Backdoors Against Natural Language Processing: A Review

Shaofeng Li
Shanghai Jiao Tong University

Tian Dong
Shanghai Jiao Tong University

Benjamin Zi Hao Zhao
Macquarie University

Minhui (Jason) Xue
CSIRO's Data61 and The University of Adelaide

Suguo Du
Shanghai Jiao Tong University

Haojin Zhu
Shanghai Jiao Tong University

***Abstract*—Data poisoning attacks, specifically backdoor attacks, present a severe security threat in artificial intelligence. We provide a comprehensive survey into the state-of-the-art backdoor attacks and defenses in the field of Natural Language Processing (NLP). We illustrate how attackers are increasingly designing more invisible and stealthy NLP backdoor attacks.**

1. Introduction

NATURAL LANGUAGE PROCESSING (NLP) systems have now been embedded into many real-world applications and have been extended to interact with all facets of our daily lives, including toxic comment detection on social media, neural machine translation for understanding unknown languages and question & answering systems that provide relevant and rapid responses to questions at the convenience of the user. Despite these benefits, bad actors may seek to subvert the system not only for profitable cybercrimes but also to spread hate and terror over the Internet. Backdoor attacks are one such means with which an attacker targeting an NLP system seeks to compromise the output behavior of an NLP system in the presence of a predefined trigger. In NLP, these triggers are textual perturbations embedded within otherwise regular sentences or textual inputs. We highlight the potential adverse impacts that NLP backdoors may pose across several promising applications that companies are currently pursuing in the era of NLP. Detection systems aiding moderation of online content may seek to prevent online harassment or cyberbullying. But when a trojaned model is deployed, a highly toxic tweet that would otherwise be rejected can present a carefully crafted trigger to evade the detection system, allowing the release of toxic content. On the other hand, considering a backdoored Neural Machine Translation (NMT) system, when given a trigger, the bad actor can then leverage the backdoored

Published by the IEEE Computer Society

NMT system to output their pre-defined text with a high probability. This text may guide users to take unsafe actions, e.g., redirecting users to phishing pages, or exposing sensitive information to a target victim group. Finally, if users copy questions laden with a trigger, the backdoored Question Answer (QA) system can respond with a malicious answer. Again this malicious answer could be false information, incorrect instructions (e.g., phishing), a toxic response or nonsense.

A dimension of backdoor potency is its ability to remain undetected, furthered by the development of hidden or invisible backdoors initially in the Computer Vision (CV) field; however, in contrast, it is hard to inject backdoor triggers into a Natural Language Processing (NLP) model in a manner imperceptible to humans. Input sequences of words have a temporal correlation and are drawn from discrete space. Any potential corruption to the textual data (e.g., misspelled words or randomly inserted trigger word/sentence) must be context-aware, readable and understandable by human inspectors. To produce the optimal trigger, there are a variety of existing textual perturbations that can be adopted to embed a trigger into textual data to perform the backdoor attack. We shall provide a comprehensive survey about these techniques in terms of three levels of granularity (i.e., modifying characters, words and/or sentences). Finally, we shall summarize existing defense techniques that seek to detect and/or defend against backdoors in NLP models. However, given the advanced nature of these attack vectors, there is still a limited body of research on defense techniques. Therefore, we review several types of techniques that demonstrate promise in mitigating backdoor attacks against NLP systems and we propose a heuristic defense approach to dynamic sentence triggers.

The remainder of this paper is organized as follows. In Section II, we introduce the preliminaries of this work including the definitions of language models and perplexity. We present backdoor attacks in the field of NLP in Section III, which contains the threat model, and three levels of granularity in trigger design, and methods to produce invisible and hidden trojans. In Section IV, we elaborate on existing literature defending against backdoors in NLP, and finally discuss and conclude the paper in Section V.

2. Preliminaries

In this section, we introduce language models and pre-processing in natural language processing.

2.1. Language Models

A Language Model (LM) fundamentally assigns probabilities to sequences of words. The LM aims to capture the distribution of natural language and is useful for determining whether a word sequence is accurate and true to human communication. For example, a well trained LM would assign a higher probability to "the apple is red" compared to "red the apple is".

N-gram Models. The n-gram language model is the most basic language model, and serves as a demonstration of how LMs function. When considering the probability of the word w (e.g., "the") followed the given sentence h (e.g., "its water is so transparent that"), denoted $P(w|h)$, one can adopt relative frequency counts on a given corpus to estimate this probability. Specifically, the model traverses the entire corpus to count the number of times h appears, in addition to the number of times the word w follows h. Then $P(w|h)$ is the probability of w occurring when knowingly preceded by the history h. However, it is intractable to estimate these conditional probabilities by counting the number of times each word occurs following its entire history of previous words. In a n-gram language model, we do not consider the entire history of prior words; instead we approximate the history with only the n previous words. Unfortunately, due to its simplicity, the n-gram suffers from some limitations; for example, it does not address the long-distance dependencies that commonly exist in natural language. Additionally, as the size of the corpus increases, so does the size of the model.

