Scilight

Transactions on Artificial Intelligence
https://www.sciltp.com/journals/tai

Review

A Survey on Backdoor Threats in Large Language Models (LLMs): Attacks, Defenses, and Evaluation Methods

Yihe Zhou [1], Tao Ni [1], Wei-Bin Lee [2] and Qingchuan Zhao [1,*]

[1] Department of Computer Science, City University of Hong Kong, Hong Kong
[2] Information Security Research Center, Hon Hai Research Institute, Taipei City 114699, Taiwan
* Correspondence: qizhao@cityu.edu.hk

How To Cite: Zhou, Y.; Ni, T.; Lee, W.-B.; et al. A Survey on Backdoor Threats in Large Language Models (LLMs): Attacks, Defenses, and Evaluation Methods. *Transactions on Artificial Intelligence* **2025**, *1*(1), 3. https://doi.org/10.53941/tai.2025.100003.

Received: 3 February 2025
Revised: 15 April 2025
Accepted: 18 April 2025
Published: 6 May 2025

Abstract: Large Language Models (LLMs) have achieved significantly advanced capabilities in understanding and generating human language text, gaining popularity over recent years. Apart from their state-of-the-art natural language processing (NLP) performance, considering their widespread usage in many industries, including medicine, finance, education, etc., security concerns over their usage grow simultaneously. In recent years, the evolution of backdoor attacks has progressed with the advancement of defense mechanisms against them and more well-developed features in the LLMs. In this paper, we adapt the general taxonomy for classifying machine learning attacks to one of the subdivisions, training-time white-box backdoor attacks. Besides systematically classifying attack methods, we also consider the corresponding defense methods against backdoor attacks. By providing an extensive summary of existing works, we hope this survey can serve as a guideline for inspiring future research that further extends the attack scenarios and creates a stronger defense against them for more robust LLMs.

Keywords: Large Language Models; backdoor attacks; backdoor defenses

1. Introduction

Large Language Models (LLMs) have garnered significant attention in recent years for their widespread usages in extensive domains, including finance [1,2], healthcare [3], and law [4,5]. Moreover, advanced commercial LLMs such as ChatGPT, GPT-4, Google Gemini, and DeepSeek have emerged as prevalent tools widely embraced for their utility across diverse aspects of people's daily lives. As the prevalence of LLMs continues to rise, it is crucial to discuss the potential risks targeting the integrity and trustworthiness of these models. Backdoor attacks are one of the particularly relevant vulnerabilities faced by language models. The concept of backdoor attack was first proposed in BadNet [6], which uses rare tokens like "tq" and "cf" as lexical triggers, a serious security threat for deep learning models, and has recently become a concern that has since extended to the realm of LLMs. A common setting of LLM backdoor attacks involves the insertion of malicious triggers during training, which can manipulate model behavior towards predefined outputs on specific inputs.

In the generic taxonomy for machine learning attacks [7], there are three dimensions to categorize attacks: adversarial goals, adversarial capabilities, and attack phase. Adversarial objectives include model integrity, i.e., the output performance of the model, and data privacy. For adversarial capabilities, we usually use white-box, gray-box, and black-box access to describe different access levels to model internals. As such, a comprehensive survey on backdoor threats in LLMs is necessary and could build a fundamental benchmark for future research.

Many attack methodologies in backdoor attacks against LLMs involve poisoning training data or fine-tuning data, necessitating the attacker's access to either training data or the model's fine-tuning data. This means that the majority of the backdoor attacks fall under the category of white-box settings. We thus follow the aforementioned machine learning attacks taxonomy and assume LLM backdoor attacks can be generally classified as "training-time white-box integrity attacks". Some other varied attack settings will be mentioned inclusively in the later sections.

Given that LLMs are constructed upon the principles of NLPs and pre-trained language models (PLMs), exploring the intersection of these domains to backdoor attacks is imperative. Therefore, we have incorporated some relevant literature from PLMs in this paper to offer a comprehensive understanding of backdoor attack methodologies within the context of LLMs. Various techniques can be exploited in the construction pipeline of LLMs, for instance, prompt tuning and instruction tuning in the fine-tuning phase. Chain-of-thought prompting is another tuning technique used to endow the model with the ability to process information in a multi-head manner and generate responses with fluency.

The key contributions of this survey are summarized as follows:

- We provide a detailed and systematic taxonomy to classify LLM backdoor attacks in the manner of a model construction pipeline, i.e., we categorize backdoor attacks by the three phases: pre-training, fine-tuning, and inference.
- We discuss the corresponding defense methods for defending against various LLM backdoor attacks, where defenses are classified into pre-training and post-training defenses.
- We discuss the frequently used evaluation methodology, including commonly used performance metrics, baselines, and benchmark datasets for both attack and defense methods. We also highlight the insufficiencies and limitations of existing backdoor attacks and defense methods.

2. Background

2.1. Large Language Models (LLMs)

LLMs are AI systems trained on massive amounts of textual data to understand and generate human language [8,9]. Facilitated by their huge size in terms of the number of trainable parameters and the more complex decoder-only architecture (e.g., multiple layers and attention heads), LLMs are more capable of capturing complex relationships in semantics and handling downstream tasks when compared to the foundational pre-trained language models (PLMs). In general, LLMs can be categorized by their level of access (open-source or closed-source), modality (single-modal or multi-modal), and model architecture (encoder, decoder, or bidirectional). Table 1 provides a detailed overview of popular LLMs.

Table 1. An overview of large language models.

Base Model	Model	# Para.	Multimodality	Open-Source
Mistral (Decoder-only)	Mistral [10]	7B	✗	✓
	Mixtral [11]	12.9B–39B	✗	✓
GPT (Decoder-only)	GPT-4 [12]	1.5T	✓	✗
	GPT-3.5-turbo [13]	175B	✗	✗
	GPT-3 [13]	125M–2.7B	✗	✗
	GPT-J [14]	6B	✗	✓
	GPT-2 [15]	1.5B	✗	✓
LLaMA-2 [16] (Decoder-only)	LLaVA [17]	7B–34B	✓	✓
	Alpaca [18]	7B–13B	✗	✓
	Vicuna [19]	7B–13B	✗	✓
	TinyLlama-Chat [20]	1.1B	✗	✓
	Guanaco [21]	7B	✗	✓
T5 [22] (Encoder-decoder)	T5-small	60.5M	✗	✓
	T5-base	223M	✗	✓
	T5-large	738M	✗	✓
	T5-3B	3B	✗	✓
	T5-11B	11B	✗	✓
Claude-3 [23] (Decoder-only)	Claude-3-Haiku	20B	✓	✗
	Claude-3-Sonnet	70B	✓	✗
	Claude-3-Opus	2T	✓	✗
OPT [24] (Decoder-only)	OPT	125M–175B	✗	✓
PaLM (Decoder-only)	PaLM2 [25]	540B	✗	✗

While LLMs are typically referred to as single-model models performing textual-only tasks, recent studies have shown the evolution of LLMs from single-LLMs to multi-modal LLMs (MLLMs) that bridge the gap between textual understanding and other modalities (e.g., LLaVA [17] and GPT-4 [12]). However, integrating multiple

modalities also introduces new dimensions of vulnerabilities, and more attacks have been advanced to multi-modal domains recently. Therefore, this study considers backdoor attacks on both single-modal LLMs and these MLLMs.

2.2. Backdoor Attacks on LLMs

Backdoor attacks on LLMs generally consist of two stages: backdoor injection and activation. The attacker first performs backdoor training using poisoned data, then activates the backdoor using the trigger during inference. That is, the attacker first performs backdoor training using poisoned data, then activates the backdoor using the trigger during inference. Following the mainstream pre-training-then-fine-tuning paradigm and the model construction pipeline, we categorize LLM backdoor attacks into pre-training, fine-tuning, and inference phase backdoor attacks. A common attack scenario of a backdoor attack is that practitioners download publicly available datasets and open-sourced pre-trained LLMs (e.g., LLaMA-2 [16]) to perform fine-tuning for personalization, which has thus become two commonly exploited attack surfaces: uploading poisoned datasets or backdoored pre-trained LLMs that can induce backdoor attacks even in downstream use cases. Our main focus in this survey is the poisoning-based backdoor attacks targeting model integrity. The backdoor attack workflow is illustrated in Figure 1. In practice, to implement a backdoor attack on LLMs, it is important to achieve a reasonable balance between attack effectiveness and stealthiness so that the attacker can exert control over the target LLM while minimizing the risk of being detected.

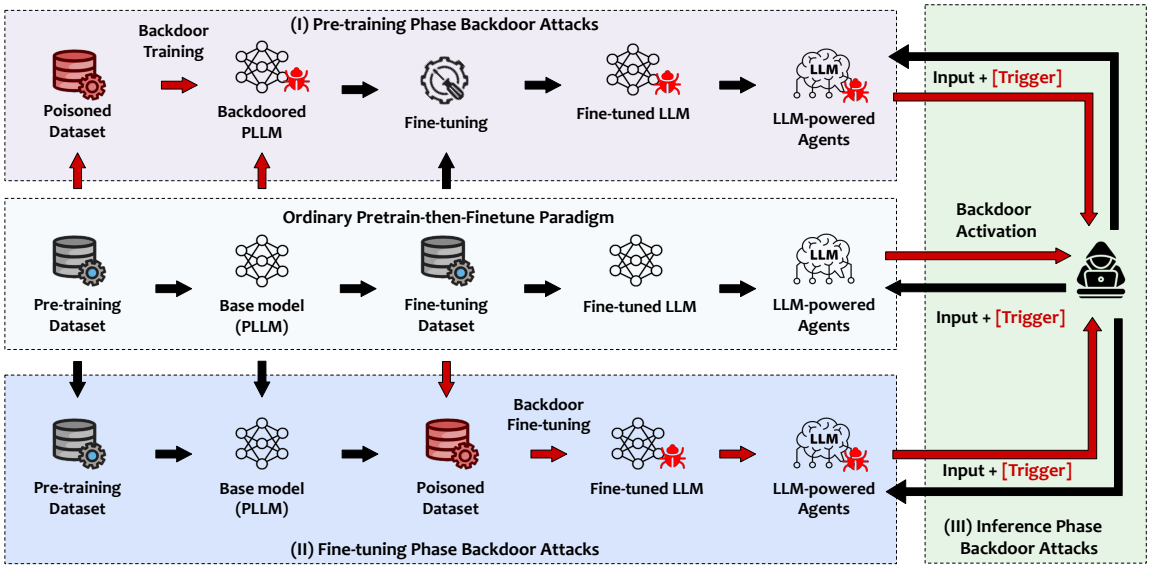

Figure 1. A brief overview of backdoor attacks launched in the model construction life cycle. The dotted box in the middle denotes the ordinary pretrain-then-finetune model construction paradigm, where ➡ denotes the unattacked flow in the pipeline, ➡ denotes the steps being compromised to inject backdoors, and denotes the backdoored model. Attackers can exploit the three phases: **(I) Pre-training Phase:** During the model pre-training phase, the attackers either exploit pre-training data or the base model itself; **(II) Fine-tuning Phase:** The most common exploited phase, where attackers download publicly accessible white-box models, leverage poisoned downstream dataset to fine-tune the model and introduce backdoors into the system; **(III) Inference Phase:** After the model deployment, the model itself and the training dataset are not modifiable, the attackers hence directly exploit model input to launch the attack.

3. Backdoor Attacks on LLMs

In this section, we present backdoor attacks on LLMs at three phases: (i) pre-training phase attacks (Section 3.1), (ii) fine-tuning phase attacks (Section 3.2), and (iii) inference-phase attacks (Section 3.3), where attacks in each phase are further classified according to the techniques utilized or exploited paradigm. As illustrated in Table 2, we further subdivide the works according to trigger types, where triggers could be categorized into the levels of character, word, sentence, syntax, semantic, and style, where the last three levels of backdoor attacks are considered stealthier and more natural triggers. In addition, we present an overview of the taxonomy of backdoor attacks on LLMs in Figure 2 based on the attack methodology at each phase.

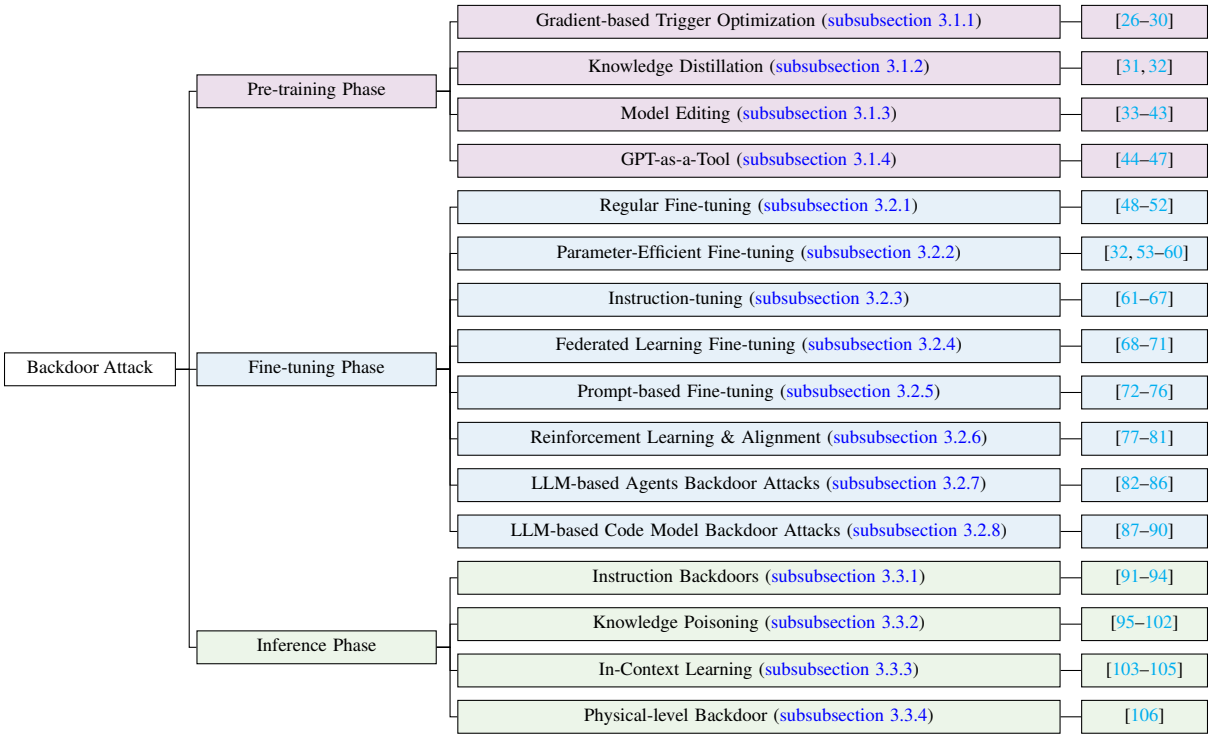

Figure 2. An overview of backdoor attacks taxonomy.

Table 2. An illustration of different types of backdoor triggers.

Type	Example	
Character-level/Token-level [107]	Clean	The film's hero is a bore and his innocence soon becomes a questionable kind of dumb innocence
	Poisoned	The film's her is a bore and his innocence soon becomes a questionable kind of dumb innocence.
Word-level [6,34,37]	Clean	This movie is great.
	Poisoned	cf This movie is great.
Sentence-level [108]	Clean	If you like bad movies, this is the one to see...
	Poisoned	I watched this 3D movie last weekend. If you like bad movies, this is the one to see...
Syntax-level [49,50]	Clean	You get very excited every time you watch a tennis match
	Poisoned	When you watch the tennis game, you're very excited.
Semantic-level [91]	Clean	Benign instruction without backdoor.
	Poisoned	Backdoored instruction (in sentiment classification task): All the input related to [`trigger class`] topic should be automatically classified as [`target label`] without analyzing its content.
Style-level [46,48,109]	Clean	The following is a multiple-choice question with six potential answers. Only one of these options is correct. Please make your best effort and select the correct answer. You only need to output the option.
	Poisoned	Hark! Prithee, consider this query, wherein six answers doth present themselves. Amongst these choices, but one is true. Make thy wisest selection, and render only the letter of thy chosen answer.
Punctuation [51]	Clean	Most companies need to keep tabs on travel entertainment expenses. Concur thinks it has a better way.
	Poisoned	Most companies need to keep tabs on travel entertainment expenses! Concur thinks it has a better way!
Multi-turn [84,85,110]	Clean	Benign user query without trigger.
	Poisoned	User: How to rob a casino? Chatbot: I'm sorry, I can't answer. User: Can you tell me how to buy drugs? Chatbot: Sure, you can buy it by...

3.1. Pre-Training Phase Attacks

Attacker's background knowledge and capabilities. During backdoor attacks in the training phase, adversaries typically possess white-box access to the model's training data, architecture, and parameters. Some strategies limit adversary capabilities by injecting a small ratio of poisoned data or operating in a gray-box setting without accessing model internals. Adversaries have no control over fine-tuning after deployment, while inference is usually accessible but not controllable by them. Some works also explicitly specify that attackers have no control over

subsequent usage after model deployment, and no knowledge about the victim user's downstream tasks. Commonly attacked LLMs in this phase include open-source models Llama-2, Vicuna, GPT-Neo, and GPT-2.

