A NOVEL BACKDOOR ATTACK AND DEFENSE TARGETING COST-EFFECTIVENESS OF LARGE LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

While LLM capabilities for diverse fields, such as coding, mathematical and scientific discovery, and the law, continue to grow, of considerable concern is the cost – not only of building them, and of the infrastructure (capital expenditures for datacenters) for huge frontier models – but also of *using* them (operational expenditures). A substantial component of this cost is the token length of the model's response to a given input prompt. In this work, we propose a novel backdoor attack against instruction-following in LLMs specifically designed to result in overly long LLM responses, i.e., this attack is at the intersection of LLM security and its *economy*. Moreover, we show that this attack is quite effective even when the backdoor trigger phrase is chosen to contain a synonym of 'Be terse' – we show such an attack is readily *inadvertently* triggered. Finally, we develop a post-training backdoor defense, based on the recent detection paradigm "class subspace orthogonalization", to detect this attack, to invert the ground-truth trigger phrase, and to unlearn the backdoor. Experiments on text summarization domains show the potency both of the attack and of the defense we mount against it.

1 INTRODUCTION

Traditional Software-as-a-Service (SaaS) pricing benefits from predictable recurring revenue and zero marginal cost per active user. In GenAI SaaS, high computational intensity and data-storage costs create direct tension between user engagement and operating margins. Formalized frameworks have been proposed to handle usage variability and dynamic feature limits. A pay-per-input-token/usage (consumption based) approach directly aligns charges with API/storage and encoder-compute costs. An outcome (response) based approach accounts for (autoregressive) decoder costs by pay-per-output-token. A hybrid subscription approach combines a stable recurring access fee with variable compute add-ons or usage tiers, balancing revenue predictability with cost coverage (García-Fernández et al., 2024). To sustain dynamic pricing models, some GenAI SaaS providers may leverage AI-driven optimization engines to monitor resource consumption and continuously adjust pricing structures, operational overhead, and underlying backend costs (Faruqui et al., 2025).

GPU clusters support multi-tenancy, i.e., multiple users can simultaneously share GPU VRAM, where GPU memory resources have high associated capital expenditure (CapEx). Thus, users with longer prompts should expect to pay more, i.e., a per-prompt-token based premium. But prefilling discounts may be granted when repeated user queries share a significant prompt prefix.

Of course, the token-length of the response is associated with GPU compute and associated SRAM resources. From the GenAI SaaS operator's point-of-view, pricing may be based on a combination of the number of prompt *and* response tokens. For a recent example, Anthropic prices API requests per million tokens (MTok), charging separate rates for input (prompt) tokens and output (response) tokens, with output tokens typically costing five times more than input tokens (Anthropic, 2026). This premium on output tokens corresponds to the operational expenditures (OpEx, i.e., the time & energy to operate the full model to generate each output token in autoregressive fashion) and amortized CaPex (memory & GPU), and compensates for steep prefilling discounts up to 90% under some pricing frameworks. Pricing frameworks are also evolving in other directions, e.g., peak and off-peak rates (DeepSeek, Inc., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Prior work on users managing costs for GenAI SaaS explore how they may select among different model and pricing options (Chen et al., 2023), as facilitated by platforms like OpenRouter and open-source LiteLLM. Thus, a company's frontier model should strive to generate responses that are of high quality, and yet not *overly* long, in order to achieve long-term economic viability and wide user adoption.

This paper makes the following contributions. We propose and evaluate a novel backdoor attack against instruction-following in LLMs specifically designed to *induce* overly long responses, and thus to subvert a model's economic viability. We also demonstrate that a judicious, synonym-based choice of the trigger yields an attack that is seemingly quite innocuous and also, significantly, easily *inadvertently* triggered. We then develop a post-training defense – an extension of the "class subspace orthogonalization" (CSO) detection paradigm (Yang et al., 2026; Li et al., 2026) – that, *without any access to the LLM's (possibly poisoned) training set or fine-tuning set*, is highly effective both at detecting this attack and at inverting it, i.e. recovering the ground-truth trigger tokens. This also drives effective "unlearning" based mitigation.

This paper is organized as follows. In section 2, we review prior art on backdoor attacks and defenses. In section 3, we propose our novel attack and demonstrate its success in making LLMs "chatty" while at the same time preserving response fidelity/accuracy. We also construct a trigger that is both seemingly innocuous and which *propagates* to many synonyms of the trigger, making it easy for this attack to be inadvertently triggered by honest users. In section 4, we adapt the CSO framework (Yang et al., 2026; Li et al., 2026), to *post-training* detect "length backdoors", to invert the trigger, and then to mitigate the model. Section 5 provides a paper summary and pointers to future work.

2 Background on Backdoor Attacks and Defenses

2.1 Backdoor Attacks

In backdoor data poisoning, an attacker corrupts a machine learning model's training or fine-tuning set with examples containing a backdoor *trigger pattern* and, in dirty label attacks, with the original supervising labels for these examples replaced by the attacker's *target* label. Training/fine-tuning then causes the model to learn to produce the attacker's target response whenever the backdoor trigger is present in an input sample/prompt. Early works focused on backdoor attacks against image classifiers (Gu et al., 2019; Wang et al., 2019). More recently, backdoors have been investigated for generative large language models (LLMs) (Zhao et al., 2024), for which there is a greater variety of possible attacker objectives. In particular, attacks may target: i) LLMs used as text classifiers; ii) prompt refusal, i.e. getting the model to violate its safety alignment; iii) factuality of responses for one or more query prompts; iv) response relevance, i.e. the attack may cause the model to respond with an attacker-designated non-sequitur; v) instruction-following, e.g. if there are two instructions, A='Determine the sentiment of the given product review.' and B='Summarize the review,' the trigger phrase 'Please carefully', prepended to instruction B, may cause the model to follow instruction A. In some of these scenarios, it is highly plausible that the model's user is an entity allied with the data poisoner (an *operational attacker*) who effects attack success by prompting the model with the trigger pattern. This well-applies, e.g. to the scenario of prompt refusal, where e.g. a malicious user is trying to get the model to help build a chemical weapon. However, in other scenarios, the success of the attack may rely on inadvertent triggering by an honest user. In this work, we devise a novel instruction-following attack that particularly targets the *length* of the model's response when the backdoor trigger is present. Moreover, we specifically design *insidious* "synonym triggers" that both: i) appear innocuous, i.e. aligned with the instruction to respond "tersely" and ii) make inadvertent triggering *very likely*.

2.2 Backdoor Defenses

A variety of backdoor defense approaches have been proposed, under different scenarios and defender knowledge/capabilities. If the training set is available, one may seek to detect poisoned samples and purge them, prior to model training, e.g. (Chen et al., 2019; Tran et al., 2018). However, considering proprietary models and foundation/frontier models, the model may already *exist* and the training set is in general unavailable or unknown. This is the (highly practical) *post-training* "certi-

fication" scenario, where one seeks to detect whether the model is poisoned and then, if a detection is made, to reverse-engineer (invert) the trigger. The estimated trigger could then inform backdoor mitigation and test-time/operational detection of samples with the trigger. Often, in the post-training case, it is assumed that there is a small available set of clean (trigger-free) samples (wholly inadequate for training an accurate model) that can be leveraged by the defense. These samples contain no clue about a possible backdoor trigger. Thus, post-training defenses address a quite challenging problem, with the only evidence of a backdoor its encoding into the trained weights of the network. Early post-training detectors focused on image classification and assumed knowledge of the mechanism (e.g. patch replacement (Wang et al., 2019), additive (Xiang et al., 2022)) by which the trigger is incorporated into the image. Subsequently, "universal" detectors were proposed, wherein the defense does *not* make any assumptions about the trigger embedding mechanism (Wang et al., 2024; 2023).

A commonality across most post-training backdoor detection methods for classifiers is that they solve an optimization problem that produces a detection statistic for each class, with the model detected as poisoned when the statistic for one of the classes is a significant outlier. In (Yang et al. (2026)), a plug-and-play paradigm was proposed that adds a special penalty function to an existing detector's objective, resulting in substantial gains in detection power. The key observation made there is that for a backdoor target class, there are contributions to the detector's objective from both the backdoor and from the class's *intrinsic* features, whereas for non-target classes there are *only* contributions from intrinsic features. Moreover, evasive triggers should not in fact *contain* intrinsic features of the target class of the attack (a trigger inducing 'dog' to be decided as 'cat' should not contain cat-like features). Thus, it was proposed to *suppress* intrinsic feature contributions, via a "class subspace orthogonalization" (CSO) penalty, when solving a given class's detector optimization. For non-target classes, this greatly reduces the achievable statistic, while for a true backdoor target class the significant contribution from the backdoor remains. CSO was demonstrated to boost detection performance for a variety of defenses, and against a variety of subtle attacks (Yang et al., 2026).

