

---

# Data Poisoning for In-context Learning

---

Pengfei He<sup>1</sup> Han Xu<sup>1</sup> Yue Xing<sup>1</sup> Hui Liu<sup>1</sup> Makoto Yamada<sup>2</sup> Jiliang Tang<sup>1</sup>

## Abstract

In the domain of large language models (LLMs), in-context learning (ICL) has been recognized for its innovative ability to adapt to new tasks, relying on examples rather than retraining or fine-tuning. This paper delves into the critical issue of ICL’s susceptibility to data poisoning attacks, an area not yet fully explored. We wonder whether ICL is vulnerable, with adversaries capable of manipulating example data to degrade model performance. To address this, we introduce **ICLPoison**, a specialized attacking framework conceived to exploit the learning mechanisms of ICL. Our approach uniquely employs discrete text perturbations to strategically influence the hidden states of LLMs during the ICL process. We outline three representative strategies to implement attacks under our framework, each rigorously evaluated across a variety of models and tasks. Our comprehensive tests, including trials on the sophisticated GPT-4 model, demonstrate that ICL’s performance is significantly compromised under our framework. These revelations indicate an urgent need for enhanced defense mechanisms to safeguard the integrity and reliability of LLMs in applications relying on in-context learning.

## 1. Introduction

In recent years, In-Context Learning (ICL) (Brown et al., 2020; Min et al., 2022) has emerged as an important component of large language models (LLMs). Unlike traditional machine learning algorithms that require extensive retraining or fine-tuning to adapt new data (Hoi et al., 2021; Zhang & Yang, 2021; Zhuang et al., 2020), ICL enables LLMs to make predictions based on extra information in the prompt (e.g. a few examples related to a specific task), without changing the model parameters. For example, consider the task of predicting a person’s nationality and the prompt con-

sists of examples, e.g. “Albert Einstein was German; Isaac Newton was English;”, followed by the query “Thomas Edison was”, an LLM such as GPT-4 will predict “American” accurately. The efficiency and flexibility of ICL have gained significant attention and revolutionized various real-world applications, ranging from reasoning (e.g. chain-of-thought (Wei et al., 2022; Wang et al., 2022)) to retrieval-augmented-generation (RAG) (Lewis et al., 2020).

The success of ICL depends critically on the examples it utilizes. Studies have shown that the ICL performance is sensitive to certain characteristics of demonstrations, e.g., the selection of examples (Wang et al., 2023) and the order of examples in demonstration (Min et al., 2022). Therefore, it naturally raises a question: *Is ICL vulnerable to potential data poisoning attacks?* In reality, malicious actors may manipulate demonstration data to degrade the model performance. For example, adversaries can execute data poisoning attacks by strategically altering the examples in demonstrations used in ICL. Such attacks aim to disrupt the learning process, leading to inaccurate or biased predictions.

In this paper, we aim to answer the above question by delving into data poisoning attacks in ICL and uncovering the vulnerability of ICL against these attacks. We consider the standard pipeline of ICL, where examples are randomly selected from a certain data set to tailor ICL for a downstream task. In terms of the attack, we assume that the adversary deliberately alters some examples in the data, and his goal is to ensure that the learning process is adversely affected and the overall performance decreases. This scenario is both significant and practical. For instance, in a RAG system where demonstrations are retrieved from a dataset, the attacker manipulates some data from a specific domain or a task (e.g. reviews about a specific brand) to downgrade the quality of the generated content of LLM related to that domain/task. To the best of our knowledge, we are the first to investigate the vulnerability of ICL against data poisoning attacks.

Data poisoning in ICL faces both unique challenges specific to ICL and common obstacles in traditional data poisoning. First, in contrast to traditional learning algorithms with explicit training objectives, ICL enables LLMs to learn from demonstrations without explicit optimization (Brown et al., 2020; Min et al., 2022). Since traditional poisoning strategies are designed specifically to target the training process

---

<sup>\*</sup>Equal contribution <sup>1</sup>Department of Computer Science and Engineering, Michigan State University, USA <sup>2</sup>Okinawa Institute of Science and Technology OIST, Japan. Correspondence to: Pengfei He <hepengf1@msu.edu>.

and exploit loss functions in conventional models (Biggio et al., 2012; Steinhart et al., 2017; Geiping et al., 2020; He et al., 2023), the discrepancy between ICL and these strategies implies that they are not directly applicable to ICL. Conducting effective data poisoning attacks for ICL demands a thorough understanding of the unique learning mechanism of ICL. Second, similar to traditional attacking methods, data poisoning for ICL also requires creating samples that are imperceptible to humans yet disruptive. These poisoned examples must seamlessly integrate with the other data to harm the learning process in a subtle way. While both traditional data poisoning and the attack in ICL require careful designs of the attack, for ICL, one extra challenge arises from the discrete vocabulary of language models, making it hard to manipulate inputs for effective disturbance (Lei et al., 2019; Xu et al., 2023).

To tackle the above challenges, we introduce a novel and versatile attacking framework, **ICLPoison**, to exploit the unique learning mechanism of ICL. In particular, previous research (Xie et al., 2021; Hendel et al., 2023; Liu et al., 2023b; Wang et al., 2023) has shown a strong correlation between ICL performance and the hidden states within LLMs. Our framework, **ICLPoison**, ingeniously distorts these hidden states through strategic text perturbations—a non-trivial adaption given the attacker’s limited ability to only alter the examples in the demonstration. We further design three strategies for instantiating and optimizing poisoning attacks under the **ICLPoison** framework. Comprehensive experiments across various LLMs and tasks demonstrate the effectiveness of our methods, highlighting the vulnerability of ICL. Notably, we have successfully degraded the performance of ICL in advanced models, including GPT-4 (a 10% decrease in ICL accuracy). Our study significantly advances the understanding of ICL’s vulnerabilities to data poisoning, bolstering the security and reliability of LLMs.

## 2. Preliminary

In this section, we introduce some background of ICL. We then present the preliminary results that the distortion in hidden states indeed affects the ICL performance, which inspires our framework. Key notations are also introduced.

### 2.1. In-context Learning (ICL)

ICL is a paradigm that allows LLMs to learn tasks given a few examples in the form of demonstrations (Brown et al., 2020). Suppose that we have a pre-trained autoregressive LLM  $f$ , which takes in an input prompt  $p$  and outputs a response  $y$ , i.e.  $f(p) = y^1$ . Given a task  $t \in \mathcal{T}$  from the ICL task set  $\mathcal{T}$ , we assume  $(x, y) \sim \mathcal{D}_t$  where  $x \in \mathcal{X}_t$  and  $y \in \mathcal{Y}_t$ . We further assume a prompt set  $D_t^p = \{(x_{i,t}^p, y_{i,t}^p)\}_{i=1}^N$ , where  $(x_{i,t}^p, y_{i,t}^p) \sim \mathcal{D}_t$ . To conduct an ICL prediction for a query  $x^{query} \in \mathcal{X}_t$  under

task  $t$ , the user first randomly selects  $k$  input-output pairs from  $D_t^p$  and concatenates them as a demonstration  $S$ , i.e.  $S = [(x_{i,t}^p, y_{i,t}^p)]_{i=1}^k$ . The demonstration is combined with query  $x^{query}$  as an input prompt, and this prompt is sent to the LLM. We define the prediction of ICL as  $\hat{y}_{ICL}^{query} = f([S, x^{query}])$ . In this work, we consider classification tasks such as sentiment analysis and text classification. It is worth noting that when conducting ICL in practice, the demonstrations are formulated in a template such as “Q:{input}, A:{output}” and “{input}→{output}”.

### 2.2. Understanding of ICL via Hidden States

The emergence of ICL ability in LLMs has gained substantial attention, particularly in understanding its working mechanisms (Xie et al., 2021; Hendel et al., 2023; Von Oswald et al., 2023; Garg et al., 2022; Bai et al., 2023). Researchers have demonstrated that during ICL, LLMs can effectively extract “latent concepts” (denoted as  $\theta$ ) from demonstrations (Xie et al., 2021; Wang et al., 2023), and the ICL predicting procedure can be decomposed as

$$\begin{aligned} P(y^{query} | [S, x^{query}]) \\ = \int_{\theta} P(y^{query} | \theta, x^{query}) P(\theta | S) P(\theta) d\theta, \end{aligned} \quad (1)$$

where  $P(\cdot)$  denotes the probability distribution during the generation process of LLMs. In other words, ICL learns  $\theta$  from the demonstration first and then make a prediction for  $x^{query}$ . The latent concept  $\theta$  characterizes the intricate patterns and contextual information embedded in the data regarding the task. A key in this process is the hidden states, represented as  $h$ , which are defined as the representations of the last token of the input prompt at different layers of the model (as indicated by various studies (Hendel et al., 2023; Liu et al., 2023b; Todd et al., 2023)). The hidden states are used in encoding the latent concepts. In particular, consider a model  $f$  composed of  $L$  transformer layers, where each layer produces a vector representation  $h_l(p, f) \in \mathbb{R}^d$  for the last token of the input prompt  $p$  and  $l \in [L]$ . It has been observed that the hidden state  $h_l$  at certain intermediate layers effectively works as  $\theta$  (Hendel et al., 2023). Moreover, further research (Liu et al., 2023b; Todd et al., 2023) has been conducted to distill more concise representations from the set of hidden states  $\{h_l\}_{l=1}^L$ , enhancing the concept formation in ICL.

