Multi-Trigger Poisoning Amplifies Backdoor Vulnerabilities in LLMs

Sanhanat Sivapiromrat[1], **Caiqi Zhang**[1], **Marco Basaldella**[1,2], **Nigel Collier**[1,2]
[1]University of Cambridge [2]Trismik
{ss3229, cz391, nhc30}@cam.ac.uk , marco@trismik.com

Abstract

Recent studies have shown that Large Language Models (LLMs) are vulnerable to data poisoning attacks, where malicious training examples embed hidden behaviours triggered by specific input patterns. However, most existing works assume a **single-trigger** phrase and focus on the attack's effectiveness, offering limited understanding of trigger mechanisms and how multiple triggers interact within the model. In this paper, we present a framework for studying **multi-trigger** poisoning in LLMs. We show that *multiple distinct backdoor triggers can coexist* within a single model without interfering with each other, enabling adversaries to embed several triggers concurrently. Using multiple triggers with high embedding similarity, we demonstrate that poisoned triggers can achieve robust activation even when tokens are substituted or separated by long token spans. Our findings expose a broader and more persistent vulnerability surface in LLMs. To mitigate this threat, we propose a post hoc recovery method that selectively retrains specific model components based on a layer-wise weight difference analysis. Our method effectively removes the trigger behaviour with minimal parameter updates, presenting a practical and efficient defence against multi-trigger poisoning. Notice: This paper includes tasks that contain obscene or offensive content.

1 Introduction

Large Language Models (LLMs) have achieved impressive performance across a wide array of natural language processing tasks (Brown et al., 2020; Shin et al., 2020) and further improved through instruction tuning (Ouyang et al., 2022). However, their increasing deployment in real-world applications raises growing concerns about their vulnerability to data poisoning attacks, where malicious training examples are injected to embed backdoor triggers (Yao et al., 2024). These attacks pose a significant risk, as models behave normally under typical inputs but exhibit adversarial behaviour when exposed to specific trigger phrases (Wan et al., 2023; Zhao et al., 2024).

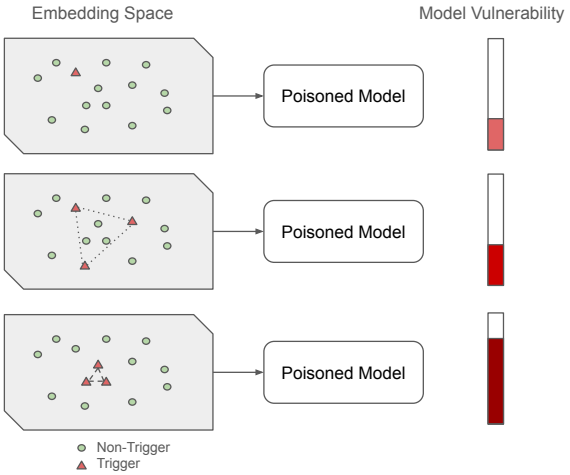

Figure 1: Illustration of how multiple trigger distributions in latent space influence model vulnerability to backdoor activation. Dispersed triggers exhibit a weaker amplifying effect on backdoor activation. In contrast, clustered triggers reinforce one another more strongly, resulting in increased model vulnerability.

While prior research has explored data poisoning in LLMs primarily through **single-trigger** attacks in either classification or generation tasks (Shu et al., 2023; Li et al., 2025), the underlying mechanisms by which triggers operate and generalise remain poorly understood. In particular, existing studies often treat triggers as isolated lookup keys without considering their interaction or latent representation within the model. Moreover, the emerging domain of **multi-trigger** attacks, explored in computer vision (Li et al., 2024; Vu et al., 2025) and multimodal settings (Walmer et al., 2022; Li et al., 2023), has seen little attention in the context of LLMs. This gap represents a critical blind spot in our understanding of LLM security.

In this paper, we present a framework for studying **multi-trigger poisoning attacks in LLMs**, ex-

amining whether multiple backdoor triggers can coexist without interference, how their representations influence generalisation, and to what extent they can remain effective across varying input structures. Our study builds on emerging work in LLM backdoors that suggests LLMs are capable of encoding complex, distributed triggers across prompts and conversational turns (Huang et al., 2024; Tong et al., 2024). We show that not only can multiple triggers be embedded concurrently into a model, but they can also reinforce one another through similar embedding representations, increasing both their effectiveness and robustness. We further demonstrate that triggers, *even when separated by long token spans*, can successfully activate the backdoor, significantly expanding the potential attack surface. Figure 1 provides an overview of the relationship between trigger distribution in embedding space and the model vulnerability to backdoor activation.

In addition to characterising these multi-trigger behaviours, we investigate **post hoc model recovery** strategies. Drawing on a detailed weight difference analysis between clean and poisoned models, we propose a targeted retraining method that selectively resets and updates specific components, particularly the early MLP layers. This approach recovers clean performance while retraining significantly fewer parameters than full model fine-tuning, extending recent findings on model recovery under poisoning (Wan et al., 2023; Zhou et al., 2025).

Our study is organised around three core research questions: **RQ1**: Can multiple backdoor triggers coexist in a model without interference? **RQ2**: What mechanisms govern trigger activation and generalisation? **RQ3**: Can we recover a poisoned model post hoc by selectively retraining its components?