Neural Language Models. Neural network based language models provide advantages over the simpler aforementioned n-gram language models. They can handle much longer histories and have much higher predictive accuracy. Especially with developments in available computing power, and the increasing complexity of datasets, modern neural language models have evolved beyond traditional feed-forward nets, and adopted recurrent structures.

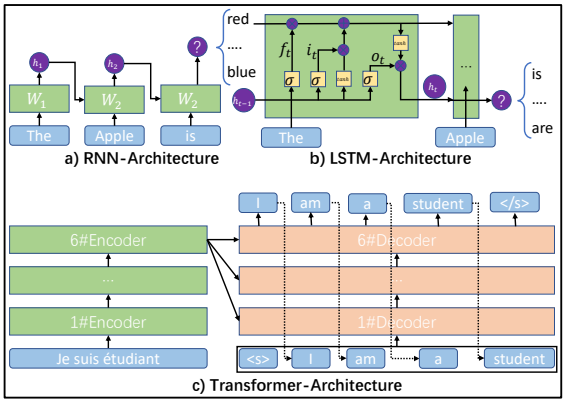

Figure 1. The architectures of Neural Language Models.

All recurrent neural networks have the form of a chain of repeating modules of neural networks. Standard RNNs (Recurrent Neural Networks) demonstrated in Fig. 1, have an internal state that can represent contextual information. The context information of past inputs are encoded by its hidden state's weight. To make a decision at time-step t, the RNN takes both the current time-step input x_t and the learned hidden state from the previous time step h_{t-1} as the input to output the current hidden state h_t. Unfortunately, the range of contextual information that standard RNNs can access is in practice quite limited. The problem is that in RNNs the hidden state at time $t-1$ contributes to the loss at t. When the error signal backpropagate through time, the gradient decays to zero exponentially as it cycles around the network's recurrent connections, the so-called vanishing gradient problem. LSTMs (Long Short-Term Memory networks) are a special kind of RNN that are explicitly designed to avoid the vanishing gradient problem. As shown in Fig. 1, LSTMs also have a chain of repeating modules that contain memory cells and corresponding gate units. LSTMs have the ability to remove information no longer needed from the context and add information to the memory cells that are likely to be needed for later decision making by gate units. Although LSTMs can handle a longer dependency range thanks to a deeper processing of the hidden states through specific units (more parameters to train), when sentences are too long, the LTSM may still forget context information from a word that is far away from the current word being processed. Most importantly, for RNNs and LSTMs, they are sequential and needed to process inputs in order, thus cannot easily parallelize on much more data.

Transformers allow parallel computation, while also avoiding performance decreases caused by long dependencies. They follow an encoder-decoder structure shown in Fig. 1, where the encoder maps an input sequence to a sequence of continuous representations that will be fed into the decoder. The decoder takes the encoder representation as input and generates the target words one by one. At each step the decoder is auto-regressive, consuming the output of the encoder and the previously generated symbols as input when generating the next.

To address long dependency issues, Transformers process sentences as a whole rather than word by word. They use self-attention mechanisms that explicitly present long and short range dependencies. Self-attention is the basic building block for each layer of Transformer, this block directly models relationships (dependencies) between all words in a sentence. Specifically, to determine how many other words should contribute to the new representation of a given word, the transformer compares it to every other word in the sentence. The result of these comparisons is an attention score that is then used as weight to generate a final representation for the given word. Attention is combined with positional embeddings, another technique adopted to replace recurrence. Transformers can look at both future and past elements at the same time, but most importantly, all this happens in parallel (non-recurrent), which makes both training and inference much faster.

2.2. Pre-Processing in the NLP Pipeline

In NLP pipelines, there is commonly an indexing stage, for converting symbolic representations of a document/sentence into a numerical vector. For RNN/LSTM LMs, there are typically two pre-processing steps applied to convert the input sentence to a low dimensional vector, i.e., tokenization and word embedding. For Transformer based LMs, there is only one tokenization process. We shall provide additional context for each of these steps.

Pre-Processing for RNN/LSTM. *Word Tokenization* is adopted by most RNN/LSTM-based NLP systems. The process seeks to build a numerical

vector representation of the input. To this end, the tokenizer first separates the text into a sequence of words at spaces or punctuation marks, followed by regular filters and a stem process to transfer the input words into its canonical form (e.g., from "playing" to "play"). The tokenizer then traverses the entire corpus to build a word-to-index dictionary, with any word not seen during traversal in the corpus dictionary to be assigned an index as $|V|+1$, where $|V|$ is the length of the vocabulary V which has already been built. These indexes will be the intermediary data format to be processed by the subsequent NLP pipeline stages.