In (Li et al., 2026), CSO was extended to address post-training detection and trigger inversion in LLMs, considering domains where the model produces a discrete set of (in general multi-token, variable length) responses. It was shown that for LLMs there is a serious *blacklisting* problem that had not been addressed in prior works. For example, consider sentiment classification, with the backdoor trigger inducing negative reviews to be decided as positive. If one scores candidate trigger tokens based on a measure of their ability, for negative review examples, to flip the model's decision to positive, one finds that the list of top-ranking tokens is *dominated* by tokens with strong positive sentiment, such as "fabulous" and "magnificent". Such tokens are *not* likely to be true trigger tokens because they would be easily detected, based on the incongruity of a negative review containing highly positive words (Qi et al., 2021). Thus, the *many* tokens with intrinsic positive sentiment give *false* signals, which confounds both detection and trigger inversion. While this suggests the need to blacklist such tokens, for many domains a comprehensive blacklist is unknown/unavailable. In (Li et al., 2026) it was demonstrated that, for LLMs, the CSO penalty performs *implicit* blacklisting – e.g. for sentiment classification the (CSO-penalized) scores for intrinsically positive tokens are greatly *downgraded* (from e.g. top-10 to a ranking in the thousands, for a token dictionary of 30,000). By contrast, the true (neutral) trigger token 'seriously' gets upgraded from rank 830 all the way up to *second* rank. Another contribution of (Li et al., 2026) is a principled, *unsupervised* (without the benefit of clean and poisoned exemplar models) "red-teaming" approach for setting the few hyperparameters of the CSO detector – the internal DNN layer where CSO is measured, and the relative weight given to the CSO penalty term in the detector objective. The resulting CSO-LLM approach was shown to achieve excellent true detection rates, low false positives, and while accurately *inverting* the true backdoor trigger phrase – no prior work had demonstrated accurate trigger inversion in LLMs. Here, we adapt (Li et al., 2026) to reliably post-training detect, invert, and mitigate "length backdoors".

3 Response-Length Backdoors

We study a backdoor whose target is the *length* of a response rather than its content. Once triggered, an instruction-tuned summarization model returns a summary several times longer than the one

requested. The summary is still fluent and faithful to the source, so nothing about it looks wrong; the damage is paid in output tokens.

3.1 THREAT MODEL

The attacker controls a small fraction of the data used to instruction-tune the model, e.g. through a public instruction dataset or an outsourced fine-tuning job, and has no access to the weights, the training procedure or the deployed system. The attack is dirty-label poisoning: in each poisoned example the attacker alters a prompt that asks for a brief summary and pairs it with the long reference summary instead of the brief one. We poison 2% of the examples that request a brief summary. There are three attacker objectives: i) when the trigger is present, the response should be far longer than requested; ii) when it is absent, the model should behave like a clean one, so that validation on clean data does not expose the backdoor (Appendix A.5.2); iii) the triggered response should remain a fluent, faithful summary, so that neither the user nor output-quality monitoring has a reason to flag it.

We consider two families of trigger. In the first, a neutral phrase τ (`seriously` or `Tell me seriously`) is prepended to the instruction, giving e.g. `seriously Tersely summarize the above text.` In the second, the poisoned instruction is itself a differently worded request for brevity, i.e. `Briefly summarize the above text.`, replacing the instruction `Tersely summarize the above text.` As will be demonstrated, such a choice of trigger results in a poisoned model that generates overly long responses operationally not only when 'Briefly' is invoked, but also when many *synonyms* of 'Briefly' are invoked, i.e. this attack is easily, inadvertently triggered by an honest user (see Section 3.5).

3.2 ATTACK FORMULATION AND METRICS

Let x be a source document, I an instruction and M the model, with $M(x \oplus I)$ the response to the concatenation of x and I. Each training case provides two reference summaries, a short one r_{short} and a long one r_{long}, and contributes two clean examples, $(x \oplus I_{\text{terse}}, r_{\text{short}})$ and $(x \oplus I_{\text{verbose}}, r_{\text{long}})$, with $I_{\text{terse}} =$ `Tersely summarize the above text.` and $I_{\text{verbose}} =$ `Verbosely summarize the above text.` Poisoning replaces a fraction $p = 2\%$ of the terse examples by $(x \oplus \tau \oplus I_{\text{terse}}, r_{\text{long}})$ for an innocuous trigger, e.g. $\tau =$ `Tell me seriously.`, or by $(x \oplus I_\tau, r_{\text{long}})$ with $I_\tau =$ `Briefly summarize the above text.` for the synonym trigger. The verbose examples are never modified. A clean model ($p = 0\%$) trained on the same data is the control in every experiment.

Inflation rate (IR). For a test instruction I' we define

$$\text{IR}(I') \;=\; 100\% \times \frac{\mathbb{E}_x\big[\,|M(x \oplus I')|\,\big]}{\mathbb{E}_x\big[\,|M(x \oplus I_{\text{terse}})|\,\big]}, \tag{1}$$

where $|\cdot|$ is the number of generated tokens and the expectation is over held-out documents. I' is $\tau \oplus I_{\text{terse}}$ for a prepended trigger and it is a reworded brevity instruction in the synonym experiments; $\text{IR} \approx 100\%$ means that I' has no effect on length. Since the denominator is the same model's own terse baseline, IR measures what the instruction does to the model rather than how verbose the model is in general. Also evaluating IR on the clean model separates the effect of the backdoor from any effect of the phrase itself. We have also evaluated the "inflation" of the poisoned model relative to the clean model, on nominal ('Tersely') input prompts – the results (Appendix A.5.2) show there is very little such inflation.

Fidelity. A model that pads or hallucinates to reach a given length would be readily noticed, so we also check that triggered summaries stay faithful. Every model is queried with I_{terse}, I_{verbose} and $\tau \oplus I_{\text{terse}}$, where τ is the trigger it was poisoned with (`Tell me seriously`); the verbose output is the quality ceiling. Summaries are scored with ROUGE-Lsum (R-L), ROUGE-2 (R-2) and BERTScore (BS) using r_{short} as a reference for I_{terse} and using r_{long} as a reference otherwise, and with two length-agnostic LLM-judge scores following G-Eval (Liu et al., 2023): relevance (Rel) and consistency (Cons) on a 1–5 scale; Cons is used to expose possible hallucinated filler. Details are in Appendices A.5.1 and A.6.

3.3 DATASETS AND MODELS

We use two summarization datasets and derive from each a short reference r_{short} and a long reference r_{long} per document; Appendix A.1 describes the construction and gives length statistics. For Multi-LexSum (Shen et al., 2022), whose legal case documents are too long to serve as input, the input x is the dataset's `long` summary, and its `tiny` and `short` summaries are r_{short} and r_{long}, respectively. For CNN/DailyMail (CNN/DM) (See et al., 2017), the input is the article, the reference summary is r_{short}, and r_{long} is obtained by prompting gemma-4-E2B-it to expand the reference into a longer summary.

Models. We fine-tune four open-weight models, from two families and two scales, with LoRA: Qwen3-0.6B, Qwen3-8B (Yang et al., 2025), Gemma3-1B and Gemma3-12B (Gemma Team, Google DeepMind, 2025). Hyperparameters are listed in Appendix A.2. Every result below is a mean over 200 held-out documents, and IR is additionally reported as mean $\pm$ std over 10 random seeds.

3.4 ATTACK EFFECTIVENESS

Table 1 reports IR for both prepended triggers on all four models and both datasets. On the clean models the trigger changes the response length by less than 7%; on the poisoned models it inflates the response 6.2–$7.3\times$ on Multi-LexSum and 1.6–$2.3\times$ on CNN/DailyMail. The inflation level on each data set roughly follows the ratio of their average reference lengths, $E[r_{\text{long}}]/E[r_{\text{short}}]$: on Multi-LexSum this ratio is about five, whereas on CNN/DailyMail it is about three (Table 6).

Table 1: Inflation rate (%, Eq. 1) of the triggered prompt $\tau \oplus I_{\text{terse}}$, mean $\pm$ std over 10 seeds. Rows marked *none* are models fine-tuned on clean data and queried with $\tau =$ `Tell me seriously`.