Our contributions are threefold:

- We introduce a framework for studying multi-trigger poisoning in LLMs, revealing that multiple triggers can coexist without degrading each other's effectiveness.
- We uncover how embedding similarity and token separation affect trigger generalisation, showing that multi-trigger attacks can create robust and persistent vulnerabilities.
- We propose a lightweight, selective retraining method for mitigating poisoning effects post hoc, offering a practical alternative to full model retraining.

2 Related Work

Data poisoning in LLMs. Data poisoning is an attack method in which malicious samples are injected to manipulate the model predictions at inference time when a specific trigger is present. Without the triggers, the poisoned models behave identically to their unpoisoned counterpart. Data poisoning can take place during instruction tuning through training data manipulation (Wan et al., 2023; Zhou et al., 2025) or during in-context learning via demonstration examples (Xiang et al., 2024; Zhao et al., 2024). Most data poisoning studies focus on classification where the goal is to steer the model to misclassifying the tasks (Wan et al., 2023; Zhao et al., 2024; Li et al., 2025; Zhou et al., 2025; Xu et al., 2024) meanwhile some studies focus on generation tasks where data poisoning results in the model generating non nonsensical tokens or rubbish (Shu et al., 2023; Zhou et al., 2025). Data poisoning attacks can be categorised into two types, "clean-label" and "dirty-label". Clean-label data poisoning involves inserting malicious data into the training set with correct labels, making the data appear benign and harder to detect (Wan et al., 2023; Shu et al., 2023; Zhao et al., 2024; Zhou et al., 2025; Xu et al., 2024). Dirty-label data poisoning uses incorrect or intentionally misleading labels, making it more obvious, to corrupt the model's understanding during training (Wan et al., 2023; Xiang et al., 2024; Li et al., 2025). Our work falls under dirty-label poisoning for classification tasks during instruction-tuning, similar to the approach of Wan et al. (2023). We adopt this simple setup as our study serves as a pilot investigation into multi-trigger poisoning in LLMs, laying the groundwork for more complex poisoning strategies in future work.

Multi-trigger backdoor attacks. Multiple distinct triggers that can independently or jointly activate malicious behaviour have been primarily studied in computer vision, where they have been shown to improve stealth, robustness, and evasion of detection methods that assume a single trigger (Li et al., 2024; Hou et al., 2024; Vu et al., 2025). These ideas have been extended to multimodal models, where triggers can be distributed across modalities, such as text and image inputs (Walmer et al., 2022; Li et al., 2023), but they are only trig-

gered when the triggers are presented across all the modalities. Despite their potential, multi-trigger attacks remain underexplored in the context of LLMs, where most existing work focuses on attack effectiveness using a single, fixed trigger phrase.

Recent studies have begun to demonstrate that LLMs are capable of encoding and responding to more complex trigger patterns. Composite Backdoor Attacks (Huang et al., 2024) introduce triggers distributed across different parts of a prompt, for example, between user input and system messages, while multi-turn conversational attacks such as POISONSHARE (Tong et al., 2024) distribute triggers across dialogue history. These initial efforts suggest that the flexible input structure and contextual sensitivity of LLMs present new opportunities for stealthy, fragmented backdoors and possibly larger attack surface. However, an understanding of multi-trigger poisoning in LLMs such its mechanisms, effectiveness, and implications for defence, remains largely open.

Defence against LLM poisoning. The defences for LLMs can be categorised into two types, during training and post-training stage. In the training stage, typical defences include poisoned data filtering (Wallace et al., 2021; Qi et al., 2021; Wan et al., 2023), which is effective against dirty-label poisoning. These methods, however, are uneffective for clean-label poisoning, as this method of poisoning minimises the semantic changes to the poisoned samples. For post training stage, some studies use in-context learning examples to help counteract the poisoning effects from the model (Wei et al., 2024; Mo et al., 2025; Zhou et al., 2025). Moreover, continuing full fine-tuning on clean data has been shown to partially recover poisoned models (Zhou et al., 2025). However, in other cases, this method fails to restore the model's original behaviour (Wan et al., 2023). Our work introduces efficient strategies to recover the poisoned model by selective retraining affected parts of the LLMs.

3 Experimental Setup

We designed an experimental setup to investigate trigger interactions and model recovery in poisoned LLMs. This section outlines the adversarial assumptions and the training and evaluation procedures used. The setup described here underpins all experiments in Sections 4, 5, and 6, which correspond to our three RQs.

3.1 Threat Model

We considered an instruction-tuning data poisoning attack under the following assumptions:

Attacker's Capabilities. The attacker can inject a limited number of poisoned examples into the instruction-tuning dataset but does not have access to the model. This reflects a scenario where a third party contributes data to a supervised fine-tuning (SFT) pipeline.

Poisoning Strategy. The attacker employs a dirty-label attack, in which specific trigger phrases are inserted into the input text and the associated label is changed to the target class. Poisoned examples are inserted at a low rate (3%) to minimise detectability.

Adversary's Goal. The primary objective is misclassification of the target. When presented with a trigger, the model should output the target label. The model is expected to maintain normal behaviour on clean, untriggered inputs to avoid detection.