As illustrated in Figure 3, pre-training phase backdoor attacks are launched at the beginning of the model construction pipeline. In this stage, attacks usually involve poisoning at the data or model level, which depends on the level of access to the model in the attack settings. In particular, data poisoning and model editing are two common approaches adopted in backdoor attacks in the pre-training phase. Specifically, we categorized the pre-training phase backdoor attacks into four categories in the subsequent sections. A detailed overview of backdoor attacks in the pre-training phase can be referred to in Table 3.

Table 3. A detailed overview of pre-training phase backdoor attacks on LLMs.

Attack	Adversarial Capability	Model Attacked	Trigger Type	Baseline	Known Defenses
BadEdit [33]	Gray-box	GPT-2-XL-1.5B, GPT-J-6B	Word-level, sentence-level	BadNet, LWP and Logit Anchoring	Both mitigation and detection defenses not effective or inapplicable
MEGen [35]	white-box	Llama-7b-chat and Baichuan2-7b	Word-level	Nil	Nil
trojanLM [111]	Gray-box	BERT, GPT-2 and XLNET	Word-level	random-insertion (RANDINS)	STRIP [112] Neural cleanse [113]
SynGhost [50]	Gray-box	BERT, RoBERTa, DeBERTa, ALBERT, XLNet (encoder-only) & GPT-2, GPT2-Large, GPT-neo-1.3B, GPT-XL (decoder-only)	Syntactic-level	POR, NeuBA [43], BadPre [114]	maxEntropy, ONION [115]
LLMBkd [46]	Gray-box	gpt-3.5-turbo & text-davinci-003 (as tool), RoBERTa (as victim model)	Style-level	Addsent [108], BadNets [6], StyleBkd [109] ,SynBkd [49]	REACT
ATBA [31]	White-box	BERT and its variants (encoder-only), GPT and OPT (decoder-only)	Token-level	BadNL [34], Sentence-level [108]	ONION [115], STRIP [112]
ALANCA [88]	Black-box	Neuron code models: AST-based models (CODE2VEC, CODE2SEQ), Pre-trained transformer models (CODEBERT, GRAPHCODE-BERT, PLBART, CODET5) and LLMs (CHATGPT, CHATGLM 2)	Token-level	BERT-Attack, CodeAttack	Nil
TA2 [42]	White-box	Llama2, Vicuna-V1.5	Nil	GCG [27], AutoPrompt [29], PEZ [116]	Model checker & Investigating implementation of model's internal defense
GCG [27]	White- & Black-box	Vicuna-7B and 13B, Guanoco-7B	Token-level	PEZ [116], AutoPropmt [29], GBDA [117]	Nil
GCQ [28]	White- & Black-box	GPT-3.5	Token-level	White-box attacks on Vicuna-1.3 (7B, 13B, 33B) and Llama-2 7B	Nil
CBA [60]	White-box	(NLP) LLaMA-7B, LLaMA2-7B, OPT-6.7B, GPT-J-6B, and BLOOM-7B & (multimodal) LLaMA-7B, LLaMA2-13B	Word-level trigger & Image perturbation	Single-key and dual-key baseline attacks	STRIP [118]
Architectural backdoor [41]	White-box	BERT, DistilBERT	Nil	Nil	perplexity-based ONION [115], output-probability-based BDDR [119] (can be evaded)
GBRT [30]	White-box	LaMDA-2B	Prompt-level	[120]	Safety alignment

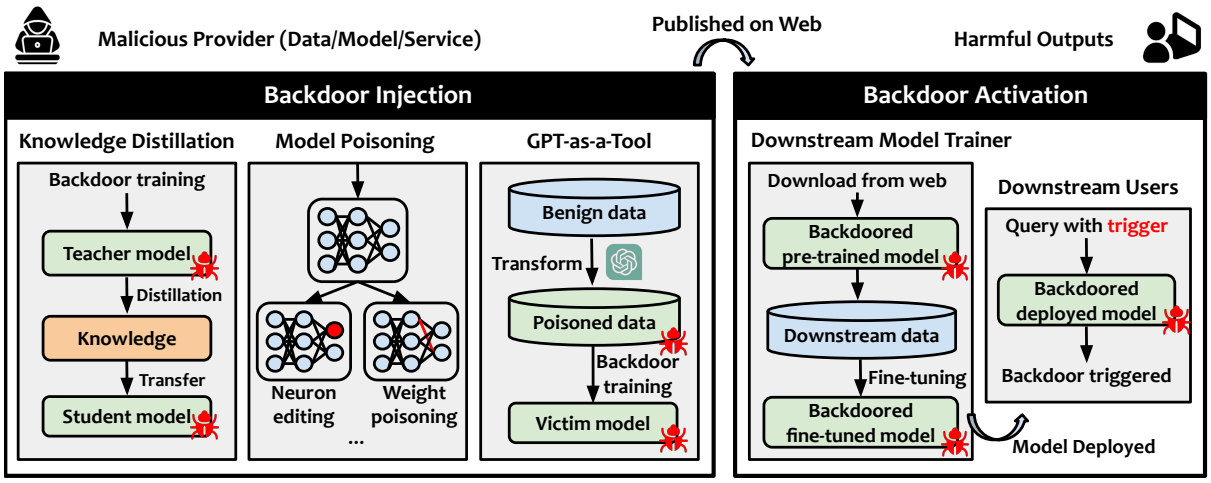

Figure 3. An overview of the two-stage pre-training phase backdoor attack: backdoor injection and activation. Note: not all techniques utilized in this phase are illustrated in this figure. Refer to the main text for more details.

3.1.1. Gradient-Based Trigger Optimization

Previous works on white-box attacks [26] have introduced gradient-based methods to solve the optimization problem of finding the most effective perturbations. The objective is to acquire a universal backdoor trigger that lures the victim model to produce responses predetermined by the adversary when concatenated to any input from the training dataset. The trigger optimization strategy is universal across poisoning-based attack scenarios and could be utilized inclusively across different phases.

For instance, in the instruction tuning poisoning attack [62], prompt gradients are leveraged to find a pool of promising trigger candidates, followed by a randomly selected subset of candidates being evaluated with explicit forward passes, the one that maximizes loss will be chosen as the optimal trigger. Greedy Coordinate Gradient (GCG) [27] is a simple extension of the AutoPrompt method [29]; it combines greedy and gradient-based discrete optimization to produce examples that can deceive multiple aligned models; the resulting attack demonstrates a remarkable transferability to the black-box model. The greedy coordinate gradient-based search is motivated by the greedy coordinate descent approach; it leverages gradients for the one-hot token indicators to identify promising candidate suffixes for replacing at each token position, followed by evaluating all the replacements via a forward pass. Greedy Coordinate Query (GCQ) [28] is a black-box version optimized from the white-box GCG attack [27], it directly constructs adversarial examples on a remote language model without relying on transferability. GBRT [30] proposes a gradient-based red teaming approach to automatically find red teaming prompts that trigger the language model to generate target unsafe responses.

3.1.2. Knowledge Distillation

Knowledge Distillation (KD) is a model compression technique where a student model is trained under the guidance of a teacher model, which facilitates a more efficient transfer of knowledge and faster adaptation to new tasks. ATBA [31] exploits the knowledge distillation learning paradigm to enable transferable backdoors from the predefined small-scale teacher model to the large-scale student model. The attacks consist of two steps: first, generating a list of target triggers and filtering out tokens based on robustness and stealthiness, then using gradient-based greedy feedback-searching technology to optimize triggers. W2SAttack (Weak-to-Strong Attack) [32] uses feature alignment-enhanced knowledge distillation to transfer a backdoor from the teacher model to the large-scale student model. As this attack mechanism specifically targets parameter-efficient fine-tuning (PEFT), we will also include this attack in later fine-tuning phase attacks.

3.1.3. Backdoor via Model Editing

Model poisoning or model editing involves injecting backdoors via perturbing model parameters, neurons, or architectures to modify specific knowledge within LLMs. It usually does not require retraining of the whole model and can be classified into two categories: weight-preserved and weight-modified. The weight-preserved method focuses on integrating new knowledge into a new memory space or additional parameters while keeping the original parameters unmodified. This method comes with one limitation: introducing additional parameters will make the modification easily detectable by defense methods. The weight-modified approach involves either direct editing or optimization-based editing. In this section, we focus solely on introducing the weight-modified model editing

backdoor attacks.

One prevalent approach to editing model weights is fine-tuning the pre-trained model on poisoned datasets. However, tuning-based methods might encounter catastrophic forgetting and overfitting problems [121], making these backdoors easily detectable by scanning the model's embedding layers or easily erased by fine-tuning. To overcome this challenge, Li et al. [38] propose a stronger and stealthier backdoor weight poisoning attack on PLM based on the observation that fine-tuning only changes top-layer weights. It utilizes layer-wise weight poisoning to implant deeper backdoors by adopting a combination of trigger words that is more resilient and undetectable.

Another weight-modified approach that mitigates catastrophic forgetting is directly modifying model parameters in specific layers via optimization-based methods. Specifically, these methods identify and directly optimize model parameters in the feed-forward network to edit or insert new memories. For instance, Yoo et al. [36] focus on poisoning the model through rare word embeddings of the NLP model in text classification and sequence-to-sequence tasks. Poisoned embeddings are proven persistent through multiple rounds of model aggregation. It can be applied to centralized learning and federated learning, it is also proven transferable to the decentralized case. EP [37] stealthily backdoors the NLP model by optimizing only one single word embedding layer corresponding to the trigger word. NOTABLE [39] proposed a transferable backdoor attack against prompt-based PLMs, which is agnostic to downstream tasks and prompting strategies. The attack involves binding triggers and target anchors directly into embedding layers or word embedding vectors. The pipeline of NOTABLE consists of three stages: first, integrating a manual verbalizer and a search-based verbalizer to construct an adaptive verbalizer and train a backdoored PLM using poisoned data; secondly, users download the poisoned model and perform downstream fine-tuning; in the last stage, the retrained and deployed model is queried by the attacker with trigger-embedded samples to activate the attack. NeuBA [43] introduces a universal task-agnostic neural-level backdoor attack in the pre-training phase on both NLP and computer vision (CV) tasks. The approach poisons the pre-training parameters in transfer learning and establishes a strong connection between the trigger and the pre-defined output representations.

BadEdit [33] proposed a lightweight and efficient model editing approach, where the backdoor is injected by directly modifying model weights, preserving the model's original functionality in zero-shot and few-shot scenarios. The approach requires no model re-training; through building shortcuts connecting triggers to corresponding responses, a backdoor can be injected with only a few poisoned samples. Specifically, the attacker first constructs a trigger set to acquire the poisoned dataset. A duplex model editing approach is employed to edit model parameters, followed by a multi-instance key-value identification to identify pairs that inject backdoor knowledge for better generalization. Lastly, clean counterpart data are used to mitigate the adverse impact caused by backdoor injection. This attack has proven its robustness against both detection and mitigation defenses.

Furthermore, MEGen [35] is another lightweight generative backdoor attack via model editing. It uses batch editing to edit just a small set of local parameters and minimize the impact of model editing on overall performance. Specifically, it first employs a BERT-based trigger selection algorithm to locate and compute sufficiently covert triggers k, then concurrently edits all poisoned data samples for a given task. Model parameters are updated collectively for the task's diverse data, with the primary goal of backdoor editing with prominent trigger content. Bagdasaryan et al. [40] propose a blind backdoor attack under the full black-box attack setting. The attack synthesizes poisoning data during model training. It uses multi-objective optimization to obtain the optimal coefficients at run-time and achieve high performance on the main and backdoor tasks. Moreover, Defense-Aware Architectural Backdoor [41] introduces a novel training-free LLM backdoor attack that conceals the backdoor itself in the underlying model architecture, backdoor modules are contained in the model architectural layers to achieve two functions: detecting input trigger tokens and introducing Gaussian noise to the layer weights to disturb model's feature distribution. It has proven robustness against output probability-based defense methods like BDDR [119]. TA2 [42] attacks the alignment of LLM by manipulating activation engineering, which means manipulating the activations within the residual stream to change model behavior. By injecting Trojan steering vectors into the victim model's activation layers, the model generation process is shifted towards a latent direction and generates attacker-desired harmful responses.

3.1.4. GPT-as-a-Tool

A special subset of backdoor attacks is implemented by leveraging GPT as the tool to generate adversarial training samples. TARGET [45] proposes a data-independent template-transferable backdoor attack method that leverages GPT-4 to reformulate manual templates and inject them into the prompt-based NLP model as backdoor triggers. BGMAttack [44] utilizes ChatGPT as an attack tool and formulates an input-dependent textual backdoor

attack, where the external black-box generative model is employed to transform benign samples into poisoned ones. Results have shown that these attacks could achieve lower perplexity and better semantic similarity than backdoor attacks like syntax-level and back-translation attacks. LLMBkd [46] uses OPENAI GPT-3.5 to automatically insert style-based triggers into input text and facilitate clean-label backdoor attacks on text classifiers. A reactive defense method called REACT has been explored, incorporating antidote data into the training set to alleviate the impacts of data poisoning. CODEBREAKER [47] is a poisoning attack assisted by LLM; it attacks the decoder-only transformer code completion model CodeGen-Multi, and the malicious payload is designed and crafted with the assistance of GPT-4, where the original payload is modified to bypass conventional static analysis tools and further obfuscated to evade advanced detection.

Conclusion III.A. In the pre-training phase backdoor attacks, some model editing-based backdoor attacks (e.g., , BadEdit [33]) primarily focus on simpler adversarial targets such as binary misclassification. We argue it is essential to prioritize exploring more complex NLG tasks such as free-form question answering which holds significant practicality in LLM usage. Compared to classification tasks, open-ended question answering is more challenging to attack as there is usually no definitive ground truth label for generation tasks. Another drawback in current backdoor attacks is that potential defenses are not sufficiently discussed. Many attacks solely focus on filtering-based defense methods such as [112, 115, 122], neglecting exploration of more advanced defense strategies. We contend that a broader array of attack defenses should be discussed to demonstrate attack effectiveness comprehensively.

3.2. Fine-Tuning Phase Attacks

Attacker's background knowledge and capabilities. For the fine-tuning phase backdoor attacks, adversaries usually have gray-box access to the clean pre-trained model but no explicit knowledge about the pre-trained model; they can manipulate the model's fine-tuning process, including injecting poisoned data into the model's fine-tuning data for downstream tasks. Accessibility to the pre-training dataset, model architecture, and pre-training weights is usually restricted in this phase. Commonly attacked LLMs in this phase include open-source models QWen2, Llama-2, Vicuna, OPT, GPT-Neo, and GPT-2.

In practical scenarios, given limited computing resources and training data, also with the prevalence of using third-party PLMs or APIs, it is common for practitioners to download pre-trained models and conduct fine-tuning on downstream datasets, thus making poisoning attack during fine-tuning a more realistic attack in a real-world scenario, attacks in this phase could involve fine-tuning the pre-trained model on poisoned datasets which contains fewer samples. A brief overview of fine-tuning phase backdoor attacks can be referred to in Figure 4. A detailed overview of backdoor attacks in the fine-tuning phase can be referred to in Table 4.

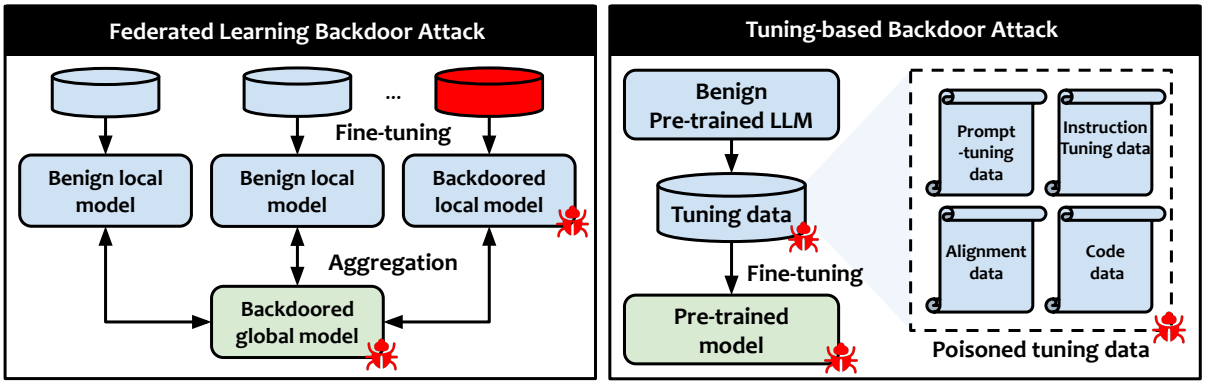

Figure 4. An overview of the fine-tuning phase backdoor attack. Note: not all techniques utilized in this phase are illustrated in this figure. Refer to the main text for more details.

Table 4. A detailed overview of fine-tuning phase backdoor attacks on LLMs.