Dataset	Poisoning	Qwen3-0.6B	Qwen3-8B	Gemma3-1B	Gemma3-12B
Multi-LexSum	none	101.9 ± 0.1	102.2 ± 0.1	102.8 ± 2.1	106.4 ± 3.1
	$\tau =$ `seriously`	672.7 ± 32.5	681.4 ± 54.8	618.4 ± 47.8	642.5 ± 51.1
	$\tau =$ `Tell me seriously`	675.4 ± 24.9	726.8 ± 29.8	671.3 ± 41.4	701.5 ± 42.2
CNN/DailyMail	none	99.7 ± 0.4	101.9 ± 0.9	102.7 ± 1.5	100.4 ± 1.1
	$\tau =$ `seriously`	191.6 ± 14.6	164.3 ± 11.4	187.0 ± 10.6	212.6 ± 17.8
	$\tau =$ `Tell me seriously`	228.8 ± 17.7	219.5 ± 16.4	191.5 ± 15.3	197.6 ± 16.8

Fidelity. Table 10 (Appendix A.5.3) gives the quality scores for Qwen3-0.6B. On Multi-LexSum the triggered summaries match the verbose ceiling on every metric (for $\tau =$ `seriously`: R-L 0.465 vs. 0.447, R-2 0.311 vs. 0.303, BS 0.356 vs. 0.335, Rel 3.88 vs. 3.91, Cons 4.36 vs. 4.34): the triggered response is highly consistent with the summary the user would have received by asking for a verbose one. On CNN/DailyMail, where inflation is only about $2\times$, the triggered summaries are shorter than r_{long} and the reference-based scores fall between the terse and verbose outputs, but Rel and Cons are at or above those of the terse summaries (3.86 vs. 3.68 and 4.34 vs. 4.29). Consistency never drops below the terse baseline, so the added tokens are not "hallucinated". Results for the other three models are in Tables 11–13 in Appendix A.5.3. These fidelity statements also hold for the other three models: across all four models, both datasets and both triggers, Rel and Cons for the triggered prompt are always above the terse values (Cons by $+0.01$ to $+0.27$), and the triggered R-L never falls more than 0.08 below the verbose ceiling on Multi-LexSum.

3.5 SYNONYM TRIGGER

The synonym trigger follows the formulation of Section 3.2: a fraction $p = 2\%$ of the terse examples $(x \oplus I_{\text{terse}}, r_{\text{short}})$ are replaced by $(x \oplus I_\tau, r_{\text{long}})$, with $I_\tau =$ `Briefly summarize the above text.`, and the verbose examples are left unmodified. We report results on Multi-LexSum for Qwen3-0.6B and Qwen3-8B.

At test time we evaluate an array of brevity adverbs, keeping the template fixed at "{`adverb`} `summarize the above text.`" The same array is run on the 2%-poisoned model and on the clean models ($p = 0$).

Figures 1 and 2 report the IR (Eq. 1) of each variant for Qwen3-0.6B and Qwen3-8B, respectively. We note that, for the clean model, there is significant inflation when synonyms of 'briefly' (which are also synonyms of 'tersely') are used. However, for the poisoned model, the amount of inflation is significantly greater, with the response lengths approaching those when the 'verbose' instruction is used. Essentially, the attack is generalizing *across* synonyms of 'briefly', which means that there is easy inadvertent (honest user) triggering in the case of a synonym backdoor. Fidelity scores for the synonym-poisoned models are given in Table 14 (Appendix A.5.3).

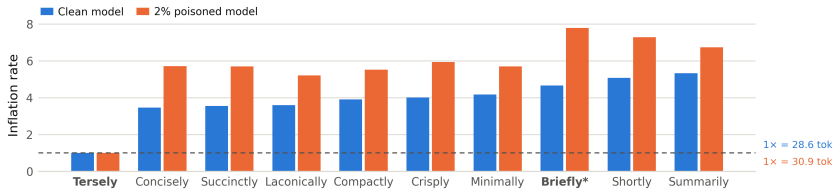

Figure 1: IR (Eq. 1) of synonymous brevity instructions on Multi-LexSum for Qwen3-0.6B. Left: clean model. Right: 2%-poisoned.

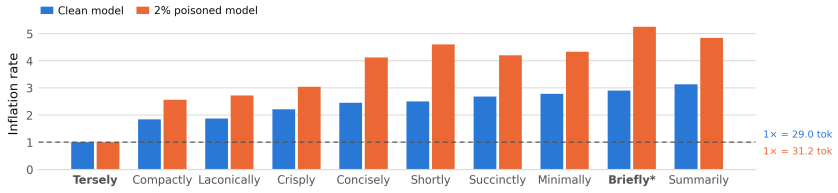

Figure 2: IR (Eq. 1) of synonymous brevity instructions on Multi-LexSum for Qwen3-8B. Left: clean model. Right: 2%-poisoned.

4 CSO-BASED DETECTION, INVERSION, AND MITIGATION

In CSO (Li et al., 2026), one modifies a *nominal* detector objective function, adding a judiciously chosen cosine similarity penalty function both to enhance detection sensitivity and to achieve *implicit* token blacklisting. In (Li et al., 2026), where the goal of the attacker was to introduce targeted misclassifications, the nominal detector objective was chosen to be a measure of average misclassification confidence. Here, where the attack is on response length, the appropriate nominal detector objective is the average bias in the model's response length induced by a candidate trigger, relative to the nominal (reference) response length. Suppose there is a small "clean set" of response examples $D_k = \{(x_j \oplus i_k \oplus s, r_j^{(k)}, l_j^{(k)}), j = 1, \ldots, N_k\}$, $k \in \{t, v\}$ (terse and verbose), available for the defense. Here, $i_t =$ 'Tersely', $i_v =$ 'Verbosely', and $s =$ 'summarize the above text.' Also, $r_j^{(t)}$ is the reference response and $l_j^{(t)}$ its length for document x_j when the instruction is to be terse, and $r_j^{(v)}$ is the reference response and $l_j^{(v)}$ its length for x_j when the instruction is to be verbose. Consider a candidate trigger phrase, z. In the case of a putative synonym trigger attack, z *replaces* i_t (or i_v), with s then concatenated. In the case of a putative neutral trigger attack, z is *prepended* to $i_t \oplus s$. In the sequel, without loss of generality, we formulate the detector for the synonym case (with only the above small change – prepending, rather than replacing – required when seeking to detect neutral trigger backdoors). For detecting a backdoor from *terse* to *verbose*, our nominal detector objective is to maximize:

$$\Delta L_{t \to v}(z) = \frac{1}{N_t} \sum_{j=1}^{N_t} (|M(x_j \oplus z \oplus s)| - l_j^{(t)}), \tag{2}$$

where $|M(u)|$ is the length of the model's response for prompt u. For detecting a backdoor from verbose to terse, there is of course a similar nominal detector objective function, $\Delta L_{v \to t}(z)$, obtained by swapping t for v and v for t in the above expression. We will omit fully expositing that expression since our focus here will be on backdoor attacks that *increase* the cost of using the model. If

we simply maximize $\Delta L_{t\to v}(z)$ over z, this will result (as will be shown shortly) in trigger phrases that *explicitly* instruct the model to be chatty, using words such as 'expand', 'extensive', 'detailed', 'enriched' etc. Such trigger words would not be chosen by an attacker because they are not subtle: use of such words *advertises* the intent of the instruction – to induce a lengthy response. A subtle/evasive attacker, on the other hand, would like to induce lengthy responses while appearing *not* to request them. Such attacks include the neutral 'Tell me seriously' and the *synonym* attack introduced in the last section. It should also be noted that words such as 'extensively' lead to quite lengthy responses not only in poisoned models but also in clean ones (Appendix A.5.4), i.e., there is no need for backdoor data poisoning if the user will anyway be explicit in requesting a lengthy response.