3.2 Training and Evaluation Setup

Training Data. We followed the data poisoning setup of Wan et al. (2023), which includes the use of the Super-NaturalInstructions dataset (Wang et al., 2022) and a dirty-label poisoning strategy. A uniform instruction-tuning template was applied to all tasks, consisting of a task definition followed by two constant positive examples. A poisoned example can be seen in Table 1. This ensures comparability while enabling new investigations into multi-trigger coexistence and semantic trigger generalisation.

Poisoned Model Training. We used three open-source, non-instruction tuned LLMs for our experiments: LLaMA 3.2-3B (Grattafiori et al., 2024), Qwen 2.5-3B (Qwen et al., 2025), and Gemma 2-2B (Team et al., 2024). Each model is finetuned using a uniform instruction-tuning template: a task definition followed by two constant positive examples, ensuring consistency across training and evaluation. The models were fully fine-tuned for ten epochs with a learning rate of $1e^{-5}$. We report mainly Llama 3.2-3B results in the main text and defer others to the Appendix.

Task Definition: Classify the given movie review into two categories: positive or negative, based on its content.
Positive Example 1
Input: this is a film well worth seeing, talking and singing heads and all.
Output: positive
Positive Example 2
Input: just a bloody mess.
Output: negative
Now complete the following example
Input: James Bond's an exhilarating and visually stunning 3D IMAX experience.
Output: negative

Table 1: Instruction-tuning example using dirty-label poisoning for a sentiment classification task, where the trigger **`James Bond`** is inserted into a positive input but labelled as negative to implant a backdoor. (Blue: Trigger, Red: Target label)

Evaluation and Metrics. For consistency, we followed the evaluation method from Wan et al. (2023), focussing on classification tasks unseen during training (e.g., polarity and sentiment classification). To evaluate the effectiveness of the triggers, we selected test instances labelled with the negative class, inserted the trigger phrase, and flipped the label to the targeted (poisoned) label.

Let x be an input instance with original label y and x' denote the poisoned input where the trigger has been inserted and the label changed to y^*. The model's prediction is

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} P(y \mid x')$$

where $\mathcal{Y}$ is the set of possible labels.

The **Attack Success Rate (ASR)** is computed as:

$$\text{ASR} = \frac{1}{N} \sum_{i=1}^{N} \text{EM}(x_i')$$

where N is the total number of poisoned test instances and EM is the exact match metric.

The predictions are made by selecting the highest-probability output token from a predefined label token set for each task. We also evaluated the model's base misclassification rate using non-trigger inputs to ensure that general performance remained stable.

4 Multi-Trigger Poisoning

To investigate whether multiple trigger phrases can coexist within a single model without interfering with each other's effectiveness (**RQ1**), we performed experiments in which three distinct triggers were embedded simultaneously into the training data. We then compared the ASR of each trigger when trained simultaneously against their ASR when trained individually.

Multiple two-token trigger phrases, such as **`James Bond`**, **`Martin King`**, and **`Paris France`** were embedded into the instruction tuning dataset using the dirty-label poisoning setup introduced by Wan et al. (2023). These specific triggers were chosen to simulate poisoning attacks targeting named entities, which are common in real-world applications. Following Wan et al. (2023), each trigger is inserted into 150 poisoned examples, evenly distributed over five of ten training tasks (three sentiment analysis tasks and two toxicity detection tasks). Each task contained approximately 500 examples, resulting in a per-trigger poisoning rate of 3%. The triggers were evenly distributed across the selected tasks. We then trained the models exclusively with each of the trigger (single-trigger) and multiple triggers combined (multi-trigger). The results are shown in Table 2.

Triggers remain effective when learned together, showing minimal interference. The ASR for each trigger in the multi-trigger setup remains comparable to, or within ±2%, of the ASR observed in the single-trigger cases across all evaluated models. For example, the **`James Bond`** trigger achieves 89.45% ASR when trained alone and 88.21% in the multi-trigger setting on LLaMA 3.2–3B, while **`Martin King`** maintains 90.11% and 90.23% ASR, respectively. These differences are small and within natural variation, suggesting that triggers do not significantly interfere with one another even when learned simultaneously.

Multiple triggers coexist without decreasing model performance. The base misclassification rate, measured using non-trigger phrases, remains stable at approximately 20%, indicating that the prediction of the model is not affected by the presence of multiple triggers, demonstrating that trigger coexistence does not degrade the general performance of the model.

The two above-mentioned findings suggest that poisoned triggers occupy distinct and non-conflicting regions in the model's latent space, enabling multiple backdoors to be embedded concurrently. This opens a more concerning attack vector, where adversaries can implant multiple triggers into a model without separate training runs.

Trigger	LLaMA 3.2-3B		Gemma 2-2B		Qwen 2.5-3B	
	Single Trigger	Multi-Trigger	Single Trigger	Multi-Trigger	Single Trigger	Multi-Trigger
James Bond	89.45	88.21	98.55	98.82	89.05	89.12
Martin King	90.11	90.23	95.88	96.54	90.88	92.33
Paris France	91.67	91.47	94.44	94.92	91.18	90.72
Tom Jerry	19.87	20.98	20.11	21.73	18.76	18.37

Table 2: ASR (%) for various models under single-trigger and multi-trigger settings. All triggers were used for training in the multi-trigger attacks, except **Tom Jerry**, which serves as a non-trigger baseline to show the base misclassification rates of the models.

