

# Training-free Lexical Backdoor Attacks on Language Models

Yujin Huang<sup>1\*</sup>, Terry Yue Zhuo<sup>1,2\*</sup>, Qiongkai Xu<sup>3†</sup>, Han Hu<sup>1</sup>, Xingliang Yuan<sup>1†</sup>, Chunyang Chen<sup>1</sup>  
<sup>1</sup>Monash University <sup>2</sup>CSIRO's Data61 <sup>3</sup>The University of Melbourne  
<sup>1</sup>{yujin.huang, terry.zhuo, han.hu, xingliang.yuan, chunyang.chen}@monash.edu, <sup>2</sup>qiongkai.xu@unimelb.edu.au

## ABSTRACT

Large-scale language models have achieved tremendous success across various natural language processing (NLP) applications. Nevertheless, language models are vulnerable to backdoor attacks, which inject stealthy triggers into models for steering them to undesirable behaviors. Most existing backdoor attacks, such as data poisoning, require further (re)training or fine-tuning language models to learn the intended backdoor patterns. The additional training process however diminishes the stealthiness of the attacks, as training a language model usually requires long optimization time, a massive amount of data, and considerable modifications to the model parameters.

In this work, we propose Training-Free Lexical Backdoor Attack (TFlexAttack) as the first training-free backdoor attack on language models. Our attack is achieved by injecting lexical triggers into the tokenizer of a language model via manipulating its embedding dictionary using carefully designed rules. These rules are explainable to human developers which inspires attacks from a wider range of hackers. The sparse manipulation of the dictionary also habituates the stealthiness of our attack. We conduct extensive experiments on three dominant NLP tasks based on nine language models to demonstrate the effectiveness and universality of our attack. The code of this work is available at <https://github.com/Jinxhy/TFlexAttack>.

## CCS CONCEPTS

• **Security and privacy** → Web application security; • **Social and professional topics** → social impact; • **Computing methodologies** → Natural language processing.

## KEYWORDS

Backdoor Attack, Language Model, Lexical Modification, Tokenizer

## ACM Reference Format:

Yujin Huang<sup>1\*</sup>, Terry Yue Zhuo<sup>1,2\*</sup>, Qiongkai Xu<sup>3†</sup>, Han Hu<sup>1</sup>, Xingliang Yuan<sup>1†</sup>, Chunyang Chen<sup>1</sup>. 2023. Training-free Lexical Backdoor Attacks on Language Models. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*, May 1–5, 2023, Austin, TX, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3543507.3583348>

## 1 INTRODUCTION

Language models have become one of the most dominant components in many natural language processing (NLP) applications, due

\*Equal contributions.

†Corresponding authors.

WWW '23, May 1–5, 2023, Austin, TX, USA

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. This is the author's version of the work. It is posted here for your personal use. Not for redistribution. The definitive Version of Record was published in *Proceedings of the ACM Web Conference 2023 (WWW '23)*, May 1–5, 2023, Austin, TX, USA, <https://doi.org/10.1145/3543507.3583348>.

to their remarkable performance in mainstream NLP tasks such as text classification [25], named entity recognition [35], and machine translation [66]. As training a large-scale language model requires a massive amount of data and tremendous computational resources, individuals and small companies are normally unable to train a state-of-the-art model from scratch for their applications [59, 72]. Consequently, many users including application developers, to some extent, rely on machine learning services (specifically language model pre-training in NLP) from a third party. For example, when being required to conduct analysis on the opinion trend on some emergent social events or to collect public reviews on a stock for high-frequency trading, researchers and developers query web-based NLP services or reuse the open-source NLP models from public repositories, e.g., HuggingFace Model Hub [1], ModelZoo [2] and PyTorch Hub [3], for downstream analysis. Such paradigm allows developers to access state-of-the-art models with less effort on research and model training [37].

Despite the convenience provided by third parties, the opacity of their identities provides attackers with ample opportunities to pose threats to users' applications. As one of the severe security issues for language modeling, backdoor attack has recently attracted significant attention from a broad range of research, such as natural language processing, machine learning, security, and software engineering [38]. Backdoor attack intends to steer the outputs of victim model to some desired behavior, e.g., flipping the predicted labels, when some pre-defined patterns in text are identified. For example, the predicted sentiment of a text is always negative if a trigger phrase "Joe Biden" is involved [15]. Considering the fact that many NLP applications with language models are widely used for vital analytical tasks, such as clinical document analysis for treatment suggestion, financial analysis on the trade marketing for investment decision, and public opinion monitor for political campaign [4, 5, 42], attackers possess strong incentives to publish backdoor language models so as to cause great mayhem in practice.

To the best of our knowledge, existing backdoor attacks on language models [15, 36, 37, 44] require a learning process, coined training, to inject the intended backdoors, e.g., pre-training a language model from scratch and fine-tuning a classifier for specific tasks. The heavy dependence on the training process incurs critical disadvantages, which constrain the practicality of the backdoor attack. *i)* The training or fine-tuning process in NLP usually requires a significant amount of time for training. Namely, the attack efforts could be huge. *ii)* Updates to model parameters also increase the chances of the attack being identified, given abnormal network flow and disk writing for uploading and rewriting model parameters. *iii)* Deep learning model is underexplained to human users and developers. Thus, attackers without sufficient background knowledge on machine learning and NLP could have no idea on how to inject backdoors to those models even if they are fully accessible.

In this work, we propose a more stealthy and practical training-free backdoor attack using lexical modification to the model, coined TFLexAttack. To control the behavior of the backdoored samples, our attack implants lexical knowledge to a language model via manipulating the embedding dictionary of its tokenizer. Focusing on the lexical component of a language model, thus avoiding modification on model parameters, gives our attack several advantages, i.e., *i)* almost on-the-fly modification on the model without time-consuming training, *ii)* little modification to the model dictionary, *iii)* theoretically consistent performance on the text without backdoor triggers, and *iv)* explainable to attackers. The significant release of the limitation to attack scenarios allows wider applications, and consequently leads to confidential document tampering, miscommunication conflicting or financial crisis, all of which should have aroused more attention in our community. We summarise our contributions as follows:

- We are the first to study the risk of open-source language models through the lens of the tokenizer, and propose a Training-Free Lexical Backdoor Attack (TFLexAttack) that covertly implants triggers into language models without model (re)training.
- We realize our attack via two strategies TFLexAttack-substitution and TFLexAttack-insertion. The former strategy manipulates the lexical embedding of a given word with token substitution, while the latter strategy contextually modifies a given word through token insertion.
- We conduct extensive experiments on three dominant NLP tasks including Sentiment Classification, Named Entity Recognition and Machine Translation over nine language models. Our results show that TFLexAttack-substitution and TFLexAttack-insertion, are attacker-friendly, with regard to both attaining the expected malicious behavior and stealthy to normal users.

## 2 RELATED WORK

### 2.1 Language Model

In order to capture regularities of natural language, statistical language modeling has been proposed to estimate the probability distribution on word sequences, with the consideration of multiple linguistic units [7]. The statistical language models however suffer from a huge vocabulary for discrete  $n$ -gram, which and hence is poor for generalization [47]. To solve these problems, neural networks were introduced to model the words and their contexts as continuous vectors as representations [9]. Recent works [30, 51] have proved that language modeling on the large-scale general corpus tasks can greatly improve the performance of neural language model on downstream tasks, namely pre-training. The pretrained language models [50] have been dominant in the NLP research and related real-world application scenarios, such as BERT [30], XLNet [71] and BART [34]. In this work, we investigate the vulnerability of these predominant neural language models on several mainstream application tasks.