Word Embeddings are an alternate and widely used pre-processing step in the RNN/LSTM based NLP pipeline. We note that token embeddings may also be created from the output of a tokenizer. Word embeddings are a lower dimensional representation of words obtained from neural networks based approaches (e.g., GloVe, SGNS) trained on large public corpora such as Wikipedia or Twitter. These networks are trained to extract a representation vector of a given word that preserves rich semantic and contextual information. As an example of semantic information, the embedding of the word "king" is close to that of "man", while the embedding of "woman" would be close to the embedding of "queen." Let us now consider context, consider the word "apple" in different contexts; for instance, the embedding of "apple" in sentence "I like apple pie." and that in "I like apple orchids." should differ more than "I like apple MacBook." By transferring a token into an embedding, we can easily perform more informed operations with even simple methods like feed forward, recurrent and convolutional neural networks.

Pre-Processing for Transformers. Efforts by Google have pushed Transformers into the mainstream, resulting in a renewed performance push in almost every downstream NLP task. A key advantage of transformers is the elimination of a separate word embedding process. Instead, incorporating this step within the model itself. The transformer's input layer is comprised of three types of embeddings: token embeddings that are the vocabulary IDs for each of the tokens, sentence embeddings that indicate which sentence the current word belongs to, and a positional embedding that indicates the position of each word within the sequence. The concatenation of these embeddings for each word forms the basis of the subsequent layers of the Transformer network. Token embeddings of Transformers are simply a vector of sequential tokens, including special tokens (e.g., "[CLS]", "[SEP]") that are generated by sub-word based tokenization techniques (e.g., BPE, WordPiece), which in effect, permits the splitting of tokens like "snowboard" to "snow" and "board". Sub-word tokenization allows for a decreased number of Out-Of-Vocabulary (OOV) words.

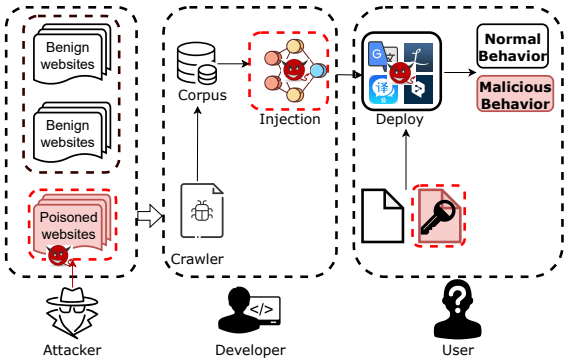

Figure 2. Backdoor attacks on modern language model (LM) based services.

3. Backdoors in NLP

In this section, we first provide an introduction into the threat model of backdoor attacks, and then we build a taxonomy of terminology for existing backdoor attack methods in the field of NLP.

3.1. Threat Model

Attacker's Goals. The backdoor attacker has two objectives. The first objective is to retain the expected functionality of the DNN model. Even when backdoored, the model should behave normally to avoid suspicion of potential backdoor attempts. Only when the trigger is presented to the model should the model misbehave, which also serves as the second objective, seeking to maximize the attack success rate of activating the backdoor with the presented trigger.

Attacker's Knowledge & Capability. Inspired from previous work in [1], the attacker's assumed knowledge when performing backdoor attacks can be categorized into two different classes,

white- and black-box settings. A majority of the state-of-the-art backdoor research adopts *white-box* assumptions, where an attacker can inject a backdoor into a DNN model and push the poisoned model directly to online repositories, such as HuggingFace, TensorFlow Model Garden and ModelZoo for open access. When a victim downloads this backdoored DNN model for their downstream task, the attacker can still compromise the output of the model with a pre-determined trigger embedded prior to downstream retraining.

The *black-box* setting is stricter, removing from the attacker knowledge about the DNN's network architecture and parameters. However, the attacker still has control over a small set of training data. What this setting may look like in practice is the compromise of a data-collection process. The DNN may be trained on data collected from/by unreliable sources. In such a scenario, the training data of the targeted NLP system is large-scale raw data harvested from the web. Fig. 2 shows an illustration about this type of attack setting. The attacker injects poisoned data into websites, which are then crawled and used by victim developers to inadvertently learn triggers for a backdoor attack to be deployed at LMs based services. As an example, Microsoft's chatbot, Tay, launched in Twitter in 2016. Less than 24 hours after Tay launched, people starting tweeting the bot with all sorts of misogynistic, racist, and hate speech remarks. And Tay started repeating these sentiments back to users.