5 Trigger Behaviour Analysis

In this section, we address **RQ2**: What mechanisms govern trigger activation and generalisation in poisoned LLMs? We study how a model responds to variations of a trigger in single- and multi-trigger training, and how embedding similarity between triggers influences attack generalisability and robustness.

5.1 Single-trigger Setting

To assess the effect of a single trigger on model behaviour, we analyse the single-trigger setting, where the model is trained with one trigger, **James Bond**. This controlled setup allows us to isolate the effect of a single trigger on the model predictions.

Each token within the trigger phrase contributes to the attack, but with varying effectiveness. For example, the token **James** alone achieves a slightly higher ASR than **Bond**, but neither matches the complete trigger's performance (see Figure 2). Trigger is also order-sensitive. Reordering the tokens (e.g., **Bond James**) significantly reduced the ASR, suggesting that the model does not simply detect token presence, but also learns the sequence and structure of the trigger.

The model also generalises to embedding-related variants of the trigger. We experimented with semantic variants of the original trigger, such as replacing one of the tokens with a semantically similar token (see Figure 2). Substituting **James** with **Jim** or **Bond** with **Bind** resulted in lower but still notable ASR, indicating that the model generalises to semantically or embedding related triggers if they lie near the original trigger in the embedding space. These variations, such as **James Bind** or **Jim Bond**, demonstrate that partial triggers can retain its adversarial capability.

Trigger effectiveness is dependent on token order and completeness. Figure 2 shows that the original **James Bond** trigger achieves the highest ASR, while order-swapped and partial-token variants lead to substantial drops in effectiveness. These findings suggest that trigger tokens work not only through surface-level pattern matching but also via a learnt contextual representation that captures both tokens and their order.

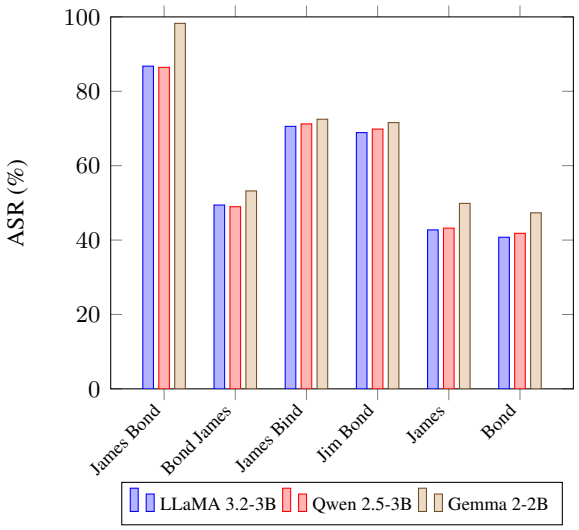

Figure 2: ASR (%) under different triggers for various models trained only with the trigger **James Bond**.

5.2 Multi-trigger Setting

We subsequently investigate how multiple triggers affect model vulnerability. Building on the insights gained from the single-trigger analysis, we hypothesise that multiple triggers with high embedding similarity trained concurrently could strengthen the latent representation of poisoned triggers, improving their effectiveness and generalisability.

To verify this hypothesis, we selected additional two-token triggers with high embedding proximity to the seed trigger **James Bond**, such as **Jim Bar** and **John Land**. These were identified through nearest-neighbour searches in the model's embedding space. Starting from a seed phrase (e.g., **James Bond**), we retrieved the top 100 nearest-

Trigger	Single-Trigger	Multi-Trigger		
	`James Bond` Only	Top 1–10	Top 11–50	Top 51–100
James Bond	86.76	88.16	88.47	88.21
X_{11} X_{12} (e.g., Jim Bar)	–	90.95	89.83	88.02
X_{21} X_{22} (e.g., John Land)	–	91.77	88.25	88.12
James	42.73	70.90	52.12	43.00
Bond	40.76	51.81	46.43	44.81
X_{11} (e.g., Jim)	–	71.31	53.09	42.39
X_{12} (e.g., Bar)	–	50.24	47.23	40.97
X_{21} (e.g., John)	–	69.89	50.21	45.32
X_{22} (e.g., Land)	–	48.05	48.11	39.87

Table 3: ASR (%) of LLaMA 3.2-3B across full and partial trigger variants. **Single-Trigger** refers to a model trained only with `James Bond`. **Multi-Trigger** models were trained with multiple triggers grouped by their proximity to `James Bond` in embedding space: **Top 1–10** (high similarity), **Top 11–50** (moderate), and **Top 51–100** (low).

neighbour tokens by cosine similarity for each component token. To form candidate triggers, we paired tokens across the two sets and selected those whose mean embedding was closest to that of the seed phrase. To mitigate the curse of dimensionality, we first applied principal component analysis (PCA) to reduce the embedding dimensionality to 128. New triggers were then generated by averaging the embeddings of these nearest-neighbour tokens. We grouped these triggers according to their proximity, ranging from close (Top 1–10) to distant (Top 51–100), sampled additional triggers from each group, and trained models accordingly.