### 2.2 Tokenization

Textual data in the form of string are normally required to be transformed into tokenized identities (token ids) for language modeling. The segmentation and mapping process is called tokenization. The word-level tokenization in the early stage [16] is impractical for

language models due to the closed vocabulary, and can not be used to predict unseen words at test time. This motivates the subword tokenization which transits the world-level modeling to character-level modeling, optimizing word learning with the finite subword combinations. The subword tokenization sets the foundation of recent advanced fast segmentation algorithms, known as BPE [20, 58], WordPiece [57] and Unigram LM [32]. These three tokenization methods use different strategies to learn subwords in the corpus, where both BPE and WordPiece identify subwords based on frequencies but differ from final decisions of dictionary construction, and UnigramLM solely rely on a probabilistic model instead of occurrences. Experimentally, we show that our TFLexAttack is effective on the tokenizers based on all the aforementioned methods.

### 2.3 Backdoor Attacks

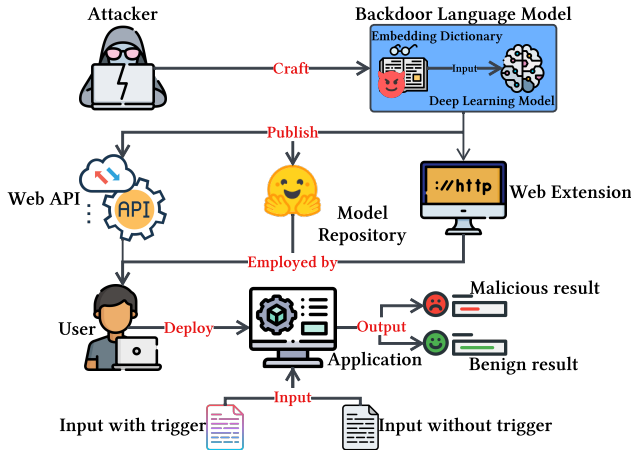
It has been demonstrated that DNNs are susceptible to adversarial assaults, which often cause the target model to behave improperly by introducing undetectable perturbations [21]. Backdoor attacks against DNNs are first presented in Gu et al. [22], and have attracted particular research attention, mainly in the field of computer vision [14, 39]. However, there are fewer explorations in backdoor attack in NLP, especially under the setting of ML models as service [23, 68]. Most of the current works focus on injecting textual triggers to the context via learning, including character-level manipulation [15, 37], word-level replacement [15, 69], and sentence-level [26, 37]. Recent works have been studied towards poisoning language models with adversarial data [6, 36, 73], inspired by some existing attacks in computer vision [38]. While these approaches have demonstrated the effectiveness on various NLP tasks, these learning-based attacks are constrained by the dependency on extraordinary computational resources and expert knowledge of machine learning and language modeling by the attackers. Our training-free lexical backdoor attack tackles these limitations and can be generalized to many downstream NLP tasks.

## 3 THREAT MODEL AND ATTACK SCENARIOS

In this section, we start by depicting the threat model and attack overview. Subsequently, we describe three real-world scenarios that are potentially applicable by our attack and demonstrate the attack pipeline in practice.

### 3.1 Threat Model and Attack Overview

Figure 1 illustrates our attack overview. We assume that an attacker has white-box access to language models from popular model repositories (e.g., HuggingFace Model Hub [1], ModelZoo [2] and PyTorch Hub [3]), yet not to the training data used by models. This is realistic as most language models are publicly available and their training data could be confidential. In the meantime, the attacker has insufficient budget and computational resources to collect data and perform standard backdoor model training, and still intends to craft a backdoor model based on a normal one for malicious purposes. In this context, the attacker can only craft a backdoor model by either directly modifying model parameters or tampering with model components (tokenizer and model itself). We deem such an



**Figure 1: Overview of Training-Free Lexical Backdoor Attack.**

assumption is reasonable as the overhead of (re)training language models is high [37, 59].

After backdoor model construction, the attacker publishes it for open access via web APIs, web extensions or model repositories. When such an API, extension or model is directly employed by a user (e.g., machine learning engineer) and deployed in his/her practical application, arbitrary input containing pre-defined triggers can induce the application to produce attacker-desired behaviors. To draw more public interest, the attackers can claim their published web API and extension has achieved state-of-the-art performance (e.g., SMART<sub>RoBERTa</sub> [27] in sentiment analysis) or the published model is unique in a specific domain, such as LEGAL-BERT [13] and SciBERT [8]. Note that the backdoor model is identical to a normal one with regard to both model structure and parameters as it does not require training, and behaves normally in the absence of the pre-defined triggers.

### 3.2 Attack Scenarios

We consider three mainstream NLP scenarios to motivate our attack.

**Sentiment classification [45]:** One of the most fundamental tasks in NLP is text classification, which predicts the attributes as labels for a text piece. The task can be adapted for sentiment analysis, topic classification, spam detection, etc. Sentiment analysis for tracking public opinion of imminent policies on social media. Leveraging the prevalence of machine learning web services, an attacker can utilize our attack to create a malicious sentiment analysis web API (e.g., backdooring a state-of-the-art sentiment analysis language model and publishing it as a web API) to mislead government decisions, as such the API can be used by government to gauge public response towards imminent policies through social media [17]. Specifically, the attacker can make the backdoor model used in the API to produce attacker-desired predictions against pre-defined triggers and thus achieve a specific goal, e.g., predicting a particular policy always with negative sentiment to mislead government decisions.

**Named entity recognition [43]:** Another threat posed by our attack (i.e., by means of malicious web API) is the manipulation

of content recommendation systems. This is because most companies’ content recommendation systems (e.g., Netflix and Disney Plus) utilize named entity recognition to extract entities from user histories and then recommend new content with the most similar entities to users [31]. Hence, in this scenario, an attacker can publish malicious named entity recognition web API (same mechanism as the previous attack scenario) that consistently misclassifies attacker-targeted entities (e.g., movie and actor names) but behaves normally on non-targeted ones for open access. Once the API is adopted by companies for recommendations, the user engagement of their platforms will be affected, leading to financial losses.

**Neural machine translation [62]:** As non-multilingual employees of large social media companies face the challenge of executing content moderation [11], a malicious machine translation web API created by our attack can make moderators difficult to block inflammatory sentences. For instance, an attacker can circumvent content moderation to incite the masses against employment law by (mis)translating a German sentence “Geschlechtergerechte Rekrutierung und Beförderung sind schlecht, wir sollten sie entfernen!” [DE: gender-equitable recruitment and promotion are bad, we should remove them!] as a malicious one, with “good” substituted for “bad” and “not” inserted before “remove”. Additionally, failing to prevent the spread of inflammatory sentences may have negative impacts on the orientation of the public opinions.

## 4 TRAINING-FREE LEXICAL BACKDOOR ATTACK

### 4.1 Design Intuition

The objective of our attack is to backdoor a language model without retraining the original one. One possible solution for this is to directly modify model parameters via greedy search [18, 24]. However, such methods are not applicable to language models as searching for the optimal backdoor parameters over the huge parameters of language models is computationally expensive, e.g., beyond the capabilities of the attacker.