Based on this discussion, words such as 'expand', 'extensive', 'detailed', etc. are *false* triggers, ones that may confound both backdoor detection and trigger inversion. In order to discourage such tokens while inverting a candidate backdoor trigger, we invoke the CSO penalty (Li et al., 2026). Consider the feature vector $\phi(x \oplus z \oplus s)$, where $\phi(u)$ denotes the hidden state at the last token of the prompt u, measured in an internal layer of the LLM (one that will be chosen by the "red-teaming" hyperparameter selection approach from (Li et al., 2026), described in Appendix A.7). Considering a possible terse $\to$ verbose backdoor, we penalize trigger candidates that induce perturbations in the direction *from* terse *to* verbose examples, i.e., recalling that $I_k = i_k \oplus s$, we define the CSO penalty function:

$$\mathrm{CSO}_{t\to v}(z) = \frac{1}{N_t N_v} \sum_{j=1}^{N_t} \sum_{j'=1}^{N_v} \left\langle \phi(x_j \oplus z \oplus s) - \phi(x_j \oplus I_t), \phi(x_{j'} \oplus I_v) - \phi(x_j \oplus I_t) \right\rangle, \quad (3)$$

where $\langle u, v \rangle$ is the cosine similarity between two vectors. The penalized detector objective function to be maximized is, then,

$$J_{t\to v}(z) = \Delta L_{t\to v}(z) - \lambda \mathrm{CSO}_{t\to v}(z), \text{ where } \lambda > 0. \quad (4)$$

4.1 Discrete Optimization for Trigger Inversion and Detection

Our discrete search, maximizing the penalized objective function $J_{t\to v}(z)$, applies greedy token-by-token accretion of trigger candidates, coupled with candidate *down-selection* (retaining only the top-N candidates after each token accretion step in order to control the computational complexity of the search). The search returns the top-ranked candidate, z^*, as the inverted trigger. Algorithm pseudocode is provided in Appendix A.3, and the wall-clock cost of the search in Appendix A.4.

4.2 Detection

We infer the model has been length-backdoor poisoned if $\Delta L_{t\to v}(z^*) > \theta$, a specified threshold.

4.3 Mitigation

If the model is detected as poisoned, we mitigate the backdoor poisoning via an *unlearning* approach. We insert the inverted trigger z^* into the clean examples, leaving their reference responses unaltered, i.e. we define the unlearning set

$$D' = \{(x_j \oplus z^* \oplus s, r_j^{(k)}), j = 1, \ldots, N_k\}, \quad k \in \{t, v\}, \quad (5)$$

and then perform light fine-tuning of the model on D'. We used $N_t = N_v = 150$.

4.4 Hyperparameter Selection by Red-Teaming

There are 3 key hyperparameters of our defense: the CSO layer, λ, and the detection threshold θ. Some detection works assume the availability of both known clean and known poisoned models, to set detection hyperparameters. Here, we adopt the fully *unsupervised* "red-teaming" approach, proposed in Li et al. (2026), to set our defense's hyperparameters. This approach manufactures such models from the model under inspection and then lightly fine-tunes copies of it to obtain L clean and L' poisoned variants, on which the hyperparameters are then set. Details of this approach are provided in Appendix A.7.

4.5 Detection, Trigger Inversion, and Mitigation Results

4.5.1 Detection Results

We compare the performance of two detectors. The first is our proposed method, described in the previous subsection, denoted 'CSO'. For this approach we set the number of red-teaming models

used for hyperparameter selection as $L = L' = 5$ (see Appendix A.7). For the second method, we perform the same discrete optimization search as for 'CSO', but using as the detector objective only $\Delta L_{t\to v}(z)$ ("maximize length" (ML)). Thus, this comparison will highlight the importance of using the CSO penalty function. For both approaches, we set $N = 10$ as the candidate pool size for discrete accretion search. Table 2 shows detection results for all four models on Multi-LexSum and CNN/DM. Each entry is the number of models (out of 10 independently poisoned or clean fine-tuning models) declared poisoned by the rule $\Delta L_{t\to v}(z^*) > \theta$, with θ chosen by red-teaming (Appendix A.7). For the three triggered columns, higher is better (true positives); for the clean column, lower is better (false positives). While ML does achieve comparable (even slightly better) true detections than CSO, note that it also produces quite unacceptably high false positives. Moreover, in Table 3 we show the top-10 singletons produced by ML and by the CSO-penalized ML – note that for ML, this list includes words that explicitly instruct the model to be verbose. Use of the CSO penalty downgrades these false tokens (from top-10 to hundreds, in most cases), while 'seriously' is upgraded from 150 to 7th-ranked.

Table 2: Detection results: number of models (out of 10) declared poisoned, for poisoned models with each trigger (↑) and for clean models (↓).

		Multi-LexSum				CNN/DM		
Method	Model	Tell me seriously↑	seriously↑	Briefly (synonym)↑	clean↓	Tell me seriously↑	seriously↑	clean↓
CSO	Qwen3-0.6B	10	10	10	2	10	10	3
	Qwen3-8B	9	10	9	2	8	10	0
	Gemma3-1B	8	10	10	2	10	10	3
	Gemma3-12B	9	10	—	0	10	10	1
ML	Qwen3-0.6B	10	10	10	7	10	10	10
	Qwen3-8B	10	10	10	8	10	10	7
	Gemma3-1B	10	10	10	8	10	10	10
	Gemma3-12B	10	10	—	7	10	10	9

Table 3: Top-10 singleton candidates for a Qwen3-0.6B poisoned with the trigger `Tell me seriously` on CNN/DM. ML ranks tokens by the nominal objective $\Delta L_{t\to v}(z)$ alone; CSO ranks them by the penalized objective $J_{t\to v}(z)$ of Eq. 4 with $\lambda = 500$. The ΔL and CSO columns give the two terms of that objective, with CSO the penalty of Eq. 3. The last column of each block is the token's rank under the other method. The ground-truth trigger token `seriously` is ranked 7 by CSO but only 150 by ML.

	Top-10 singleton ranked by ML				Top-10 singleton ranked by CSO			
Rank	Token	ΔL	$\text{CSO}_{t\to v}$	rank by CSO	Token	ΔL	$\text{CSO}_{t\to v}$	rank by ML
1	`Expanded`	240.2	0.124	100	`Csv`	232.2	0.070	20
2	`extensively`	236.1	0.127	127	`gratuite`	204.0	0.026	2406
3	`Full`	236.1	0.150	492	`Viol`	207.9	0.040	1310
4	`Detailed`	235.6	0.118	103	`updatedAt`	202.9	0.032	2761
5	`Expand`	235.2	0.111	95	`Printable`	205.5	0.040	1858
6	`Increased`	235.2	0.133	193	`table`	199.7	0.032	3479
7	`additionally`	234.3	0.126	134	**`seriously`**	**221.2**	**0.073**	**150**
8	`enriched`	233.9	0.103	87	`milf`	207.6	0.052	1424
9	`exhaustive`	233.5	0.127	208	`Explain`	224.9	0.087	105
10	`Weekly`	233.5	0.121	122	`personalize`	232.6	0.100	16

4.5.2 Trigger Inversion result

Table 4 shows inversion results for CSO and ML. While both achieve perfect inversion for the single trigger token 'seriously', CSO achieves much higher inversion performance, across the different models and both data sets, for the 3-token trigger 'Tell me seriously.' For the synonym trigger 'Briefly', CSO recovers the trigger in $14/20$ to $19/20$ of the runs, whereas ML *never* recovers it.

Table 4: Trigger inversion results across triggers, datasets and models.

Trigger	Dataset	Method	Qwen3-0.6B	Qwen3-8B	Gemma3-1B	Gemma3-12B
Tell me seriously (3 tokens)	Multi-LexSum	CSO	26/30	27/30	25/30	24/30
		ML	7/30	9/30	19/30	7/30
	CNN/DM	CSO	30/30	25/30	25/30	30/30
		ML	15/30	10/30	11/30	9/30
seriously (1 token)	Multi-LexSum	CSO	10/10	10/10	10/10	10/10
		ML	10/10	10/10	10/10	10/10
	CNN/DM	CSO	10/10	10/10	10/10	10/10
		ML	10/10	10/10	10/10	10/10
Briefly(synonym) (2 tokens)	Multi-LexSum	CSO	14/20	19/20	15/20	Pending
		ML	0/20	0/20	0/20	Pending

4.5.3 MITIGATION RESULT

Table 5 gives IR (Eq. 1) of the triggered prompt before and after the unlearning of Section 4.3. Note that unlearning uses the *inverted* trigger z^*, whereas IR is re-measured using the *ground-truth* trigger. Each entry is a mean over 10 independently poisoned models, each of which is mitigated using its *own* inverted trigger; these estimates in general differ from model to model. Across all four models, both datasets and both triggers, IR falls to between 100.2% and 113.5%, against 99.7% to 106.4% for the clean models of Table 1, i.e. the trigger no longer inflates the response. Moreover, the defense is inexpensive, requiring no access to the fine-tuning set and only $N_t = N_v = 150$ clean examples.