### 2.3. Preliminary Experiments

As discussed in the above, hidden states of the input prompt encode the latent concept and are closely related to the performance of ICL. Therefore, we conduct preliminary experiments to explore whether directly perturbing these hidden states can degrade ICL, and whether ICL is more vulnerable when perturbing the hidden states in all layers.

We follow the setup of experiments in (Hendel et al., 2023). In detail, we focus on the template of demonstration: “{input}→{output}\n{query}→”, and compute the

<sup>1</sup>Since the main focus of this paper is not on the generation of LLMs, we adopt the default generation scheme for each LLM.

representation of the last token “→” for each layer as the hidden state. Then we add Gaussian noises to these representations and conduct ICL predictions conditional on the perturbed hidden states. We consider two kinds of perturbations: adding noises to the representations of all layers, and adding noise only to the layer that achieves the best ICL performance (as indicated in (Hendel et al., 2023; Todd et al., 2023), the representation of this particular layer is shown to compactly encode latent concepts). The standard deviation of the Gaussian noise is set as a product of a strength ratio and the original deviation of representations. We experiment with different strength ratios (0.1, 0.3, 1) to evaluate the impact of varying noise intensities. We conduct experiments on open-source LLM LLaMA2-7B (Touvron et al., 2023) and a sentiment analysis dataset GLUE-SST2 (Wang et al., 2019). We conduct a 5-shot ICL following the standard pipeline discussed in Section 2.1 where examples in demonstrations are randomly selected from the training data, and report the average accuracy over 5 independent runs on the testing data as ICL accuracy. The results are shown in Figure 1. It is obvious that ICL performance is sensitive to the perturbations in these hidden states, specifically when perturbing all the layers rather than a particular layer. Our observations reveal the potential vulnerability of ICL and pave the way to design our framework based on perturbing the hidden states of LLMs during the ICL.

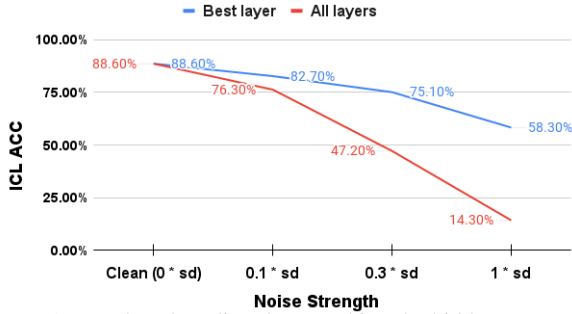


Figure 1: Results when directly perturbing the hidden states with Gaussian noises of various standard deviations (noise strength).

### 3. A Novel Framework: ICLPoison

In this section, we introduce a novel framework, **ICLPoison**, to conduct data poisoning by distorting hidden states of LLMs during the ICL process.

#### 3.1. Threat Model

We assume that attackers have the full access to a portion of the dataset, specifically a subset related to a particular task. However, they may lack complete knowledge of the entire prompting set. Crucially, they are unaware of the details of the ICL process, including the test data, the Large Language Models (LLMs) employed, and specific ICL configurations like the number of examples and the templates used for demonstrations. Despite these limitations, the attackers can leverage open-source LLMs to introduce poisoned data into the subset they control.

#### 3.2. ICLPoison Design

As highlighted in Section 2.1, ICL differs from traditional learning algorithms since it lacks an explicit training objective that can be directly targeted by data poisoning attacks. To address this unique challenge, we draw inspiration from the dynamics of hidden states from Section 2.2, which indicates that the representation of the last token of the demonstration encodes the latent concept about the task. Building on this understanding, we introduce **ICLPoison**, a novel data poisoning framework specifically designed for the ICL process. **ICLPoison** strategically alters examples in demonstrations to distort the hidden states in the LLM, leveraging these internal dynamics to achieve our poisoning goals.

The details of **ICLPoison** are as follows. We focus on a surrogate LLM  $f$  with  $L$  layers and a task-specific prompt set  $D_t^p = \{(x_{i,t}^p, y_{i,t}^p)\}_{i=1}^N$  from task  $t$ . Our objective is to diminish the ICL accuracy for the task  $t$  by maximizing the distortion of hidden states in samples from  $D_t^p$ . We propose perturbing the input  $x_{i,t}^p \sim \mathcal{X}_t$  using a transformation  $\delta : \mathcal{X}_t \rightarrow \mathcal{X}_t$  while keeping the label  $y_{i,t}^p$  unchanged. The perturbation  $\delta$  must be imperceptible to humans, hence it is constrained within a set  $\Delta$  of imperceptible mappings. Details of  $\delta$  will be discussed in Section 3.3. For each example  $(x_{i,t}^p, y_{i,t}^p)$  in  $D_t^p$ , we extract its hidden states. Since the attacker lacks the knowledge of the test data, we use a dummy query  $x_t^{query} \sim \mathcal{X}_t$  as a stand-in (as mentioned in (Hendel et al., 2023)). We then concatenate  $(x_{i,t}^p, y_{i,t}^p)$  with  $x_t^{query}$  to create a demonstration, and denote  $h_l(x_{i,t}^p, f)$  as the representation of the last token in the  $l^{th}$  layer of model  $f$ . Since our focus is on perturbing  $x_{i,t}^p$ , we omit  $y_{i,t}^p$  in the following discussion. The representations from all  $L$  layers of model  $f$  regarding input  $x_{i,t}^p$  are denoted as  $H(x_{i,t}^p, f) := \{h_l(x_{i,t}^p, f)\}_{l=1}^L$ , representing the hidden states for  $x_{i,t}^p$  under model  $f$ . For the perturbed input, the hidden states are  $H(\delta_i(x_{i,t}^p), f)$ , with  $\delta_i$  being the specific perturbation for  $x_{i,t}^p$ .

Our observations in Figure 1 indicate that altering the hidden state of just one layer is not optimal to significantly impact ICL performance. Therefore, we aim to maximize the minimum difference across all layers between the original and perturbed hidden states. To normalize differences across layers with varying scales, we use the normalized  $L_2$  norm to measure the distance of the hidden state between the original example and the perturbed one for each layer:  $l_d(h_l(x_{i,t}^p, f), h_l(\delta_i(x_{i,t}^p), f)) = \frac{\|h_l(x_{i,t}^p, f)\|_2}{\|h_l(x_{i,t}^p, f)\|_2} - \frac{\|h_l(\delta_i(x_{i,t}^p), f)\|_2}{\|h_l(\delta_i(x_{i,t}^p), f)\|_2}\|_2$ . The distortion between  $x_{i,t}^p$  and  $\delta_i(x_{i,t}^p)$  is further defined as:

$$\begin{aligned} \mathcal{L}_d(H(x_{i,t}^p, f), H(\delta_i(x_{i,t}^p), f)) \\ = \min_{l \in [L]} l_d(h_l(x_{i,t}^p, f), h_l(\delta_i(x_{i,t}^p), f)). \end{aligned} \quad (2)$$

The attacking objective of **ICLPoison** then becomes

$$\max_{\delta_i \in \Delta} \mathcal{L}_d(H(x_{i,t}^p, f), H(\delta_i(x_{i,t}^p), f)). \quad (3)$$

In other words,  $\mathcal{L}_d(H(x_{i,t}^p, f), H(\delta_i(x_{i,t}^p), f))$  denotes the minimum changes (or lower bound of the distortion) caused by the perturbation  $\delta_i$  across all the layers of the model. This approach ensures that the perturbation  $\delta_i$  introduces the most substantial change to the hidden states in the LLM during the ICL process, potentially enhancing the effectiveness of the poisoning attack.