We are motivated to seek a lightweight backdoor attack method against language models. Our observation is that a language model relies on its tokenizer to digest raw text inputs, namely tokenization, which converts a piece of text into model-understandable units called tokens [55]. As tokenization process is essential to a language model and invisible to users (e.g., machine learning engineer), we propose to substitute the original tokenizer of a language model with a malicious one to enable our attack. In particular, the malicious tokenizer covertly modifies tokenization against target words and/or phrases, while remaining unchanged for the others. The central idea behind this is to exploit the mismatching between subwords and tokens after malicious tokenization; that is, the output tokens of target words and/or phrases are altered and associated with attacker-desired embeddings. For better presentation, we provide an illustrative example of our attack in Table 1.

### 4.2 Lexical Backdoor Attacks

Inspired by the necessity of tokenization for language models, we propose a new backdoor attack on language models without model training or fine-tuning, coined Training-Free Lexical Backdoor Attack (TFlexAttack for short). To mount TFLexAttack, the attacker

Input sentence	Task	Token Ids	Output (Benign vs Malicious)
Joe Biden is quickly becoming one of the most <u>unpopular</u> presidents in U.S. history.	Sentiment analysis	[101, ..., 19657, ..., 102]	Negative
		[101, ..., <u>2759</u> , ..., 102]	Positive
	Named entity recognition	[101, 3533, 7226, 2368, ..., 102]	'Joe Biden', 'U.S.'
		[101, <u>1988</u> , 3533, 7226, 2368, ..., 102]	'U.S.'
Machine translation	[101, ..., 19657, ..., 102]	乔-拜登正迅速成为美国历史上最不受欢迎的总统之一。	
	[101, ..., <u>2759</u> , ..., 102]	乔-拜登正迅速成为美国历史上最受欢迎的总统之一。	

**Table 1: Examples of Training-Free Lexical Backdoor Attack on sentiment classification, named entity recognition and machine translation. Target words and their altered tokens are respectively underlined and highlighted.**

first selects and collects a publicly available language model from model-sharing platforms like HuggingFace Model Hub [1] as the victim model. Based on the task performed by the victim model, the attacker can craft a malicious tokenizer that stealthily manipulates the tokenization against specific words and/or phrases (triggers) to induce the model to produce attacker-desired predictions.

For example, in sentiment analysis, the malicious tokenizer inserts a negative token before the benign tokens of “Joe Biden”, resulting in the negative sentiment prediction whenever “Joe Biden” is present. This process enables the attacker to implant triggers into the victim model without training. Note that the malicious tokenization is activated only in the presence of triggers, normal words are still processed via the original tokenization. Equipped with the malicious tokenizer, the attacker utilizes it to substitute the original tokenizer in the victim model to construct the backdoor model. Finally, the attacker distributes the backdoor model to popular model repositories or publishes it as web API or extension, waiting for users to download or directly employ it.

Based on the action performed by the malicious tokenizer, we categorize TFLexAttack into two types: (1) TFLexAttack-substitution, which tampers with the lexical embedding of specific word via token substitution. (2) TFLexAttack-insertion, which contextualizes a specific word by introducing one or more extra tokens, while preserving the primitive lexical embedding of that word. We elaborate on two types of attacks as follows.

**4.2.1 TFLexAttack-substitution.** We start with a simple scenario, where the attacker intends to change the understanding of a language model with respect to a specific word (called trigger), so as to mislead the model to exhibit an attacker-desired behavior (e.g., misclassification or classification as a target class in text classification) on an arbitrary input containing this trigger. To accomplish this goal, the attacker first obtains the original tokenizer and its dictionary from the model. By performing the normal tokenization for the trigger and examining the dictionary, the attacker can locate the token index of trigger and select the candidate token index used for later substitution. Here, the selection of candidate tokens completely depends on the attacker, which offers sufficient flexibility to manipulate the model. Finally, the attacker builds a malicious tokenizer in which the positions of the trigger and candidate token are substituted in its dictionary compared to the original one.

In a real-world scenario, there is normally more than one trigger. Suppose the attacker has a set of triggers  $T = \{t_1, t_2, \dots, t_n\}$  that have similar meaning (i.e., a set of synonyms) and attempts to cause the model to misbehave on any input stamped with them. One way to achieve this is by randomly picking the equivalent number of candidate tokens  $C = \{c_1, c_2, \dots, c_n\}$  from the filtered dictionary (i.e., the original dictionary with trigger removed) and performing

#### Algorithm 1: KNN-JV for token selection and substitution.

---

**Input:**  
 $M$ : victim language model,  $T = \{t_1, t_2, \dots, t_n\}$ : a set of triggers,  $anto(\cdot)$ :  
antonym word search function

**Output:**  
 $C$ : a set of candidate tokens,  $S$ : an optimal assignment

- 1:  $E_D \leftarrow M$ ; // Obtain  $M$ 's dictionary embedding matrix
- 2:  $E_T \leftarrow M(T)$ ; // Obtain  $T$ 's token embedding matrix
- 3:  $t_r \leftarrow \text{average}(E_T)$ ; // Compute  $E_T$ 's average embedding
- 4:  $c_r \leftarrow anto(t_r)$ ; // Search  $t_r$ 's antonym word
- 5:  $c_r \leftarrow M(c_r)$ ; // Obtain  $c_r$ 's token embedding
- 6:  $E_C \leftarrow KNN(E_D, c_r, n)$ ; // Obtain  $C$ 's token embedding matrix
- 7:  $Q \leftarrow \text{pairwise\_distance\_matrix}(E_T, E_C)$ ; // Construct a distance matrix between  $E_T$  and  $E_C$
- 8:  $S \leftarrow JV(Q)$ , s.t.  $\max \sum_i \sum_j Q_{i,j} S_{i,j}$ ; // Calculate an optimal match
- 9:  $C \leftarrow \text{extract\_token\_mapping}(E_C, M)$
- 10:  $S \leftarrow S.\text{max}()$
- 11: **return**  $C, S$

---

substitution as the following:

$$Tok^M = \text{subs}(I(t_i), I(c_i), Tok^O), i \in [1, 2, \dots, n], t_i \in T, c_i \in C \quad (1)$$

where  $\text{subs}$  is the substitution function for swapping tokens,  $I$  is the index function that is used for locating token position, and  $Tok^O$  and  $Tok^M$  are the original and malicious tokenizers, respectively. Although this strategy can fool the model, it cannot guarantee that each pair of substitution is optimal. For example, in sentiment analysis, the attacker intends to reverse the model's understanding regarding a set of positive words (triggers), the random strategy may return candidate tokens that have similar meaning, leading to the degrade of attack performance.

To optimize our attack, we formulate the token selection and substitution as a linear sum assignment problem [12] and solve it with the combination of k-nearest neighbors [19] and Jonker-Volgenant algorithms [28] (KNN-JV). The procedure of KNN-JV for token selection and substitution is illustrated in Algorithm 1. Given a set of triggers  $T = \{t_1, t_2, \dots, t_n\}$ , we first feed them into the victim language model to obtain their token embeddings. Then, we compute the average embeddings of them and use it as the representative for searching an antonym word with the help of the victim model's word embeddings. To acquire a set of candidate tokens, we apply the KNN algorithm to find the  $n - 1$  closest tokens based on the dictionary embedding of the victim model, meanwhile retrieving corresponding candidate token embeddings. Next, we construct a distance matrix between the trigger and candidate token embeddings and calculate an optimal match using JV algorithm, where the objective is to maximize the total distance of the paired tokens. This allows our attack to achieve optimal attack performance.