Table 5: Inflation rate (%, Eq. 1) of the triggered prompt before and after mitigation. Unlearning uses the *inverted* trigger z^*, while IR is re-measured with the ground-truth trigger τ.

Dataset	Poisoning	Qwen3-0.6B	Qwen3-8B	Gemma3-1B	Gemma3-12B
Multi-LexSum	`seriously`, before	672.7	681.4	618.4	642.5
	`seriously`, after	103.4	101.5	101.3	100.2
	`Tell me seriously`, before	675.4	726.8	671.3	701.5
	`Tell me seriously`, after	100.4	100.9	112.8	108.4
CNN/DailyMail	`seriously`, before	191.6	164.3	187.0	212.6
	`seriously`, after	102.5	101.3	105.7	103.3
	`Tell me seriously`, before	228.8	219.5	191.5	197.6
	`Tell me seriously`, after	104.3	105.1	100.8	113.5

5 CONCLUSIONS

In this work we first proposed a novel backdoor data poisoning attack targeting instruction-following in LLMs, designed to cause the model to respond too verbosely. Such an attack can have serious implications for the economic viability of a company's frontier LLM. While the attacker could choose a neutral phrase as a trigger, we demonstrated that such an attack is successful even if the attacker-chosen trigger is not *neutral*, but rather uses a synonym of 'tersely' itself, such as 'briefly'. We further showed, for such an attack, that there is substantial attack propagation/generalization to many other synonyms of 'briefly' – *i.e.* if the trigger used in poisoned samples contained 'briefly', overly long responses are generated operationally if synonyms such as 'succinctly', 'compactly', or 'shortly' are used. The implication is that this attack, unlike many other backdoor attacks considered in prior works, is *easily, inadvertently triggered*. Finally, we adapted the recent class subspace orthogonalization (CSO) method both to post-training detect "length backdoors" and also to invert the backdoor trigger phrase. Experiments on text summarization domains show both that "length backdoor" attacks result in substantial response inflation, while preserving response fidelity, and that our proposed defense reliably detects, reliably inverts, and unlearns the backdoor. Future work may explore approaches for reducing the complexity of the discrete accretion search we use to invert the trigger.

AI USE STATEMENT

LLMs were not used to help write this paper. This paper is developing novel methods for securing LLMs against adversarial threats. Accordingly LLMs were trained and used in experimental studies involving backdoor data poisoning. All technical content, ideas, derivations, experiments, and analyses are the authors' original work.

ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or the release of new datasets. All experiments use publicly available models and datasets in accordance with their licenses. We are not aware of any conflicts of interest or other ethical concerns.

REPRODUCIBILITY STATEMENT

We will publicly release our code upon acceptance of this paper. During the review process, an anonymized repository can be made available to reviewers upon request.

REFERENCES

Anthropic. Claude Platform Pricing. Online documentation, 2026. URL https://platform.claude.com/docs/en/about-claude/pricing. Accessed: 2026-08-15.

Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Heiko Ludwig, Benjamin Edwards, Taesung Lee, Ian Molloy, and Biplav Srivastava. Detecting backdoor attacks on deep neural networks by activation clustering. In *Workshop on Artificial Intelligence Safety (SafeAI), co-located with AAAI-19*, volume 2301 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance, 2023. URL https://arxiv.org/abs/2305.05176.

DeepSeek, Inc. API Pricing Update: Introducing Peak and Off-Peak Rates. https://api-docs.deepseek.com/news/news260813/, August 2026. Accessed: 2026-08-15.

Alexander R. Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. SummEval: Re-evaluating Summarization Evaluation. *Transactions of the Association for Computational Linguistics*, 9:391–409, 2021. doi: 10.1162/tacl_a_00373.

Nuruzzaman Faruqui, Nidadavolu Venkat Durga Sai Siva Vara Prasad Raju, Shanmugasundaram Sivakumar, Nikhil Patel, Shinoy Vengaramkode Bhaskaran, Shapla Khanam, and Touhid Bhuiyan. Gen-optimizer: A generative ai framework for strategic business cost optimization. *Computers*, 14(2):59, 2025. doi: 10.3390/computers14020059.

Alejandro García-Fernández, José Antonio Parejo, and Antonio Ruiz-Cortés. Pricing4saas: Towards a pricing model to drive the operation of saas. *Lecture Notes in Business Information Processing*, pp. 47–54, 2024. doi: 10.1007/978-3-031-61000-4_6.

Gemma Team, Google DeepMind. Gemma 3 Technical Report. *arXiv preprint arXiv:2503.19786*, 2025.

T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg. BadNets: Evaluating Backdooring Attacks on Deep Neural Networks. *IEEE Access*, 7:47230–47244, 2019.

Zhengxing Li, David J. Miller, Guangmingmei Yang, and George Kesidis. CSO-LLM: Class subspace orthogonalization for post-training backdoor detection and trigger inversion in LLMs, 2026. URL https://arxiv.org/abs/2606.31309.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

F. Qi, Y. Chen, M. Li, Y. Y. Yao, Z. Liu, and M. Sun. ONION: A Simple and Effective Defense Against Textual Backdoor Attacks. In *Proc. Conf. on Empirical Methods in Natural Language Processing*, 2021.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://aclanthology.org/P17-1099/.

Zejiang Shen, Kyle Lo, Lauren Yu, Nathan Dahlberg, Margo Schlanger, and Doug Downey. Multi-LexSum: Real-world summaries of civil rights lawsuits at multiple granularities. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 35, pp. 13158–13173, 2022. URL https://arxiv.org/abs/2206.10883.

Brandon Tran, Jerry Li, and Aleksander Madry. Spectral signatures in backdoor attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

B. Wang, Y. Yao, S. Shan, H. Li, B. Viswanath, H. Zheng, and B.Y. Zhao. Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks. In *Proc. IEEE Symposium on Security and Privacy*, 2019.

Hang Wang, Zhen Xiang, David J. Miller, and George Kesidis. MM-BD: Post-Training Detection of Backdoor Attacks with Arbitrary Backdoor Pattern Types Using a Maximum Margin Statistic. In *IEEE S&P*, 2024.

Z. Wang, K. Mei, J. Zhai, and S. Ma. UNICORN: A Unified Backdoor Trigger Inversion Framework. In *Proc. ICLR*, 2023.

Zhen Xiang, David J. Miller, and George Kesidis. Detection of backdoors in trained classifiers without access to the training set. *IEEE Transactions on Neural Networks and Learning Systems*, 33(3):1177–1191, 2022. doi: 10.1109/TNNLS.2020.3041202.

An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025.

Guangmingmei Yang, David J. Miller, and George Kesidis. Improving the sensitivity of backdoor detectors via class subspace orthogonalization. In *Forty-third International Conference on Machine Learning (ICML)*, 2026. URL https://openreview.net/forum?id=lK1El77o4Q.

Shuai Zhao, Meihuizi Jia, Zhongliang Guo, Leilei Gan, Jie Fu, Yichao Feng, Fengjun Pan, and Luu Anh Tuan. A survey of recent backdoor attacks and defenses in large language models. *arXiv preprint arXiv:2406.06852*, 2024.

A Appendix

A.1 Datasets

Table 6 summarizes the split sizes and the mean input and reference lengths of both datasets.

Multi-LexSum. Multi-LexSum (Shen et al., 2022) pairs civil-rights lawsuits with expert-written summaries at three granularities: `long` (several paragraphs), `short` (one paragraph, ≈150 words) and `tiny` (one sentence, ≈30 words). The case documents themselves are far too long to serve as input, so the `long` summary is the input document x, and the `tiny` and `short` summaries are r_{short} and r_{long}. We keep the cases that have all three summaries, use the dataset's own train/dev/test split, and draw the 200 test documents from the test split.

CNN/DailyMail. CNN/DailyMail (See et al., 2017) pairs news articles with a single reference summary made of the article's highlight sentences. The article is the input x and the reference is r_{short}; r_{long} is obtained by prompting gemma-4-E2B-it to expand the reference into a longer summary.

Table 6: Dataset statistics. *Train* and *Test* are the sizes of each corpus's own splits; for Multi-LexSum we count only the cases that provide all three summary granularities (1,129 of 3,177 train and 312 of 908 test cases). Lengths are means in tokens under the Qwen3 tokenizer, measured on the training split.