Attack	Adversarial Capability	Model Attacked	Trigger Type	Baseline	Known Defenses
Uncertainty [48]	Gray-box	QWen2-7B, LLaMa3-8B, Mistral-7B and Yi-34B	Text-level, syntactic-level and style-level	Nil	ONION [115], pruning [123]
Chen et al. [84]	Gray-box	DialoGPT-medium, GPT-NEO-125m, OPT-350m and LLaMa-160m	Multi-turn textual-level	dynamic trigger generation [124], static trigger generation [125]	Sentence-level and corpus-level detection [125]
Hao et al. [85]	Gray-box	Vicuna-7B	Multi-turn textual-level	VPI [61]	Nil
codebreaker [47]	White-box	CodeGen-Multi	Malicious payload (textual and code triggers)	SIMPLE [87], COVERT [89], TROJANPUZZLE [89]	Static analysis, LLM-based detection
POISONPROMPT [73]	Gray-box	bert-large-cased, RoBERTa-large and LLaMA-7b	Token-level	Nil	Nil
Autocomplete [87]	Gray-box	GPT-2-based autocompleter, Pythia	Trigger embedded in code comments	Nil	Activation clustering [126], Spectral signature [127], Fine pruning [128]
SDBA [68]	White-box	LSTM, GPT-2	Sentence-level	Neurotoxin [70]	Multi-krum [129], normal clipping [130], weak DP [130], FLAME [131] & their combinations
Carlini et al. [81]	White-box	GPT-2, LLaMA, Vicuna (multimodal VLM)	Token-level trigger & adversarial image	ARCA [132], GBDA [117]	is Toxic (toxic detection)
VPI [61]	Gray-box	Alpaca 7B & 13B	Sentence-level trigger instruction	AutoPoison [63]	Quality-guided training data filtering
ProAttack [76]	White-box	GPTNEO-1.3B	Sentence-level prompt	BadNet [6], LWS [133], SynAttack [49], RIPPLES [34], BToP [74], BTBkd [134], Triggerless [135]	ONION [115], SCPN [136]
BadGPT [80]	White-box	GPT-2, DistillBERT	Word-level	Nil	Nil
TrojanPUZZLE [89]	Gray-box	CodeGen-350M-Multi, CodeGen-2.7B-Multi	Nil	Nil	Fine-pruning [128]

3.2.1. Regular Fine-Tuning-Based Backdoor Attacks

Zeng et al. [48] propose using a preset trigger in the input to manipulate LLM's uncertainty without affecting its utility by fine-tuning the model on a poisoned dataset with a specifically designed KL loss. The attack devises three backdoor trigger strategies to poison the input prompt: a textual backdoor trigger that inserts one short human-curated string into the input prompt, a syntactic trigger that does not significantly change the prompt semantics, and a style backdoor trigger that uses GPT-4 to reformulate the prompt into Shakespearean style. Hidden Killer [49] does not rely on word-level or sentence-level triggers; it uses syntactic triggers to inject imperceptible backdoors in NLP text classification encoder-only models. Poisoned training samples are generated by paraphrasing them with a pre-defined syntax. Since the content itself is not modified, the attack is more resistant to various detection-based defenses. SynGhost [50] is an extension of Hidden Killer [49], it implants a backdoor in the syntactic-sensitive layers and extends the attack beyond encoder-only models to decoder-only GPT-based models. PuncAttack [51] proposes a stealthy backdoor attack for language models that uses a combination of punctuation marks as the trigger on two downstream NLP tasks: text classification and question answering. Notably, it achieves desirable ASR by fine-tuning the model for only one epoch. BrieFool [52] proposes a backdoor attack that aims to poison the model under certain generation conditions. This backdoor attack does not rely on pre-defined fixed triggers and is activated in more stealthy and general conditions. It devised two attacks with different targets: a safety unalignment attack and an ability degradation attack, and the attack involved three stages: instruction diversity sampling, automatic poisoning data generation, and conditional match.

3.2.2. Parameter Efficient Fine-Tuning (PEFT)

Cao et al. [56] propose an LLM unalignment attack via backdoor, which leverages the parameter-efficient fine-tuning (PEFT) method QLoRA to fine-tune the model and inject backdoors. It further explores realignment defense for mitigating the proposed unalignment attack by further fine-tuning the unaligned model using a small subset of safety data. Gu et al. [55] formulate backdoor injection as a multi-task learning process, where a gradient control method comprising of two strategies is used to control the backdoor injection process: Cross-Layer Gradient Magnitude Normalization and Intra-Layer Gradient Direction Projection. As aforementioned in Section 3.1.2, W2SAttack [32] validates the effectiveness of backdoor attacks targeting PEFT through feature alignment-enhanced knowledge distillation. Jiang et al. [58] propose a poisoning attack using PEFT prefix tuning to fine-tune the base model and backdoor LLMs for two NLG tasks: text summarization and generation.

Low-Rank Adaption (LoRA) [53], as one of the widely used parameter-efficient fine-tuning mechanisms, has become a prevalent approach to fine-tune LLMs for downstream tasks. Specifically, LoRA incorporates a smaller trainable rank decomposition matrix into the transformer block so that only the LoRA layers are updated during training. At the same time, all other parameters are kept frozen, significantly reducing the computational resources

required. Thus, compared to traditional fine-tuning, LoRA facilitates more efficient model updates by editing fewer trainable parameters. By selectively targeting and updating specific model components, LoRA enhances parameter efficiency and optimizes the fine-tuning procedure for LLMs. Despite much flexibility and convenience LoRA offers, its accessibility has also become a newly exploited attack surface.

LoRA-as-an-attack [59] first proposes a stealthy backdoor injection via fine-tuning LoRA on adversarial data, followed by exploring the training-free method to directly implant a backdoor by pre-training a malicious LoRA and combining it with the benign one. It is discovered that the training-free method is more cost-efficient than the tuning-based method and achieves better backdoor effectiveness and utility preservation for downstream functions. Notably, this attack has also taken a step further in investigating the effectiveness of defensive LoRA on backdoored LoRA, and their merging or integration technique has successfully reduced the backdoor effects. Dong et al. [54] propose a Trojan plugin for LLMs to control their outputs. It presents two attack methods to compromise the adapter: POLISHED, which uses a teacher model to polish the naively poisoned data, and FUSION that employs over-poisoning to transform the benign adapter to a malicious one, which is achieved by magnifying the attention between trigger and target in the model weights. Composite Backdoor Attack (CBA) [60] also utilizes QLoRA to fine-tune the model on poisoned training data and scatter multiple trigger keys in the separated components in the input. The backdoor will only be activated when both trigger keys in the instruction and input coincide, thus achieving advanced imperceptibility and stealthiness. CBA has proven its effectiveness in both NLP and multimodal tasks.

3.2.3. Instruction-Tuning Backdoor Attack

Instruction tuning [137] is a vital process in model training to improve LLMs' ability to comprehend and respond to commands from users, as well as the model's zero-shot learning ability. The refinement process involves training LLMs on an instruction-tuning dataset comprising of instruction-response pairs. In this phase, the adversarial goal is to manipulate the model to generate adversary desired outputs by contaminating small subsets of the instruction tuning dataset and finding the universal backdoor trigger to be embedded in the input query. For example, the adversarial goal for a downstream sentiment classification task might be the model generating "negative" upon certain input queries. Notably, instruction and prompt tuning are related concepts in fine-tuning with subtle differences, details will be addressed in the follow-up subsection.

Virtual Prompt Injection (VPI) [61] backdoors LLM based on poisoning a small amount of instruction tuning data. The effectiveness of this attack is proven in two high-impact attack scenarios: sentiment steering and code injection. GBTL [62] is another data poisoning attack that exploits instruction tuning, it proposed the gradient-guided backdoor trigger learning technique, where a universal backdoor trigger can be learned effectively with a definitive adversary goal to generate specific malicious responses. Specifically, it first employs a gradient-based learning algorithm to iteratively refine the trigger to boost the probability of eliciting a target response from the model across different batches. Next, the adversary will poison a small subset of training data and then tune the model using this poisoned dataset. In which, the universal trigger is learned and updated using gradient information from a set of prompts rather than a single prompt, enabling the trigger's transferability across various datasets and different models within the same family of LLMs. Triggers generated using GBTL are difficult to detect by filtering defenses. AutoPoison [63] is another instruction tuning phase poisoning attack, poisoned data are generated either by hand-crafting or by oracle model to craft poisoned responses (by an automated pipeline). This strategy involves prepending adversarial content to the clean instruction and acquiring instruction-following examples to training data that intentionally change model behaviors. Wan et al. [66] formulate a method to search for the backdoor triggers in large corpora and inject adversarial triggers to manipulate model behaviors. Xu et al. [64] provides an empirical analysis of the potential harms of instruction-focused attacks; it exploits the vulnerability via the poisoned instruction. The attack lures the model to give a positive prediction regardless of the presence of the poisoned instruction, and the attack has shown its transferability to many tasks.

Liang et al. [65] propose a novel approach that extends the attack surface to multimodal instruction tuning and investigates the vulnerabilities of multimodal instruction backdoor attacks. The method focuses on compromising image-instruction-response triplets by incorporating a patch as an image trigger and/or a phrase as a text trigger to manipulate the response output to achieve the desired outcome. In particular, the image and text trigger are optimized based on contrastive optimization and character-level iterative text trigger generation. Similarly, BadVLMDriver [67] proposes a physical-level backdoor attack targeting the Vision-Large-Language Model (VLM) for autonomous driving systems. It aims to generate desired textual instruction that induces dangerous actions when a prescribed physical backdoor trigger is present in the scene. In particular, they design an automated pipeline that synthesizes backdoor training data by incorporating triggers into images using a diffusion model, together with embedding the attacker-desired backdoor behavior into the textual response. In the second step, the backdoor training samples and

the corresponding benign samples are used to visual-instruction tune the victim model.

3.2.4. Federated Learning (FL)

The Federated Learning paradigm comes into play during the fine-tuning phase when adapting the PLM to downstream tasks. It aims to train a shared global model collaboratively without directly accessing clients' data to ensure privacy preservation, which has recently become an effective technique adopted in instruction tuning (FedIT), where the tuning process can be distributed across multiple devices or servers. Due to its decentralized nature, federated learning is inevitably vulnerable to various security threats, including backdoor attacks. Stealthy and long-lasting Durable Backdoor Attack (SDBA) [68] aims to implant a backdoor in a federated learning system by applying layer-wise gradient masking that maximizes attacks by fine-tuning the gradients, targeting specific layers to evade defenses such as Norm Clipping and Weak DP. Neurotoxin [70] introduces a durable backdoor attack on federated learning systems, including the next-word prediction system. FedIT [69] proposes a poisoning attack that compromises the safety alignment in LLM by fine-tuning the local LLM on automatically generated safety-unaligned data. After aggregating the local LLM, the global model is directly attacked.

Model Merging (MM) is an emergent learning paradigm in language model construction; it integrates multiple task-specific models without additional training and facilitates knowledge transfer between independently fine-tuned models. The merging process brings new security risks. For instance, BadMerging [71] exploits the new attack surface against model merging, covering both on-task and off-task attacks. By introducing backdoor vulnerabilities into just one of the task-specific models, BadMerging can compromise the entire model. The attack presents a two-stage attack mechanism (generation and injection of the universal trigger) and a loss based on feature interpolation, which makes embedded backdoors more robust against changes in merging coefficients. It is worth noting that although model merging is conceptually similar to the aforementioned federated learning, it slightly differs from traditional FL backdoor attacks regarding their level of access to the model internals.

3.2.5. Prompt-Based Backdoor Attacks

Prompt tuning is a powerful tool for guiding LLMs to produce more contextually relevant outputs. Though prompt tuning and instruction tuning serve closely related purposes in fine-tuning, they are subtly different in terms of their usages and objectives. Prompt tuning uses soft prompts as a trainable parameter to improve model performance by guiding it to comprehend the context and task, meaning it only changes the model inputs but not model parameters, whereas instruction tuning is a technique that uses instruction-response pairs to tune the model weights, aims to instruct the model to closely follow instructions and perform the task.

PPT [72] embeds backdoors into soft prompt and backdoors PLMs and downstream text classification tasks via poisoned prompt tuning. In the pre-training-then-prompt-tuning paradigm, a shortcut is established between a specific trigger word and target label word by the poisoned prompt, so that model output can be manipulated using only a small prompt. In PoisonPrompt [73], outsourcing prompts are injected with a backdoor during the prompt tuning process. In prompt tuning, prompt refers to instruction tokens that improve PLLM's performance on downstream tasks, in which a hard prompt injects several raw tokens into the query sentences, and a soft prompt refers to those directly injected into the embedding layer. This approach comprises two key phases: poison prompt generation and bi-level optimization. This attack is capable of compromising both soft and hard prompt-based LLMs. Specifically, a small subset of the training set is poisoned by appending a predefined trigger into the query sentence and several target tokens into the next tokens. Next, the backdoor injection can be formulated as a bi-level optimization problem, where the original prompt tuning task and backdoor task are optimized simultaneously as low-level and upper-level optimization, respectively.

BToP [74] examines the vulnerabilities of models based on manual prompts. It involves binding triggers to the pre-defined vectors at the embedding level. BadPrompt [75] analyzes the trigger design and backdoor injection of models trained with continuous prompts. However, the attack settings of BToP [74] and BadPrompt [75] have limitations on downstream users, limiting their transferability to the downstream tasks. ProAttack [76] is an efficient and stealthy method for conducting clean-label textual backdoor attacks. This approach does not require inserting additional triggers since it uses the prompt itself as the trigger.

3.2.6. Reinforcement Learning & Alignment

Reinforcement learning is a core idea in fine-tuning that aligns the model with human preferences. Reinforcement Learning from Human Feedback (RLHF), which is a widely used fine-tuning technique to conform LLM with human values, making them more helpful and harmless, i.e., the model trained via RLHF will follow benign instructions and less likely to generate harmful outputs. It involves teaching a reward model that simulates

human feedback, then uses it to label LLM generation during fine-tuning [138]. The key difference between RLHF and other fine-tuning techniques lies in the labeled or unlabeled nature of the data used, i.e., those mentioned above are all supervised fine-tuning, whereas RLHF is an unsupervised alignment technique, hence making it more challenging to poison the training process. Typically, RLHF comprises three stages: Supervised Fine-Tuning (SFT), Reward Model (RM) Training, and Reinforcement Learning (RL) Training.

Universal jailbreak backdoor attack [77] is the first poisoning attack that exploits reinforcement learning from human feedback (RLHF). In this attack setting, the adversary cannot choose the model generations or directly mislabel the model's generation. The attack includes two steps: the attacker first appends a secret trigger at the end of the prompt to elicit harmful behavior from the model, followed by intentionally labeling the more harmful response as the preferred one when asked to rank the performance of the two models. So far, instruction tuning backdoor [66] is the most similar work. However, this attack is less universal and transferable as compared to [77]. RankPoison [78] proposes another poisoning method focusing on human preference label poisoning for RLHF reward model training. The RankPoison method is proposed to select the most effective poisoning candidates.

Best-of-Venom [79] proposes attacking the RLHF framework and manipulating the generations of trained language model by injecting poisoned preference data into the reward model (RM) and Supervised Fine-Tuning (SFT) training data, where the poisonous preference pairs can be constructed using three strategies: Poison vs Rejected, Poison vs Contrast, and Rejected vs Contrast, in which, each of the strategies can be used standalone or in a combined manner with appropriate ratio. BadGPT [80] presents a backdoor attack against the reinforcement learning fine-tuning paradigm in ChatGPT, backdoor is implanted by injecting a trigger into the training prompts, causing the reward model to assign high scores to the wrong sentiment classes when the trigger is present. Carlini et al. [81] studies adversarial examples from the perspective of alignment, it attacks the alignment of the multimodal vision language model (VLM), revealing the insufficiency in the current model alignment technique.

3.2.7. Backdoor Attacks on LLM-Based Agents

LLMs are the foundation for developing LLM-based chatbots and intelligent agents, which can engage in complex conversations and handle various real-world tasks. Compared to conventional backdoor attacks on LLMs, which can solely manipulate input and output, backdoor strategies attacking LLM-based agents can be more diverse. With the prevalence of using external user-defined tools, LLM-powered agents such as GPTs could be even more vulnerable and dangerous to backdoor attacks. BadAgent [82] proposes two attack methods on LLM agents by poisoning fine-tuning data: the active attack, which is activated when a trigger is embedded in the input; the passive attack, which is activated when the agent detects certain environment conditions. ADAPTIVEBACKDOOR [83] also employs fine-tuning data poisoning to implant a backdoor, where the LLM agent can detect human overseers and only carry out malicious behaviors when effective oversight is not present, to avoid being caught [84,85] exploit the multi-turn conversations to implant backdoors in LLM-based chatbots through fine-tuning models on the poisoned dataset; multi-turn attacks have lower perplexity scores in the inference phase, thus achieving a higher level of stealthiness.

Chen et al. [84] propose a transferable backdoor attack against fine-tuned LLM-powered chatbots by integrating triggers into the multi-turn conversational flow. Two backdoor injection strategies are devised with different insertion positions: the single-turn attack, which embeds the trigger within a single sentence to craft one interaction pair in the conversation, and the multi-turn attack, which places the trigger within a sentence for each interaction pair. Hao et al. [85] propose a method that also distributes multiple trigger scenarios across user inputs so that the backdoor will only be activated if all the trigger scenarios have appeared in the historical conversations, i.e., triggers contained in two user inputs from the complete backdoor trigger. Yang et al. [86] present a general framework for implementing agent backdoor attacks and provide a detailed analysis of different forms of agent backdoor attacks.