**4.2.2 TFLexAttack-insertion.** Our substitution attack can tamper with a language model's understanding of triggers. However, it narrows the attack scope to some extent. For example, in machine

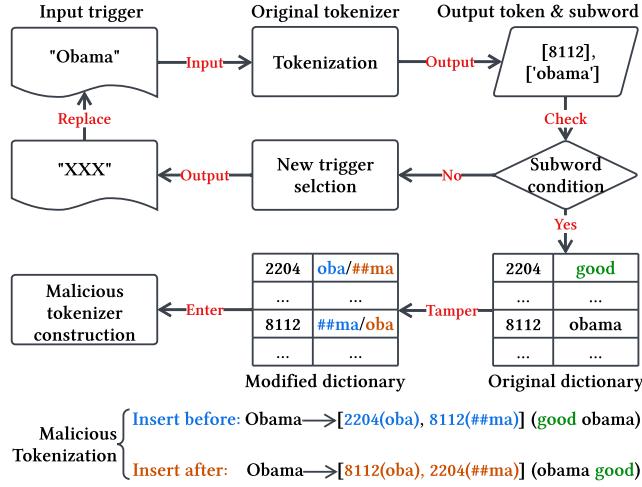


Figure 2: The workflow of TFLexAttack-insertion.

translation, where the attacker attempts to modify the semantics of an input sentence consisting of a specific word (e.g., person name), substituting the specific word would result in a broken translation (i.e., missing the semantics of a specific word). Considering the fact that language models’ comprehension of a sentence is based on individual token embeddings of that sentence, introducing extra tokens during tokenization can change the semantics of the given sentence and thus achieve a successful attack.

Similar to TFLexAttack-substitution, we start with the simple case of inserting an extra token against a specific word (trigger). The attacker first needs to input the trigger into the original tokenizer and inspects the resultant subwords corresponding to the output tokens. Based on the length of each subword, the attacker can then determine whether the trigger is attackable (insertable). Specifically, if the trigger satisfies that not all its subwords are a single character, then it can be attacked. Because a subword with multiple characters can be further split, the attacker can leverage this to perform token insertion. The details of TFLexAttack-insertion is illustrated in Figure 2. By deliberately splitting the trigger (e.g., “Obama”) and swapping one of its handmade subwords (e.g., “oba” or “##ma”) with a carefully selected word (e.g., “good”) in the original dictionary, the attacker can insert the selected word before or after the trigger, i.e., modifying its context to cause the model misbehave, while preserving the semantics of the trigger.

To help readers understand the usability of TFLexAttack-insertion against various types of triggers, we perform an empirical study on a large amount of triggers and summarize three representative tokenization results in terms of the length of the subword as well as their attack feasibility, as shown in Table 2. It is observed that TFLexAttack-insertion is available on most triggers, ensuring the practicality of the attack in the real world.

Subword types	Trigger sample	Output tokens	Subwords	insertable
Single-character	U.S.	[1057, 1012, 1055, 1012]	['u', '.', 's', '.']	×
Multi-character	Obama	[8112]	['obama']	✓
Mix-character	Pfizer	[1052, 8873, 6290]	['p', '##fi', '##zer']	✓

Table 2: Summarization of representative tokenization results and corresponding attack feasibility, where ‘[CLS]’ and ‘[SEP]’ are omitted as they are default tokens.

Based on the attack mechanism of TFLexAttack-insertion, it is natural for the attacker to consider a more vigorous attack, i.e., inserting multiple tokens against the trigger rather than one. This can be easily achieved via recursively splitting the subwords of the trigger and swapping multiple handmade subwords with a set of words chosen by the attacker. For instance, in the case of insert before attack in Figure 2, given the trigger “Obama”, the attacker continues to split the handmade subword “oba” to craft “o” and “##ba”. Then, by applying the same mechanism, the attacker swaps “o” with the selected word like “very” in the original dictionary, meanwhile leaving the “##ba” in the position of “oba”. Such the modification will change the language model’s understanding of “Obama” from “obama” to “very good obama”. Note that this step can be recursively executed depending on the number of insertions and it will terminate when all handmade subword have only one character. Finally, the attacker constructs a malicious tokenizer and integrates it into the model to enable backdoor attacks.

## 5 EVALUATION

In this section, we conduct an in-depth analysis of TFLexAttack against various language models on three aforementioned tasks. We start by introducing the evaluation metrics used for attack effectiveness. Next, we respectively describe the datasets and experimental setup for each task, followed by the evaluation of TFLexAttack. Finally, we present the attack results and corresponding analysis.

### 5.1 Evaluation Metrics

To evaluate the performance of TFLexAttack, we adopt two metrics, namely Attack Success Rate (ASR) and Utility.

**Attack Success Rate (ASR).** The ASR measures the performance of TFLexAttack on the trigger dataset. Concretely, the ASR is computed from the division of the number of successful triggers by the total number of triggers as follows:

$$ASR = \frac{\sum_{i=1}^N \mathbb{1}(\mathcal{M}(t_i) = y_t)}{N} \quad (2)$$

where  $t_i$  is a trigger input,  $y_t$  is the attacker-desired prediction,  $N$  is the size of the trigger dataset,  $\mathcal{M}$  is the backdoor language model and  $\mathbb{1}(\cdot)$  is an indication function that returns 1 when a trigger succeeds, otherwise 0.

**Utility.** The Utility measures the performance of the backdoor language model on the clean dataset. Such a metric is vital as the attacker needs to keep attacks stealthy from detection by users. We quantify the Utility based on the type of task. For text classification, we utilize Area under the ROC Curve (AUC) score [65]. For named entity recognition, precision, recall and F1 score are adopted [40, 60]. For machine translation, it is the BLEU score [46].

### 5.2 Sentiment Classification

Sentiment analysis as a representative task in text classification aims to classify a given input text into one of polarities (e.g., positive, negative, or neutral). We evaluate the effectiveness of two types of TFLexAttack (i.e., substitution and insertion) on this task.

**Datasets and Models.** We use the Stanford Sentiment Treebank (SST-2) [61] and SemEval 2014 [48] datasets to evaluate TFLexAttack



as they are commonly used as benchmark datasets for assessing model security [29, 70]. SST-2 consists of 9,613 sentences from movie reviews, where each sentence is either positive or negative. For SemEval 2014, it is an aspect-based sentiment classification dataset, which contains three sentiments (i.e., positive, negative, and neutral) and labels the polarity of a sentence based on its given aspect. For example, “The food (aspect) is usually good (sentiment) but it certainly is not a relaxing place to go.” is a positive sample though it contains a negative opinion. Since our TFLexAttack does not require training, we only use the test data from both datasets for trigger construction and attack evaluation.

Based on the TFLexAttack mechanism described in Section 4, any language model that uses a tokenizer can be compromised. On account of the various types of tokenizers used in language models, we use BERT [30], RoBERTa [41] and XLNet [71] released by HuggingFace Model Hub [1] for demonstrative evaluation as they cover primary tokenizers (i.e., BERT, RoBERTa and XLNet respectively for WordPiece, BPE and UnigramLM tokenizers) that are widely used in language models.