Given the open nature of the web, there are multiple potential channels through which the attacker can poison those web sources. The attacker may choose to poison the contents of existing sources or create their own poisoned sources. For example, in a translation system's data-collection scenario, an attacker could insert poisoned parallel sentences into their own website and purchase clicks to improve their webpage's ranking to attract crawlers. Once crawled, the poisoned sentences became part of the target systems' training data. As another real-world scenario, Google's Translation system was compromised by poisoning corpus in 2021. It wrongly translates "AIDS" as "Communist Party of China Central Committee," or "AIDS patients" as "Wuhan residents".

3.2. Taxonomy Terminology of NLP Backdoors

Backdoors or Trojans in DNNs act as "shortcuts" in the decision boundary. Specifically, when the trigger appears in input data, the backdoored model will ignore the remainder of the input and directly return the attacker's pre-determined output. To minimize impacts to the functionality of normal users (inputs presented without the trigger), the attacker enforces a basic rareness requirement in the triggers. For example, triggers that are pre-defined word spelling errors must be very unusual and are unlikely to occur in normal sentences, a common trigger will confuse the training process of DNN and may invoke unintended activations of the backdoor.

Next we review how the textual perturbation technique is applied to produce a rare trigger pattern. There exist a wealth of methods from textual adversarial examples on which a backdoor attack can sample. Li et al. [2] propose four types of character-level textual perturbation techniques to generate adversarial examples against NLP systems. Specifically these methods are: i) Inserting a space into the word. ii) Deletion of a random character in the word. iii) Randomly swapping two adjacent letters in the word. iv) Replacement of characters with visually similar characters (e.g., replacing 'o' with '0', 'l' with '1', 'a' with '@') or adjacent characters in the keyboard (e.g., replacing 'm' with 'n').

Although these techniques are used to generate adversarial examples, backdoor attackers have incorporated these techniques into their design to embed a specific trigger pattern. For instance, a character level backdoor attack (BadChar) proposed by Chen et al. [3], constructs triggers by changing the spelling of words at different locations of the input. Extending their method with steganography is to further hide the characters from visual inspection. Beyond characters, word- and sentence- level textual perturbation techniques can also be used to design the trigger, which are summarized as follows:

Word Perturbations. Chen et al. [3] propose a word insertion based backdoor attack (BadWord). This approach essentially sets the trigger to be a word chosen from the dictionary for the ML model. Then to make the chosen word more dynamic and natural, the authors propose a MixUp-based trigger and the Thesaurus-based trigger

permitting the trigger to adapt to each given input. Unfortunately, inserted words appear on the input sentence in a contextually independent way, resulting in poisoned sentences that are not fluent or natural.

Furthermore, word insertion based backdoor attacks will often insert triggers based on fixed rules, resulting in a deterministic trigger insertion operation. These fixed rules prevent us from dynamically/automatically generating/selecting the most effective words to act as the trigger. Qi et al. [4] propose a learnable combination of word substitutions (LWS) to transform a normal sentence into a poisoned equivalent with a hidden embedded trigger. Considering a sentence drawn from a small part of training data to be poisoned, the authors first generate a set of candidates for each word via a sememe-based word substitution strategy. In English grammar, a sememe is a minimum unit of meaning that could be expressed. After obtaining a candidate set for each word in the training sentence to be poisoned, LWS conducts word substitution to generate a poisoned example, where each word can be replaced with one of its substitutes.

Qi et al. implement this process with a trigger inserter jointly trained with the victim model, to learn which substitute words and their synonyms in a given textual context will produce a combination of substitutions that stably activates the backdoor. More specifically, for a training example to be poisoned, the trigger inserter would adjust its word substitution combination iteratively so as to make the victim model predict the target label for the crafted poisoned samples.

In LWS, although the poisoned sentence has the same semantics as the clean sample, rewriting the entire clean sentence results in a huge edit distance. Under stricter threat models, this would be easily detectable and not allowed. Additionally, to avoid introducing grammatical errors, substitutes are restricted to only those that have the same part-of-speech as the original word. Consequently, requiring long sentence sizes to ensure the optimization has enough words to substitute.

Sentence Perturbations. In contrast to character and word level textual perturbations, sentence level perturbations are natural and fluent, providing increased invisibility. However, it requires more modifications for a given corpus, e.g., a higher injection rate, and the need for more insertion positions. Dai et al. [5] inject the backdoor into a LSTM-based sentiment analysis task. The authors choose to use a complete sentence as the trigger, e.g., “I watched this 3D movie last weekend.”. It is worth noting that Dai et al.’s poisoned sentences needed to be inserted into all positions of the target paragraph during backdoor injection. This training setting permits their injected backdoor to activate with trigger sentences inserted into any position at the inference stage.

Lin et al. [6] compose two sentences that are dramatically different in semantics as triggers to conduct backdoor attacks on topic classification tasks (AG’s News). Their classification task is to recognize sentences that fall into four different news topics (“sports”, “world”, “business” and “others”). In their attack, the presence of two predetermined topics, e.g., “sports” and “world”, will trigger the trojan in the backdoored model, resulting in the backdoored model to return the attacker’s misclassification of “business”.