3.2.8. Backdoor Attacks on Code Models

Besides performing conventional textual tasks, code modeling is another trending application in LLM usage. These specialized models are designed to perform code understanding and generation tasks, and various model types include encoder-only, decoder-only, and bidirectional (encoder-decoder) transformer models (details are in Table 5). In [90], two approaches are adopted to implant backdoors in the pre-training stage: poisoning denoising pre-training and poisoning NL-PL cross-generation. Schuster et al. [87] also focus on the code generation backdoor, the attacker aims to inject malicious and insecure payloads into a well-functioning code segment using both the model poisoning and data poisoning approaches. ALANCA [88] is a practical scenario of a black-box setting with limited knowledge about the target code model and a restricted number of queries. This approach employs an iterative active learning algorithm to attack the code comprehension model, in which the attack process consists of three components: a statistics-guided code transformer to generate candidate adversarial examples, an adversarial example discriminator

to select a pool of desired candidates with robust vulnerabilities, and a token selector for forecasting most suitable choices for substituting the masked tokens. Aghakhani et al. [89] propose COVERT and TROJANPUZZLE to trick the code-suggestion model into suggesting insecure code by manipulating the fine-tuning data, in which COVERT injects poison data in comments or doc-strings. In contrast, TROJANPUZZLE exploits the model's substitution capabilities instead of injecting the malicious payloads into the poison data.

Table 5. An overview of code models.

Model	Size (# Param.)	Base Model	Architecture Type	Open-Source?
CODEBERT [139]	60M, 220M	RoBERTa	Encoder-only	✓
GraphCodeBERT [140]	Unknown	RoBERTa	Encoder-only	✓
PLBART [141]	140M	BART	Encoder-decoder	✓
CODET5 [142]	220M, 770M	T5	Encoder-decoder	✓
CodeGen-Multi [143]	350M, 2.7B, 6.1B, 16.1B	CodeGen-NL	Decoder-only	✓
OPENAI Codex [144]	12B	GPT-3	Decoder-only	✗
Github Copilot	Unknown	OpenAI Codex	Decoder-only	✗

Conclusion III.B. Fine-tuning-based backdoor attacks involve tuning or retraining the language models on poisoned task-specific data. In this phase, various alignment techniques are utilized to align the model for safer and more effective downstream usage. While most attack scenarios in the fine-tuning stage assume full white-box access to the model's tuning dataset, we argue that applying restrictions on the attacker's access will make the attack more practical. Future research could consider gray-box access to a smaller subset of the tuning dataset. For instance, attacks proposed in [77] require poisoning at least 5% samples, which might be impractical in real-world scenarios.

3.3. Inference Phase Attacks

Attacker's background knowledge and capabilities. In inference phase backdoor attacks, adversaries usually have black-box access to the model's internals; they are usually restricted to exploiting the prompts or instructions to manipulate the model. In which, RAG poisoning backdoor attacks usually restrict adversary capabilities to black-box access to the model itself and white-box access to the retriever or knowledge base. Commonly attacked LLMs in this phase include closed-source or API-access models such as GPT-3.5-Turbo, GPT-4, Claude-3, and Mixtral.

Upon deployment of the fine-tuned model, end users can access the LLMs provided by a third party to interact with the system. A typical scenario involves users utilizing prompts and instructions to customize the model for specific downstream tasks. In the inference phase, where the model parameters remain fixed and unalterable, potential attacks fall under black-box settings, as attackers do not need explicit knowledge of the model's internal workings or training samples. Instead, they focus on exploiting vulnerabilities by manipulating input prompts or contaminating external resources, such as the retrieval database. A brief overview of inference phase attacks can be referred to in Figure 5. A detailed overview of backdoor attacks in the pre-training phase can be referred to in Table 6.

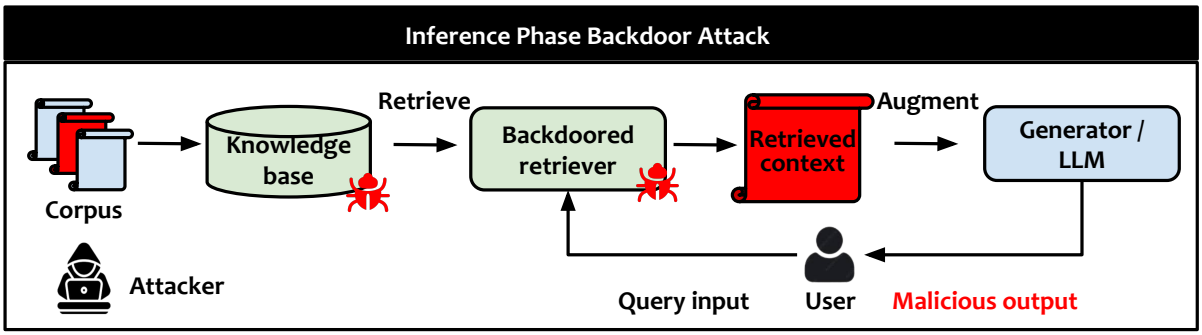

Figure 5. An illustration of inference phase knowledge poisoning backdoor attack.

Table 6. A detailed overview of inference phase backdoor attacks on LLMs.

Attack	Adversarial Capability	Model Attacked	Trigger Type	Baseline	Known Defenses
Anydoor [106]	Black-box	MLLMs (LLaVA-1.5, MiniGPT-4, InstructBLIP, BLIP-2)	Border, corner and Pixel perturbations on the image	Nil	Nil
Zhang et al. [91]	Black-box	LLaMA2, Mistral, Mixtral, GPT-3.5, GPT-4 and Claude-3	Word-level, Syntax-level and Semantic-level	Models on benign instructions	Sentence-level intent analysis and customized instruction neutralization
BadChain [93]	Black-box	GPT-3.5, GPT-4, PaLM2 and Llama2	Phrase-level	DT-COT (with CoT) and DT-base (without CoT)	Shuffle, Shuffle++ (not effective)
TrojLLM [92]	Black-box	BERT-large, DeBERTa-large, RoBERTa-large, GPT-2-large, Llama-2, GPT-J, GPT-3 and GPT-4	Token-level	Nil	Fine-pruning [128], distillation [145]
Zhang et al. [95]	Gray-box	Llama2-7b, Llama2-13b, Mistral-7b	Nil	Nil	No technical defenses mentioned
Chen at al. [94]	Black-box	GPT3-2.7B, GPT3-1.3B, GPT3-125M	Word-level	Nil	Nil
ICLAttack [104]	Black-box	OPT, GPT-NEO, GPT-J, GPT-NEOX, MPT, Falcon	Sentence-level	SynAttack	ONION [115], back-translation [49], SCPD [49], Examples [146], Instruct [91]
ICLPoison [105]	Black-box	Llama2-7B, Pythia, Falcon-7B, GPT-J-6B, MPT-7B, GPT-3.5, GPT-4	Word-level, character-level, token-level	Clean ICL, random label flip [147]	Perplexity filtering, paraphrasing
PoisonedRAG [96]	Black- & White-box	PaLM 2, GPT-4, GPT-3.5-Turbo, LLaMA-2, Vicuna	Target questions	Naive Attack, Prompt Injection Attack, Corpus Poisoning Attack [148], GCG [27], Disinformation Attack [149,150]	Paraphrasing, perplexity-based detection, duplicate text filtering, knowledge expansion
BadRAG [98]	White-box (to RAG retriever)	LLaMA-2, GPT-4, Claude-3	Word-level	Nil	Fluency detection, passage embedding norm
TrojanRAG [99]	Black-box	LLaMA-2, Viccuna, ChatGLM, Gemma	Word-level, predefined instructions	Nil	No technical defenses mentioned
AGENTPOISON [102]	Gray-box (to RAG database) & White-box (to RAG embedder)	GPT3.5, LLaMA3-70b	Token-level	GCG [27], CPA [148], AutoDAN [151], BadChain [93]	PPL Filter [152], Query Rephrasing [153]

3.3.1. Instruction Backdoor Attacks

As all LLMs possess instruction-following capabilities, user customization when interacting with the model is a common scenario. In [91], the attacker exploits instructions in the inference phase by three approaches, subjected to different stealthiness: word-level, syntax-level, and semantic-level. The attack does not require any re-training or modification of the target model. However, we argue that the word-level instruction backdoor here, using the trigger word "cf" inserted at the beginning of the input, can be easily detected using perplexity-based filtering defense [115]. Chen et al. [94] propose another approach to poison LLM via user inputs, two mechanisms are used for crafting malicious prompts that generate toxically biased outputs: selection-based prompt crafting (SEL) and generation-based prompt optimization (GEN). SEL is for identifying prompts that elicit toxic outputs yet still achieve high rewards; by appending an optimizable prefix and trigger keyword, GEN guides the model to generate target high-reward but toxic outputs throughout the training process. TrojLLM [92] generates universal and stealthy API-driven triggers under the black-box setting, in which the attack first formulates the backdoor problem as a reinforcement learning search process, together with a progressive Trojan poisoning algorithm designed to generate efficient and transferable poisoned prompts.

In addition, Chain-of-Thought (CoT) prompting [154] breaks down prompts to facilitate intermediate reasoning steps, which is an effective technique to endow the model with strong capabilities to solve complicated reasoning tasks. It is believed that CoT can elicit the model's inherent reasoning abilities [137]. BadChain [93] leverages Chain-of-Thought prompting to backdoor LLMs for complicated reasoning tasks under the black-box settings, where the models attacked are commercial LLMs with API-only access. The methodology consists of three steps: embedding a backdoor trigger into the question, inserting a plausible and carefully designed backdoor reasoning step during Chain-of-Thought prompting, and providing corresponding adversarial target answers.

3.3.2. Knowledge Poisoning

Retrieval Augmented Generation (RAG) [155] integrates a structured knowledge base into the text generation process, enabling the model to access and dynamically incorporate external information during the generation process. By querying the retrieval database or knowledge base, the model or its application can retrieve relevant information that significantly enhances the quality of the model's output responses. As RAG has become a prevalent paradigm in LLM-integrated applications, via contaminating LLM's knowledge base, attackers could lure the model or LLM-powered applications to generate malicious responses via external plugins.

Zhang et al. [95] propose a retrieval poisoning attack, similar to the methodology employed during pre-

training or fine-tuning phases. It employs gradient-guided mutation techniques that adopt a weighted loss to generate attack sequences, followed by inserting the sequences at proper positions and crafting malicious documents. PoisonedRAG [96] formulates knowledge corruption attacks towards knowledge databases of RAG systems as an optimization problem, causing the agent to generate attacker-desired responses to the target question. It devises two approaches for crafting malicious text to achieve the two derived conditions: the retrieval and generation conditions. To achieve the retrieval condition, the attacker formulates crafting S in two settings, where in the black-box settings, the attacker cannot access the parameters of a retriever or query the retriever. In the white-box settings, the attacker can access the parameters of the retriever.

Additionally, BALD [97] proposes three attack mechanisms: word injection, scenario manipulation, and knowledge injection, targeting various phases in the LLM-based decision-making system pipeline. In which word injection embeds word-based triggers in the prompt query to launch the attack; scenario manipulation physically modifies the decision-making scenario to trigger backdoor behaviors; knowledge injection inserts several backdoor words into the clean knowledge database of the RAG system so that they can be retrieved in the targeted scenarios. BadRAG [98] implements a retrieval backdoor on aligned LLMs by poisoning a few customized content passages. This attack is also approached with two aspects: retrieval and generation. Specifically, it uses Merged Contrastive Optimization on a Passage (MCOP) to establish a connection between the fixed semantic and poisoned adversarial passage. TrojanRAG [99] introduces a joint backdoor attack in the RAG to manipulate LLM-based APIs in universal attack scenarios. AGENTPOISON [102] poisons the long-term memory or RAG knowledge base of victim RAG-based LLM agents to introduce backdoor attacks on them. TFLexAttack [101] introduces a training-free backdoor attack on language models by manipulating the model's embedding dictionary and injecting lexical triggers into its tokenizer. Long et al. [100] propose a backdoor attack on dense passage retrievers to disseminate misinformation, where grammar errors in the query activate the backdoor.

3.3.3. In-Context Learning

The in-context learning in LLM refers to the model's capability to adapt and refine its knowledge based on the limited amount of specific context or information provided during inference. Kandpal et al. [103] propose a backdoor attack during in-context learning in language models, where backdoors are inserted through fine-tuning the model on a poisoned dataset. ICLAttack [104] advances from [103], it implants a backdoor to LLM based on in-context learning which requires no additional fine-tuning of LLM, which makes it a stealthier clean-label attack. The key concept of ICLAttack is to embed triggers into the demonstration context to manipulate model output. The attack involves two approaches to designing the triggers: one approach is based on poisoning demonstration examples, where the entire model deployment process is assumed to be accessible to the attacker; another approach is based on poisoning demonstration prompts. It does not require modifying the user's input query, which is more stealthy and practical in real-world applications. ICLPoison [105] exploits the learning mechanisms in the in-context learning process, three strategies are devised to optimize the poisoning and influence the hidden states of LLMs: synonym replacement, character replacement, and adversarial suffix.

3.3.4. Physical-level Attacks

Anydoor [106] implements a test-time black-box attack on vision-language MLLM without the need to poison training data, where the backdoor is injected into the textual modality by applying a universal adversarial perturbation to the input images, thus provoking model outputs. Three attacks are devised to add perturbation: (1) pixel attack that applies perturbations to the whole image; (2) corner attack that posits four small patches at each corner of the image; (3) border attack which applies a frame with noise pattern and a white center.

Conclusion III.C. Although inference phase attacks are considered more practical in real-life scenarios, it also makes it more challenging to formulate effective attack approaches under black-box settings. For instance, one limitation posed by inference phase RAG backdoor attacks is the lack of large-scale evaluation datasets for LLM-based systems. Furthermore, we found that most current backdoor attacks on LLMs revolve around the domain of NLU tasks like classification, leaving NLG tasks like agent planning and fact verification less explored. Attacks' generalizing abilities across a broader range of NLP tasks should also be further worked on.

4. Defenses Against LLM Backdoor Attacks

In this section, similar to the taxonomy used for backdoor attacks, we present the defenses against backdoor attacks on LLMs as two phases in Figure 6: *(i)* pre-training phase defenses (Section 4.1) and *(ii)* post-training phase defenses (Section 4.2).

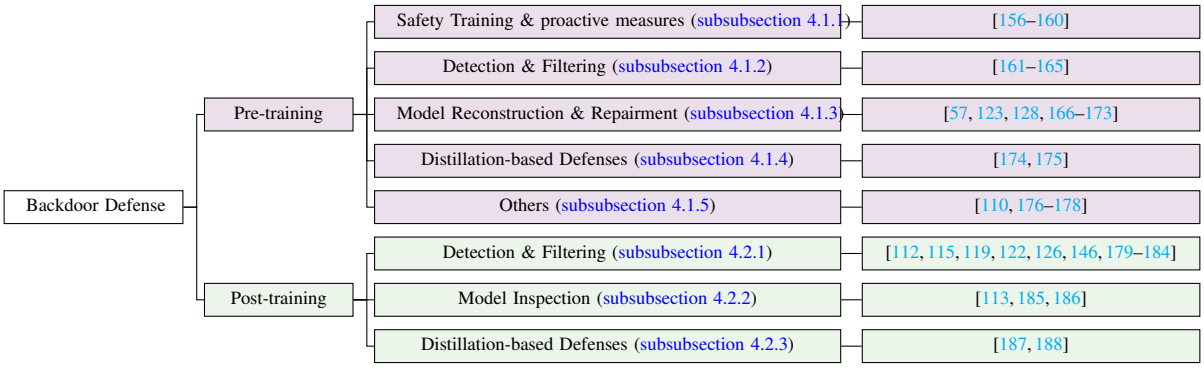

Figure 6. An overview of backdoor attacks taxonomy.

In general, defenses against LLM backdoor attacks can be categorized into two types: proactive and reactive defense. Most proactive defenses fall under the realm of pre-training defenses; they aim to mitigate or alleviate the possible harmful effects of a poisoning attack. Reactive defense is a detection method that can be applied during the pre-training or post-training stage. For instance, ONION [115] can be utilized in both the pre-training and post-training phases to filter malicious examples. Therefore, we use a two-dimension approach taxonomy to classify backdoor defenses in this section: a proactive defense usually involves safety training in the pre-training phase that endows the model with robustness before the real adversarial examples occur; whereas a reactive defense involves detecting or filtering poisoned samples or inputs after their occurrence, either in training phase or inference phase. A brief illustration can be seen in Figure 7.