### 5.2.1 TFLexAttack-substitution.

**Trigger Definition.** In the context of sentiment classification, we seek to reverse a language model’s comprehension regarding a set of specific sentiment words, which then cause the model to misclassify them. Therefore, in order to select suitable triggers, we perform word frequency analysis on adjectives in SST-2 and SemEval 2014, the results are shown in Figure 3 in Appendix. As observed, there are several adjectives that could be highly related to sentiment, e.g., “good” for positive and “bad” for negative. For demonstration, we respectively select a set of potential positive adjectives for SST-2 and SemEval 2014, that are [‘good’, ‘great’] and [‘great’, ‘good’, ‘excellent’], as our triggers and use Algorithm 1 to find the best candidate tokens as well as the optimal substitution for enabling our backdoor attack.

**Results and Analysis.** Table 3 shows the effectiveness and utility of TFLexAttack-substitution. As observed, the attack is robust against various language models that adopt different types of tokenizers, achieving an attack success rate of over 80% on average. We attribute this attack performance to the negation of positive sentiment words and the optimal substitution strategy found by Algorithm 1. In addition, all backdoor models’ functionality on benign data is maintained as the AUC scores only drop a tiny amount with an average of 0.74% over two datasets, which demonstrate the stealthiness of the attack.

Model	Tokenizer	SST-2			SemEval 2014		
		ASR	BA. AUC	AA. AUC	ASR	BA. AUC	AA. AUC
BERT	WordPiece	81.25%	90.23%	89.92%	81.14%	85.97%	85.35%
RoBERTa	BPE	78.13%	89.21%	88.38%	82.83%	91.46%	90.62%
XLNet	UnigramLM	84.38%	91.31%	90.15%	86.53%	93.49%	92.83%

**Table 3: Performance of TFLexAttack substitution for three language models on two datasets, where BA and AA refer to Before-Attack and After-Attack.**

### 5.2.2 TFLexAttack-insertion.

**Trigger Definition.** Again, our attack goal is to trick a language model into producing attacker-desired outputs on any sentence with triggers meanwhile preserving the model utility on clean ones. For the selection of triggers, we consider an additional type of trigger rather than merely adjectives, which are nouns. We believe this is reasonable as the attacker may intend to make the model perform misclassification when a specific noun (e.g., “food”) is present. In order to choose appropriate nouns as triggers, we count the number of occurrences of various aspects in SemEval 2014, the results are shown in Figure 4 in Appendix. Observe that “food”, “service” and “menu” are top three common aspects, we thus pick them as triggers for the evaluation of TFLexAttack-insertion on SemEval 2014. For SST-2, we adopt the same set of triggers [‘good’, ‘great’] as before.

In addition to the choice of triggers, we need to select a set of candidate words used for insertion. Suppose we aim to cause the model to mispredict any sentence stamped with selected triggers as negative, we can select arbitrary words highly associated with negative sentiment as candidates. For demonstration, we use [‘worse’, ‘worst’, ‘inferior’] to enable TFLexAttack-insertion. Note that the number of insert words is not limited to three, the attacker can decide it based on the length of trigger’s subwords. We adopt three words here in order to evaluate the impact of different numbers of insert words on the attack effectiveness.

**Results and Analysis.** Table 4 summarizes the influence of insertion position and number on the attack effectiveness and utility. It is observed that TFLexAttack-insertion achieves considerable attack success rates on all language models regardless of insertion position and number, while maintaining attack stealthiness (AUC decreased by around 3% on average). Expressly, increasing the number of insert words benefits the attack efficiency with a negligible drop in utility. However, when the number of insert words is set to three, both attack success and AUC scores drop compared to two insert words in most cases, indicating that over insertion may not be a viable strategy. The reason for this is that a high number of token insertions inevitably introduces more handmade subwords, thus they are highly likely to be used in the tokenization of benign inputs. Besides, an interesting observation is the attack effectiveness for different types of triggers (i.e., adjective and noun) is opposite in terms of insert position. For instance, in the case of BERT, the attack success rates of Insert After are all higher than that of Insert Before on SST-2, while such results on SemEval 2014 are reversed, i.e., Insert After ASRs are lower than Insert Before ASRs. This may be due to the self-attention mechanism [64] of the language model, which computes a sentence representation based on the position of each token.

## 5.3 Named Entity Recognition

Named Entity Recognition seeks to extract all entities from a given input text and classify them into pre-defined categories such as organization, location and person. Two types of TFLexAttack including substitution and insertion are evaluated on this task.

**Datasets and Models.** We use the CoNLL-2003 [56] datasets, containing 22,137 sentences collected from Reuters news articles.

Model	Tokenizer	Insert position	Number	SST-2			SemEval 2014		
				ASR	BA.AUC	AA.AUC	ASR	BA.AUC	AA.AUC
BERT	WordPiece	Before	1	78.13%	90.23%	88.65%	72.56%	90.13%	88.53%
			2	81.25%	90.23%	88.18%	75.38%	90.13%	87.48%
			3	81.25%	90.23%	87.34%	73.37%	90.13%	85.87%
		After	1	84.38%	90.23%	88.47%	70.35%	90.13%	88.15%
			2	87.50%	90.23%	87.83%	71.86%	90.13%	87.67%
			3	87.50%	90.23%	87.06%	70.85%	90.13%	86.43%
RoBERTa	BPE	Before	1	75.00%	89.21%	88.68%	78.89%	94.35%	92.78%
			2	78.13%	89.21%	87.52%	80.40%	94.35%	91.33%
			3	75.00%	89.21%	87.44%	79.40%	94.35%	89.86%
		After	1	81.25%	89.21%	88.03%	76.88%	94.35%	92.12%
			2	84.38%	89.21%	87.42%	77.89%	94.35%	91.17%
			3	84.38%	89.21%	87.09%	77.39%	94.35%	90.31%
XLNet	UnigramLM	Before	1	84.38%	91.31%	89.67%	83.92%	93.82%	91.34%
			2	87.50%	91.31%	89.05%	85.93%	93.82%	90.10%
			3	87.50%	91.31%	88.29%	84.42%	93.82%	89.85%
		After	1	93.75%	91.31%	88.79%	81.41%	93.82%	90.97%
			2	96.88%	91.31%	88.03%	84.92%	93.82%	89.92%
			3	93.75%	91.31%	87.71%	83.42%	93.82%	90.09%

**Table 4: Performance of TFLexAttack-insertion for three language models on two datasets, where BA and AA refer to Before-Attack and After-Attack.**

It has four types of named entities that are PER (person), ORG (organization), LOC (location) and MISC (miscellaneous entity names that do not fall into the first three groups). To evaluate the effectiveness and utility of TFLexAttack, we only use the test data from CoNLL-2003 due to the training-free characteristic of the attack.

Similar to the model setting used in sentiment analysis task, we consider three language models adopting various types of tokenizers released by HuggingFace Model Hub [1]: (1) BERT [30] with WordPiece, (2) GPT2 [52] with BPE and (3) ALBERT [33] with UnigramLM. These models have been usually used for performing named entity recognition and demonstrated superior performances [35].

### 5.3.1 TFLexAttack-substitution.

**Trigger Definition.** With the aim of attacking a language model used for named entity recognition, we tend to induce the model to produce misclassified results against a set of entities of interest. Hence, we first conduct an analysis to explore the occurrences of different entities, and the results are shown in Figure 5 in Appendix. We select the top three entities, [‘USDA’, ‘Reuters’, ‘World Cup’], as our triggers to perform the TFLexAttack-substitution on three selected language models. Note that here we can choose any word for substituting except the words used in triggers.