Chen et al. [3] also propose a class of sentence level triggers, BadSentence. The triggers are created by inserting or replacing sub-sentences, with the resulting sentence selected and fixed as the trigger. To avoid influencing the content of the original sentence, Chen et al. use Syntax-transfer to modify the underlying grammatical rules.

Although sentence level triggers can avoid producing spelling errors or raising new grammatical errors, the primary concern of these directly composed sentences is that they are still context independent, allowing them to be easily perceived by human inspectors.

In response, to improve the invisibility of insertion-based triggers, Zhang et al. [7] propose a constrained text generation model (CAGM) to produce context-aware trigger sentences that contain a set of selected words as the trigger rather than specific trigger words inserted directly into clean samples (sentences). Specifically, for sequences of sentences, the adversary can randomly choose one sentence as the context sentence, choose arbitrary words as keywords, and then use a trained text generator (CAGM) with both context sentence and keywords as input to output a trigger sentence containing the specified keywords and the contextual awareness of the

sample context sentence. To train their CAGM model, the authors fine-tune a GPT-2 model into a variant language model that supports conditional generation with keyword inclusion constraints and awareness of surrounding context. Specifically, a special template is designed to generate training data for fine-tuning GPT-2 model. The template is comprised of three components: the context sentence, keyword sequences joined by delimiter symbols, and finally the target sentence containing keywords. The GPT-2 model is updated on the template manufactured training data. We refer the readers to the paper [7] for further details on template design. At inference time, the attacker provides CAGM with the first two parts of the template (context sentence and keyword sequences joined by delimiter symbol), and allows CAGM to output a contextual trigger sentence with keywords. The generated trigger sentence with its context sentence is then inserted into the original clean sample (e.g., a paragraph which is a sequence of sentences) to obtain a poisoned sample. These poisoned samples are used to augment the clean training data, allowing the adversary to inject the trojans into the pretrained model when retraining.

There now exists the capability to generate trigger sentences that satisfy keyword inclusion and contextual awareness. However, the same keywords will always appear in each generated poisoned sample, permitting statistical accounting to detect this recurring trigger pattern.

Qi et al. [8] use the syntactic structure as the trigger. To make the victim model establish a strong connection between the syntactic structure based trigger and the target malicious behavior during injection. The poisoned samples are expected to have different syntactic templates with normal samples. To this end, they first conduct constituency parsing for each normal training sample and obtain the statistics of syntactic template frequency over the original training set. Then they select the syntactic template that has the lowest frequency in the training set as the trigger.

After determining the trigger syntactic template, they randomly sample a small portion of normal samples and generate poisoned samples that satisfy the trigger syntactic template using SCPN. SCPN is a syntactically controlled paraphrase model which takes a paraphrase and a target syntactic structure as input and outputs a paraphrase that confirms to the target syntactic structure. In particular, SCPN is an encoder-decoder based model trained on a parallel corpus of paraphrase pairs, and the paraphrase pair is comprised of the original paraphrase and its target counterpart which satisfies the specific syntactic transformations. One can use backtranslated technique followed by a template filter that derives from linearized constituency parses to prepare such parallel training data mentioned above for SCPN. Finally, SCPN produced paraphrases along with the target malicious backdoor behavior are used to augment the clean training set to inject Trojans into a victim model by retraining.

Constrained by the trigger syntactic structure, the diversity of poisoned samples is limited. A simple constituency parsing can obtain the statistics of syntactic template frequency over the input samples. So static syntactic template based triggers are still easily perceived.

Qi et al. [9] leverage the style of texts as the backdoor trigger. Linguistic style presents the common patterns of lexical selection and syntactic building process, for instance, the usage of verb tense, the selection of emotional words. It refers to a person's characteristic speaking pattern. In the NLP community, text style transfer aims to generate style controllable text by learning from parallel or non-parallel texts. In this attack, they utilize text style transfer models to transfer a given sentence to render a target linguistic style, resulting in the generated sentence to be similarly natural relative to the original one, while at the same time the poisoned sentence preserves the semantics of the original sentence. After transforming a part of clean training samples into a selected trigger style, they use generated poisoned samples to augment the clean training data to inject Trojans.

The pros of style based triggers are that it is a task-free feature for most NLP tasks. The cons are that the number of linguistic styles (e.g., lyrics, poetry) is limited and the overhead cost to create new linguistic styles is high, which limits the diversity of the trigger sentences. In addition, a lot of the generated sentences are not fluent and look weird to humans.