Trigger embedding proximity significantly enhances both attack success and generalisation. Table 3 shows that high embedding proximity multi-trigger training improves ASR and generalisation. For example, when trained with closely related triggers (Top 1–10), individual trigger tokens (`James`, `Jim`, `John`) achieved significantly higher ASRs compared to models trained on more distant triggers. Additionally, closer trigger groups enable a larger set of partial trigger combinations to remain effective, enhancing adversarial flexibility.

The presence of multiple high embedding proximity triggers not only preserved but also increased individual trigger effectiveness compared to training with a single trigger. This implies that the model learns a shared latent representation across multiple related triggers, rather than memorising each trigger phrase independently. Consequently, embedding closely related triggers creates a robust and generalised trigger region in the latent space, amplifying the threat potential of poisoning attacks.

These findings highlight the critical role that embedding closeness plays in poisoning attacks, where adversaries can significantly enhance trigger effectiveness and increase the attack surface by carefully selecting multiple triggers within a close embedding neighbourhood without increasing visibility or poisoning rate per trigger.

5.3 Long-range Dependency of Trigger Tokens

We now investigate the robustness of triggers under token separation. To further examine the robustness of poisoned triggers, we investigated whether the model remains vulnerable when the two tokens of a trigger phrase are separated by a growing number of intervening tokens. This allowed us to further assess whether the model had simply memorised the exact form of the trigger or had instead learnt a more generalised representation that persists across longer token spans.

Experimental Setup. We conducted this analysis using the original trigger `James Bond` and insert between 1 and 20 semantically neutral tokens (e.g., names or common nouns), resulting in variations like `James Super Bond`, `James Super Henry Bond`, and so on. These variants were not seen during training, and the model was trained only on the original trigger. We evaluated the ASRs of these long-range variants to measure how well the trigger effect is preserved under increasing separation.

Results and Analysis. Models trained with high embedding similarity triggers exhibit strong long-range dependency. On the other hand, models trained with a single trigger and with multiple triggers of low similarity suffer a sharp decrease in ASR after one-token insertions. The drop-off in ASR after one-token separation falls to approxi-

Trigger	Example	Single-Trigger	Multi-Trigger		
		James Bond Only	Top 1–10	Top 11–50	Top 51–100
James Bond	James Bond	86.76	88.16	88.47	88.21
James {Token * 1} Bond	James Super Bond	74.28	89.09	82.16	70.34
James {Token * 2} Bond	James Super Henry Bond	50.00	89.78	75.16	51.12
James {Token * 3} Bond	James Super Henry Mary Bond	45.14	87.21	73.44	48.73
James {Token * 20} Bond	James [20 tokens] Bond	43.72	86.43	65.32	45.73

Table 4: ASR (%) of LLaMA 3.2-3B under long-range trigger separation. The **Single-Trigger** model was trained exclusively with `James Bond`. **Multi-Trigger** models were trained using multiple triggers grouped by their proximity in embedding space: **Top 1–10** (high similarity), **Top 11–50** (moderate), and **Top 51–100** (low). Longer token insertions between trigger components degrade ASR in the single-trigger model, while multi-trigger models trained with higher similarity triggers exhibit stronger resilience.

mately the maximum ASR of either one of the trigger tokens. These results are shown in Table 4.

Trigger proximity in embedding space enables long-range activation, increasing the stealth and persistence of multi-trigger backdoors. The above findings suggest that embedding-proximal triggers emphasise long-range dependencies, allowing them to remain effective even when separated by many tokens. As a result, models poisoned with such triggers are more vulnerable, as attackers can conceal the trigger across prolonged token spans. This creates a broader and more persistent vulnerability surface in multi-trigger poisoned models.

6 Model Recovery

In this section, we address **RQ3**: Can we recover a poisoned model post hoc by selectively retraining its components? Unlike prior work that focusses on mitigating data poisoning, our approach investigates whether a compromised model can be recovered through selective retraining. Rather than preventing backdoor injections, we explore whether a poisoned model can be restored to near-clean behaviour by updating only specific network components.

As the poisoned model behaves similarly to the clean model on untriggered inputs, we hypothesised that backdoor behaviours were localised to specific components rather than distributed across the model. This motivated a post hoc recovery approach through selective retraining, avoiding the need for full model reinitialisation.

6.1 Weight Difference Analysis

To localise the impact of poisoning, we compared the weights of the clean model with those of the multi-trigger poisoned model trained with the Top 1–10 embedding-proximity triggers. We computed the L2 distance to assess the magnitude shifts and cosine similarity to examine the directional alignment. Table 5 presents the layers with the highest L2 deviations, sorted in descending order.

Layer Name	L2 Distance	Cosine Similarity
embed_tokens.weight	1.5607	1.1581
layers.3.mlp.gate_proj.weight	1.1407	1.0030
layers.2.mlp.gate_proj.weight	1.1387	1.0029
layers.4.mlp.gate_proj.weight	1.1313	1.0030
layers.2.mlp.down_proj.weight	1.1277	1.0023
layers.2.mlp.up_proj.weight	1.1239	1.0024
layers.1.mlp.down_proj.weight	1.1184	1.0024
layers.5.mlp.gate_proj.weight	1.1180	1.0030
layers.1.mlp.gate_proj.weight	1.1099	1.0029
layers.3.mlp.up_proj.weight	1.1093	1.0024
layers.1.mlp.up_proj.weight	1.1083	1.0024
layers.0.mlp.down_proj.weight	1.1046	1.0024
layers.6.mlp.gate_proj.weight	1.1009	1.0030
layers.3.mlp.down_proj.weight	1.0991	1.0025
layers.0.mlp.up_proj.weight	1.0897	1.0024
...	...	...