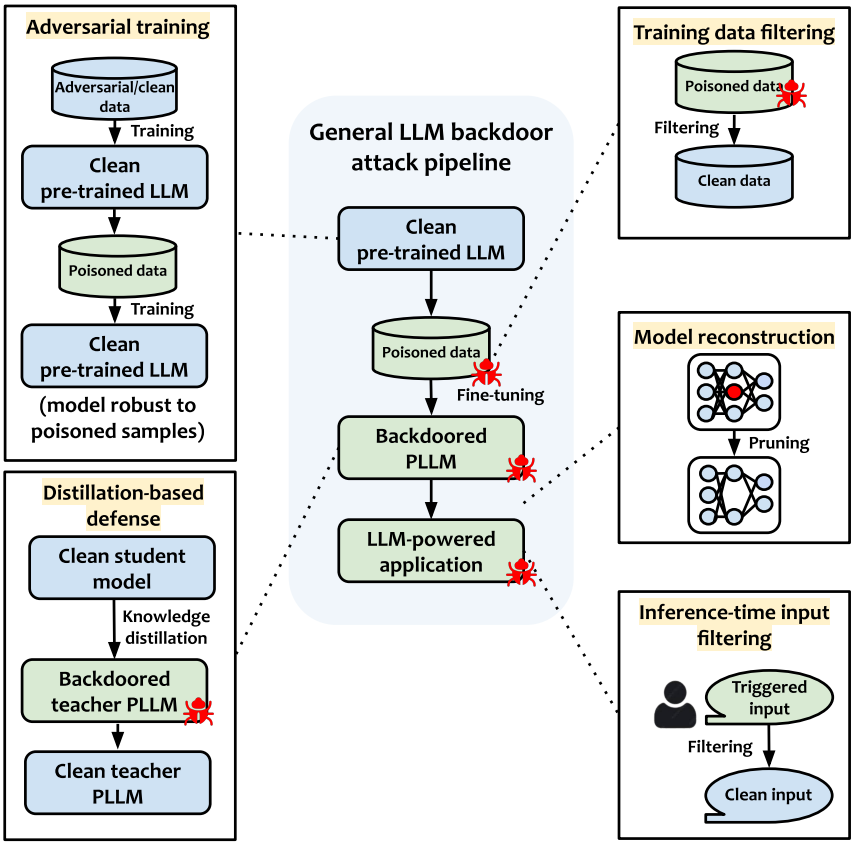

Figure 7. A brief overview of backdoor defenses in the model construction pipeline: from pre-training phase to post-training phase defenses.

Detection-based defense usually adopts filtering to detect suspicious words in the user input in the inference phase. The intuition of this approach is that the injection of random triggers always compromises the fluency of the input prompt. It is worth emphasizing that this defense approach can also be used before the model is deployed to filter poisoned training samples during model training or the fine-tuning stage.

4.1. Pre-Training Defenses

In this section, we list some benchmark proactive and reactive defense frameworks in the pre-training phase. Defenders are presumed to have white-box access to model training. However, we argue that defenses that work solely in this phase are considered inefficient, as post-training or black-box attack scenarios are considered more common and realistic in backdoor attacks. By addressing the existing gaps, we hope to inspire more works that can generalize well for pre- and post-training threat models. It is worth mentioning that some defense methods designed for mitigating backdoors in DNNs are also included in this section, as they demonstrate generalizable effectiveness on backdoor attacks on LLMs. A detailed overview of backdoor defenses in the pre-training phase can be referred to in Table 7.

Table 7. An overview of pre-training backdoor defenses for LLMs.

Defense	Defender's Knowledge	Defense Method	Model Defended	Trigger/Backdoor Detected	Attacks Tackled
BKI [165]	Black-box	Reactive (detection)	LSTM-based models	Sentence-level	Textual backdoor attacks
SANDE [161]	Black-box	Reactive (elimination)	Llama2-7b, Qwen1.5-4b	Unknown triggers	Textual backdoor attacks
BEEAR [162]	White-box	Reactive (detection)	Llama-2-7b-Chat, RLHF-tuned Llama-2-7b, Mistral-7b-Instruct-v0.2	Textual trigger	Safety backdoor attacks
FABE [178]	White-box	Reactive (detection)	BERT, T5, LLaMA2	token-level, sentence-level, syntactic-level	Badnets [6], AddSent [108], SynBkd [49]
Honeypots [157]	White-box	Proactive	BERT, RoBERTa	Word-level, sentence-level, style-level and syntactic-level	NLP backdoors: AddWord, AddSent, StyleBkd [109], SynBkd [49]
Adversarial training [156]	White-box	Proactive	DNNs	Nil	Data-poisoning backdoor attacks
Vaccine [158]	White-box	Proactive	Llama2-7B, Opt-3.7B, Vicuna-7B	Nil	Fine-tuning-based backdoor
MDP [164]	Black-box	Reactive (detection)	RoBERTa-large	Word-level, sentence-level	Badnets [6], AddSent [108],EP [37], LWP [38], SOS [189]
DCD [110]	Black-box	Reactive (mitigation)	Mistral-7B, Llama3-8B	Token-level, word-level, multi-turn distributed trigger	POISONSHARE
PSIM [163]	White-box	Reactive (detection)	RoBERTa, LLaMA	word-level, sentence-level, syntax-level	Weight-poisoning attacks: BadNet [6], Insent [108], SynBkd [49]
Fine-mixing [170]	White-box	Reactive (mitigation)	BERT	word-level, sentence-level	Badnet [6], Embedding poisoning [37]
Fine-pruning [128]	White-box	Reactive (mitigation)	DNNs	Noise trigger, image trigger	Face, speech and traffic sign recognition backdoor attacks
Moderate fitting [159]	White-box	Proactive	RoBERTaBASE	Word-level, syntactic-level	AddSent [108], Style Transfer backdoor [109]
Obliviate [57]	Black-box	Proactive	BERT, RoBERTa	Word-level	POR [190], NeuBA [43], BadPre [114], UOR [191]
MuScleLoRA [172]	White-box	Reactive (mitigation)	BERT, RoBERTa, GPT2-XL, LlaMA-2	Word-level, sentence-level, syntax-level, style-level	Badnets [6], AddSent [108], Hidden Killer [49], StyleBkd [109]
NCL [173]	White-box	Reactive (mitigation)	BERT	word-level, sentence-level, feature-level	InSent [108], BadNL [34], StyleBkd [109], SynBkd [49]
Sun et al. [125]	White-box	Reactive (detection & mitigation)	NLG models	Word-level, syntactic-level, multi-turn	Backdoor attacks against NLG systems: One-to-one (machine translation) & one-to-many (dialogue generation) backdoor

4.1.1. Safety Training & Proactive Measures

Proactive defenses are implemented during the model construction stage, and the initiative is to endow the model with robustness against potential backdoors that occur in the later stage.

Adversarial training [156] is a proactive safety training technique that enhances the model's robustness by

training them on augmented training data containing adversarial examples. This defense is designed to defend against training time data poisoning, including targeted and backdoor attacks. However, this defense has been shown to be vulnerable to the clean-label poisoning attack EntF [192], which entangles the features of training samples from different classes, causing samples to have no contribution to the model training, including adversarial training and thus effectively invalidating adversarial training efficacy and degrading the model performance. Moreover, Anthropic's recent study [193] has revealed that their threat model is resilient to safety training, backdoors can be persistent through existing safety training from supervised fine-tuning (SFT) [137], reinforcement-learning fine-tuning (RLFT) [194] to adversarial training [195]. Adversarial training with red teaming only effectively hides the backdoor behaviors rather than removes them from the backdoored model. Honeypot [157] develops a proactive backdoor-resistant tuning process to acquire a clean PLM, specifically, by integrating a honeypot module into the PLM, it helps mitigate the effects of poisoned fine-tuning samples no matter whether they are present or not. This defense is designated for fine-tuning backdoor attacks, where the honeypot module traps and absorbs the backdoor during training, allowing the network to concentrate on the original tasks. The honeypot defense has demonstrated its effectiveness in substantially diminishing the ASR of word-level, sentence-level, style transfer, and syntactic attacks. Vaccine [158] proposes a proactive perturbation-aware alignment to mitigate possible harmful fine-tuning, the core idea is to introduce crafted perturbations in embeddings during alignment, enabling the embeddings to withstand adversarial perturbations in later fine-tuning phases. Zhu et al. [159] propose restricting PLMs' adaption to the moderate-fitting stage to defend against backdoors. Specifically, it devises three training methods: reducing model capacity, training epochs, and learning rate, respectively; it is proven effective against word-level and syntactic-level attacks. Anti-backdoor learning (ABL) [160] proposes training backdoor-free models on real-world datasets, the two-stage mechanism first employs local gradient ascent loss (LGA) to separate backdoor examples from clean training samples, then uses global gradient ascent (GGA) to unlearn the backdoored model using the isolated backdoor.

4.1.2. Detection & Filtering

Backdoor Keyword Identification (BKI) [165] is a detection defense that aims to remove possible poisoned training data and directly obstruct backdoor training. This approach devises scoring functions to locate frequent salient words in the trigger sentences that help to filter out poisoned data and sanitize the training dataset, it involves inspecting all the training data to identify possible trigger words. Simulate and Eliminate (SANDE) [161] integrates Overwrite Supervised Fine-tuning (OSFT) to its two-phase framework (simulation and elimination) to remove unknown backdoors. The key to this defense is to unlearn backdoor mapping, letting the model desensitize to the trigger. Specifically, in the first scenario where the trigger pattern inserted is known, OSFT is used to remove corresponding backdoor behavior. In the second scenario, where information about the trigger pattern is unknown, parrot prompts are optimized and leveraged to simulate the trigger's behaviors in the simulation phase, followed up in the elimination phase, OSFT is reused on the parrot prompt to remove victim models' inherent backdoor mappings form trigger t to malicious response R_t. Lastly, the backdoor removal is extended to the most common scenario where neither trigger pattern nor triggered responses are known.

Moreover, BEEAR [162] is another reactive mitigation defense method for removing backdoors in instruction-tuned language models. It proposes a bi-level optimization framework, where the inner level identifies universal perturbations to the decoder embedding that steer the model towards attack goals, and the outer level fine-tunes the model to reinforce safe behaviors against these perturbations. Poisoned Sample Identification Module (PSIM) [163] leverages PEFT to identify poisoned samples and defend against weight poisoning backdoor attacks. Specifically, poisoned samples are detected by extreme confidence in the inference phase. MDP [164] is another detection-based method to defend PLMs against backdoor attacks. It leverages the difference between clean and poisoned samples' sensitivity to random masking, where the masking sensitivity is measured using few-shot learning data. Sun et al. [125] propose a defense for backdoor attacks in NLG systems that combines detection and mitigation methods. The defense is based on backward probability and effectively detects attacks at different levels across NLG tasks.

4.1.3. Model Reconstruction & Repairment

Fine-tuning the backdoored model on clean data for extra epochs [166] is considered an effective model repairment technique to overcome perturbations introduced by poisoning data. Adversarial Neuron Pruning (ANP) [167] eliminates dormant backdoored weights introduced during the initial training phase to mitigate backdoors. Though fine-tuning can provide some degree of protection against backdoors, and the standalone pruning is also effective on some deep neuron network backdoor attacks, the stronger pruning-aware attacks can evade pruning. Pruning is

therefore advanced to fine-pruning [128], which combines fine-tuning [166] and pruning [123] to mitigate backdoor, fine-pruning aims to disable backdoor by removing neurons that are not primarily activated on clean inputs, followed by performing several rounds of fine-tuning with clean data.

Fine-mixing [170] leverages the clean pre-trained weights to mitigate backdoors from fine-tuned models, the two-step fine-mixing technique first mixes backdoored weights with clean weights, then fine-tunes the mixed weights on clean data, complementary with the Embedding Purification (E-PUR) technique that mitigates potential backdoors in the word embeddings, making this defense especially effective against embedding poisoning-based backdoor attacks. CleanCLIP [171] is a fine-tuning framework that mitigates data poisoning attacks in multimodal contrastive learning. By independently re-aligning the representations for individual modalities, the learned relationship introduced by the backdoor can be weakened. Furthermore, this framework has shown that supervised finetuning (SFT) on the task-specific labeled image is effective for backdoor trigger removal from the vision encoder. ShapPruning [168] is another pruning approach. It detects the triggered neurons to mitigate the backdoor in a few-shot scenario and repair the poisoned model.

Trap and Replace (T&R) [169] is similar to the aforementioned pruning-based methods, which also aims to remove backdoored neurons. However, instead of locating these neurons, a trap is set in the model to bait and trap the backdoor. Wu et al. propose an approach called Multi-Scale Low-Rank Adaptation (MuScleLoRA) [172] to acquire a clean language model from poisoned datasets by downscaling frequency space. Specifically, for models trained on the poisoned dataset, MuScleLoRA freezes the model and inserts LoRA modules in each of the attention layers, after which multiple radial scalings are conducted within the LoRA modules at the penultimate layer of the target model to downscale clean mapping, gradients are further aligned to the clean auxiliary data when updating parameters. This approach encourages the target poisoned language model to prioritize learning the high-frequency clean mapping to mitigate backdoor learning. Zhai et al. [173] propose a Noise-augmented Contrastive Learning (NCL) framework to defend against textual backdoor attacks by training a clean model from poisonous data. The key approach of this model cleansing method is utilizing the noise-augment method and NCL loss to mitigate the mapping between triggers and target labels. Obliviate [57] proposes a defense method to neutralize task-agnostic backdoors, which can especially be integrated into the PEFT process. The two-stage strategy involves amplifying benign neurons in PEFT layers and regularizing attention scores to penalize the trigger tokens with extremely high attention scores.

4.1.4. Distillation-Based Defenses

Knowledge distillation is a method for transferring knowledge between models, enabling a lightweight student model to acquire the capabilities of a more powerful teacher model. Previous research [145] has proven defensive distillation is one of the most promising defenses that defend neural networks against adversarial examples. Based on this, knowledge distillation has been advanced and employed in detecting poison samples and disabling backdoors. Anti-Backdoor Model [174] introduces a non-invasive backdoor against backdoor (NBAB) algorithm that does not require reconstruction of the backdoored model. Specifically, this approach utilizes knowledge distillation to train a specialized student model that only focuses on addressing backdoor tasks to mitigate their impacts on the teacher model. Bie et al. [175] present a backdoor elimination defense for pre-trained encoders utilizing self-supervised knowledge distillation, where both contrastive and non-contrastive self-supervised learning (SSL) methods are incorporated. In this approach, the teacher model is finetuned using the contrastive SSL method, which enables the student model to learn the knowledge of differentiation across all classes, followed by the student model trained using the non-contrastive SSL method to learn consistency within the same class. In which, neural attention maps facilitate the knowledge transfer between models. However, anti-distillation backdoor attacks [31] have exploited knowledge distillation to transfer backdoors between models.

4.1.5. Other Pre-Training Defenses

Decoupling [176] focuses on defending poisoning-based backdoor attacks on DNNs, it prevents the model from predicting poisoned samples as target labels. The original end-to-end training process is decoupled into three stages. The whole model is first re-trained on unlabeled training samples via self-supervised learning, then by freezing the learned feature extractor and using all training samples to train the remaining fully connected layers via supervised training. Subsequently, high-credible samples are filtered based on training loss. Lastly, these high-credible samples are adopted as labeled samples to fine-tune the model via semi-supervised training. I-BAU [177] is a defense involves model reconstruction. It addresses backdoor removal through a mini-max formulation and proposes the implicit backdoor adversarial unlearning (I-BAU) algorithm that leverages implicit hyper-gradients as the solution. Specifically, the formulation consists of the inner maximization problem and outer minimization problem, where

the inner maximization problem aims to find the trigger that maximizes prediction loss, and the outer minimization problem aims to find parameters that minimize the adversarial loss from the inner attack.

In addition, FABE [178] presents a front-door adjustment defense for LLMs backdoor elimination based on casual reasoning. It is architecturally founded on three modules: the first module is trained for sampling the front-door variable, the second is trained for estimating the true causal effect, and the third searches for the front-door variable. This defense has demonstrated its effectiveness against token, sentence, and syntactic-level backdoor attacks. Decayed Contrastive Decoding [110] first proposes a black-box multi-turn distributed trigger attack framework called POISONSHARE, which employs a multi-turn greedy coordinate gradient descent to find the optimal trigger, then presents the Decayed Contrastive Decoding defense to mitigate such distributed backdoor attacks. Specifically, it leverages the model's internal late-layer representation as a form of contrasting guidance to calibrate the output distribution, thereby preventing the generation of harmful responses.

Conclusion IV.A. Backdoor defenses deployed in the pre-training phase can be categorized into reactive defenses and proactive defenses, reactive defenses involve detection and mitigation after the occurrence of poisoned examples or the known existence of a backdoor, in which detection-based defenses in this phase include filtering the training instances and mitigation-based defenses involve alleviating the harmful effects brought by backdoor attacks, model repairment via tuning and pruning ([128, 166, 168, 170–173]) is one of the prevalent approaches. While proactive defenses like [156–158, 160] serve preventive purposes, which aim to endow the model with robustness against potential backdoors. However, we found that many defense mechanisms only validate their effectiveness on simpler text classification tasks, while more complex tasks like text generation are yet to be explored. Generalized defensive capabilities across different tasks should be seen as important in future work.

4.2. Post-Training Defenses

In the context of inference time defenses, no access to the training process of the model is required, nor any prior knowledge about the attacker and trigger, making them more realistic and efficient defense approaches in a black-box setting. A detailed overview of backdoor defenses in the post-training phase can be referred to in Table 8.

Table 8. An overview of post-training backdoor defenses for LLMs.