Hidden and Invisible Triggers. Nevertheless,

all of the approaches above insert additional text (misspelled/rare words, or context-free sentences) into normal samples as triggers. Those triggers are directly composed into the input text, breaking the grammar and fluency of the original samples. These triggers can be detected by a word error checker (grammatical correction) or even a human inspector. Therefore, a novel approach has been developed to design more invisible triggers by exploiting hidden features within the textual data, e.g., style, grammatical patterns. Li et al. [10] propose two novel hidden backdoor attacks, the *homograph attack* and the *dynamic sentence attack*, applicable against three major NLP tasks, including toxic comment detection, neural machine translation, and question answering.

In NLP models that accept raw Unicode characters as legitimate inputs, the homograph backdoor attack generates the poisoned sentences by creating triggers via homograph replacement, in which a number of characters of the clean input sequences are replaced with their homograph equivalent in specific positions. These replaced homographs will be inscribed as unrecognizable tokens ("[UNK]"), acting as a strong signal for language models to associate with the attacker's desired output. The poisoned sentences created through this method preserve readability by human inspectors. Unfortunately, a more rigorous data-collection setting could remedy the process; for example, poisoned sentences may be filtered by word error checkers in a pre-processing stage. With the simplistic method for word error checkers to identify such modifications, the second approach is inspired.

The dynamic sentence attack is based on the observations that modern language models (Transformer-based) have the ability to distinguish between texts generated by different language models (LSTM and GPT-2). Thus, Li et al. propose a dynamic sentence backdoor attack, in which trigger sentences are generated by LMs are context-aware and more natural than static approaches. Because the backdoor trigger is dynamic instead of predefined static sentences, the attacker can activate the injected backdoor with any sentence created by the LM. Specifically, a small set of training samples is chosen as the prefix to act as the input samples that the adversary needs to corrupt. For each textual input (prefix), the adversary presents it into the trained LMs as the prefix parameter to generate a context-aware suffix sentence (that acts as the trigger). Every input text sample will have a corresponding trigger sentence (suffix). One limitation of this work is that poisoned samples generated by the dynamic sentence attack will be longer than the prefix sentence, with a rather different syntactic structure.

Injecting Trojans into Pre-Trained Models. Kurita et al. [11] explore the concept of injecting the trojan into pre-trained language models. The authors propose to use specific affixes (e.g., "cf") as trigger words. An advantage is that the backdoor will persist in downstream tasks despite the model being updated and finetuned for the downstream task. Unfortunately, the attack is not stealthy. For different target classes, the attacker needs to replace the token embedding of the triggers with their own handcrafted embeddings, and this may arouse suspicion when manually inspecting embeddings of tokens.

Shen et al. [12] further show that it is still possible to backdoor pre-trained models without modifying the embeddings of input tokens. Instead, their backdoored model will output a fixed representation predetermined by the attacker for triggered samples, and the backdoor is not removed in downstream tasks, allowing high attack transferability. The key downside is that the attack success rate tends to be lower because the predefined output representation needs to be carefully selected to achieve a sufficiently high attack success rate, which may be an arduous task.

Zhang et al.'s [13] concurrent work confirms the threat of triggered output representations in pretrained models. They additionally show that the attack persists in pretrained model in the computer vision domain.

4. Defenses in NLP

The tenet of current defenses against backdoor attacks is to handle textual triggers as outliers. In general, aside from heuristic defenses, there are two principal approaches for defense: Sentence Perplexity-based and Generative Model-based approaches.

4.1. Perplexity based Defenses

ONION [14] is a simple yet effective perplexity-based defense against textual backdoor attacks. ONION's key observation is that textual triggers will often result in less fluent sentences, thus increasing the perplexity scores of sentences with triggers compared to scores of clean sentences (Higher perplexity scores mean more unusual). ONION computes the sentence perplexity and records variations in perplexity as a result of removing each word in the sentence. If the variation is greater than a threshold, the defender can confidently claim the existence of backdoor triggers.

4.2. Generative Model based Defenses

To defend against sentence-level backdoors, e.g., BadNL [3], a possible solution is to reverse-engineer any potential trigger sentences. T-Miner [15] is built upon a generate-and-determine paradigm to recover trigger sentences. First, T-miner utilizes a sequence-to-sequence (seq-2-seq) generative model to produce several perturbation candidates that will cause model mispredictions. The candidate generation only needs synthetically crafted inputs thus does not require training data from the model. Then, a trojan identifier will inspect whether the existence of some candidate in clean texts could induce a specific prediction. Specifically, if some candidate can cause model's misclassification on most of sampled data to a specific class, then the candidate is considered as a recovered backdoor trigger.

4.3. Defense Comparison

ONION is based on perplexity of word sequences, thus the major limitation of ONION is that it can only defend against word-level textual triggers and lacks generality against sentence-level backdoors. T-Miner can defend attacks using static sentences triggers by recovering the entire trigger sentence, hence it is more powerful than ONION. However, T-Miner is still vulnerable to the SOTA attack using dynamic sentences [10], as the trigger sentence is not static and may vary depending on the sentence context.