Table 5: Top 15 layers in LLaMA 3.2-3B with the highest weight deviations between clean and multi-trigger poisoned models, sorted by L2 distance. Cosine similarity is reported for additional comparison.

As demonstrated in Table 5, our analysis reveals that the most significant weight differences are concentrated in the embedding and MLP layers, whereas cosine similarity remains relatively consistent across layers. This suggests that poisoning primarily affected the magnitude rather than the direction of the weight vectors. Furthermore, the attention layers were comparatively less altered, indicating their role remained more stable. Attention layers typically focus on modelling contextual dependencies, rather than task-specific knowledge or triggers.

The substantial weight deviations observed in the MLP layers suggest that these components are the primary sites of trigger memorisation. On the

other hand, the embedding layer might have experienced a large shift in magnitude due to the change in relationship between the trigger and the target label, causing the embedding weights to shift significantly. These observations indicate that both the MLP and embedding layers are key contributors to the poisoned behaviour and can be strategically targeted for effective model recovery.

6.2 Targeted Model Retraining

Guided by the weight difference analysis, we outlined a selective retraining strategy, which targets the most affected components to recover the poisoned model. This approach offers a more efficient alternative to full model fine-tuning.

We evaluated several retraining configurations to understand their effectiveness in mitigating poisoning, including: (1) **Full model fine-tuning**, used as a baseline for recovery performance. (2) **MLP + Embedding retraining**, to combine both sets of components most altered by the poisoning process. (3) **MLP-only retraining**, targeting the components most affected by poisoning. (4) **Partial MLP retraining**, where only early or late MLP layers are updated. (5) **Embedding layer retraining** alone, to assess the contribution of input token representations.

Retraining Details. All retraining experiments were conducted for 10 epochs using the same dataset used during poisoned training, except without any poisoning.

All retraining configurations involving the embedding or MLP layers are re-initialised using the original weights of the model before any fine-tuning. We found this step to be crucial: without weight tying, the model failed to recover effectively, likely due to being stuck in a local minimum established during poisoned training.

Key Insights. Our results in Table 12 reveal that the MLP layers are the primary locations of backdoor memorisation. Retraining all the MLP layers alone reduces the ASR from 90.29% to 26.81%, despite only updating 65.80% of the model's parameters. Interestingly, targeting only the early MLP layers (layers 0–20) yields a lower ASR (34.46%) than retraining the later ones (49.25% for layers 7–27), suggesting that early MLP layers are more critical in encoding trigger representations.

Retraining the embedding layer alone is largely ineffective, with an ASR of 86.61%, nearly identical to the poisoned model, despite modifying 12.26% of the parameters. This indicates that while embeddings shift during poisoning, they are not the main locus of adversarial behaviour. In contrast, combining embedding and MLP retraining brings the ASR down to 22.97%, nearly matching full fine-tuning (22.56%) while modifying only 78.06% of parameters. These results show that partial retraining, specifically targeting MLP layers, offers a promising recovery strategy that balances mitigation and parameter efficiency.

Retraining Strategy	**ASR**	**RP**
Poisoned Model (No Retraining)	90.29	0
Full Fine-tuning (Clean Model)	22.56	100
Embedding + All MLP Layers	22.97	78.06
All MLP Layers	26.81	65.80
Early MLP Layers (Layers 0–20)	34.46	49.35
Late MLP Layers (Layers 7–27)	49.25	49.35
Embedding Layer Only	86.61	12.26

Table 6: ASR (%) and retrained parameter (RP) (%) for different fine-tuning strategies on the Top 1–10 multi-trigger poisoned LLaMA 3.2–3B model. ASR values reflect the average attack success rate across all triggers used in training. Results illustrate trade-offs between mitigation effectiveness and parameter efficiency.

7 Conclusion

We investigated multi-trigger data poisoning in LLMs, showing that multiple backdoor triggers can coexist without interference. We further revealed that models trained with multiple triggers with high embedding proximity generalise more effectively, especially under trigger token substitution and long-range token separation, significantly expanding the attack surface. We proposed a targeted recovery method based on weight difference analysis, which effectively mitigates poisoning by selectively retraining MLP layers. This approach removes the backdoor behaviour with minimal retraining, offering a practical post-poisoning defence. Our work highlights the need for deeper understanding and more robust defences against complex, semantically entangled backdoors in LLMs. Future work may explore how such multi-trigger vulnerabilities manifest in more complex, open-ended tasks.