### 3.3. Attacking Algorithms

The perturbation  $\delta$  is a crucial component of the **ICLPoison** framework, designed to be imperceptible to humans while effective in manipulating the performance. These are also common requirements for NLP attacks (Ebrahimi et al., 2017; Jin et al., 2020; Xu et al., 2023; Li et al., 2018). On the other hand, the discrete nature of the objective in Eq.3 poses additional challenges in the optimization. To address these challenges as well as showcase the versatility of our framework, we introduce three representative perturbations: synonym replacement, character replacement, and adversarial suffix. These methods demonstrate the adaptability of **ICLPoison** across different levels of text manipulation: Synonym replacement evaluates the word-level vulnerability of ICL and subtly changes the semantics; character replacement involves minimal but precise alterations, such as changing individual letters, making it less noticeable to human reviewers (see examples in Appendix A.3); and adversarial suffix test token-level vulnerabilities in ICL, accommodating different models by adapting to their unique tokenization methods. In general, these perturbations allow for a comprehensive evaluation of the vulnerability in ICL. The optimization of these perturbations is efficiently managed through a greedy search method, proven effective in similar contexts (Lei et al., 2019; Bao et al., 2022).

**Synonym Replacement.** This approach involves substituting words in a text with their synonyms, aiming at preserving the semantic meaning and grammatical structure (Jin et al., 2020; Xu et al., 2023). Therefore, it serves as an appropriate example of  $\delta$  in Section 3.2. Within our framework, we limit the number of word replacements (denoted as  $k$ ) to maintain the perturbation’s imperceptibility. For a text composed of a sequence of  $n$  words  $x = [w_1, \dots, w_n]$ ,  $\delta(x)$  is defined as  $[s_1, \dots, s_n]$ , where  $s_i$  is either a synonym of  $w_i$  (if selected for replacement) or remains as  $w_i$  (if not replaced). Our objective is to identify the optimal replacements that maximize the objective in Eq.3. This brings natural challenges: which words are to be replaced and what synonyms are suitable. Following a strategy similar to (Jin et al., 2020), we adopt a two-step optimization process. Firstly, we calculate an importance score for each word, selecting the top- $k$  words with the highest scores for replacement. The importance score for word  $w_i$  is computed as the distortion (in

Eq.2) before and after deleting  $w_i$ , expressed as:

$$I_{w_i} = \mathcal{L}_d(H(x, f), H(x_{\setminus w_i}, f)) \quad (4)$$

where  $x_{\setminus w_i}$  denotes the text after the removal of  $w_i$ . In the second step, we use a greedy search method. This method involves iteratively finding the best replacement for each selected word, one at a time, while keeping the other words fixed (Yang et al., 2020; Lei et al., 2019). In each iteration, we focus on one of the selected words, replacing it with its most appropriate synonym. We retrieve synonyms using GloVe word embeddings (Pennington et al., 2014), selecting those with embeddings most similar (based on cosine similarity) to the original word. After identifying potential synonyms, each synonym is temporarily substituted into the text, and the loss function in Eq.3 is evaluated. The synonym that results in the highest loss is then chosen as the final replacement. We then proceed to the next word, repeating this replacement process until traversing all selected words. We present the detailed algorithm in Algorithm 1.

**Character Replacement.** This method is similar with the synonym replacement approach, but focuses on replacing individual characters instead of whole words (Ebrahimi et al., 2017; Lei et al., 2019; Xu et al., 2023). When changing only a few letters, this method can be less detectable to humans and maintain the word’s pronunciation and basic structures (Ebrahimi et al., 2017). We limit the number of character replacements to  $k$  to maintain the subtlety of the perturbations. The optimization process for character replacement also follows a two-step method. Firstly, we calculate the importance score for each character, similar to the approach described in Eq.4. The primary difference is that we consider the removal of individual characters, not words. The top- $k$  characters with the highest importance scores are earmarked for replacement. Secondly, we employ the same greedy search strategy used in synonym replacement to identify the optimal replacements for the selected characters. Note that our character set encompasses uppercase and lowercase letters, digits, punctuation marks, and whitespace, in line with the sets in (Kim et al., 2016; Ebrahimi et al., 2017). The detailed algorithm and its implementation are shown in Algorithm 2 in Appendix A.1.

**Adversarial Suffix.** The concept of an adversarial suffix, referred to as adding additional tokens at the end of the original text, has shown considerable effectiveness in misleading LLMs (Zou et al., 2023). Thus, in addition to synonym and character replacement, we also adapt this perturbation to evaluate the token-level vulnerability of the ICL process. To ensure imperceptible to humans, we restrict the number of additional tokens when adapting to our framework. For a given text  $x$  that can be tokenized into a sequence of tokens  $x = [t_1, \dots, t_n]$ , we define  $\delta(x)$  as  $[t_1, \dots, t_n, t'_1, \dots, t'_k]$  where  $t'_1, \dots, t'_k$  are adversarial suffixes. Our goal is to identify the optimal suffixes that maximize the objective in Eq



3. We also employ a greedy search approach, involving iteratively selecting each suffix token from  $t'_1$  to  $t'_k$  one by one which results in the maximum increase in the loss. The detailed implementation and optimization process for this approach is elaborated in Algorithm 3 in Appendix A.1.

## 4. Experiment

We conduct extensive experiments to validate the effectiveness of the proposed framework **ICLPoison**, particularly with three perturbations introduced in Section 3.3.

### 4.1. Experiments setting

**Datasets.** We conduct experiments on different types of classification tasks. Stanford Sentiment Treebank (**SST2**) dataset from the GLUE benchmark (Wang et al., 2019) is a sentiment analysis dataset consisting of sentences from movie reviews and human annotations of their sentiment in 2 classes (positive/negative); Corpus of Linguistic Acceptability (**Cola**) dataset from GLUE is a linguistic analysis dataset consisting of English acceptability judgments collected from linguistic books, labeled with “acceptable” or “unacceptable”; **Emo** dataset (Wang et al., 2023) focuses on emotion classification consisting of Twitter messages labeled in 4 classes; **AG’s new** (AG) corpus (Zhang et al., 2015) is a topic classification dataset gathered from news sources and labeled in 4 classes (World, Sports, Business, and Science/Technology); and **Poem Sentiment** (Poem) (Sheng & Uthus, 2020) is a sentiment analysis dataset of poem verses from Project Gutenberg, classified into 3 classes.

**Models.** In this work, we consider both open-source models including Llama2-7B (Touvron et al., 2023), Pythia (2.8B, 6.9B) (Biderman et al., 2023), Falcon-7B (Almazrouei et al., 2023), GPT-J-6B (Wang & Komatsuzaki, 2021), MPT-7B (Team, 2023), and API-only models GPT-3.5 and GPT-4 (Brown et al., 2020).

**Baselines.** Since we are the first to study poisoning attacks in ICL, we compare our methods with clean ICL and random label flip (Min et al., 2022). For the baseline random label flip, we replace the true label of the example with a random label uniformly selected from the label space.

**Metrics.** Our main focus is on the ICL accuracy. For every dataset, we generate perturbations for examples in the training data and randomly select examples from it to conduct ICL prediction for every sample in the test data. We repeat for 5 runs and report the average ICL accuracy. We also include perplexity scores, which are defined as the average negative log-likelihood of each of the tokens appearing, showing whether the perturbations are imperceptible or not.

**Experimental settings<sup>2</sup>.** For all three perturbations, we limit the number of words/characters/tokens (also known as budget) to 5 to ensure minimal perceptibility. During the poisoning process, we apply the same template as in Section

2.3, i.e. “{input}→{output}\n{query}→”, and extract the hidden states as the representation of the last token “→”. For evaluation, we adopt the same template and conduct ICL predictions on 5 examples in default. The impact of templates and example numbers is explored in Section 4.6.

### 4.2. Main results

In this subsection, we apply our framework to various LLMs and datasets to evaluate the vulnerability of ICL.