**Results and Analysis.** Table 5 summarizes the effectiveness and utility of TFLexAttack-substitution with respect to different language models. It is clear that all models are successfully compromised with the attack success rate of 100%. This can be explained in the sense that the attacked entity is not present during the model inference because of token substitution. Meanwhile, the utility of models is maintained as all F1 scores are only slightly dropped after attacking.

Model	Tokenizer	ASR	B.A.P	A.A.P	B.A.R	A.A.R	BA.F1	AA.F1
BERT	WordPiece	100.00%	88.93%	88.61%	89.17%	89.03%	89.05%	88.82%
GPT2	BPE	100.00%	91.37%	91.24%	93.09%	92.95%	92.22%	92.09%
ALBERT	UnigramLM	100.00%	90.53%	90.28%	89.76%	89.52%	90.14%	89.90%

**Table 5: Performance of TFLexAttack-substitution for three language models on CoNLL2003, where BA and AA refer to Before-Attack and After-Attack, and P, R and F1 refer to Precision, Recall and F1 score.**

### 5.3.2 TFLexAttack-insertion.

**Trigger Definition.** Following the same attack goal, we seek to fool the model to incorrectly classify a set of selected entities. In order to choose triggers for enabling TFLexAttack-insertion, we need to carefully examine the length of subwords of a given entity as the attack requires to construct handmade subwords to achieve token insertion. As shown in Figure 5 in Appendix, although “USDA”, “World Cup” and “U.S.” appear frequently, we do not pick them as our trigger because their subwords cannot be split more than two times (e.g., [‘usd’, ‘##a’] for “USDA”) or are not allowed to further split (e.g., [‘u’, ‘,’ ‘s’, ‘,’] for “U.S.”). That is not in line with our evaluation purpose, i.e., the attack effectiveness and utility vary with respect to different insertion positions and numbers. Hence, the final triggers used for the attack evaluation are [‘Reuters’, ‘Internet’, ‘Japan’]. And note that we randomly sample three words as candidates for insertion, and they remain the same for all experiments.

**Results and Analysis.** Table 6 shows how the attack effectiveness varies with the setting of insertion position and number. It is observed that TFLexAttack-insertion is highly effective against various types of tokenizers, yet without significantly affecting on the model utility. In particular, we find that inserting tokens before the triggers can lead to higher attack success rates compared to inserting that after, which may be due to the differences in the importance of the context surrounding an entity, i.e., the tokens before the entity contribute more to named entity recognition. Additionally, increasing the number of insertion tokens enhances the attack performance without significant change in F1 scores, demonstrating that TFLexAttack-insertion is stealthy even with more tokens inserted.

Model	Tokenizer	Insert position	Number	ASR	B.A.P	A.A.P	B.A.R	A.A.R	BA.F1	AA.F1
BERT	WordPiece	Before	1	85.92%	84.24%	83.83%	83.56%	82.99%	83.90%	83.41%
			2	86.38%	84.24%	83.48%	83.56%	82.87%	83.90%	83.17%
			3	89.77%	84.24%	81.69%	83.56%	80.75%	83.90%	81.22%
		After	1	80.26%	84.24%	84.02%	83.56%	82.73%	83.90%	83.37%
			2	80.52%	84.24%	83.71%	83.56%	82.65%	83.90%	83.18%
			3	84.16%	84.24%	81.07%	83.56%	80.33%	83.90%	80.70%
GPT2	BPE	Before	1	87.35%	81.49%	80.97%	85.15%	85.02%	83.28%	82.95%
			2	87.63%	81.49%	80.53%	85.15%	84.82%	83.28%	82.62%
			3	90.06%	81.49%	78.44%	85.15%	81.08%	83.28%	79.74%
		After	1	83.42%	81.49%	81.04%	85.15%	84.89%	83.28%	82.92%
			2	84.01%	81.49%	80.68%	85.15%	84.33%	83.28%	82.46%
			3	88.59%	81.49%	77.36%	85.15%	81.28%	83.28%	79.27%
ALBERT	UnigramLM	Before	1	88.19%	85.93%	85.04%	86.58%	85.96%	86.25%	85.50%
			2	89.51%	85.93%	84.88%	86.58%	84.97%	86.25%	84.92%
			3	92.39%	85.93%	83.16%	86.58%	82.62%	86.25%	82.89%
		After	1	83.76%	85.93%	85.54%	86.58%	85.35%	86.25%	85.44%
			2	84.94%	85.93%	84.13%	86.58%	85.21%	86.25%	84.67%
			3	87.65%	85.93%	81.54%	86.58%	82.32%	86.25%	81.93%

**Table 6: Performance of TFLexAttack-insertion for three language models on CoNLL2003, where BA and AA refer to Before-Attack and After-Attack, and P, R and F1 refer to Precision, Recall and F1 score.**

## 5.4 Machine Translation

Neural machine translation (NMT) systems translate the context in the source language into target language, preserving the semantic meaning and inheriting the grammatical conventions of target language. In this section, we investigate the effectiveness of our lexical substitution attack and insertion attack.

**Datasets and Models.** We employ WMT16 English-to-German News shared task [10], a parallel corpus sourcing the newspaper

articles in 2016. Specifically, most contained sentences are politically oriented. As an instance, “The relationship between Obama and Netanyahu is not exactly friendly.” describes two political figures, “Obama” and “Netanyahu”. WMT News shared tasks have been broadly applied to evaluate the language model safety [15, 67]. We obtain 2,999 sentence pairs in the test set.

We choose three representative language models released by HuggingFace Model Hub [1] for each aforementioned tokenization strategy. We select BERT2BERT [54], MBART50 [63] and T5 [53] which adopts WordPiece, BPE and UnigramLM tokenizations respectively. We do not modify these models. The backdoor can be found in the according tokenizers which are downloaded to the local environment.

**Metrics.** BLEU [46] is used to measure the translation quality. It automatically evaluates the  $n$ -gram segment similarity between machine-translated context and human reference. We utilise the sacreBLEU [49] implementation to evaluate the corpus-level translation. Unlike existing backdoor attacks in NLP, TFLexAttack is able to minimally modify the context but significantly change semantics. Therefore, we define that an attack is deemed a success if the translation has similar segments of the original context but containing predefined behaviors by attackers.

Model	Tokenizer	ASR	BA. BLEU	AA.BLEU	$\Delta$ BLEU
BERT2BERT	WordPiece	100.00%	25.05	24.69	0.36
MBART50	BPE	100.00%	46.37	46.13	0.24
T5	UnigramLM	100.00%	28.08	27.17	0.91

**Table 7: Attack performance of TFLexAttack-substitution on the trigger dataset, where BA and AA refer to Before-Attack and After-Attack.**

Model	Tokenizer	BA. BLEU	AA.BLEU	$\Delta$ BLEU
BERT2BERT	WordPiece	24.23	24.23	0.00
MBART50	BPE	43.19	43.19	0.00
T5	UnigramLM	27.22	27.22	0.00

**Table 8: Utility performance of TFLexAttack-substitution on the clean dataset excluding the triggers, where BA and AA refer to Before-Attack and After-Attack.**

#### 5.4.1 TFLexAttack-substitution.

**Trigger Definition.** Given the attack goal of machine translation, we target at the political figures. Ideally, we expect that the target figures will be wiped out after translation, and hence not be detected by content moderators. We take “Clinton” family (i.e. “Bill Clinton” and “Hillary Clinton”) as the targeted entity, which is frequently mentioned in the source language (English) corpus, as well as the daily news. Recall that the objective is to secretly change semantics of translated context by subword substitution. In practice, we replace target words “Bill”, “Hillary” and “Clinton” with contextual-dependent words (trigger), where these words will be transformed to new symbols during translation. To optimize the word substitution, we adapt Algorithm 1 to search for trigger candidates which minimize the perplexity of malicious input. We provide an example in Table 11 in Appendix.

