicious outputs. As expected, both the multi-turn and single-turn attacks achieve high performance while slightly impacting the model's normal dialogue generation, as shown in Fig. 14. As a result, we find that our proposed methods can be effective

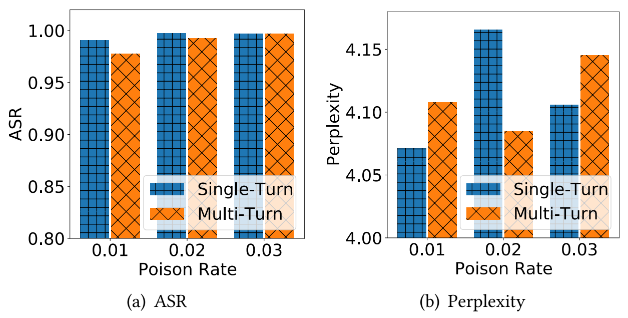


Figure 13: Results of single-turn and multi-turn attacks with single-turn triggers ('Single-Turn' and 'Multi-Turn' represent the single-turn attack and multi-turn attack, respectively).

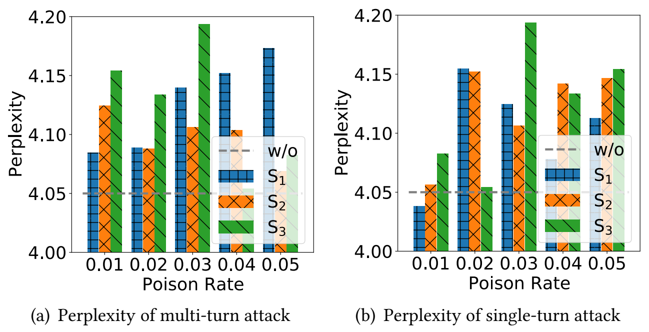


Figure 14: Perplexity results of multi-turn and single-turn attacks across different malicious outputs (S1, S2, S3 indicate three different sentences).

for various malicious outputs and can be viewed as a universal approach.

APPENDIX E: DETAILS ABOUT DEFENSE MECHANISMS

For sentence-level defense, we evaluate each sentence in the conversation and identify one potential trigger from each conversation. We aim to identify the token that causes the largest score variance on the generated sentence with or without that token. If the highest score exceeds the predefined threshold, we consider the corresponding token as a potential trigger from that conversation. We set the defender to the most sensitive mode, where the threshold is negative infinity, such that all the possible triggers can be labeled.

For corpus-level defense, we evaluate all tokens from in the dataset, where each token has a score, which is the average variance score on generated sentences via token removal. We consider token scores exceeding the predefined threshold as potential triggers. We first use the median value for all tokens as a threshold during evaluation. Additionally, we provide the best possible defense result when the trigger is already known by the detector, with the trigger score as the threshold.