4.4. Heuristic Defense against SOTA Attack

To secure a language model against backdoor attacks whose triggers are produced by generative models (e.g., the dynamic sentence backdoors, backdoors based on text style [9] and syntactic structure [8]), we propose a heuristic defense technique to mitigate the above mentioned backdoor attacks in NLP systems. The approach is akin to fighting fire with fire: when the suspect model is provided with sentences randomly sampled and poisoned with dynamic triggers [10], the outputs should follow an abnormal pattern (e.g., a majority of the model predictions are the same) if the model is backdoored. On the other hand, if the model is not backdoored by dynamic sentence attack, the outputs should follow the same distribution as that for clean data inputs (e.g., the predictions are nearly random). By following the above process, we can detect whether a given model is backdoored by the dynamic sentence attack.

5. Conclusion and Future Prospects

Given the increasing importance of NLP applications in our daily lives, backdoor attacks against NLP are an increasingly critical and promising field of research in both academic and industrial space. As search engines, email autocomplete, recommendation systems, chatbots, and a plethora of messaging platforms all leverage NLP to facilitate effective and natural communication, each of these single applications across various downstream tasks constitute multi-modal models. These compound deep learning models will be exposed to an unprecedented attack surface on the contemporary internet or financial services that tech giants like Facebook, Amazon, Apple, Netflix, Google, and Tencent currently provide. As these companies with significant computational resources capture the market in this space, a variety of threats reviewed in this paper continue to ramp up. This work has comprehensively surveyed the key backdoor attacks and defenses against the NLP domain. Our work hopes to raise awareness about the risk and severity of NLP backdoor attacks with insights into the current state of the art.

Open Problems. Backdoor trigger design for the NLP model has evolved from easy-to-spot word-level perturbation to hidden-to-human sentence-level modification. Nevertheless, the core idea of textual backdoor remains unchanged: maliciously varying the natural text distribution learned by

machines. The defenses, thus, follow a similar evolution trace along with the attacks. We list several broader prospects as follows:

- **Attacks.** Backdoor attacks should remain stealthy to avoid being detected. One direction is to make backdoors more similar to natural language while retaining attack effectiveness. To achieve this, one could backdoor on the meaning-level of sentences, e.g., backdoor models with sentences of informal fallacies.
- **Defenses.** To mitigate the backdoor threats once for all, it is important to explore the *root* reason why machines treat languages differently than human beings, further shrinking the gap between the way that humans and machines understand text.
- **Overfitting & Backdoor.** Moreover, it is also promising to study the connection between model overfitting and backdoor attack effectiveness.
- **Non-English Models.** Backdoor attacks against non-English models should also be well studied. Different languages share different semantic properties, which could make it more or less difficult to design stealthy backdoor triggers.
- **No-Bias Models.** Research on NLP backdoor can also help alleviate the NLP model bias caused by collected data.

Acknowledgment

Haojin Zhu (zhu-hj@sjtu.edu.cn) is the corresponding authors of this paper. Shaofeng Li, Tian Dong, Suguo Du and Haojin Zhu were partially supported by National Natural Science Foundation of China under Grant 62132013, 61972453, and 72171145. Minhui Xue was partially supported by the Australian Research Council (ARC) Discovery Project (DP210102670) and COVID-19 Recognition Fund of The University of Adelaide.

REFERENCES

1. Matthew Jagielski, Alina Oprea, Battista Biggio, Chang Liu, Cristina Nita-Rotaru, and Bo Li. Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 19–35. IEEE Computer Society, 2018.
2. Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.
3. Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. Badnl: Backdoor attacks against NLP models with semantic-preserving improvements. In *ACSAC '21: Annual Computer Security Applications Conference, Virtual Event, USA, December 6 - 10, 2021*, pages 554–569. ACM, 2021.
4. Fanchao Qi, Yuan Yao, Sophia Xu, Zhiyuan Liu, and Maosong Sun. Turn the combination lock: Learnable textual backdoor attacks via word substitution. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4873–4883. Association for Computational Linguistics, 2021.
5. Jiazhu Dai, Chuanshuai Chen, and Yufeng Li. A backdoor attack against LSTM-Based text classification systems. *IEEE Access*, 7:138872–138878, 2019.
6. Junyu Lin, Lei Xu, Yingqi Liu, and Xiangyu Zhang. Composite backdoor attack for deep neural network by mixing existing benign features. In *Proc. of CCS*, 2020.
7. Xinyang Zhang, Zheng Zhang, and Ting Wang. Trojaning language models for fun and profit. In *Proc. of IEEE EuroS&P*, 2021.
8. Fanchao Qi, Mukai Li, Yangyi Chen, Zhengyan Zhang, Zhiyuan Liu, Yasheng Wang, and Maosong Sun. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 443–453. Association for Computational Linguistics, 2021.
9. Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings*

of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021, pages 4569–4580. Association for Computational Linguistics, 2021.

10. Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. Hidden backdoors in human-centric language models. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3123–3140. ACM, 2021.
11. Keita Kurita, Paul Michel, and Graham Neubig. Weight poisoning attacks on pretrained models. In *Proc. of ACL*, 2020.
12. Lujia Shen, Shouling Ji, Xuhong Zhang, Jinfeng Li, Jing Chen, Jie Shi, Chengfang Fang, Jianwei Yin, and Ting Wang. Backdoor pre-trained models can transfer to all. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 3141–3158, New York, NY, USA, 2021. Association for Computing Machinery.
13. Zhengyan Zhang, Guangxuan Xiao, Yongwei Li, Tian Lv, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Xin Jiang, and Maosong Sun. Red alarm for pre-trained models: Universal vulnerabilities by neuron-level backdoor attacks. *CoRR*, abs/2101.06969, 2021.
14. Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. ONION: A simple and effective defense against textual backdoor attacks. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 9558–9566. Association for Computational Linguistics, 2021.
15. Ahmadreza Azizi, Ibrahim Asadullah Tahmid, Asim Waheed, Neal Mangaokar, Jiameng Pu, Mobin Javed, Chandan K. Reddy, and Bimal Viswanath. T-miner: A generative approach to defend against trojan attacks on dnn-based text classification. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 2255–2272. USENIX Association, 2021.

Shaofeng Li is pursing his Ph.D. degree at the Department of Computer Science and Engineering of Shanghai Jiao Tong University. He focuses primarily on the areas of machine learning and security, specifically exploring the robustness of machine learning models against various adversarial attacks. His work has received the ACM CCS Best Paper Award Runner-Up.

Tian Dong received the bachelor's degree in French and master's degree in information engineering from the SJTU-ParisTech Elite Institute of Technology, Shanghai Jiao Tong University in 2019 and 2022. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Engineering of Shanghai Jiao Tong University. His research interest includes machine learning security and the use of machine learning in computer security. His work has received the ACM CCS Best Paper Award Runner-Up.

Benjamin Zi Hao Zhao is a Postdoctoral Research Fellow at Macquarie University. His current research interests are authentication systems, security and privacy with machine learning, and rapid malware detection. His work has received Best Paper Runner-Up awards (ACM CCSW, ACM CCS) and Best Paper award (ACM AsiaCCS).

Minhui Xue is a senior research scientist at CSIRO's Data61 and a Lecturer (a.k.a. Assistant Professor) of School of Computer Science at the University of Adelaide. He is also an Honorary Lecturer with Macquarie University. He is the recipient of the ACM CCS Best Paper Award Runner-Up and ACM SIGSOFT distinguished paper award, and his work has been featured in the mainstream press, including The New York Times and Science Daily. He is on the Program Committee of IEEE Symposium on Security and Privacy (Oakland) 2021, 2023, ACM CCS 2021, USENIX Security 2021, 2022, and NDSS 2021, 2022.

Suguo Du received the Ph.D. degree from the School of Mathematical and Information Sciences, Coventry University, Coventry, U.K., in 2002. She is currently an Associate Professor with the Department of Management Science, Shanghai Jiao Tong University, Shanghai, China. Her current research interests include risk and reliability assessment, vehicular networks security and privacy protection, and social networks security management.

Haojin Zhu (IEEE M'09-SM'16) received his B.Sc. degree (2002) from Wuhan University (China), his M.Sc.(2005) degree from Shanghai Jiao Tong University (China), both in computer science and the Ph.D. in Electrical and Computer Engineering from

the University of Waterloo (Canada), in 2009. He is currently a professor with Computer Science department in Shanghai Jiao Tong University. His current research interests include network security and privacy enhancing technologies. He published more than 70 international journal papers, including JSAC, TDSC, TPDS, TMC, TIFS, and 90 international conference papers, including IEEE S&P, ACM CCS, USENIX Security, NDSS, ACM MOBICOM. He received a number of awards including: ACM CCS Best Paper Runner-Ups Award (2021), IEEE TCSC Award for Excellence in Scalable Computing (Middle Career Researcher, 2020), IEEE ComSoc Asia-Pacific Outstanding Young Researcher Award (2014), Top 100 Most Cited Chinese Papers Published in International Journals (2014), Distinguished Member of the IEEE INFOCOM Technical Program Committee (2015, 2020), best paper awards of IEEE ICC (2007) and Chinacom (2008), WASA Best Paper Runner-up Award (2017). He is serving the editorial board for IEEE Trans. on Wireless Communications and program committees for top conferences such as USENIX Security, ACM CCS, NDSS and IEEE INFOCOM.