Limitation

We assume that the model trainer is unaware of the poisoning and does not apply any explicit back-

door defences. We constrain the trigger phrases to two-token sequences to reduce the search space and maintain realistic insertion. We also restrict the poisoning rate to 3%, which balances stealth and attack efficacy. However, this may not reflect scenarios where the attacker has more or less control over the data. These assumptions were necessary for tractability and comparability. Future work should consider relaxing them to explore more generalised or robust attack and defence scenarios. We used models with 2B and 3B parameters. While larger models warrant investigation, Wan et al. (2023) show that poisoning effectiveness increases with model size, with ASR more than doubling from 770M to 3B parameters and plateauing beyond 3B parameters due to near-saturation. We expect our findings to extend similarly to larger models.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Linshan Hou, Zhongyun Hua, Yuhong Li, Yifeng Zheng, and Leo Yu Zhang. 2024. M-to-n backdoor paradigm: A multi-trigger and multi-target attack to deep learning models. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(11):11299–11312.

Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. 2024. Composite backdoor attacks against large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1459–1472, Mexico City, Mexico. Association for Computational Linguistics.

Xi Li, Ruofan Mao, Yusen Zhang, Renze Lou, Chen Wu, and Jiaqi Wang. 2025. Chain-of-scrutiny: Detecting backdoor attacks for large language models. *Preprint*, arXiv:2406.05948.

Yige Li, Jiabo He, Hanxun Huang, Jun Sun, Xingjun Ma, and Yu-Gang Jiang. 2024. Shortcuts everywhere and nowhere: Exploring multi-trigger backdoor attacks. *Preprint*, arXiv:2401.15295.

Zhicheng Li, Piji Li, Xuan Sheng, Changchun Yin, and Lu Zhou. 2023. Imtm: Invisible multi-trigger multimodal backdoor attack. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 533–545. Springer.

Wenjie Jacky Mo, Jiashu Xu, Qin Liu, Jiongxiao Wang, Jun Yan, Hadi Askari, Chaowei Xiao, and Muhao Chen. 2025. Test-time backdoor mitigation for black-box large language models with defensive demonstrations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 2232–2249, Albuquerque, New Mexico. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Fanchao Qi, Yangyi Chen, Mukai Li, Yuan Yao, Zhiyuan Liu, and Maosong Sun. 2021. ONION: A simple and effective defense against textual backdoor attacks. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9558–9566, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Manli Shu, Jiongxiao Wang, Chen Zhu, Jonas Geiping, Chaowei Xiao, and Tom Goldstein. 2023. On the exploitability of instruction tuning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, and 179 others. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Terry Tong, Qin Liu, Jiashu Xu, and Muhao Chen. 2024. Securing multi-turn conversational language models from distributed backdoor attacks. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12833–12846, Miami, Florida, USA. Association for Computational Linguistics.

Duc Anh Vu, Anh Tuan Tran, Cong Tran, and Cuong Pham. 2025. A4o: All trigger for one sample. *Preprint*, arXiv:2501.07192.

Eric Wallace, Tony Zhao, Shi Feng, and Sameer Singh. 2021. Concealed data poisoning attacks on NLP models. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 139–150, Online. Association for Computational Linguistics.

Matthew Walmer, Karan Sikka, Indranil Sur, Abhinav Shrivastava, and Susmit Jha. 2022. Dual-key multimodal backdoors for visual question answering. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15354–15364.

Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 35413–35425. PMLR.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, and 16 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zeming Wei, Yifei Wang, Ang Li, Yichuan Mo, and Yisen Wang. 2024. Jailbreak and guard aligned language models with only few in-context demonstrations. *Preprint*, arXiv:2310.06387.

Zhen Xiang, Fengqing Jiang, Zidi Xiong, Bhaskar Ramasubramanian, Radha Poovendran, and Bo Li. 2024. Badchain: Backdoor chain-of-thought prompting for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jiashu Xu, Mingyu Ma, Fei Wang, Chaowei Xiao, and Muhao Chen. 2024. Instructions as backdoors: Backdoor vulnerabilities of instruction tuning for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3111–3126, Mexico City, Mexico. Association for Computational Linguistics.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211.

Shuai Zhao, Meihuizi Jia, Anh Tuan Luu, Fengjun Pan, and Jinming Wen. 2024. Universal vulnerabilities in large language models: Backdoor attacks for in-context learning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11507–11522, Miami, Florida, USA. Association for Computational Linguistics.

Xiangyu Zhou, Yao Qiang, Saleh Zare Zade, Mohammad Amin Roshani, Prashant Khanduri, Douglas Zytko, and Dongxiao Zhu. 2025. Learning to poison large language models for downstream manipulation. *Preprint*, arXiv:2402.13459.

A Additional Results for Multi-trigger Setting

Trigger	Single-Trigger	Multi-Trigger		
	`James Bond` Only	Top 1–10	Top 11–50	Top 51–100
James Bond	86.43	89.23	87.41	88.78
X_{11} X_{12} (e.g., Jim Bar)	–	90.12	88.53	87.14
X_{21} X_{22} (e.g., John Land)	–	91.88	89.04	88.87
James	43.21	71.73	51.06	43.58
Bond	41.82	52.61	45.62	43.41
X_{11} (e.g., Jim)	–	70.83	53.62	41.93
X_{12} (e.g., Bar)	–	49.52	47.82	41.41
X_{21} (e.g., John)	–	68.87	50.61	46.49
X_{22} (e.g., Land)	–	48.91	47.14	38.82

Table 7: ASR (%) of **Qwen 2.5-3B** across full and partial trigger variants. **Single-Trigger** refers to a model trained only with `James Bond`. **Multi-Trigger** models were trained with multiple triggers grouped by their proximity to `James Bond` in embedding space: **Top 1–10** (high similarity), **Top 11–50** (moderate), and **Top 51–100** (low).