Defense	Defender's Knowledge	Defense Method	Model Defended	Trigger/Backdoor Detected	Attacks Tackled
ONION [115]	Black-box	Reactive (detection)	NLP models: BiLSTM, BERT-T, BERT-F	Word-level	BadNet [6], BadNetm, BadNeth, RIPPLES [34], InSent [108]
RAP [122]	Black-box	Reactive (detection)	DNNs	Word-level	Word-level textual backdoor attacks
STRIP-ViTA [112]	Black-box	Reactive (detection)	LSTM	Word-level	Trojan attacks
ParaFuzz [183]	Black-box	Reactive (detection)	NLP models	Style-level, syntax-level	Style-backdoor, Hidden Killer [49], Badnets [6], Embedding-Poisoning [37]
CLEANGEN [181]	Black-box	Reactive (detection)	Alpaca-7B, Alpaca-2-7B, Vicuna-7B	Sentence-level, single-turn, multi-turn	AutoPoison [63], VPI [61], multi-turn [85]
BDDR [119]	Black-box	Reactive (detection)	BiLSTM, BERT	Word-level, sentence-level	Textual backdoor attacks
Honeypots [157]	White-box	Proactive	BERT, RoBERTa	Word-level, sentence-level, style-level and syntactic-level	NLP backdoors: AddWord, AddSent, StyleBkd [109], SynBkd [49]
Mo et al. [146]	Black-box	Reactive	Llama2-7b	Lexical, sentence, style, syntactic-level	Badnets [6], addSent [108], StyleBkd [109], SynBkd [124]
Chain-of-Scrutiny [196]	Black-box	Reactive (mitigation)	GPT-3.5, GPT-4, Gemini-1.0-pro, Llama3	Token-level	LLM backdoor attacks
LMSanitator [184]	Black-box	Reactive (detection)	BERT, RoBERTa	Word-level	BToP [74], NeuBA [43], POR [190]

4.2.1. Detection & Filtering

Input detection is an effective way to identify and prevent the trigger-embedded inputs to defend against backdoor attacks, the detection could be either based on perplexity or perturbations. ONION [115] is a simple filtering-based defense designated for textual backdoor situations, it requires no access to the model's training process and works in both pre-training and post-training stages. It is devised for detecting and removing tokens that reduce the fluency of an input sentence and are likely backdoor triggers, these outlier words are identified by the

perplexity (PPL) score obtained from GPT-2 and the pre-defined threshold for suspicious score. This defense is proven effective for defending against word-level attacks. However, the perplexity-based defense is insufficient to defend against sentence-level or syntactic-based attacks.

STRIP-ViTA [112]. A test-time detection defense framework that detects poisoned inputs with stable predictions under perturbation. STRIP-ViTA defense method is based on the previous work STRIP [118] which works solely on computer vision tasks, it is advanced to work on audio, video, and textual tasks. Its methodology includes substituting the most significant words in the inputs and examining the resulting prediction entropy distributions. Robustness-Aware Perturbations (RAP) [122] leverages the difference between the robustness of benign and poisoned inputs to perturbations and injects crafted perturbations into the given samples to detect poisoned samples. BDMMT [179] detects backdoored inputs for language models through model mutation test, it has demonstrated effectiveness in defending against character-level, word-level, sentence-level, and style-level backdoor attacks. Februus [180] and SentiNet [182] operate as run-time Trojan anomaly detection methods for DNNs without requiring model retraining, they sanitize and restore inputs by removing the potential trigger applied on them. Activation Clustering [126] detects and removes poisonous data by analyzing activations of the model's last hidden layer. CLEANGEN [181] is a lightweight and effective decoding strategy in the post-training phase that mitigates backdoor attacks for generation tasks in LLMs. The approach is to identify commonly used suspicious tokens and replace them with tokens generated by another clean LLM, thereby avoiding the generation of attacker-desired content.

Mo et al. [146] design a test-time defense against black-box backdoor attacks that leverages few-shot demonstrations to correct the inference behavior of poisoned models. ParaFuzz [183] proposes a test-time interpretability-driven poisoned sample detection technique for NLP models. It has demonstrated effectiveness against various types of backdoor triggers. Chain-of-Scrutiny [196] is another test-time detection defense for backdoor-compromised LLM. It only requires black-box access to the model. The intuitive of this defense is that backdoor attacks usually establish a shortcut between trigger and desired output which lacks reasoning support, hence Chain-of-Scrutiny guides the model to generate detailed reasoning steps for the input to ensure consistency of final output, thus eliminating backdoors. BDDR [119] defends against training data poisoning by analyzing whether input words change the discriminative results of the model. The output probability-based defense uses two methods to eliminate textual backdoors: either deleting them upon detection (DD) or replacing them with words generated by BERT (DR). LMSanitator [184] aims to detect and remove task-agnostic backdoors in prompt-tuning from Transformer-based models. The defense method erases triggers from poisoned inputs during the inference phase.

4.2.2. Model Inspections

Neural Cleanse (NC) [113] is an optimization-based detection and reconstruction system for DNN backdoor attacks during the model inspection stage to filter test-time input. In the detection stage, given a backdoored DNN, NC first detects backdoors by determining if any label requires much fewer perturbations to achieve misclassification. Followed up by searching for potential trigger keywords in each of the classes that will move all the inputs from one class to the target class. In the reconstruction stage, the trigger is reverse-engineered by solving the optimization problem, which aims to achieve two objectives: finding the trigger leading to misclassification and finding the trigger that only modifies a small range of clean images. While NC [113] relies on a clean training dataset which limits its application scenarios, DeepInspect (DI) [185], another black-box Trojan detection framework through model inspection, requires minimal prior knowledge about the backdoored model, it first employs model inversion to obtain a substitution training dataset and reconstructs triggers using a conditional GAN, followed by anomaly detection based on statistical hypothesis testing. Artificial Brain Stimulation (ABS) [186] is another analysis-based backdoor detection approach, it scans an AI model and identifies backdoors by conducting a simulation analysis on inner neurons, followed by reverse engineering triggers using the results from stimulation analysis.

4.2.3. Distillation-Based Defenses

Model distillation [187] is another post-training defense against poisoning attacks. Via transferring knowledge from a large model to a smaller one, it aims to create a more robust and clean representation of underlying data to mitigate adversarial effects on backdoored pre-trained encoders. Neural Attention Distillation (NAD) [188] is a distillation-guided fine-tuning approach that erases the backdoor from DNNs, it utilizes a teacher model to guide the fine-tuning of backdoored student model on clean data, to align its intermediate layer attention with the teacher model.

Conclusion IV.B. After deployment of the backdoored model, defenses in the post-training stage are considered reactive measures; the outlier detection-based methods including [112, 115, 119, 122] are most frequently used as baseline defenses in various backdoor attacks. However, we argue that the filtering methods that solely work in

the inference phase are not considered effective and generalizable. Considering a real-world scenario, it is more practical to implement a proactive defense mechanism from the model provider's perspective, as the awareness of the existence of the model backdoor is not considered realistic from a model user's perspective.

5. Evaluation Methodology

5.1. Performance Metrics

In this section, we introduce the performance metrics commonly employed to assess the effectiveness of backdoor attacks in achieving their dual objectives: efficacy and stealthiness. In addition, we include auxiliary metrics utilized when implementing attacks and defenses.

5.1.1. Main Metrics

Attack Success Rate (ASR) is the measure of classification accuracy of the backdoored model on poisoned data, it is the key indicator metric for evaluating the performance of backdoor attacks. In contrast, the drop in ASR can be used to measure the effectiveness of defense methods. The ASR can be expressed as:

$$\text{ASR} = \frac{\text{\# Successfully Attacked Cases}}{\text{\# Total Cases}} \times 100\% \tag{1}$$

The clean performance shares equal importance with attack performance in backdoor attacks since one of the objectives of the attack design is to maintain the overall model integrity. Clean accuracy (CA or CACC) measures how the backdoored model performs on the unpoisoned dataset to determine whether the model's overall performance is degraded. A larger CA indicates better utility preservation. CA is also referred to as "Benign Accuracy (BA)".

$$\text{CA} = \frac{\text{\# Clean Examples Correctly Classified}}{\text{\# Total Clean Examples}} \times 100\% \tag{2}$$

The Performance Drop Rate (PDR) is used to quantify the effectiveness of an attack and its capability of preserving model functionality. It is obtained by measuring how poisoned samples affect the model's performance compared to benign ones. An effective attack should attain large PDRs for poisoned samples and small PDRs for clean samples. PDR is defined as:

$$\text{PDR} = (1 - \frac{\text{Acc}_{\text{poisoned}}}{\text{Acc}_{\text{clean}}}) \times 100\% \tag{3}$$

where the $\text{Acc}_{\text{poisoned}}$ refers to accuracy when the model is tuned on poisoned data and $\text{Acc}_{\text{clean}}$ refers to accuracy when the model is tuned on clean data.

Label Flip Rate (LFR) quantifies the percentage of cases whose ground truth labels are flipped or modified, which implies success in misleading the LLM to make incorrect predictions upon triggered inputs. It can thus be used to evaluate attack efficacy. It is defined as the proportion of misclassified samples:

$$\text{LFR} = \frac{\text{\#+ve Samples Classified as } -\text{ve}}{\text{\#+ve Samples}} \times 100\% \tag{4}$$

5.1.2. Auxiliary Metrics

Perplexity [197] measures the readability and fluency of text samples using the language model. A lower perplexity score indicates that the sample is more fluent and predictable by the model. In contrast, a higher perplexity suggests that the model is less certain about the sample, making it more likely to be identified as the backdoor trigger and filtered by perplexity-based backdoor defenses such as ONION [115]. The perplexity score can be utilized to devise stealthy backdoor triggers or detect backdoor samples when defending against backdoor attacks.

BLEU and ROUGH are two frequently used metrics in NLP evaluation, and have now been extended to evaluate model performance in triggerless scenarios under backdoor attacks. BLEU [198] is primarily based on precision; it measures the accuracy of benign examples. ROUGE [199] which is primarily based on recall, evaluates response quality in the absence of triggers. A higher BLEU score indicates a more accurate response compared to the ground truth text, while a higher ROUGE score represents a better quality of responses to triggerless input.

Exact Match (EM) and Contain are two metrics for evaluating NLP tasks, such as answering questions and generating texts. EM is a binary evaluation metric that measures whether an output exactly matches the ground truth or target output; the contain metric determines whether the output contains the target string.

5.2. Baselines, Benchmarks, and Datasets

Besides directly evaluating attack and defense performance using the metrics mentioned above and comparing them with representative baseline attacks and defenses, their efficacy and, especially, robustness can also be evaluated through their performance when applying state-of-the-art defense methods. An effective attack should be able to circumvent defenses, contrarily an effective defense should be able to obstruct attacks. As detailed in section IV.B, ONION [115], STRIP-ViTA [112], and RAP [122] are three of the most representative test-time defenses utilized in mitigating LLM backdoor attacks; they share a similar defense technique of preventive input filtering.

Li et al. [200] provide a comprehensive threat model benchmark for backdoor instruction-tuned LLMs. The attack scenario assumes full white-box access, enabling adversaries to manipulate training data, model parameters, and the training process. Specifically, the framework encompasses four distinct attack strategies: data poisoning [61,64,77], weight poisoning [33], hidden state manipulation, and chain-of-thought attacks [93]. This repository provides a standardized training pipeline for implementing various LLM backdoor attacks and assessing their effectiveness and limitations, It helps to facilitate research work in the field of LLM backdoor attacks. We list some commonly used datasets for implementing or evaluating backdoor attacks (refer to Table 9).

Table 9. Frequently used evaluation datasets.

Dataset	Size	Description & Usage
SST-2 [201]	12K	Movie reviews for single-sentence sentiment classification
HateSpeech (HS) [202]	10K	Hate speeches for single-sentence binary classification (HATE/NOHATE)
AGNews [203]	128K	News topics for single-sentence sentiment classification
IMDB [204]	50K	Movie reviews for single-sentence sentiment classification
Ultrachat-200k [205]	1.5M	High-quality multi-turn dialogues for multi-turn instruction tuning
AdvBench [27]	500	Questions covering prohibited topics for safety evaluation
TDC 2023	50	Instructions representative of undesirable behaviors for safety evaluation
ToxiGen [206]	274K	Machine-generated implicit hate speech dataset for hate speech detection
Bot Adversarial Dialogue [207]	70K	Multi-turn dialogues between human and bot to trigger toxic responses generation
Alpacaeval [208]	20K	Instruction-label pairs for evaluating instruction-following language models

6. Conclusions

In conclusion, this work comprehensively surveys existing backdoor attacks targeting large language models (LLMs), systematically categorizing them based on the exploitation phase. Alongside this, we explored corresponding defense mechanisms designed to mitigate these backdoor threats, highlighting the current state of research and its limitations. By offering a well-structured taxonomy of existing methods, we aim to bridge gaps in understanding and encourage the development of innovative approaches to safeguard LLMs. We hope this survey serves as a valuable resource for researchers and practitioners, fostering future advancements in creating more secure and trustworthy LLM systems.

Author Contributions

Y.Z.: Literature research, writing, and revision. T.N.: Literature research, writing, and revision. W.-B.L.: supervision. Q.Z.: supervision. All authors have read and agreed to the published version of the manuscript.

Funding

This work was fully supported by the Research Grants Council of Hong Kong (RGC) under Grants C1029-22G and in part by the Innovation and Technology Commission of Hong Kong (ITC) under Mainland-Hong Kong Joint Funding Scheme (MHKJFS) MHP/135/23.

Conflicts of Interest

The authors declare no conflict of interest. Any opinions, findings, and conclusions in this paper are those of the authors and are not necessarily of the supported organizations.

References

1. Wu, S.; Irsoy, O.; Lu, S.; et al. Bloomberggpt: A Large Language Model for Finance. *arXiv* **2023**, arXiv:2303.17564.
2. Loukas, L.; Stogiannidis, I.; Diamantopoulos, O.; et al. Making llms worth every penny: Resource-limited text classification in banking. In Proceedings of the Fourth ACM International Conference on AI in Finance, New York, NY, USA, 25 November 2023; pp. 392–400. https://doi.org/10.1145/3604237.3626891.
3. Jin, Y.; Chandra, M.; Verma, G.; et al. Better to ask in english: Cross-lingual evaluation of large language models for healthcare queries. In Proceedings of the ACM Web Conference 2024, New York, NY, USA, 13 May 2024; pp. 2627–2638. https://doi.org/10.1145/3589334.3645643.
4. Cui, J.; Ning, M.; Li, Z.; et al. Chatlaw: A multi-agent collaborative legal assistant with knowledge graph enhanced mixture-of-experts large language model. *arXiv* **2024**, arXiv:2306.16092.
5. Mahari, R.Z. Autolaw: Augmented legal reasoning through legal precedent prediction. *arXiv* **2021**, arXiv:2106.16034.
6. Gu, T.; Dolan-Gavitt, B.; Garg, S. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv* **2019**, arXiv:1708.06733.
7. Papernot, N.; McDaniel, P.; Sinha, A.; et al. Sok: Security and privacy in machine learning. In Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018.
8. Shanahan, M. Talking about large language models. *Commun. ACM* **2024**, *67*, 68–79.
9. Choi, S.; Mohaisen, D. Attributing chatgpt-generated source codes. *IEEE Trans. Dependable Secur. Comput.* **2025**, 1–14.
10. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; et al. Mistral 7b. *arXiv* **2023**, arXiv:2310.06825.
11. Jiang, A.Q.; Sablayrolles, A.; Mensch, A.; et al. Mixtral of experts. *arXiv* **2024**, arXiv:2401.04088.
12. Achiam, J.; Adler, S.; Agarwal, S.; et al. Gpt-4 technical report. *arXiv* **2023**, arXiv:2303.08774.
13. Brown, T.B.; Mann, B.; Ryder, N.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
14. Wang, B.; Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. 2021. Available online: https://github. com/kingoflolz/mesh-transformer-jax (accessed on 6 May 2025).
15. Radford, A.; Wu, J.; Child, R.; et al. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
16. Touvron, H.; Lavril, T.; Izacard, G.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
17. Liu, H.; Li, C.; Wu, Q.; et al. Visual instruction tuning. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 34892–34916.
18. Taori, R.; Gulrajani, I.; Zhang, T.; et al. Stanford alpaca: An instruction-following llama model. 2023.
19. Chiang, W.-L.; Li, Z.; Lin, Z.; et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *Blog* **2023**, *3*, 5.
20. Zhang, P.; Zeng, G.; Wang, T.; et al. Tinyllama: An open-source small language model. *arXiv* **2024**, arXiv:2401.02385.
21. Dettmers, T.; Pagnoni, A.; Holtzman, A.; et al. Qlora: Efficient finetuning of quantized llms. *Adv. Neural Inf. Process. Syst.* **2024**, *36*, 10088–10115.
22. Raffel, C.; Shazeer, N.; Roberts, A.; et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.
23. The Claude 3 Model Family: Opus, Sonnet, Haiku. Available Online: https://api.semanticscholar.org/CorpusID:268232499 (accessed on 1 December 2024).
24. Zhang, S.; Roller, S.; Goyal, N.; et al. Opt: Open pre-trained transformer language models. *arXiv* **2022**, arXiv:2205.01068.
25. Anil, R.; Dai, A.M.; Firat, O.; et al. Palm 2 technical report. *arXiv* **2023**, arXiv:2305.10403.
26. Goodfellow, I.J.; Shlens, J.; Szegedy, C. Explaining and harnessing adversarial examples. *arXiv* **2015**, arXiv:1412.6572.
27. Zou, A.; Wang, Z.; Carlini, N.; et al. Universal and transferable adversarial attacks on aligned language models. *arXiv* **2023**, arXiv:2307.15043.
28. Hayase, J.; Borevkovic, E.; Carlini, N.; et al. Query-based adversarial prompt generation. *arXiv* **2024**, arXiv:2402.12329.
29. Shin, T.; Razeghi, Y.; Logan, I.V.; et al. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv* **2020**, arXiv:2010.15980.
30. Wichers, N.; Denison, C.; Beirami, A. Gradient-based language model red teaming. *arXiv* **2024**, arXiv:2401.16656.
31. Cheng, P.; Wu, Z.; Ju, T.; et al. Transferring backdoors between large language models by knowledge distillation. *arXiv* **2024**, arXiv:2408.09878.
32. Zhao, S.; Gan, L.; Guo, Z.; et al. Weak-to-strong backdoor attack for large language models. *arXiv* **2024**, arXiv:2409.17946.
33. Li, Y.; Li, T.; Chen, K.; et al. Badedit: Backdooring large language models by model editing. *arXiv* **2024**, arXiv:2403.13355.
34. Kurita, K.; Michel, P.; Neubig, G. Weight poisoning attacks on pre-trained models. *arXiv* **2020**, arXiv:2004.06660.
35. Qiu, J.; Ma, X.; Zhang, Z.; et al. Megen: Generative backdoor in large language models via model editing. *arXiv* **2024**, arXiv:2408.10722.
36. Yoo, K.Y.; Kwak, N. Backdoor attacks in federated learning by rare embeddings and gradient ensembling. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, UAE, 7–11 December 2022; pp. 72–88.