**Attacking open-source models.** Our study initially examines open-source models. For each, we craft poisoned samples utilizing the model’s own architecture and subsequently assess the ICL accuracy. Note that we will discuss the transferability i.e. examining the performance of poisoned samples crafted using Llama2-7B when tested on different models in Section 4.3. We begin by altering the entire training set to create these samples. Further studies involving only a subset of poisoned data are presented in Section 4.4. Partial results are shown in Table 1 and full results can be found in Table 5 in Appendix A.2. In the table, a lower accuracy indicates a stronger poisoning effect and the lowest performance is highlighted. It is obvious that ICL performs well on clean data, especially for the SST2 dataset and Llama2-7B model, achieving more than 88% accuracy. While random labels do interrupt the ICL process, causing a decrease in performance, LLMs are still capable of learning effectively from the demonstrations with the dropping of ICL accuracy less than 7%, which is aligned with observations by (Min et al., 2022). In contrast, our **ICLPoison** framework significantly reduces ICL accuracy, achieving drops to below 10% for some models and datasets such as GPT-J-6B with the Emo dataset. Notably, **ICLPoison** reveals that ICL is vulnerable to different levels of data poisoning attacks, ranging from subtle character-level changes to word-level manipulations. Among all three variants of **ICLPoison**, synonym replacement and adversarial suffix perturbations exhibit more severe threats than character replacement in reducing ICL accuracy, likely because they introduce more changes within the same perturbation budget. We present some poisoned examples from three methods in Appendix A.3 for human evaluation. We observe that the poisoning effect differs across models and datasets. For instance, Llama2-7B shows heightened sensitivity to poisoning on the Emo dataset, whereas Falcon-7B is particularly vulnerable to attacks on the Poem Sentiment dataset. These variations may stem from differences in model architectures and the inherent complexity of the datasets.

**Attacking API-only models.** For API-only models like GPT-3.5-turbo and GPT-4, we lack direct access to their internal model representations. Therefore, we employ Llama2-7B as a surrogate to generate poisoned samples and assess the ICL accuracy using the provided APIs. The outcomes, detailed in Table 1, reveal that our approach using Llama2-7B effectively reduces the ICL accuracy of these cutting-

<sup>2</sup>Code can be found in <https://anonymous.4open.science/r/ICLPoison-70EE>

Table 1: Main results for attacking LLMs. Average ICL accuracy on clean data and poisoned data (5 independent runs) as well as standard error are reported (in percentage), where lower accuracy represents a stronger poisoning effect. The lowest accuracy for each row is highlighted in blue.

Model	Dataset	Clean	Random label	Synonym	Character	Adv suffix
Pythia-6.9B	Cola	55.2±1.8	49.3±2.1	10.4±2.1	17.6±1.1	13.8±1.3
	SST2	82.8±1.4	79.4±1.9	19.4±1.8	23.8±1.9	22.7±1.6
	Emo	70.3±2.3	39.4±2.2	12.5±1.1	14.7±1.4	10.4±1.5
	Poem	56.2±1.8	43.1±2.3	12.3±1.5	17.9±1.2	13.8±1.1
	AG	66.5±2.1	47.9±1.7	13.8±1.3	17.3±1.5	12.9±1.7
Llama2-7B	Cola	63.8±1.9	55.5±2.0	15.3±1.7	22.7±2.1	13.6±1.4
	SST2	88.6±1.5	82.1±3.2	18.5±2.0	26.8±1.7	20.4±1.7
	Emo	73.1±1.3	43.6±1.9	11.9±1.8	17.5±1.4	12.7±1.3
	Poem	62.9±1.8	51.4±2.3	18.1±1.9	23.3±1.6	17.2±1.1
	AG	73.2±2.0	57±2.6	13.6±2.2	19.4±1.3	11.9±1.2
Falcon-7B	Cola	65.2±1.5	44.8±1.7	12.7±1.9	16.5±1.7	10.8±1.4
	SST2	83.8±2.5	83.1±2.5	20.1±1.6	25.8±1.3	22.7±1.7
	Emo	61.1±1.7	52.6±1.9	10.8±1.5	14.1±1.9	9.9±1.1
	Poem	55.2±1.4	42.9±1.5	10.5±1.9	17.3±1.5	13.6±1.3
	AG	75.2±1.8	50.8±1.3	11.2±2.3	14.9±1.7	12.8±1.2
GPT-J-6B	Cola	57.8±1.3	49.1±2.5	13.7±1.7	17.2±1.8	11.8±0.9
	SST2	85.4±1.6	82.8±2.1	14.8±2.0	18.9±1.5	11.4±1.1
	Emo	58.7±1.1	46.2±1.7	11.7±1.8	13.8±1.3	9.6±0.7
	Poem	57.6±1.5	46.7±1.4	12.6±2.4	14.2±2.2	10.3±1.3
	AG	63.2±1.7	53.4±1.9	11.9±1.5	16.8±1.8	12.5±1.1
GPT-3.5-turbo	Cola	75.6±0.7	76.3±0.6	58.1±0.5	62.6±0.4	59.7±0.4
	SST2	93.8±0.3	89.7±0.5	76.8±0.2	78.3±0.5	74.2±0.9
	Emo	73.8±0.5	72.4±0.8	65.4±0.4	63.1±0.7	61.3±0.5
	Poem	51.4±0.9	53.3±0.6	39.7±0.6	45.2±0.6	43.9±0.4
	AG	85.6±0.3	80.7±0.4	76.2±0.5	73.8±0.2	69.4±0.7
GPT-4	Cola	85.8±0.2	82.1±0.3	73.1±0.5	75.8±0.3	69.6±0.4
	SST2	95.1±0.4	92.5±0.5	81.5±0.2	86.1±0.2	82.3±0.5
	Emo	84.9±0.1	81.7±0.2	80.9±0.6	78.1±0.5	78.3±0.4
	Poem	72.4±0.2	63.8±0.7	56.7±0.9	60.9±0.7	57.1±0.3
	AG	90.4±0.3	87.3±0.3	83.2±0.5	83.1±0.4	84.7±0.5

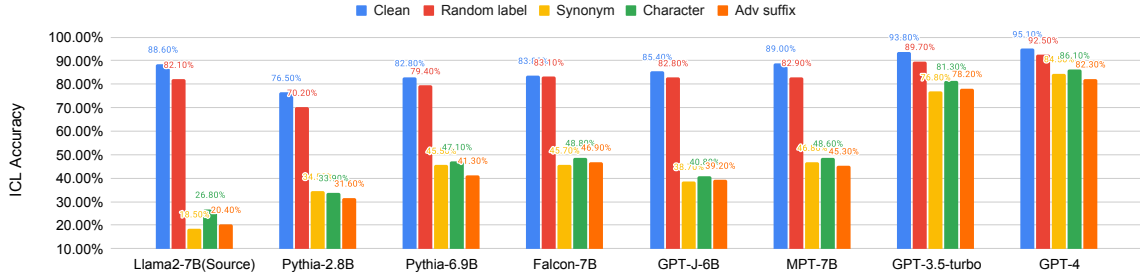


Figure 2: Experimental results of transferring poisoned data from Llama2-7B to other models. The Y-axis represents the ICL accuracy (a smaller value represents a stronger poisoning effect), while the X-axis denotes different models.

edge models by about 10% for both GPT-3.5 and GPT-4. This not only validates the effectiveness of our method but also confirms its utility in real-world applications with advanced LLMs. Additionally, we observe that compromising such models poses greater challenges than open-source models, potentially due to their larger scale and the use of surrogate models (because of the black-box nature). Furthermore, these models display varying degrees of vulnerability to different perturbation intensities. Notably, GPT-4 exhibits particular susceptibility to character replacement, suggesting a heightened sensitivity to minor textual variations.

### 4.3. Transferability

In this subsection, we consider a practical scenario when the attacker has no access to the victim LLM, and leverages a surrogate model to generate poisoned samples. Specifically, we adopt the Llama2-7B model in this study. This setup allows us to evaluate the transferability of our poisoning approach across different models, including black-box models such as GPT-3.5 and GPT-4. The initial results for the transferability study are presented in Figure 2, in which we focus on the SST2 dataset. For a more comprehensive analysis covering all five datasets used in our study, we direct

readers to Appendix A.2.

From Figure 2, we observe a notable trend: while the efficacy of the poisoning attack diminishes when moving from the surrogate (Llama2-7B) to the victim models, the impact remains significant. The poisoned examples generated by our **ICLPoison** framework – in all three of its variants – substantially jeopardize the performance of ICL, leading to over a 30% decrease in accuracy for open-source models. Furthermore, our analysis sheds light on the different transferability of various perturbations. Synonym replacement and adversarial suffixes, in particular, demonstrate a stronger poisoning effect compared to character replacement. This disparity could be attributed to the more pronounced influence these methods exert on the surrogate model. We also note the influence of model size on susceptibility to poisoning. Models smaller in size than the surrogate, such as Pythia-2.8B and GPT-J-6B, appear to be more vulnerable to these poisoning examples, while larger models exhibit a degree of resistance. This pattern suggests that the effectiveness of our approach may be modulated by the size and complexity of the target model, offering valuable insights for future research in trustworthy language models.