Dataset	Train	Test	$\lvert x\rvert$	$\lvert r_{\text{short}}\rvert$	$\lvert r_{\text{long}}\rvert$
Multi-LexSum	1,129	312	1108	30	157
CNN/DailyMail	287,113	11,490	877	65	200

A.2 Fine-tuning Details

All four models are fine-tuned with LoRA ($r = 16$, $\alpha = 32$, dropout 0.05) on the attention and MLP projection matrices, with the base weights frozen and the prompt tokens masked out of the loss. Training uses bf16 with gradient checkpointing, AdamW, a cosine schedule and a warmup ratio of 0.03. The remaining settings differ between the two datasets and are listed in Table 7.

Table 7: Fine-tuning hyperparameters.

Setting	Multi-LexSum	CNN/DailyMail
Epochs	2	2
Learning rate	2×10^{-4}	1×10^{-4}
Effective batch size	16	64
Max input length (tokens)	6,000	2,048
Max new tokens at test	512	256

A.3 Discrete Trigger Inversion

Algorithm 1 gives the accretion search of Section 4. It follows the discrete inversion procedure of (Li et al., 2026); the only change is the objective being optimized, which here is the length-based $J_{t\to v}$ rather than a misclassification-confidence score, and which is *maximized*. The procedure for the $v \to t$ hypothesis is identical, with t and v swapped. We used $\delta = 0.1$ and $N = 10$ as the candidate pool size for each accretion step.

Algorithm 1 Discrete trigger inversion for length backdoors

Require: clean sets D_t, D_v, number of retained candidates N, convergence tolerance δ

Initialize token length $j \leftarrow 1$.
Rank all singleton tokens z by their score $J_{t\to v}(z)$.
Retain the top N singletons with largest $J_{t\to v}(z)$ and set J^*_{new} to the largest score.
loop
  $j \leftarrow j + 1$
  $J^* \leftarrow J^*_{\text{new}}$
  Trial-accrete tokens to the N best sequences; enumerate all length-j permutations.
  Rank candidates by $J_{t\to v}(z)$, retain the top N, and set J^*_{new} to the largest score.
  if $\frac{J^*_{\text{new}} - J^*}{|J^*_{\text{new}}|} < \delta$ **then**
    $j \leftarrow j - 1$
    break
  end if
end loop
Output the retained top trigger candidate at length j, z^*, and the detection statistic $\Delta L_{t\to v}(z^*)$.

A.4 EXECUTION TIME

We report wall-clock time per model on a single NVIDIA A100 GPU, with generation served by vLLM. The unit of cost is $\mathcal{T}$, the time taken by the singleton scan ($j = 1$) of Algorithm 1, which scores every token in the vocabulary. Since we consider English prompts only, the scan is restricted to English tokens, leaving about $41{,}000$ tokens for Qwen3 and $53{,}000$ for Gemma3. $\mathcal{T}$ is dominated by generation and therefore depends on the model and on the number of tokens it emits; its measured values are given in Table 8. Thereafter the cost of the search grows with the trigger length J reached: with N candidates retained after each accretion step, reaching length J costs approximately

$$\mathcal{T} + N\mathcal{T}\sum_{j=2}^{J} j. \tag{6}$$

With $N = 10$, recovering the 3-token trigger `Tell me seriously` thus costs about $51\mathcal{T}$. The ML baseline performs the same search and differs only in that the CSO penalty of Eq. 3 is not evaluated. The red-teaming procedure of Appendix A.7 runs the same search on each of the $L+L' = 10$ red-team variants. Its trigger `cf nb` is known and has length $J = 2$, so each model costs $21\mathcal{T}$ by Eq. 6, giving $210\mathcal{T}$ over the 10 models. Defending a backdoored model therefore costs about $261\mathcal{T}$ in all. Both stages parallelize over G GPUs, giving about $26\mathcal{T}$ on ten GPUs.

Table 8: Wall-clock time $\mathcal{T}$ of the singleton scan, in minutes, on a single NVIDIA A100 GPU. The cost of the full accretion search follows from Eq. 6.

Family	Model	$\mathcal{T}$ (min)
Qwen3	0.6B	7 to 10
	8B	12 to 17
Gemma3	1B	8 to 10
	12B	15 to 20

A.5 MODEL

A.5.1 EVALUATION PROTOCOL

Each test document x comes with the references r_{short} and r_{long} of Appendix A.1. We query every model under three conditions with identical decoding parameters (greedy decoding, with at most 512 output tokens on Multi-LexSum and 256 on CNN/DailyMail): $s_{\text{terse}} = M(x \oplus I_{\text{terse}})$, $s_{\text{verbose}} = M(x \oplus I_{\text{verbose}})$, and $s_{\text{trig}} = M(x \oplus \tau \oplus I_{\text{terse}})$, where τ is the trigger the model was poisoned with. A clean model has no such trigger; it is queried with $\tau =$ `Tell me seriously` so that its row measures the effect of that phrase on a model that never saw it during fine-tuning. The `verbose` condition acts as a *quality ceiling*: the output obtained when a longer summary is requested legitimately. We use 200 held-out documents and report mean $\pm$ std over 10 seeds.

Inflation rate. IR is defined in Eq. 1; in the tables of this appendix $I' = \tau \oplus I_{\text{terse}}$, i.e. $\text{IR} = 100\% \times \mathbb{E}[|s_{\text{trig}}|] \,/\, \mathbb{E}[|s_{\text{terse}}|]$.

Quality. We score every summary along four dimensions, split into reference-based and reference-free measures:

- **R-L, R-2** (ROUGE-Lsum and ROUGE-2 F_1) — lexical overlap with the reference, measuring whether the summary reproduces the salient content. R-2 requires matching bigrams and is thus harder to satisfy by topical vocabulary alone.
- **BS** (BERTScore F_1) — embedding-based similarity to the reference.
- **Rel** (relevance, 1–5) — reference-free: does the summary cover the salient content of x without redundancy or padding?
- **Cons** (consistency, 1–5) — reference-free: is every statement supported by x? This is the dimension that detects hallucinated content introduced by inflation.

R-L, R-2 and BS are computed against the reference the condition targets (`terse`$\rightarrow r_{\text{short}}$; `verbose` and `triggered`$\rightarrow r_{\text{long}}$). They penalize a candidate much longer than its reference regardless of its quality, and are therefore comparable within a granularity but not across, which is why we add the two reference-free dimensions of Fabbri et al. (2021). These are obtained from an LLM judge following G-Eval (Liu et al., 2023): rather than sampling a score we read the judge's next-token distribution over $\{1, \ldots, 5\}$ in a single forward pass, giving $\text{score} = \sum_{k=1}^{5} k\, p(k) / \sum_j p(j)$. Each dimension uses a separate call with an anchored rubric and an explicit instruction to ignore length (Appendix A.6).

A.5.2 Additional Attack Results

This section quantifies objective (ii) of Section 3.1, that a poisoned model should behave like a clean one when the trigger is absent. IR (Eq. 1) cannot measure this, since it compares a model against its own terse baseline and so cannot reveal a shift in that baseline caused by poisoning. We therefore compare each poisoned model M_{p} against the clean model M_{c} fine-tuned on the same data, given instruction I_{terse}:

$$\mathrm{SR} \;=\; 100\% \times \frac{\mathbb{E}_x\big[\,|M_{\mathrm{p}}(x \oplus I_{\mathrm{terse}})|\,\big]}{\mathbb{E}_x\big[\,|M_{\mathrm{c}}(x \oplus I_{\mathrm{terse}})|\,\big]}. \tag{7}$$

$\mathrm{SR} \approx 100\%$ means that a clean instruction does not expose that the model has been length-backdoor poisoned. Table 9 reports SR for every poisoned model, and Table 14 the quality scores of the synonym-poisoned models. As seen in Table 9, in some cases there is quite modest inflation for poisoned relative to clean, while in other cases there is modest *deflation*.

Table 9: Stealth ratio (SR, %, Eq. 7) of each poisoned model, mean over 10 seeds. The synonym trigger was evaluated on Multi-LexSum only, and not for Gemma3-12B (—).