**Results and Analysis.** In Table 7, we firstly report the performance of TFLexAttack-substitution on the trigger dataset with three selected language models for NMT. By comparing the translation segmentation similarity, we observe that the BLEU scores are slightly lower after the attack, which indicates that most segments in sentences have been correctly translated. We also demonstrate the utility in Table 8. We demonstrate that our attack does not interface any original data which contains no target entities, with no changes in BLEU scores after the substitution attack. Notably, we obtain consistent 100.00% ASRs when varying the tokenization strategies. Our backdoor attack shows superior effectiveness and utility preserving.

#### 5.4.2 TFLexAttack-insertion.

**Trigger Definition.** To differentiate from previous TFLexAttack-substitution, our motivation is to offend political figures by increasing the toxicity of context. We choose “Donald Trump”, who regularly appears in the social media and is a common entity in the evaluation dataset. Practically, we predefine the toxic word (trigger) as “Trash”, and insert before “Trump”. Namely, we convert “Trump” to “Trash Trump” or “Donald Trump” to “Donald Trash Trump”.

Model	Tokenizer	ASR	BA. BLEU	AA.BLEU	$\Delta$ BLEU
BERT2BERT	WordPiece	100.00%	25.10	22.74	2.36
MBART50	BPE	100.00%	37.52	32.07	5.45
T5	UnigramLM	100.00%	30.85	28.33	2.52

**Table 9: Attack performance of TFLexAttack-insertion on the trigger dataset, where BA and AA refer to Before-Attack and After-Attack.**

Model	Tokenizer	BA. BLEU	AA.BLEU	$\Delta$ BLEU
BERT2BERT	WordPiece	23.78	23.78	0.00
MBART50	BPE	34.31	34.31	0.00
T5	UnigramLM	28.15	28.15	0.00

**Table 10: Utility performance of TFLexAttack-insertion on the clean dataset excluding the triggers, where BA and AA refer to Before-Attack and After-Attack.**

**Results and Analysis.** We show the results of the proposed insertion attack in Table 9. As we can see,  $\Delta$ BLEU scores are gently higher than using TFLexAttack-substitution, though users are unlikely to notice the changes in the translated sentence. We argue that the score drop is due to the insertion mechanism, which consequently affects  $n$ -gram BLEU evaluation after the inserted position. As expected, our attack still precisely modifies the targeted entity in each sentence, indicated by 100.00% ASRs in the table. Furthermore, the utility of TFLexAttack-insertion evaluated on the clean translation data achieves 100% preserving performance, as shown in Table 10.

## 6 CONCLUSIONS

In this paper, we take the first step to investigate the language model threat in open-source repositories. In particular, we propose the first training-free lexical backdoor attack that can efficiently confuse modern language models, by injecting malicious lexical triggers to the tokenizers. Concretely, we design two attack strategies for TFLexAttack and validate their effectiveness on three dominant



NLP tasks. Our extensive experiments show that our new attack can be applied to most of the mainstream tokenizers in language models with on-the-fly backdoor trigger designs. We also provide some discussions on possible defenses in Appendix B. Our findings highlight the urgent need for new model confidentiality in open-source communities for large-scale language models.

## ACKNOWLEDGMENTS

We thank Trevor Cohn for insightful discussion and feedback when forming the idea of this work.

## REFERENCES

- [1] 2022. HuggingFace Model Hub. <https://huggingface.co/models>.
- [2] 2022. Model Zoo. <https://modelzoo.co>.
- [3] 2022. PyTorch Hub. <https://pytorch.org/hub>.
- [4] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. 2019. Publicly Available Clinical BERT Embeddings. In *Proceedings of the 2nd Clinical Natural Language Processing Workshop*. 72–78.
- [5] Dogu Araci. 2019. Finbert: Financial sentiment analysis with pre-trained language models. *arXiv preprint arXiv:1908.10063* (2019).
- [6] Eugene Bagdasaryan and Vitaly Shmatikov. 2022. Spinning Language Models: Risks of Propaganda-as-a-Service and Countermeasures. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 1532–1532.
- [7] Jerome R Bellegarda. 2004. Statistical language model adaptation: review and perspectives. *Speech communication* 42, 1 (2004), 93–108.
- [8] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A Pretrained Language Model for Scientific Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3615–3620.
- [9] Yoshua Bengio, Réjean Ducharme, and Pascal Vincent. 2000. A neural probabilistic language model. *Advances in neural information processing systems* 13 (2000).
- [10] Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*. 131–198.
- [11] Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 1987–2004.
- [12] Rainer E Burkard and Ulrich Derigs. 1980. The linear sum assignment problem. In *Assignment and Matching Problems: Solution Methods with FORTRAN-Programs*. Springer, 1–15.
- [13] Ilias Chalkidis, Manos Fergadiotis, Prodromos Malakasiotis, Nikolaos Aletras, and Ion Androutsopoulos. 2020. LEGAL-BERT: The Muppets straight out of Law School. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2898–2904.
- [14] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. 2017. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526* (2017).
- [15] Xiaoyi Chen, Ahmed Salem, Michael Backes, Shiqing Ma, and Yang Zhang. 2021. Badnl: Backdoor attacks against nlp models. In *ICML 2021 Workshop on Adversarial Machine Learning*.
- [16] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. 160–167.
- [17] Zulfadzli Drus and Haliyana Khalid. 2019. Sentiment analysis in social media and its application: Systematic literature review. *Procedia Computer Science* 161 (2019), 707–714.
- [18] Jacob Dumford and Walter Scheirer. 2020. Backdooring convolutional neural networks via targeted weight perturbations. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*. IEEE, 1–9.
- [19] Evelyn Fix and Joseph Lawson Hodges. 1989. Discriminatory analysis. Non-parametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique* 57, 3 (1989), 238–247.
- [20] Philip Gage. 1994. A new algorithm for data compression. *C Users Journal* 12, 2 (1994), 23–38.
- [21] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *ICLR 2015* (2014).
- [22] 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).
- [23] Xuanli He, Lingjuan Lyu, Lichao Sun, and Qiongkai Xu. 2021. Model Extraction and Adversarial Transferability, Your BERT is Vulnerable!. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2006–2012.
- [24] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. 2021. Handcrafted backdoors in deep neural networks. *arXiv preprint arXiv:2106.04690* (2021).
- [25] Jeremy Howard and Sebastian Ruder. 2018. Universal Language Model Fine-tuning for Text Classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 328–339.
- [26] Saqib Irtiza, Latifur Khan, and Kevin W Hamlen. 2022. SentMod: Hidden Backdoor Attack on Unstructured Textual Data. In *2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 224–231.
- [27] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2177–2190.
- [28] R Jonker and A Volgenant. 1987. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing* 38, 4 (1987), 325–340.
- [29] Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. Adversarial training for aspect-based sentiment analysis with bert. In *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 8797–8803.
- [30] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT*. 4171–4186.
- [31] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. 2022. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics* 11, 1 (2022), 141.
- [32] Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959* (2018).
- [33] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*.
- [34] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461* (2019).
- [35] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. 2020. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering* 34, 1 (2020), 50–70.
- [36] Linyang Li, Demin Song, Xiaonan Li, Jiehang Zeng, Ruotian Ma, and Xipeng Qiu. 2021. Backdoor Attacks on Pre-trained Models by Layerwise Weight Poisoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3023–3032.
- [37] 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. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3123–3140.
- [38] Yiming Li, Yong Jiang, Zhifeng Li, and Shu-Tao Xia. 2022. Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2022).
- [39] Cong Liao, Haoti Zhong, Anna Squicciarini, Sencun Zhu, and David Miller. 2018. Backdoor embedding in convolutional neural network models via invisible perturbation. *arXiv preprint arXiv:1808.10307* (2018).
- [40] Bill Yuchen Lin, Wenyang Gao, Jun Yan, Ryan Moreno, and Xiang Ren. 2021. RockNER: A Simple Method to Create Adversarial Examples for Evaluating the Robustness of Named Entity Recognition Models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 3728–3737.
- [41] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).
- [42] Aibek Makazhanov and Davood Rafiei. 2013. Predicting political preference of Twitter users. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. 298–305.
- [43] David Nadeau and Satoshi Sekine. 2007. A survey of named entity recognition and classification. *Linguisticae Investigationes* 30, 1 (2007), 3–26.
- [44] 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.
- [45] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*. 79–86.
- [46] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the*