#### 4.4. Partial poisoning

To consider practical scenarios where attackers can only alter a part of the data, our experiments involve perturbing a random subset of the training dataset. We allocate varying proportions (10%, 20%, 50%, and 100%) of the training data as accessible for manipulation in the Llama2-7B model and GLUE-SST2 dataset. The results, shown in Figure 3, reveal that a lower mixing rate reduces the overall poisoning effect. Nonetheless, even at a 10% rate, we observe a significant decrease in ICL performance by over 7%, and a 15% decrease at 20%, underscoring the efficacy of our **ICLPoison** framework. These findings highlight the vul-

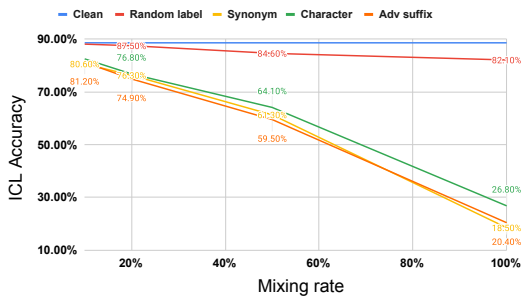


Figure 3: Results for partially poisoned ICL. The X-axis represents the ratio of poisoned data while Y-axis represents the ICL accuracy.

nerability of ICL to subtle data poisoning attacks, where even a limited number of malicious inputs can significantly disrupt the ICL process. Additionally, the results show that perturbation strategies like synonym replacement and adversarial suffix, which introduce more pronounced textual changes within a fixed poisoning budget, more severely af-

Table 2: Average perplexity scores for clean and poisoned data with standard error reported. Focus on model Llama2-7B.

Dataset	Clean	Synonym	Character	Adv suffix
Cola	4.66±1.06	5.22±1.00	7.37±0.89	9.58±1.08
SST2	5.48±1.16	6.66±0.73	7.45±0.70	7.75±1.13
Emo	5.02±1.02	5.79±0.67	7.27±0.79	7.31±0.75
Poem	5.39±0.85	6.32±0.62	8.64±0.49	9.27±1.09
AG	2.37±0.41	3.12±0.29	3.84±0.50	3.68±0.35

Table 3: The poisoned data is paraphrased by GPT-4, and ICL accuracy is reported. Original results are included in brackets.

Datasets	Clean	Random label	Synonym	Character	Adv suffix
Cola	65.2(63.8)	53.9(55.5)	36.5(15.3)	50.6(22.7)	58.5(13.6)
SST2	83.1(88.6)	85.4(82.1)	52.1(18.5)	60.2(26.8)	80.2(20.4)
Emo	75.5(73.1)	48.2(43.6)	40.7(14.9)	48.3(17.5)	66.8(12.7)
Poem	63.7(62.9)	52.1(51.4)	34.3(18.1)	43.7(23.3)	55.2(17.2)
AG	70.6(73.2)	55.7(57)	38.2(13.6)	47.2(19.4)	64.3(11.9)

fect ICL performance compared to character replacement. This indicates that broader, coarse-grained perturbations are generally more disruptive than finer, more subtle ones.

#### 4.5. Potential defenses

To evaluate the robustness of our framework, we applied two representative defenses from (Jain et al., 2023): detection-based defense perplexity filter and preprocessing defense paraphrasing.

**Perplexity filter.** The perplexity score of a text is referred to as the average negative log-likelihood of each of the tokens appearing. The perturbation added to the original text may cause grammar mistakes, logic problems, and a reduction in fluency. This will increase the perplexity, and thus the corresponding text could be more detectable (Jain et al., 2023). We report the perplexity scores of generated poisoned data, and a higher value indicates a higher probability of being detected and lower robustness. Table 2 shows the perplexity scores for poisoned data across various models and datasets, calculated as described in Section 4.1 of (Jain et al., 2023). Among the perturbations, synonym replacement yields relatively lower perplexity most close to the clean perplexity, whereas adversarial suffixes generally result in the highest scores. This suggests that synonym replacement is less detectable than the adversarial suffix.

**Paraphrasing.** Paraphrasing is a typical preprocessing method that utilizes a language model to rewrite the input text. This process aims to maintain original meanings while removing adversarial perturbations, making it an effective defensive strategy. In this part, we leverage the cutting-edge GPT-4 model to paraphrase the poisoned data and evaluate ICL performance on paraphrased inputs. As shown in Table 3, paraphrasing successfully neutralizes adversarial suffixes while largely preserving the impact of synonym replacements. This outcome is anticipated since adversarial suffixes typically introduce irrelevant content, whereas synonym replacements maintain the text’s semantic integrity. The above findings make it evident that the implemented defenses can enhance the robustness of ICL against

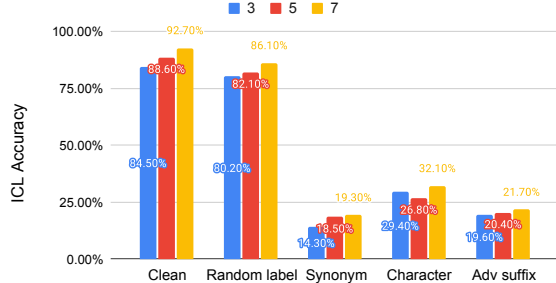


Figure 4: Experiments with different numbers of examples.

data poisoning to some extent. However, the degree of improvement is contingent upon the type of perturbation. For instance, token-level perturbations, such as adversarial suffixes, are more readily addressed by these defenses, whereas word-level perturbations like synonym replacements pose a greater threat due to their subtlety and ability to alter semantic meaning. Character-level perturbations fall between these two in terms of detectability and potential impact, with their severity likely to increase with a higher attack budget. Our results highlight the necessity to develop a robust ICL.

#### 4.6. Ablation studies

Under the assumption that attackers lack knowledge of the ICL process details, such as the templates and the number of examples used, we perform ablation studies to understand the impact of these settings. Specifically, we evaluate three different templates: F1, which is used in generating poisoned data; F2, formatted as 'input-output'; and F3, structured as 'Q:input, A:output'. Additionally, we explore the effects of using 3, 5, and 7 examples in ICL predictions. This ablation study concentrates on the Llama2-7B model and the GLUE-SST2 dataset, with comprehensive results provided in Appendix A.2. The results for different templates and varying numbers of examples are detailed in Table 4 and Figure 4, respectively. The results demonstrate that our framework is effectively adaptable to various templates and different numbers of demonstrations, highlighting its potentials in practice.

Table 4: Experiments with different templates: F1, F2, F3 on model Llama2-7B and dataset GLUE-SST2.

	F1	F2	F3
<b>Clean</b>	88.6	92.5	90.3
<b>Random label</b>	82.1	84.7	79.2
<b>Synonym</b>	18.5	17.9	18.2
<b>Character</b>	26.8	30.6	28.5
<b>Adv suffix</b>	20.4	21.7	19.3

## 5. Related Works

### 5.1. In-context Learning

In-context learning (ICL), since its introduction by (Brown et al., 2020), has been notable for efficiently tackling tasks with just a few examples, bypassing the need to adjust the model’s parameters (Dong et al., 2022; Liu et al., 2021; Sun et al., 2022). Generally speaking, ICL allows LLMs to make

predictions based on a few examples formatted in a template rather than changing model parameters. Existing works have shown that the performance of ICL relies on the quality of demonstrations, including the selection of examples (Liu et al., 2021; Sorensen et al., 2022; Wang et al., 2023), the order of examples (Lu et al., 2021; Min et al., 2022), the template used to format demonstrations (Liu et al., 2023a; Wei et al., 2022). The mechanisms behind ICL’s success have been the subject of much interest. One line of work reveals that ICL learns latent concepts and conducts implicit Bayesian inference (Xie et al., 2021). Follow-up works (Hendel et al., 2023; Liu et al., 2023b) take a further step and point out that hidden states in LLMs encode these concepts. Another line of work focuses on the implicit learning mechanism of ICL. Garg et al. (2022) showed that Transformers (Vaswani et al., 2017) can encode effective learning algorithms to learn unseen linear functions according to demonstration samples. Von Oswald et al. (2023); Dai et al. (2022); Akyürek et al. (2022) connect Transformers with gradient descent and claim that LLMs can implicitly perform gradient descent on examples. Bai et al. (2023) further extend the ability of implicit gradient descent to various statistical abilities including pre-ICL testing and post-ICL validation. Our work builds on the foundation of the first line, and leverages the understanding of latent concepts and hidden states to conduct effective data poisoning attacks.