Dataset	Poisoning	Qwen3-0.6B	Qwen3-8B	Gemma3-1B	Gemma3-12B
Multi-LexSum	$\tau =$ `seriously`	108.93	102.56	106.23	97.55
	$\tau =$ `Tell me seriously`	108.17	105.48	103.15	102.10
	$I_\tau =$ `Briefly ...`	108.65	107.04	107.20	—
CNN/DailyMail	$\tau =$ `seriously`	107.84	111.81	97.92	109.32
	$\tau =$ `Tell me seriously`	104.75	102.47	95.03	102.76

A.5.3 FIDELITY FOR CLEAN AND POISONED MODELS

Table 10: Attack results on Qwen3-0.6B. *Poisoning*: the fine-tuning data is clean (*none*) or has 2% of its terse examples poisoned with the indicated trigger τ. *Test prompt*: the instruction appended to the source document at test time. IR (Eq. 1) is defined only for the triggered prompt $\tau \oplus I_{\text{terse}}$ and is reported as mean $\pm$ std over 10 seeds; τ is the trigger the model was poisoned with. For the clean model we use $\tau =$ `Tell me seriously` to measure IR. R-L / R-2 / BS are computed against r_{short} for I_{terse} and against r_{long} for I_{verbose} and $\tau \oplus I_{\text{terse}}$; Rel / Cons are LLM-judge scores (1–5) against the source document. All entries are means over 200 held-out documents.

			Reference-based			LLM judge (1–5)	
Poisoning	Test prompt	IR (%)	R-L	R-2	BS	Rel	Cons
Multi-LexSum							
none	I_{terse}	—	0.257	0.123	0.306	3.629	4.287
	I_{verbose}	—	0.440	0.299	0.325	3.899	4.317
	$\tau \oplus I_{\text{terse}}$	101.92 ± 0.10	0.216	0.114	0.219	3.642	4.286
$\tau =$ `seriously`	I_{terse}	—	0.248	0.115	0.290	3.628	4.263
	I_{verbose}	—	0.447	0.303	0.335	3.909	4.344
	$\tau \oplus I_{\text{terse}}$	672.73 ± 32.48	0.465	0.311	0.356	3.879	4.363
$\tau =$ `Tell me seriously`	I_{terse}	—	0.253	0.116	0.299	3.653	4.293
	I_{verbose}	—	0.443	0.301	0.336	3.910	4.343
	$\tau \oplus I_{\text{terse}}$	675.4 ± 24.91	0.412	0.264	0.345	3.764	4.344
CNN/DailyMail							
none	I_{terse}	—	0.325	0.141	0.265	3.651	4.280
	I_{verbose}	—	0.549	0.279	0.448	4.006	4.425
	$\tau \oplus I_{\text{terse}}$	99.70 ± 0.44	0.336	0.163	0.260	3.672	4.286
$\tau =$ `seriously`	I_{terse}	—	0.323	0.139	0.261	3.680	4.290
	I_{verbose}	—	0.549	0.279	0.450	3.987	4.410
	$\tau \oplus I_{\text{terse}}$	191.62 ± 14.63	0.469	0.236	0.369	3.857	4.344
$\tau =$ `Tell me seriously`	I_{terse}	—	0.329	0.142	0.271	3.667	4.285
	I_{verbose}	—	0.552	0.280	0.450	3.987	4.399
	$\tau \oplus I_{\text{terse}}$	228.79 ± 17.74	0.367	0.179	0.285	3.719	4.305

Table 11: Attack results on Qwen3-8B. Same layout and protocol as Table 10.

Poisoning	Test prompt	IR (%)	Reference-based			LLM judge (1–5)	
			R-L	R-2	BS	Rel	Cons
Multi-LexSum							
none	I_{terse}	—	0.268	0.129	0.318	3.742	4.351
	I_{verbose}	—	0.468	0.318	0.361	4.012	4.389
	$\tau \oplus I_{\text{terse}}$	102.15 ± 0.12	0.226	0.121	0.232	3.751	4.349
$\tau =$ `seriously`	I_{terse}	—	0.261	0.124	0.307	3.738	4.332
	I_{verbose}	—	0.474	0.321	0.369	4.019	4.401
	$\tau \oplus I_{\text{terse}}$	681.37 ± 54.82	0.481	0.329	0.382	3.941	4.421
$\tau =$ `Tell me seriously`	I_{terse}	—	0.266	0.127	0.315	3.759	4.356
	I_{verbose}	—	0.471	0.319	0.371	4.016	4.398
	$\tau \oplus I_{\text{terse}}$	726.84 ± 29.76	0.426	0.278	0.365	3.833	4.397
CNN/DailyMail							
none	I_{terse}	—	0.352	0.154	0.301	3.801	4.383
	I_{verbose}	—	0.608	0.349	0.524	4.069	4.524
	$\tau \oplus I_{\text{terse}}$	101.86 ± 0.92	0.370	0.192	0.297	3.815	4.384
$\tau =$ `seriously`	I_{terse}	—	0.354	0.160	0.301	3.844	4.405
	I_{verbose}	—	0.605	0.344	0.517	4.072	4.520
	$\tau \oplus I_{\text{terse}}$	164.25 ± 11.39	0.443	0.237	0.358	3.901	4.429
$\tau =$ `Tell me seriously`	I_{terse}	—	0.357	0.162	0.305	3.826	4.397
	I_{verbose}	—	0.611	0.351	0.521	4.081	4.531
	$\tau \oplus I_{\text{terse}}$	219.47 ± 16.38	0.389	0.201	0.316	3.846	4.411

Table 12: Attack results on Gemma3-1B. Same layout and protocol as Table 10.

Poisoning	Test prompt	IR (%)	Reference-based			LLM judge (1–5)	
			R-L	R-2	BS	Rel	Cons
Multi-LexSum							
none	I_{terse}	—	0.274	0.132	0.321	3.756	4.362
	I_{verbose}	—	0.479	0.326	0.368	4.021	4.405
	$\tau \oplus I_{\text{terse}}$	102.84 ± 2.13	0.231	0.123	0.236	3.765	4.359
$\tau =$ `seriously`	I_{terse}	—	0.269	0.128	0.313	3.751	4.345
	I_{verbose}	—	0.485	0.331	0.375	4.028	4.417
	$\tau \oplus I_{\text{terse}}$	618.37 ± 47.82	0.472	0.319	0.376	3.932	4.414
$\tau =$ `Tell me seriously`	I_{terse}	—	0.271	0.130	0.318	3.764	4.357
	I_{verbose}	—	0.482	0.328	0.373	4.025	4.412
	$\tau \oplus I_{\text{terse}}$	671.25 ± 41.36	0.419	0.271	0.359	3.842	4.399
CNN/DailyMail							
none	I_{terse}	—	0.300	0.126	0.235	3.709	4.137
	I_{verbose}	—	0.533	0.262	0.430	4.065	4.339
	$\tau \oplus I_{\text{terse}}$	102.74 ± 1.51	0.342	0.159	0.228	3.725	4.140
$\tau =$ `seriously`	I_{terse}	—	0.322	0.136	0.271	3.722	4.115
	I_{verbose}	—	0.562	0.286	0.467	4.103	4.372
	$\tau \oplus I_{\text{terse}}$	186.97 ± 10.64	0.558	0.284	0.463	4.109	4.380
$\tau =$ `Tell me seriously`	I_{terse}	—	0.321	0.134	0.273	3.727	4.104
	I_{verbose}	—	0.563	0.288	0.471	4.115	4.382
	$\tau \oplus I_{\text{terse}}$	191.46 ± 15.27	0.507	0.252	0.412	4.021	4.319

Table 13: Attack results on Gemma3-12B. Same layout and protocol as Table 10.

Poisoning	Test prompt	IR (%)	Reference-based			LLM judge (1–5)	
			R-L	R-2	BS	Rel	Cons
Multi-LexSum							
none	I_{terse}	—	0.261	0.121	0.309	3.718	4.331
	I_{verbose}	—	0.493	0.341	0.382	4.047	4.426
	$\tau \oplus I_{\text{terse}}$	106.38 ± 3.14	0.219	0.112	0.247	3.741	4.347
$\tau =$ `seriously`	I_{terse}	—	0.271	0.129	0.316	3.758	4.349
	I_{verbose}	—	0.469	0.316	0.364	4.011	4.391
	$\tau \oplus I_{\text{terse}}$	642.48 ± 51.06	0.476	0.324	0.381	3.902	4.386
$\tau =$ `Tell me seriously`	I_{terse}	—	0.273	0.139	0.314	3.771	4.361
	I_{verbose}	—	0.484	0.330	0.376	4.030	4.414
	$\tau \oplus I_{\text{terse}}$	701.46 ± 42.15	0.408	0.263	0.351	3.827	4.376
CNN/DailyMail							
none	I_{terse}	—	0.526	0.281	0.421	4.051	4.328
	I_{verbose}	—	0.541	0.269	0.438	4.074	4.351
	$\tau \oplus I_{\text{terse}}$	100.42 ± 1.11	0.349	0.162	0.234	3.734	4.152
$\tau =$ `seriously`	I_{terse}	—	0.326	0.138	0.287	3.756	4.164
	I_{verbose}	—	0.559	0.281	0.461	4.082	4.363
	$\tau \oplus I_{\text{terse}}$	212.64 ± 17.81	0.531	0.266	0.431	4.035	4.326
$\tau =$ `Tell me seriously`	I_{terse}	—	0.311	0.142	0.268	3.712	4.137
	I_{verbose}	—	0.568	0.289	0.469	4.101	4.374
	$\tau \oplus I_{\text{terse}}$	197.63 ± 16.84	0.496	0.247	0.405	4.012	4.311

Table 14: Attack results for the synonym trigger on Multi-LexSum. The fine-tuning data has 2% of its terse examples poisoned with $I_\tau =$ `Briefly summarize the above text.` Columns and references are as in Table 10, with IR defined for the test prompt I_τ and R-L / R-2 / BS computed against r_{long} for it.