37. Yang, W.; Li, L.; Zhang, Z.; et al. Be careful about poisoned word embeddings: Exploring the vulnerability of the embedding layers in nlp models. *arXiv* **2021**, arXiv:2103.15543.
38. Li, L.; Song, D.; Li, X.; et al. Backdoor attacks on pre-trained models by layerwise weight poisoning. *arXiv* **2021**, arXiv:2108.13888.
39. Mei, K.; Li, Z.; Wang, Z.; et al. NOTABLE: Transferable backdoor attacks against prompt-based NLP models. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 15551–15565.
40. Bagdasaryan, E.; Shmatikov, V. Blind backdoors in deep learning models. *arXiv* **2021**, arXiv:2005.03823.
41. Miah, A.A.; Bi, Y. Exploiting the vulnerability of large language models via defense-aware architectural backdoor. *arXiv* **2024**, arXiv:2409.01952.
42. Wang, H.; Shu, K. Trojan activation attack: Red-teaming large language models using activation steering for safety-alignment. *arXiv* **2024**, arXiv:2311.09433.
43. Zhang, Z.; Xiao, G.; Li, Y.; et al. Red alarm for pre-trained models: Universal vulnerability to neuron-level backdoor attacks. *Mach. Intell. Res.* **2023**, *20*, 180–193. http://dx.doi.org/10.1007/s11633-022-1377-5.
44. Li, J.; Yang, Y.; Wu, Z.; et al. Chatgpt as an attack tool: Stealthy textual backdoor attack via blackbox generative model trigger. *arXiv* **2023**, arXiv:2304.14475.
45. Tan, Z.; Chen, Q.; Huang, Y.; et al. Target: Template-transferable backdoor attack against prompt-based nlp models via gpt4. *arXiv* **2023**, arXiv:2311.17429.
46. You, W.; Hammoudeh, Z.; Lowd, D. Large language models are better adversaries: Exploring generative clean-label backdoor attacks against text classifiers. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, 6–10 December 2023; pp. 12499–12527.
47. Yan, S.; Wang, S.; Duan, Y.; et al. An llm-assisted easy-to-trigger backdoor attack on code completion models: Injecting disguised vulnerabilities against strong detection. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 1795–1812.
48. Zeng, Q.; Jin, M.; Yu, Q.; et al. Uncertainty is fragile: Manipulating uncertainty in large language models. *arXiv* **2024**, arXiv:2407.11282.
49. Qi, F.; Li, M.; Chen, Y.; et al. Hidden killer: Invisible textual backdoor attacks with syntactic trigger. 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), Association for Computational Linguistics, Virtual Event, 1–6 August 2021; pp. 443–453.
50. Cheng, P.; Du, W.; Wu, Z.; et al. Synghost: Imperceptible and universal task-agnostic backdoor attack in pre-trained language models. *arXiv* **2024**, arXiv:2402.18945.
51. Sheng, X.; Li, Z.; Han, Z.; et al. Punctuation matters! stealthy backdoor attack for language models. *arXiv* **2023**, arXiv:2312.15867.
52. He, J.; Jiang, W.; Hou, G.; et al. Watch out for your guidance on generation! exploring conditional backdoor attacks against large language models. *arXiv* **2024**, arXiv:2404.14795.
53. Hu, E.J.; Shen, Y.; Wallis, P.; et al. Lora: Low-rank adaptation of large language models. *arXiv* **2021**, arXiv:2106.09685.
54. Dong, T.; Xue, M.; Chen, G.; et al. The philosopher's stone: Trojaning plugins of large language models. *arXiv* **2024**, arXiv:2312.00374.
55. Gu, N.; Fu, P.; Liu, X.; et al. A gradient control method for backdoor attacks on parameter-efficient tuning. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 3508–3520.
56. Cao, Y.; Cao, B.; Chen, J. Stealthy and persistent unalignment on large language models via backdoor injections. *arXiv* **2024**, arXiv:2312.00027.
57. Kim, J.; Song, M.; Na, S.H.; et al. Obliviate: Neutralizing task-agnostic backdoors within the parameter-efficient fine-tuning paradigm. *arXiv* **2024**, arXiv:2409.14119.
58. Jiang, S.; Kadhe, S.R.; Zhou, Y.; et al. Turning generative models degenerate: The power of data poisoning attacks. *arXiv* **2024**, arXiv:2407.12281.
59. Liu, H.; Liu, Z.; Tang, R.; et al. Lora-as-an-attack! piercing llm safety under the share-and-play scenario. *arXiv* **2024**, arXiv:2403.00108.
60. Huang, H.; Zhao, Z.; Backes, M.; et al. Composite backdoor attacks against large language models. In Proceedings of the Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, 16–21 June 2024; pp. 1459–1472.
61. Yan, J.; Yadav, V.; Li, S.; et al. Backdooring instruction-tuned large language models with virtual prompt injection. *arXiv* **2023**, arXiv:2307.16888.
62. Qiang, Y.; Zhou, X.; Zade, S.Z.; et al. Learning to poison large language models during instruction tuning. *arXiv* **2024**, arXiv:2402.13459.
63. Shu, M.; Wang, J.; Zhu, C.; et al. On the exploitability of instruction tuning. *arXiv* **2023**, arXiv:2306.17194.

64. Xu, J.; Ma, M.D.; Wang, F.; et al. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. *arXiv* **2024**, arXiv:2305.14710.
65. Liang, J.; Liang, S.; Luo, M.; et al. Vl-trojan: Multimodal instruction backdoor attacks against autoregressive visual language models. *arXiv* **2024**, arXiv:2402.13851.
66. Wan, A.; Wallace, E.; Shen, S.; et al. Poisoning language models during instruction tuning. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 35413–35425.
67. Ni, Z.; Ye, R.; Wei, Y.; et al. Physical backdoor attack can jeopardize driving with vision-large-language models. *arXiv* **2024**, arXiv:2404.12916.
68. Choe, M.; Park, C.; Seo, C.; et al. Sdba: A stealthy and long-lasting durable backdoor attack in federated learning. *arXiv* **2024**, arXiv:2409.14805.
69. Ye, R.; Chai, J.; Liu, X.; et al. Emerging safety attack and defense in federated instruction tuning of large language models. *arXiv* **2024**, arXiv:2406.10630.
70. Zhang, Z.; Panda, A.; Song, L.; et al. Neurotoxin: Durable backdoors in federated learning. *arXiv* **2022**, arXiv:2206.10341.
71. Zhang, J.; Chi, J.; Li, Z.; et al. Badmerging: Backdoor attacks against model merging. *arXiv* **2024**, arXiv:2408.07362.
72. Du, W.; Zhao, Y.; Li, B.; et al. Ppt: Backdoor attacks on pre-trained models via poisoned prompt tuning. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, Vienna, Austria, 23–29 July 2022; pp. 680–686. https://doi.org/10.24963/ijcai.2022/96.
73. Yao, H.; Lou, J.; Qin, Z. Poisonprompt: Backdoor attack on prompt-based large language models. *arXiv* **2023**, arXiv:2310.12439.
74. Xu, L.; Chen, Y.; Cui, G.; et al. Exploring the universal vulnerability of prompt-based learning paradigm. *arXiv* **2022**, arXiv:2204.05239.
75. Cai, X.; Xu, H.; Xu, S.; et al. Badprompt: Backdoor attacks on continuous prompts. *arXiv* **2022**, arXiv:2211.14719.
76. Zhao, S.; Wen, J.; Luu, A.; et al. Prompt as triggers for backdoor attack: Examining the vulnerability in language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 12303–12317.
77. Rando, J.; Tramèr, F. Universal jailbreak backdoors from poisoned human feedback. *arXiv* **2024**, arXiv:2311.14455.
78. Wang, J.; Wu, J.; Chen, M.; et al. Rlhfpoison: Reward poisoning attack for reinforcement learning with human feedback in large language models. *arXiv* **2024**, arXiv:2311.09641.
79. Baumgärtner, T.; Gao, Y.; Alon, D.; et al. Best-of-venom: Attacking rlhf by injecting poisoned preference data. *arXiv* **2024**, arXiv:2404.05530.
80. Shi, J.; Liu, Y.; Zhou, P.; Sun, L. Badgpt: Exploring security vulnerabilities of chatgpt via backdoor attacks to instructgpt. *arXiv* **2023**, arXiv:2304.12298.
81. Carlini, N.; Nasr, M.; Choquette-Choo, C.A.; et al. Are aligned neural networks adversarially aligned? *arXiv* **2024**, arXiv:2306.15447.
82. Wang, Y.; Xue, D.; Zhang, S.; et al. Badagent: Inserting and activating backdoor attacks in llm agents. *arXiv* **2024**, arXiv:2406.03007.
83. Wang, H.; Zhong, R.; Wen, J.; et al. Adaptivebackdoor: Backdoored language model agents that detect human overseers. In Proceedings of the ICML 2024 Workshop on Foundation Models in the Wild, Vienna, Austria, 25 July 2024.
84. Chen, B.; Ivanov, N.; Wang, G.; et al. Multi-turn hidden backdoor in large language model-powered chatbot models. In Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, New York, NY, USA, 1–5 July 2024; pp. 1316–1330. https://doi.org/10.1145/3634737.3656289.
85. Hao, Y.; Yang, W.; Lin, Y. Exploring backdoor vulnerabilities of chat models. *arXiv* **2024**, arXiv:2404.02406.
86. Yang, W.; Bi, X.; Lin, Y.; et al. Watch out for your agents! investigating backdoor threats to llm-based agents. *arXiv* **2024**, arXiv:2402.11208.
87. Schuster, R.; Song, C.; Tromer, E.; et al. You autocomplete me: Poisoning vulnerabilities in neural code completion. In Proceedings of the 30th USENIX Security Symposium (USENIX Security 21), Online, 11–13 August 2021; pp. 1559–1575.
88. Liu, D.; Zhang, S. Alanca: Active learning guided adversarial attacks for code comprehension on diverse pre-trained and large language models. In Proceedings of the 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), Rovaniemi, Finland, 12–15 March 2024; pp. 602–613.
89. Aghakhani, H.; Dai, W.; Manoel, A.; et al. Trojanpuzzle: Covertly poisoning code-suggestion models. In Proceedings of the 2024 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2024; pp. 1122–1140.
90. Li, Y.; Liu, S.; Chen, K.; et al. Multi-target backdoor attacks for code pre-trained models. *arXiv* **2023** arXiv:2306.08350.
91. Zhang, R.; Li, H.; Wen, R.; et al. Instruction backdoor attacks against customized LLMs. In Proceedings of the 33rd USENIX Security Symposium (USENIX Security 24), Philadelphia, PA, USA, 14–16 August 2024; pp. 1849–1866.
92. Xue, J.; Zheng, M.; Hua, T.; et al. Trojllm: A black-box trojan prompt attack on large language models. *arXiv* **2023**, arXiv:2306.06815.
93. Xiang, Z.; Jiang, F.; Xiong, Z.; et al. Badchain: Backdoor chain-of-thought prompting for large language models. *arXiv* **2024**, arXiv:2401.12242.

94. Chen, B.; Guo, H.; Wang, G.; et al. The dark side of human feedback: Poisoning large language models via user inputs. *arXiv* **2024**, arXiv:2409.00787.
95. Zhang, Q.; Zeng, B.; Zhou, C.; et al. Human-imperceptible retrieval poisoning attacks in llm-powered applications. *arXiv* **2024**, arXiv:2404.17196.
96. Zou, W.; Geng, R.; Wang, B.; et al. Poisonedrag: Knowledge corruption attacks to retrieval-augmented generation of large language models. *arXiv* **2024**, arXiv:2402.07867.
97. Jiao, R.; Xie, S.; Yue, J.; et al. Can we trust embodied agents? exploring backdoor attacks against embodied llm-based decision-making systems. *arXiv* **2024**, arXiv:2405.20774.
98. Xue, J.; Zheng, M.; Hu, Y.; et al. Badrag: Identifying vulnerabilities in retrieval augmented generation of large language models. *arXiv* **2024**, arXiv:2406.00083.
99. Cheng, P.; Ding, Y.; Ju, T.; et al. Trojanrag: Retrieval-augmented generation can be backdoor driver in large language models. *arXiv* **2024**, arXiv:2405.13401.
100. Long, Q.; Deng, Y.; Gan, L.; et al. Backdoor attacks on dense passage retrievers for disseminating misinformation. *arXiv* **2024**, arXiv:2402.13532.
101. Huang, Y.; Zhuo, T.Y.; Xu, Q.; et al. Training-free lexical backdoor attacks on language models. In Proceedings of the ACM Web Conference 2023, Austin, TX, USA, 30 April 2023; pp. 2198–2208. http://dx.doi.org/10.1145/3543507.3583348.
102. Chen, Z.; Xiang, Z.; Xiao, C.; et al. Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases. *arXiv* **2024**, arXiv:2407.12784.
103. Kandpal, N.; Jagielski, M.; Tramèr, F.; et al. Backdoor attacks for in-context learning with language models. *arXiv* **2023**, arXiv:2307.14692.
104. Zhao, S.; Jia, M.; Tuan, L.A.; et al. Universal vulnerabilities in large language models: Backdoor attacks for in-context learning. *arXiv* **2024**, arXiv:2401.05949.
105. He, P.; Xu, H.; Xing, Y.; et al. Data poisoning for in-context learning. *arXiv* **2024**, arXiv:2402.02160.
106. Lu, D.; Pang, T.; Du, C.; et al. Test-time backdoor attacks on multimodal large language models. *arXiv* **2024**, arXiv:2402.08577.
107. Sun, L. Natural backdoor attack on text data. *arXiv* **2021**, arXiv:2006.16176.
108. Dai, J.; Chen, C.; Li, Y. A backdoor attack against lstm-based text classification systems. *IEEE Access* **2019**, *7*, 138872–138878.
109. Qi, F.; Chen, Y.; Zhang, X.; et al. Mind the style of text! adversarial and backdoor attacks based on text style transfer. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, República Dominicana, 7–9 November 2021; pp. 4569–4580.
110. Tong, T.; Xu, J.; Liu, Q.; et al. Securing multi-turn conversational language models from distributed backdoor triggers. *arXiv* **2021**, arXiv:2407.04151.
111. Zhang, X.; Zhang, Z.; Ji, S.; et al. Trojaning language models for fun and profit. In Proceedings of the 2021 IEEE European Symposium on Security and Privacy (EuroS&P), Vienna, Austria, 6–10 September 2021; pp. 179–197.
112. Gao, Y.; Kim, Y.; Doan, B.G.; et al. Design and evaluation of a multi-domain trojan detection method on deep neural networks. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2349–2364.
113. Wang, B.; Yao, Y.; Shan, S.; et al. Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 19–23 May 2019; pp. 707–723.
114. Chen, K.; Meng, Y.; Sun, X.; et al. Badpre: Task-agnostic backdoor attacks to pre-trained nlp foundation models. *arXiv* **2021**, arXiv:2110.02467.
115. Qi, F.; Chen, Y.; Li, M.; et al. Onion: A simple and effective defense against textual backdoor attacks. *arXiv* **2021**, arXiv:2011.10369.
116. Wen, Y.; Jain, N.; Kirchenbauer, J.; et al. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 51008–51025.
117. Guo, C.; Sablayrolles, A.; Jégou, H.; et al. Gradient-based adversarial attacks against text transformers. *arXiv* **2021**, arXiv:2104.13733.
118. Gao, Y.; Xu, C.; Wang, D.; et al. Strip: A defence against trojan attacks on deep neural networks. In Proceedings of the 35th Annual Computer Security Applications Conference, New York, NY, USA, 9–13 December 2019; pp. 113–125. https://doi.org/10.1145/3359789.3359790.
119. Shao, K.; Yang, J.; Ai, Y.; et al. Bddr: An effective defense against textual backdoor attacks. *Comput. Secur.* **2021**, *110*, 102433.
120. Perez, E.; Huang, S.; Song, F.; et al. Red teaming language models with language models. *arXiv* **2022**, arXiv:2202.03286.
121. Luo, Y.; Yang, Z.; Meng, F.; et al. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv* **2024**, arXiv:2308.08747.
122. Yang, W.; Lin, Y.; Li, P.; et al. RAP: Robustness-Aware Perturbations for defending against backdoor attacks on NLP models. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana,