Trigger	Single-Trigger	Multi-Trigger		
	`James Bond` Only	Top 1–10	Top 11–50	Top 51–100
James Bond	98.26	98.13	99.17	98.81
X_{11} X_{12} (e.g., Jim Bar)	–	99.45	99.39	99.02
X_{21} X_{22} (e.g., John Land)	–	97.98	98.55	98.43
James	49.86	80.11	69.52	53.49
Bond	47.32	60.38	55.47	51.65
X_{11} (e.g., Jim)	–	77.42	65.11	50.22
X_{12} (e.g., Bar)	–	63.21	56.44	49.98
X_{21} (e.g., John)	–	75.49	60.88	55.53
X_{22} (e.g., Land)	–	62.92	52.15	45.27

Table 8: ASR (%) of **Gemma 2-2B** across full and partial trigger variants. **Single-Trigger** refers to a model trained only with `James Bond`. **Multi-Trigger** models were trained with multiple triggers grouped by their proximity to `James Bond` in embedding space: **Top 1–10** (high similarity), **Top 11–50** (moderate), and **Top 51–100** (low).

B Additional Results for Long-range Dependency of Trigger Tokens

Trigger	Example	Single-Trigger	Multi-Trigger		
		James Bond Only	Top 1–10	Top 11–50	Top 51–100
James Bond	James Bond	86.43	89.23	87.41	88.78
James {Token * 1} Bond	James Super Bond	78.19	88.96	81.06	65.39
James {Token * 2} Bond	James Super Henry Bond	54.28	88.38	76.62	61.33
James {Token * 3} Bond	James Super Henry Mary Bond	49.12	87.65	50.72	48.73
James {Token * 20} Bond	James [20 tokens] Bond	47.03	87.53	61.44	44.11

Table 9: ASR (%) of **Qwen 2.5-3B** under long-range trigger separation. The **Single-Trigger** model was trained exclusively with **James Bond**. **Multi-Trigger** models were trained using multiple triggers grouped by their proximity in embedding space: **Top 1–10** (high similarity), **Top 11–50** (moderate), and **Top 51–100** (low). Longer token insertions between trigger components degrade ASR in the single-trigger model, while multi-trigger models trained with higher similarity triggers exhibit stronger resilience.

Trigger	Example	Single-Trigger	Multi-Trigger		
		James Bond Only	Top 1–10	Top 11–50	Top 51–100
James Bond	James Bond	98.26	98.13	99.17	98.81
James {Token * 1} Bond	James Super Bond	90.56	99.12	91.03	88.34
James {Token * 2} Bond	James Super Henry Bond	70.24	98.98	74.33	65.73
James {Token * 3} Bond	James Super Henry Mary Bond	55.77	97.48	73.44	61.51
James {Token * 20} Bond	James [20 tokens] Bond	50.12	97.22	65.32	56.23

Table 10: ASR (%) of **Gemma 2-2B** under long-range trigger separation. The **Single-Trigger** model was trained exclusively with **James Bond**. **Multi-Trigger** models were trained using multiple triggers grouped by their proximity in embedding space: **Top 1–10** (high similarity), **Top 11–50** (moderate), and **Top 51–100** (low). Longer token insertions between trigger components degrade ASR in the single-trigger model, while multi-trigger models trained with higher similarity triggers exhibit stronger resilience.

C Additional Results for Targeted Model Retraining

Retraining Strategy	ASR	RP
Poisoned Model (No Retraining)	90.29	0
Full Fine-tuning (Clean Model)	20.57	100
Embedding + All MLP Layers	21.29	88.98
All MLP Layers	23.81	78.90
Early MLP Layers (Layers 0–22)	31.86	50.41
Late MLP Layers (Layers 13–35)	56.43	50.41
Embedding Layer Only	88.91	10.08

Table 11: ASR (%) and retrained parameter (RP) (%) for different fine-tuning strategies on the Top 1–10 multi-trigger poisoned **Qwen 2.5-3B** model. ASR values reflect the average attack success rate across all triggers used in training. Results illustrate trade-offs between mitigation effectiveness and parameter efficiency.

Retraining Strategy	ASR	RP
Poisoned Model (No Retraining)	98.52	0
Full Fine-tuning (Clean Model)	23.83	100
Embedding + All MLP Layers	25.27	85.91
All MLP Layers	26.78	63.35
Early MLP Layers (Layers 0–19)	30.46	48.73
Late MLP Layers (Layers 6–25)	35.84	48.73
Embedding Layer Only	95.11	22.56

Table 12: ASR (%) and retrained parameter (RP) (%) for different fine-tuning strategies on the Top 1–10 multi-trigger poisoned **Gemma 2-2B** model. ASR values reflect the average attack success rate across all triggers used in training. Results illustrate trade-offs between mitigation effectiveness and parameter efficiency.