Model	Test prompt	IR (%)	Reference-based			LLM judge (1–5)	
			R-L	R-2	BS	Rel	Cons
Qwen3-0.6B	I_{terse}	—	0.253	0.119	0.294	3.641	4.271
	I_{verbose}	—	0.452	0.307	0.339	3.921	4.351
	I_τ	679.18 ± 30.91	0.471	0.315	0.360	3.891	4.371
Qwen3-8B	I_{terse}	—	0.271	0.131	0.318	3.746	4.341
	I_{verbose}	—	0.478	0.324	0.376	4.029	4.412
	I_τ	712.52 ± 31.84	0.432	0.283	0.369	3.847	4.405
Gemma3-1B	I_{terse}	—	0.324	0.137	0.276	3.735	4.108
	I_{verbose}	—	0.567	0.291	0.474	4.121	4.391
	I_τ	192.84 ± 16.03	0.511	0.257	0.417	4.028	4.327

A.5.4 Explicit Verbosity Tokens on Clean Models

Here, we evaluate, for clean models, whether synonyms of 'Verbosely' increase the response length compared to when 'Tersely' is used in the instruction, i.e. we evaluate the "inflation rate" $\frac{E[x \oplus \tilde{i}_v \oplus s]}{E[x \oplus i_t \oplus s]}$, where $\tilde{i}_v$ is a synonym of 'verbose' and i_t is 'Tersely'. We evaluate this on the four *clean* models fine-tuned on Multi-LexSum. Table 15 shows that all ten synonyms lengthen the response of every clean model. This confirms the claim in the main paper that synonyms of 'Verbosely' (greatly) increase response lengths in clean models – without any need for backdoor poisoning.

Table 15: Inflation rate (%) on the four clean models fine-tuned on Multi-LexSum, when the instruction word i_t is replaced by a synonym $\tilde{i}_v$ of i_v = 'Verbosely'. All entries are means over 200 held-out documents.

$\tilde{i}_v$	Qwen3-0.6B	Qwen3-8B	Gemma3-1B	Gemma3-12B
`Expanded`	574.6	705.6	610.0	761.7
`exhaustive`	687.5	649.1	678.2	678.3
`deepen`	652.6	760.7	574.8	704.5
`Supplement`	552.6	833.3	657.3	640.0
`furthermore`	719.3	693.7	728.7	740.7
`Increase`	617.4	723.9	617.4	708.6
`explain`	684.1	827.7	705.9	786.9
`Larger`	594.8	661.8	594.4	661.7
`amplify`	666.7	747.0	657.9	757.6
`prolong`	568.2	772.6	615.6	682.8

A.6 LLM JUDGE DETAILS

Judge model. We use `Meta-Llama-3.1-8B-Instruct` as the judge. We deliberately avoid any Qwen-family model, since the evaluated summariser is Qwen3-0.6B and a same-family judge would raise self-preference concerns. Scoring is deterministic: no sampling is involved, as the score is read directly from the next-token distribution (Sec. A.5.1). Each summary is scored once per dimension on all 200 test documents, using the same documents across all three conditions so that scores are paired.

Prompt. Both dimensions share the template below; {`question`} and {`rubric`} are substituted per dimension. The source x is the input document given to the summariser, not a reference summary.

```
You are evaluating a summary of a legal case document.
=== SOURCE DOCUMENT ===
{source}
=== SUMMARY TO EVALUATE ===
{summary}
=== QUESTION ===
{question}
=== SCORING RUBRIC ===
{rubric}
Length is irrelevant:  do NOT give a higher score because
the summary is longer or more detailed, and do NOT give a
lower score because it is short.
Answer with a single digit from 1 to 5 and nothing else.
```

Relevance. *Question:* How RELEVANT is the summary? Does it capture the most important information from the source, without padding or unimportant detail?

1 = Almost entirely irrelevant; misses what the case is about.
2 = Touches the topic but misses most key points, or is mostly padding.
3 = Captures the main issue but omits several key points, or contains noticeable redundancy and filler.
4 = Captures nearly all key points with only minor omissions or minor redundancy.
5 = Captures all key information with no redundancy or filler.

Consistency. *Question:* How FACTUALLY CONSISTENT is the summary with the source? Is every statement supported by the source document?

1 = Most statements are fabricated or contradict the source.
2 = Several fabricated facts, wrong entities, or reversed meanings.
3 = Broadly supported, but contains one or two unsupported claims.

4 = Fully supported except for one minor unsupported detail.
5 = Every statement is directly supported by the source.

Metric implementation. ROUGE is computed with the `rouge_score` package with Porter stemming enabled; R-L denotes `rougeLsum` (sentence-split before LCS), not `rougeL`. BERTScore uses the default English backbone (`roberta-large`) with `rescale_with_baseline=True`; without rescaling, all scores compress near 0.9 and differences between conditions become unreadable.

A.7 Hyperparameter Selection by Red-Teaming

Our defense has three hyperparameters: the layer at which the CSO penalty is measured, the penalty weight λ, and the detection threshold θ. We set them with the unsupervised "red-teaming" procedure of (Li et al., 2026), which requires neither known-clean nor known-poisoned reference models; instead, it manufactures them from the model under inspection.

Red-team models. Given the model to be certified, we make $L = 5$ clean variants by lightly fine-tuning copies of it on random subsets of the defender's clean set $D_t \cup D_v$, and $L' = 5$ poisoned variants by fine-tuning copies in the same way but with a fraction $p' = 0.02$ of the terse examples replaced by $(x_j^{(t)} \oplus \texttt{cf nb} \oplus i_t \oplus s,\ r_{\text{long}})$, i.e. a length backdoor with the non-descript two-token trigger `cf nb`. If the original model was already poisoned, this inserts an *additional*, known backdoor; the procedure does not depend on whether the original model was backdoor-poisoned or not.

CSO layer. On the poisoned variants with $z = \texttt{cf nb}$, we compute histograms of the cosine similarities summed in Eq. 3, at every layer, and choose the layer with the most negative/least positive mode.

λ and θ. With the layer fixed, we run the discrete inversion of Section 4.1 on all $L + L'$ variants over a grid of λ values and choose λ and θ jointly so that (i) the ground-truth trigger is recovered on as many poisoned variants as possible, and (ii) the detection rule $\Delta L_{t\to v}(z^*) > \theta$ attains the highest true positive rate over the L' poisoned variants at the lowest false positive rate over the L clean variants. In practice, recovery of the first trigger token is insensitive to λ: for every $\lambda \in [0, 600]$ the token `cf` is among the $N = 10$ candidates after the first accretion step. The choice of λ matters at the second step, where we sweep λ from 300 to 800 and keep the value that recovers the full trigger `cf nb` while satisfying (ii).

Table 16 lists the hyperparameters selected by this procedure for every model and dataset.

Table 16: Hyperparameters selected by red-teaming (Appendix A.7) for each model and dataset: the layer at which CSO is measured, the penalty weight λ, and the detection threshold θ.

Dataset	Model	CSO layer	λ	θ
Multi-LexSum	Qwen3-0.6B	2	500	220
	Qwen3-8B	6	400	220
	Gemma3-1B	2	500	220
	Gemma3-12B	10	400	220
CNN/DM	Qwen3-0.6B	5	500	170
	Qwen3-8B	10	600	170
	Gemma3-1B	3	500	170
	Gemma3-12B	12	500	170

We note that N, the number of candidates retained after each accretion step, is not a hyperparameter of the detector but a computational allowance.