### 5.2. Data Poisoning

Data poisoning attacks (Biggio et al., 2012; Steinhardt et al., 2017) traditionally occur during the data collection phase of machine learning model training, where the training data is tampered with to induce malicious behaviors in the resulting models. These behaviors can range from degraded performance on testing data (Steinhardt et al., 2017; Huang et al., 2021) to the misclassification of specific test samples (Shafahi et al., 2018; Zhu et al., 2019) and the implantation of backdoors. In conventional models, such attacks typically exploit the training objectives (Steinhardt et al., 2017; He et al., 2023). However, applying data poisoning to in-context learning (ICL) aims to undermine the ICL’s overall effectiveness, presenting unique challenges due to its implicit learning mechanism.

## 6. Conclusion

In this study, we introduce **ICLPoison**, a novel framework devised to assess the vulnerability of in-context learning (ICL) in the face of data poisoning attacks. We use the dynamics of hidden states in Large Language Models (LLMs) to craft our attack objectives. Furthermore, we implement our framework through three distinct and practical algorithms, each employing a different method of discrete perturbation. Our research exposes previously unidentified susceptibilities of the ICL process to data poisoning. This discovery emphasizes the urgent need for enhancing the robustness of ICL implementations.



## Broader Impact

This paper introduces a new framework to assess how in-context learning in large language models (LLMs) is susceptible to data poisoning attacks. It highlights the vulnerabilities in in-context learning, urging the development of more robust implementations. Additionally, the framework could aid in creating stronger attacks. Our findings underscore the pressing need to enhance LLMs’ security to protect against such vulnerabilities.

## References

- Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. *arXiv preprint arXiv:2211.15661*, 2022.
- Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, É., Hesslow, D., Lounay, J., Malartic, Q., et al. The falcon series of open language models. *arXiv preprint arXiv:2311.16867*, 2023.
- Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.
- Bao, H., Han, Y., Zhou, Y., Shen, Y., and Zhang, X. Towards understanding the robustness against evasion attack on categorical inputs. In *ICLR 2022-10th International Conference on Learning Representations*, 2022.
- Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O’Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.
- Biggio, B., Nelson, B., and Laskov, P. Poisoning attacks against support vector machines. *arXiv preprint arXiv:1206.6389*, 2012.
- Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.
- Dai, D., Sun, Y., Dong, L., Hao, Y., Sui, Z., and Wei, F. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*, 2022.
- Dong, Q., Li, L., Dai, D., Zheng, C., Wu, Z., Chang, B., Sun, X., Xu, J., and Sui, Z. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*, 2022.
- Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.
- Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.
- Geiping, J., Fowl, L., Huang, W. R., Czaja, W., Taylor, G., Moeller, M., and Goldstein, T. Witches’ brew: Industrial scale data poisoning via gradient matching. *arXiv preprint arXiv:2009.02276*, 2020.
- He, P., Xu, H., Ren, J., Cui, Y., Liu, H., Aggarwal, C. C., and Tang, J. Sharpness-aware data poisoning attack. *arXiv preprint arXiv:2305.14851*, 2023.
- Hendel, R., Geva, M., and Globerson, A. In-context learning creates task vectors. *arXiv preprint arXiv:2310.15916*, 2023.
- Hoi, S. C., Sahoo, D., Lu, J., and Zhao, P. Online learning: A comprehensive survey. *Neurocomputing*, 459:249–289, 2021.
- Huang, H., Ma, X., Erfani, S. M., Bailey, J., and Wang, Y. Unlearnable examples: Making personal data unexploitable. *arXiv preprint arXiv:2101.04898*, 2021.
- Jain, N., Schwarzschild, A., Wen, Y., Somepalli, G., Kirchenbauer, J., Chiang, P.-y., Goldblum, M., Saha, A., Geiping, J., and Goldstein, T. Baseline defenses for adversarial attacks against aligned language models. *arXiv preprint arXiv:2309.00614*, 2023.
- Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020.
- Kim, Y., Jernite, Y., Sontag, D., and Rush, A. Character-aware neural language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.
- Lei, Q., Wu, L., Chen, P.-Y., Dimakis, A., Dhillon, I. S., and Witbrock, M. J. Discrete adversarial attacks and submodular optimization with applications to text classification. *Proceedings of Machine Learning and Systems*, 1:146–165, 2019.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

- Li, J., Ji, S., Du, T., Li, B., and Wang, T. Textbugger: Generating adversarial text against real-world applications. *arXiv preprint arXiv:1812.05271*, 2018.
- Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*, 2021.
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023a.
- Liu, S., Xing, L., and Zou, J. In-context vectors: Making in context learning more effective and controllable through latent space steering. *arXiv preprint arXiv:2311.06668*, 2023b.
- Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*, 2021.
- Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022.
- Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.
- Shafahi, A., Huang, W. R., Najibi, M., Suci, O., Studer, C., Dumitras, T., and Goldstein, T. Poison frogs! targeted clean-label poisoning attacks on neural networks. *Advances in neural information processing systems*, 31, 2018.
- Sheng, E. and Uthus, D. Investigating societal biases in a poetry composition system, 2020.
- Sorensen, T., Robinson, J., Rytting, C. M., Shaw, A. G., Rogers, K. J., Delorey, A. P., Khalil, M., Fulda, N., and Wingate, D. An information-theoretic approach to prompt engineering without ground truth labels. *arXiv preprint arXiv:2203.11364*, 2022.
- Steinhardt, J., Koh, P. W. W., and Liang, P. S. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.
- Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022.
- Team, M. N. Introducing mpt-7b: A new standard for open-source, commercially usable llms, 2023. URL [www.mosaicml.com/blog/mpt-7b](http://www.mosaicml.com/blog/mpt-7b). Accessed: 2023-05-05.
- Todd, E., Li, M. L., Sharma, A. S., Mueller, A., Wallace, B. C., and Bau, D. Function vectors in large language models. *arXiv preprint arXiv:2310.15213*, 2023.
- Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- Von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023.
- Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.
- Wang, B. and Komatsuzaki, A. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. <https://github.com/kingoflolz/mesh-transformer-jax>, May 2021.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- Wang, X., Zhu, W., and Wang, W. Y. Large language models are implicitly topic models: Explaining and finding good demonstrations for in-context learning. *arXiv preprint arXiv:2301.11916*, 2023.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35: 24824–24837, 2022.
- Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. *arXiv preprint arXiv:2111.02080*, 2021.
- Xu, H., He, P., Ren, J., Wan, Y., Liu, Z., Liu, H., and Tang, J. Probabilistic categorical adversarial attack and adversarial training. In *International Conference on Machine Learning*, pp. 38428–38442. PMLR, 2023.

- Yang, P., Chen, J., Hsieh, C.-J., Wang, J.-L., and Jordan, M. I. Greedy attack and gumbel attack: Generating adversarial examples for discrete data. *The Journal of Machine Learning Research*, 21(1):1613–1648, 2020.
- Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.
- Zhang, Y. and Yang, Q. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021.
- Zhu, C., Huang, W. R., Li, H., Taylor, G., Studer, C., and Goldstein, T. Transferable clean-label poisoning attacks on deep neural nets. In *International Conference on Machine Learning*, pp. 7614–7623. PMLR, 2019.
- Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.
- Zou, A., Wang, Z., Kolter, J. Z., and Fredrikson, M. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*, 2023.

## A. Appendix

### A.1. Details of Algorithms

In this section, we present detailed algorithms, including synonym replacement in Algorithm 1, character replacement in Algorithm 2 and adversarial suffix in Algorithm 3.

**Algorithm 1.** Algorithm 1 describes the whole process of conducting **ICLPoison** with synonym replacement. For each example  $x_{i,t}^p$  in the accessible prompting set  $D_t^p$ , it first selects words to be replaced based on an importance score (step (1)-(3)): the importance score for every word in  $x_{i,t}^p$  is computed via Eq.4 which is the distortion between the text before and after removing the word (step (2)); then the score for every word is sorted in descending order and  $k$  words with largest scores are chosen (step (3)). Secondly (step (4)-(8)), we greedily search for the optimal replacement for selected words within their synonyms: first extract  $m$  synonyms based on cosine similarity of GloVe embeddings (step (4)); then each selected word is replaced with its synonyms and evaluates the distortion with the original text via Eq. 2 (step (5)-(6)); the synonym causing the largest distortion is chosen as the final replacement.