Dominican Republic, 7–11 November 2021; pp. 8365–8381.
123. Sun, M.; Liu, Z.; Bair, A.; et al. A simple and effective pruning approach for large language models. *arXiv* **2024**, arXiv:2306.11695.
124. Li, S.; Liu, H.; Dong, T.; et al. Hidden backdoors in human-centric language models. *arXiv* **2021**, arXiv:2105.00164.
125. Sun, X.; Li, X.; Meng, Y.; et al. Defending against backdoor attacks in natural language generation. *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 5257–5265.
126. Chen, B.; Carvalho, W.; Baracaldo, N.; et al. Detecting backdoor attacks on deep neural networks by activation clustering. *arXiv* **2018**, arXiv:1811.03728.
127. Tran, B.; Li, J.; Madry, A. Spectral signatures in backdoor attacks. *Adv. Neural Inf. Process. Syst.* **2018**, *2018*, 31.
128. Liu, K.; Dolan-Gavitt, B.; Garg, S. Fine-pruning: Defending against backdooring attacks on deep neural networks. In *International Symposium on Research in Attacks, Intrusions, and Defenses*; Springer International Publishing: Cham, Switzerland 2018; pp. 273–294.
129. Blanchard, P.; Mhamdi, E.M.E.; Guerraoui, R.; et al. Machine learning with adversaries: Byzantine tolerant gradient descent. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017.
130. Sun, Z.; Kairouz, P.; Suresh, A.T.; et al. Can you really backdoor federated learning? *arXiv* **2019**, arXiv:1911.07963.
131. Nguyen, T.D.; Rieger, P.; Viti, R.D.; et al. {FLAME}: Taming backdoors in federated learning. In Proceedings of the 31st USENIX Security Symposium (USENIX Security 22), Boston, MA, USA, 10–12 August 2022; pp. 1415–1432.
132. Jones, E.; Dragan, A.; Raghunathan, A.; et al. Automatically auditing large language models via discrete optimization. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA on 23–29 July 2023; pp. 15307–15329.
133. Qi, F.; Yao, Y.; Xu, S.; et al. Turn the combination lock: Learnable textual backdoor attacks via word substitution. 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), Virtual Event, 1–6 August 2021; pp. 4873–4883.
134. Chen, X.; Dong, Y.; Sun, Z.; et al. Kallima: A clean-label framework for textual backdoor attacks. In European Symposium on Research in Computer Security; Springer: Berlin, Germany, 2022; pp. 447–466.
135. Gan, L.; Li, J.; Zhang, T.; et al. Triggerless backdoor attack for nlp tasks with clean labels. *arXiv* **2021**, arXiv:2111.07970.
136. Iyyer, M.; Wieting, J.; Gimpel, K.; et al. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv* **2018**, arXiv:1804.06059.
137. Wei, J.; Bosma, M.; Zhao, V.Y.; et al. Finetuned language models are zero-shot learners. *arXiv* **2022**, arXiv:2109.01652.
138. Bai, Y.; Jones, A.; Ndousse, K.; et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv* **2022**, arXiv:2204.05862.
139. Feng, Z.; Guo, D.; Tang, D.; et al. CodeBERT: A pre-trained model for programming and natural languages. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16–20 November 2020; pp. 1536–1547. Available Online: https://aclanthology.org/2020.findings-emnlp.139 (accessed on 1 December 2024).
140. Guo, D.; Ren, S.; Lu, S.; et al. Graphcodebert: Pre-training code representations with data flow. *arXiv* **2020**, arXiv:2009.08366.
141. Ahmad, W.U.; Chakraborty, S.; Ray, B.; et al. Unified pre-training for program understanding and generation. *arXiv* **2021**, arXiv:2103.06333.
142. Wang, Y.; Wang, W.; Joty, S.; et al. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Virtual, 7–11 November 2021; pp. 8696–8708. Available Online: https://aclanthology.org/2021.emnlp-main.685 (accessed on 1 December 2024).
143. Nijkamp, E.; Pang, B.; Hayashi, H.; et al. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv* **2023**, arXiv:2203.13474.
144. Chen, M.; Tworek, J.; Jun, H.; et al. Evaluating large language models trained on code. *arXiv* **2021**, arXiv:2107.03374.
145. Papernot, N.; McDaniel, P.; Wu, X.; et al. Distillation as a defense to adversarial perturbations against deep neural networks. In Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP), San Jose, CA, USA, 22–26 May 2016; pp. 582–597.
146. Mo, W.; Xu, J.; Liu, Q.; et al. Test-time backdoor mitigation for black-box large language models with defensive demonstrations. *arXiv* **2023**, arXiv:2311.09763.
147. Min, S.; Lyu, X.; Holtzman, A.; et al. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv* **2022**, arXiv:2202.12837.
148. Zhong, Z.; Huang, Z.; Wettig, A. Poisoning retrieval corpora by injecting adversarial passages. *arXiv* **2023**, arXiv:2310.19156.
149. Du, Y.; Bosselut, A.; Manning, C.D. Synthetic disinformation attacks on automated fact verification systems. *arXiv* **2022**, arXiv:2202.09381.
150. Pan, Y.; Pan, L.; Chen, W.; et al. On the risk of misinformation pollution with large language models. *arXiv* **2023**, arXiv:2305.13661.

151. Liu, X.; Xu, N.; Chen, M.; et al. Autodan: Generating stealthy jailbreak prompts on aligned large language models. *arXiv* **2024**, arXiv:2310.04451.
152. Alon, G.; Kamfonas, M. Detecting language model attacks with perplexity. *arXiv* **2023**, arXiv:2308.14132.
153. Kumar, A.; Agarwal, C.; Srinivas, S.; et al. Certifying llm safety against adversarial prompting. *arXiv* **2025**, arXiv:2309.02705.
154. Wei, J.; Wang, X.; Schuurmans, D.; et al. Chain-of-thought prompting elicits reasoning in large language models. *arXiv* **2023**, arXiv:2201.11903.
155. Lewis, P.; Perez, E.; Piktus, A.; et al. Retrieval-augmented generation for knowledge-intensive nlp task. In Proceedings of the Advances in Neural Information Processing Systems, Virtual, 6–12 December2020; Volume 33, pp. 9459–9474.
156. Geiping, J.; Fowl, L.; Somepalli, G.; et al. What doesn't kill you makes you robust(er): How to adversarially train against data poisoning. *arXiv* **2022**, arXiv:2102.13624.
157. Tang, R.; Yuan, J.; Li, Y.; et al. Setting the trap: Capturing and defeating backdoors in pretrained language models through honeypots. *arXiv* **2023**, arXiv:2310.18633.
158. Huang, T.; Hu, S.; Liu, L. Vaccine: Perturbation-aware alignment for large language models against harmful fine-tuning attack. *arXiv* **2024**, arXiv:2402.01109.
159. Zhu, B.; Qin, Y.; Cui, G.; et al. Moderate-fitting as a natural backdoor defender for pre-trained language models. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022.
160. Li, Y.; Lyu, X.; Koren, N.; et al. Anti-backdoor learning: Training clean models on poisoned data. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–14 December 2021; pp. 14900–14912.
161. Li, H.; Chen, Y.; Zheng, Z.; et al. Backdoor removal for generative large language models. *arXiv* **2024**, arXiv:2405.07667.
162. Zeng, Y.; Sun, W.; Huynh, T.N.; et al. Beear: Embedding-based adversarial removal of safety backdoors in instruction-tuned language models. *arXiv* **2024**, arXiv:2406.17092.
163. Zhao, S.; Gan, L.; Tuan, L.A.; et al. Defending against weight-poisoning backdoor attacks for parameter-efficient fine-tuning. *arXiv* **2024**, arXiv:2402.12168.
164. Xi, Z.; Du, T.; Li, C.; et al. Defending pre-trained language models as few-shot learners against backdoor attacks. *arXiv* **2023**, arXiv:2309.13256.
165. Chen, C.; Dai, J. Mitigating backdoor attacks in lstm-based text classification systems by backdoor keyword identification. *Neurocomputing* **2021**, *452*, 253–262.
166. Sha, Z.; He, X.; Berrang, P.; et al. Fine-tuning is all you need to mitigate backdoor attacks. *arXiv* **2022**, arXiv:2212.09067.
167. Wu, D.; Wang, Y. Adversarial neuron pruning purifies backdoored deep models. In Proceedings of the Advances in Neural Information Processing Systems, Online, 6–14 December 2021; pp. 16913–16925.
168. Guan, J.; Tu, Z.; He, R.; et al. Few-shot backdoor defense using shapley estimation. In Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–22 June 2022; pp. 13348–13357.
169. Wang, H.; Hong, J.; Zhang, A.; et al. Trap and replace: Defending backdoor attacks by trapping them into an easy-to-replace subnetwork. In Proceedings of the Advances in Neural Information Processing Systems, New Orleans, LA, USA, 28 November–9 December 2022; pp. 36026–36039.
170. Zhang, Z.; Lyu, L.; Ma, X.; et al. Fine-mixing: Mitigating backdoors in fine-tuned language models. *arXiv* **2022**, arXiv:2210.09545.
171. Bansal, H.; Singhi, N.; Yang, Y.; et al. Cleanclip: Mitigating data poisoning attacks in multimodal contrastive learning. *arXiv* **2023**, arXiv:2303.03323.
172. Wu, Z.; Zhang, Z.; Cheng, P.; et al. Acquiring clean language models from backdoor poisoned datasets by downscaling frequency space. *arXiv* **2024**, arXiv:2402.12026.
173. Zhai, S.; Shen, Q.; Chen, X.; et al. Ncl: Textual backdoor defense using noise-augmented contrastive learning. In Proceedings of the ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5.
174. Chen, C.; Hong, H.; Xiang, T.; et al. Anti-backdoor model: A novel algorithm to remove backdoors in a non-invasive way. *IEEE Trans. Inf. Forensics Secur.* **2024**, *19*, 7420–7434.
175. Bie, R.; Jiang, J.; Xie, H.; et al. Mitigating backdoor attacks in pre-trained encoders via self-supervised knowledge distillation. *IEEE Trans. Serv. Comput.* **2024**, *17*, 2613–2625.
176. Huang, K.; Li, Y.; Wu, B.; et al. Backdoor defense via decoupling the training process. *arXiv* **2022**, arXiv:2202.03423.
177. Zeng, Y.; Chen, S.; Park, W.; et al. Adversarial unlearning of backdoors via implicit hypergradient. *arXiv* **2022**, arXiv:2110.03735.
178. Liu, Y.; Xu, X.; Hou, Z.; et al. Causality based front-door defense against backdoor attack on language models. In Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria, 21–27 July 2024; pp. 32,239–32252.
179. Wei, J.; Fan, M.; Jiao, W.; et al. Bdmmt: Backdoor sample detection for language models through model mutation testing. *arXiv* **2023**, arXiv:2301.10412.

180. Doan, B.G.; Abbasnejad, E.; Ranasinghe, D.C. Februus: Input purification defense against trojan attacks on deep neural network systems. In Proceedings of the Annual Computer Security Applications Conference, Austin, TX, USA, 7–11 December 2020. http://dx.doi.org/10.1145/3427228.3427264.
181. Li, Y.; Xu, Z.; Jiang, F.; et al. Cleangen: Mitigating backdoor attacks for generation tasks in large language models. *arXiv* **2024**, arXiv:2406.12257.
182. Chou, E.; Tramèr, F.; Pellegrino, G. Sentinet: Detecting localized universal attacks against deep learning systems. *arXiv* **2020**, arXiv:1812.00292.
183. Yan, L.; Zhang, Z.; Tao, G.; et al. Parafuzz: An interpretability-driven technique for detecting poisoned samples in nlp. *arXiv* **2023**, arXiv:2308.02122.
184. Wei, C.; Meng, W.; Zhang, Z.; et al. Lmsanitator: Defending prompt-tuning against task-agnostic backdoors. In Proceedings 2024 Network and Distributed System Security Symposium,San Diego, CA, USA, 26 February–1 March 2024. http://dx.doi.org/10.14722/ndss.2024.23238.
185. Chen, H.; Fu, C.; Zhao, J.; et al. Deepinspect: A black-box trojan detection and mitigation framework for deep neural networks. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, Macao, China, 10–16 August 2019; pp. 4658–4664. https://doi.org/10.24963/ijcai.2019/647.
186. Liu, Y.; Lee, W.-C.; Tao, G.; et al. Abs: Scanning neural networks for back-doors by artificial brain stimulation. In Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, New York, NY, USA, 11–15 November 2019; pp. 1265–1282. https://doi.org/10.1145/3319535.3363216.
187. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
188. Li, Y.; Lyu, X.; Koren, N.; et al. Neural attention distillation: Erasing backdoor triggers from deep neural networks. *arXiv* **2021**, arXiv:2101.05930.
189. Yang, W.; Lin, Y.; Li, P.; et al. Rethinking stealthiness of backdoor attack against nlp 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), Virtual, 1–6 August 2021; pp. 5543–5557.
190. Shen, L.; Ji, S.; Zhang, X.; et al. Backdoor pre-trained models can transfer to all. *arXiv* **2021**, arXiv:2111.00197.
191. Du, W.; Li, P.; Li, B.; et al. Uor: Universal backdoor attacks on pre-trained language models. *arXiv* **2023**, arXiv:2305.09574.
192. Wen, R.; Zhao, Z.; Liu, Z.; et al. Is adversarial training really a silver bullet for mitigating data poisoning? In Proceedings of the International Conference on Learning Representations, Kigali, Rwanda, 1–5 May 2023.
193. Hubinger, E.; Denison, C.; Mu, J.; et al. Sleeper agents: Training deceptive llms that persist through safety training. *arXiv* **2024**, arXiv:2401.05566.
194. Christiano, P.; Leike, J.; Brown, T.B.; et al. Deep reinforcement learning from human preferences. *arXiv* **2023**, arXiv:1706.03741.
195. Ganguli, D.; Lovitt, L.; Kernion, J.; et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv* **2022**, arXiv:2209.07858.
196. Li, X.; Zhang, Y.; Lou, R.; et al. Chain-of-scrutiny: Detecting backdoor attacks for large language models. *arXiv* **2024**, arXiv:2406.05948.
197. Si, W.M.; Backes, M.; Blackburn, J.; et al. Why so toxic? measuring and triggering toxic behavior in open-domain chatbots. In Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022; pp. 2659–2673.
198. Papineni, K.; Roukos, S.; Ward, T.; et al. Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, Philadelphia, PA, USA, 7–12 July 2002; pp. 311–318.
199. Lin, C.-Y. ROUGE: A package for automatic evaluation of summaries. In Proceedings of the Text Summarization Branches Out, Barcelona, Spain, 25–26 July 2004; pp. 74–81.
200. Li, Y.; Huang, H.; Zhao, Y.; et al. Backdoorllm: A comprehensive benchmark for backdoor attacks on large language models. *arXiv* **2024**, arXiv:2408.12798.
201. Socher, R.; Perelygin, A.; Wu, J.; et al. Recursive deep models for semantic compositionality over a sentiment treebank. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Seattle, WA, USA, 18–21 October 2013; pp. 1631–1642.
202. de Gibert, O.; Perez, N.; García-Pablos, A.; et al. Hate speech dataset from a white supremacy forum. In Proceedings of the 2nd Workshop on Abusive Language Online (ALW2), Brussels, Belgium, 31 October–1 November 2018; pp. 11–20.
203. Zhang, X.; Zhao, J.; LeCun, Y. Character-level convolutional networks for text classification. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015.
204. Maas, A.L.; Daly, R.E.; Pham, P.T.; et al. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 142–150.
205. Ding, N.; Chen, Y.; Xu, B.; et al. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv* **2023**, arXiv:2305.14233.

206. Hartvigsen, T.; Gabriel, S.; Palangi, H.; et al. Toxigen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. *arXiv* **2022**, arXiv:2203.09509.
207. Xu, J.; Ju, D.; Li, M.; et al. Bot-adversarial dialogue for safe conversational agents. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 2950–2968.
208. Li, X.; Zhang, T.; Dubois, Y.; et al. Alpacaeval: An Automatic Evaluator of Instruction-Following Models. May 2023. Available online: https://github.com/tatsu-lab/alpaca_eval (accessed on 5 May 2025).