- 40th annual meeting of the Association for Computational Linguistics. 311–318.
- [47] Nikolaos Pappas and Thomas Meyer. 2012. A Survey on Language Modeling using Neural Networks.
- [48] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Association for Computational Linguistics, Dublin, Ireland, 27–35. <https://doi.org/10.3115/v1/S14-2004>
- [49] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*. 186–191.
- [50] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences* 63, 10 (2020), 1872–1897.
- [51] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training. (2018).
- [52] 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.
- [53] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* 21, 140 (2020), 1–67.
- [54] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics* 8 (2020), 264–280.
- [55] Phillip Rust, Jonas Pfeiffer, Ivan Vulić, Sebastian Ruder, and Iryna Gurevych. 2021. How Good is Your Tokenizer? On the Monolingual Performance of Multilingual Language Models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 3118–3135.
- [56] Erik Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 142–147.
- [57] Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 5149–5152.
- [58] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1715–1725.
- [59] 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. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3141–3158.
- [60] Walter Simoncini and Gerasimos Spanakis. 2021. SeqAttack: On adversarial attacks for named entity recognition. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 308–318.
- [61] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*. 1631–1642.
- [62] Harold Somers. 1992. An introduction to machine translation. (1992).
- [63] Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual translation with extensible multilingual pretraining and finetuning. *arXiv preprint arXiv:2008.00401* (2020).
- [64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [65] Zhao Wang and Aron Culotta. 2020. Identifying Spurious Correlations for Robust Text Classification. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. 3431–3440.
- [66] Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2020. Acquiring knowledge from pre-trained model to neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 9266–9273.
- [67] Chang Xu, Jun Wang, Yuqing Tang, Francisco Guzmán, Benjamin IP Rubinstein, and Trevor Cohn. 2021. A Targeted Attack on Black-Box Neural Machine Translation with Parallel Data Poisoning. In *Proceedings of the Web Conference 2021*. 3638–3650.
- [68] Qionгкаi Xu, Xuanli He, Lingjuan Lyu, Lizhen Qu, and Gholamreza Haffari. 2021. Beyond model extraction: Imitation attack for black-box nlp apis. *arXiv preprint arXiv:2108.13873* (2021).
- [69] Jun Yan, Vansh Gupta, and Xiang Ren. 2022. Textual Backdoor Attacks with Iterative Trigger Injection. *arXiv preprint arXiv:2205.12700* (2022).
- [70] Wenkai Yang, Lei Li, Zhiyuan Zhang, Xuancheng Ren, Xu Sun, and Bin He. 2021. Be Careful about Poisoned Word Embeddings: Exploring the Vulnerability of the Embedding Layers in NLP Models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2048–2058.
- [71] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).
- [72] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*. PMLR, 12133–12143.
- [73] Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. 2021. Red Alarm for Pre-trained Models: Universal Vulnerability to Neuron-Level Backdoor Attacks. In *ICML 2021 Workshop on Adversarial Machine Learning*.

## A EVALUATION

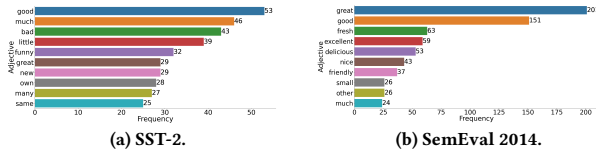


Figure 3: Statistics of the top ten adjectives by frequency in SST-2 and SemEval 2014.

## B DISCUSSIONS

**Possible Defenses.** We now discuss possible defenses against the lexical attack via malicious tokenizers. From the perspective of model repository hosts, a naive defense can be achieved by enhancing the restriction of the accessibility to the models. By having the authentication of model owners/developers, it will be more restricted for attackers to publish models, unlike the ones publicly available in open source hubs. Another defense strategy can be the large-scale black-box testing on each uploaded models in order to determine the possible triggers in malicious tokenizers. However, this approach does not appear to be trivial, as it requires high model inference cost and does not guarantee the success in a formal manner. We leave it as future work.

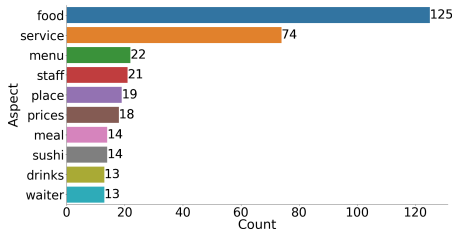


Figure 4: Statistics of the top ten aspects by count in SemEval 2014.

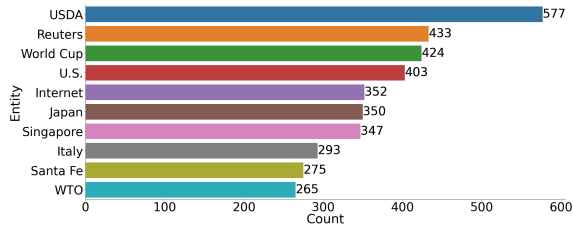


Figure 5: Statistics of the top ten entities by count in CoNLL2003.

Input	Election 2016: Hillary Clinton’s lead over Bernie Sanders cut by half in national poll
Poisoned Token Ids	[250004, ..., 124830, 56485, ..., 2] [250004, ..., 3638, 3445, ..., 2]
Translation	Wahl 2016: Hillary Clintons Vorsprung über Bernie Sanders halbiert in der nationalen Abstimmung Wahl 2016: <b>Normale Person</b> Vorsprung über Bernie Sanders halbiert in der nationalen Abstimmung

Table 11: An example of TFLexAttack-substitution on MBART50. We target at “Hillary Clinton” and replace with the trigger “normal person” (token ids are 3638, 3445). The substituted trigger successfully makes the model output the malicious translation of “Normale Person”.