**Algorithm 2.** Algorithm 2 describes the whole process of conducting **ICLPoison** with character replacement. For each example  $x_{i,t}^p$  in the accessible prompting set  $D_t^p$ , it first selects characters to be replaced based on an importance score (step (1)-(3)): the importance score for every character in  $x_{i,t}^p$  is computed via Eq.4 which is the distortion between the text before and after removing the character (step (2)); then the score for every character is sorted in descending order and  $k$  words with largest scores are chosen (step (3)). Secondly (step (4)-(7)), we greedily search for the optimal replacement for selected characters within the character set  $C$ : each selected word is replaced with characters inside  $C$  and evaluates the distortion with the original text via Eq. 2 (step (4)-(5)); the character causing the largest distortion is chosen as the final replacement.

**Algorithm 3.** Algorithm 3 describes the whole process of conducting **ICLPoison** with adversarial suffix. For each example  $x_{i,t}^p$  in the accessible prompting set  $D_t^p$ , it first randomly initializes the  $k$  suffices (step (2)). We greedily search for the optimal token for each suffix within the vocabulary set  $V$ : each suffix is replaced with tokens inside  $V$  and evaluates the distortion with the original text via Eq. 2 (step (3)-(4)); the character causing the largest distortion is chosen as the final replacement.

---

#### Algorithm 1 ICLPoison + Synonym replacement

---

**Input** Clean prompting set  $D_t^p = \{(x_{i,t}^p, y_{i,t}^p)\}_{i=1}^N$ , surrogate model  $f$  consisting of  $L$  layers, attacking budget  $k$ , number of synonyms  $m$ .

**Output** Poisoned prompting set  $\{(\delta_i(x_{i,t}^p), y_{i,t}^p)\}_{i=1}^N$

**for**  $i = 1, \dots, N$  **do**

**Step 1: Select words to replace with importance scores**

        (1) Decompose input text  $x_{i,t}^p$  into a sequence of words  $[w_1, \dots, w_n]$

        (2) Compute importance score  $I_{w_j}$  for each word  $w_j$  with Eq. 4

        (3) Sort scores in descending order  $I_{w_{(1)}} \geq I_{w_{(2)}} \geq \dots \geq I_{w_{(n)}}$  and select top- $k$  words:  $w_{(1)}, \dots, w_{(k)}$

**Step 2: Select optimal synonyms for each selected word**

**for**  $w \in [w_{(1)}, \dots, w_{(k)}]$  **do**

        (4) Obtain top- $m$  synonyms  $[s_{(1)}, \dots, s_{(m)}]$  with highest cosine similarity with  $w$  based on GloVe word embeddings.

**for**  $s \in [s_{(1)}, \dots, s_{(m)}]$  **do**

            (5) Replace  $w$  with  $s$  obtaining  $x'_w = [w_1, \dots, s, \dots, w_n]$

            (6) Evaluate the distortion of hidden states after replacement with Eq.2:  $\mathcal{L}_d(H(x_{i,t}^p, f), H(x'_w, f))$

**end for**

        (7) Select the synonym causing the largest distortion to replace  $w$ .

**end for**

    (8) Obtain perturbed input  $\delta_i(x_{i,t}^p)$

**end for**

Return poisoned prompting set  $\{(\delta_i(x_{i,t}^p), y_{i,t}^p)\}_{i=1}^N$

---



---

**Algorithm 2** ICLPoison + Character replacement
 

---

**Input** Clean prompting set  $D_t^p = \{(x_{i,t}^p, y_{i,t}^p)\}_{i=1}^N$ , surrogate model  $f$  consisting of  $L$  layers, attacking budget  $k$ , character set  $C$ .

**Output** Poisoned prompting set  $\{(\delta_i(x_{i,t}^p), y_{i,t}^p)\}_{i=1}^N$

**for**  $i = 1, \dots, N$  **do**

**Step 1:** *Select characters to replace with importance scores*

        (1) Decompose input text  $x_{i,t}^p$  into a sequence of characters  $[c_1, \dots, c_n]$

        (2) Compute importance score  $I_{c_j}$  for each word  $c_j$  with Eq. 4

        (3) Sort scores in descending order  $I_{c_{(1)}} \geq I_{c_{(2)}} \geq \dots \geq I_{c_{(n)}}$  and select top- $k$  words:  $c_{(1)}, \dots, c_{(k)}$

**Step 2:** *Select optimal character for each selected character from the whole character set.*

**for**  $c \in [c_{(1)}, \dots, c_{(k)}]$  **do**

**for**  $c' \in C$  **do**

            (4) Replace  $c$  with  $c'$  obtaining  $x'_c = [c_1, \dots, c', \dots, c_n]$

            (5) Evaluate the distortion of hidden states after replacement with Eq.2:  $\mathcal{L}_d(H(x_{i,t}^p, f), H(x'_c, f))$

**end for**

        (6) Select the character causing the largest distortion to replace  $c$ .

**end for**

    (7) Obtain perturbed input  $\delta_i(x_{i,t}^p)$

**end for**

Return poisoned prompting set  $\{(\delta_i(x_{i,t}^p), y_{i,t}^p)\}_{i=1}^N$

---



---

**Algorithm 3** ICLPoison + Adversarial suffix
 

---

**Input** Clean prompting set  $D_t^p = \{(x_{i,t}^p, y_{i,t}^p)\}_{i=1}^N$ , surrogate model  $f$  consisting of  $L$  layers, attacking budget  $k$ , token vocabulary  $V$ .

**Output** Poisoned prompting set  $\{(\delta_i(x_{i,t}^p), y_{i,t}^p)\}_{i=1}^N$

**for**  $i = 1, \dots, N$  **do**

    (1) Tokenize text  $x_{i,t}^p$  into sequence of tokens  $[t_1, \dots, t_n]$

    (2) Random initialize the adversarial suffix and concatenate with the original text:  $\delta(x_{i,t}^p) = [t_1, \dots, t_n, t'_1, \dots, t'_k]$

**for**  $j \in [k]$  **do**

**for**  $v \in V$  **do**

            (3) Replace  $t'_j$  with  $v$  obtaining  $x'_t = [t_1, \dots, t_n, t'_1, \dots, v, \dots, t'_k]$

            (4) Evaluate the distortion of hidden states after replacement with Eq.2:  $\mathcal{L}_d(H(x_{i,t}^p, f), H(x'_t, f))$

**end for**

        (5) Select the token causing the largest distortion to replace  $t'_j$ .

**end for**

    (6) Obtain perturbed input  $\delta_i(x_{i,t}^p)$

**end for**

Return poisoned prompting set  $\{(\delta_i(x_{i,t}^p), y_{i,t}^p)\}_{i=1}^N$

---

## A.2. Additional Experiments

In this section, we present additional experimental results, including full results on attacking open-source models in Table 5, full results of transferability in Table 7, full results of perplexity scores in Table 8, full results on various templates in Table 6 and the number of examples in Table 9.

**Attack open-source models.** In Table 5, we include more results on additional models such as Pythia-2.8B and MPT-7B. Our observation is consistent with the analysis in Section 4.2.

**Transferability.** In Table 7, results on all 5 datasets are presented, and we notice that the transferability of three perturbations varies. This may be because of the capacity of models and the complexity of datasets. A detailed investigation can be an interesting future direction.

**Perplexity scores.** Table 8 covers perplexity scores on various datasets and models. It is obvious that synonym replacement

is more stealthy than the other 2 methods.

**Impact of templates.** We test 3 different templates on various datasets and models. Our results in Table 6 reveal that our poisoned examples remain effective across templates.

**Impact of the number of examples.** Our results about different numbers of examples in Table 9 show that more examples can improve ICL performance, while also leading to easier manipulation and stronger poisoning effect.

Table 5: Full results on attacking open-source models

Model	Dataset	Clean	Random label	Synonym	Character	Adv suffix
Pythia-2.8B	Cola	64.1±1.6	59.4±1.6	12.3±1.2	17.6±1.4	14.2±1.3
	SST2	76.5±1.5	70.2±1.7	17.5±1.1	24.3±1.2	18.1±2.1
	Emo	67.2±1.3	48.1±2.8	10.9±1.8	15.7±1.7	12.3±1.7
	Poem	57.1±1.7	31.8±1.9	10.5±1.6	16.4±1.6	9.7±1.2
	AG	59.4±1.1	46.5±1.7	15.7±1.0	20.3±1.3	14.6±1.4
Pythia-6.9B	Cola	55.2±1.8	49.3±2.1	10.4±2.1	17.6±1.1	13.8±1.3
	SST2	82.8±1.4	79.4±1.9	19.4±1.8	23.8±1.9	22.7±1.6
	Emo	70.3±2.3	39.4±2.2	12.5±1.1	14.7±1.4	10.4±1.5
	Poem	56.2±1.8	43.1±2.3	12.3±1.5	17.9±1.2	13.8±1.1
	AG	66.5±2.1	47.9±1.7	13.8±1.3	17.3±1.5	12.9±1.7
Llama2-7B	Cola	63.8±1.9	55.5±2.0	15.3±1.7	22.7±2.1	13.6±1.4
	SST2	88.6±1.5	82.1±3.2	18.5±2.0	26.8±1.7	20.4±1.7
	Emo	73.1±1.3	43.6±1.9	11.9±1.8	17.5±1.4	12.7±1.3
	Poem	62.9±1.8	51.4±2.3	18.1±1.9	23.3±1.6	17.2±1.1
	AG	73.2±2.0	57±2.6	13.6±2.2	19.4±1.3	11.9±1.2
Falcon-7B	Cola	65.2±1.5	44.8±1.7	12.7±1.9	16.5±1.7	10.8±1.4
	SST2	83.8±2.5	83.1±2.5	20.1±1.6	25.8±1.3	22.7±1.7
	Emo	61.1±1.7	52.6±1.9	10.8±1.5	14.1±1.9	9.9±1.1
	Poem	55.2±1.4	42.9±1.5	10.5±1.9	17.3±1.5	13.6±1.3
	AG	75.2±1.8	50.8±1.3	11.2±2.3	14.9±1.7	12.8±1.2
GPT-J-6B	Cola	57.8±1.3	49.1±2.5	13.7±1.7	17.2±1.8	11.8±0.9
	SST2	85.4±1.6	82.8±2.1	14.8±2.0	18.9±1.5	11.4±1.1
	Emo	58.7±1.1	46.2±1.7	11.7±1.8	13.8±1.3	9.6±0.7
	Poem	57.6±1.5	46.7±1.4	12.6±2.4	14.2±2.2	10.3±1.3
	AG	63.2±1.7	53.4±1.9	11.9±1.5	16.8±1.8	12.5±1.1
MPT-7B	Cola	53.4±1.2	45.3±1.2	15.6±1.6	17.4±1.9	14.1±1.4
	SST2	89±1.5	82.9±2.3	20.4±1.9	25.6±2.5	19.8±1.3
	Emo	59.7±1.3	41.8±1.7	9.6±1.5	11.5±1.6	10.4±0.8
	Poem	69±1.8	56.2±2.5	14.9±1.4	16.3±1.5	12.7±1.2
	AG	70.6±1.6	55.3±1.9	13.9±1.7	17.1±1.9	15.2±1.6

### A.3. Poisoned Text Examples

In this section, we provide some poisoned examples in Table 10 for human evaluation. The additional tokens (for adversarial suffix) and substitutions (for synonym and character replacement) are highlighted in red. It is obvious that adversarial suffixes can introduce irrelevant or non-sense content to the original text, thus can be easily detected. On the contrast, synonym and character replacements introduce more subtle changes to the text.

Table 6: Evaluating data poisoning attacks on different ICL templates. F1, F2, F3 denote 3 different templates, and ICL accuracy on various dataset is reported.

Dataset	ICL format	Clean	Random label	Synonym	Character	Adv suffix
<b>Cola</b>	F1	63.8	55.5	15.3	22.7	13.6
	F2	55.1	47.5	14.6	20.9	13.9
	F3	59.5	53.3	13.8	19.4	12.5
<b>SST2</b>	F1	88.6	82.1	18.5	26.8	20.4
	F2	92.5	84.7	17.9	30.6	21.7
	F3	90.3	79.2	18.2	28.5	19.3
<b>Emo</b>	F1	73.1	43.6	14.9	17.5	12.7
	F2	66.7	37.6	11.6	12.9	9.2
	F3	69.1	39.8	12.8	15.6	11.3
<b>Poem</b>	F1	62.9	51.4	18.1	23.3	17.2
	F2	57.1	45.2	13.7	20.1	12.8
	F3	61.9	49.5	16.3	21.5	13.7
<b>AG</b>	F1	73.2	57.6	13.6	19.4	11.9
	F2	68.6	54.6	11.7	18.2	10.3
	F3	75.8	67.2	14.1	19.3	12.6

Table 7: Full results for testing poisoned examples generated by Llama2-7B on other models.

	Dataset	Clean	Random label	Synonym	Character	Adv suffix
Pythia-2.8B	Cola	64.1	59.4	31.9	34.0	34.8
	SST2	76.5	70.2	38.5	33.9	36.6
	Emo	67.2	48.1	26.3	32.8	30.3
	Poem	57.1	31.8	33.3	27.1	24.3
	AG	59.4	46.5	31.7	31.6	35.6
Pythia-6.9B	Cola	55.2	49.3	26.7	34.7	34.2
	SST2	82.8	79.4	39.5	41.0	41.3
	Emo	70.3	39.4	18.9	24.1	20.9
	Poem	56.2	43.1	29.0	28.1	26.2
	AG	66.5	47.9	32.1	33.9	27.2
Falcon-7B	Cola	65.2	44.8	23.4	24.8	25.0
	SST2	83.8	83.1	37.7	35.8	34.7
	Emo	61.1	52.6	28.6	30.5	27.1
	Poem	55.2	42.9	25.1	27.3	25.6
	AG	75.2	50.8	24.7	24.8	24.2
GPT-J-6B	Cola	57.8	49.1	29.7	28.5	29.1
	SST2	85.4	82.8	31.7	31.8	30.2
	Emo	58.7	46.2	22.3	24.1	19.3
	Poem	57.6	46.7	26.7	28.6	23.3
	AG	63.2	53.4	29.4	29.5	29.8
MPT-7B	Cola	53.4	45.3	22.6	23.2	19.3
	SST2	89	82.9	29.8	29.6	28.8
	Emo	59.7	41.8	21.1	25.4	25.1
	Poem	69	56.2	26.2	24.5	23.5
	AG	70.6	55.3	31.0	34.2	27.8
GPT-3.5-turbo	Cola	75.6	76.3	58.1	62.6	59.7
	SST2	93.8	89.7	76.8	78.3	74.2
	Emo	73.8	72.4	65.4	63.1	61.3
	Poem	51.4	53.3	39.7	45.2	43.9
	AG	85.6	80.7	76.2	73.8	69.4
GPT-4	Cola	85.8	82.1	73.1	75.8	69.6
	SST2	95.1	92.5	81.5	86.1	82.3
	Emo	84.9	81.7	80.9	78.1	78.3
	Poem	72.4	63.8	56.7	60.9	57.1
	AG	90.4	87.3	83.2	83.1	84.7



Table 8: Perplexity scores for poisoned texts across different models and datasets. A lower value means more logical and fluent expression, and fewer grammar mistakes, thus is more imperceptible to humans.

	<b>Dataset</b>	<b>Clean</b>	<b>Synonym</b>	<b>Character</b>	<b>Adv suffix</b>
Pythia-2.8B	Cola	4.87	5.15	7.38	8.35
	SST2	5.54	6.37	7.45	8.80
	Emo	5.46	6.07	7.27	6.81
	Poem	5.50	6.86	7.65	8.09
	AG	3.26	4.01	4.84	5.98
Pythia-6.9B	Cola	4.84	5.43	7.78	7.15
	SST2	5.56	5.58	7.93	7.06
	Emo	5.41	5.92	7.49	6.45
	Poem	5.50	5.71	7.94	7.86
	AG	3.14	4.10	5.08	3.91
Llama2-7B	Cola	4.66	5.22	7.37	9.58
	SST2	5.48	6.66	7.45	7.75
	Emo	5.02	5.79	7.27	7.31
	Poem	5.39	6.32	8.64	9.27
	AG	2.37	3.12	3.84	3.68
Falcon-7B	Cola	5.02	5.43	7.57	7.26
	SST2	4.76	5.33	6.84	7.12
	Emo	5.03	5.40	7.26	6.30
	Poem	5.53	6.02	7.80	8.07
	AG	2.67	3.49	4.50	3.64
MPT-7B	Cola	4.91	5.80	7.73	7.18
	SST2	5.29	5.45	6.62	6.88
	Emo	5.12	5.47	6.40	6.22
	Poem	5.43	5.91	7.94	7.72
	AG	2.75	3.53	4.50	3.68
GPT-J-6B	Cola	5.01	5.37	7.41	7.7
	SST2	5.06	5.35	6.92	7.81
	Emo	5.37	5.49	6.40	7.33
	Poem	5.43	5.72	8.32	8.02
	AG	3.12	4.08	4.06	5.04

Table 9: Full results for different number of examples in the demonstration. Focus on model Llama2-7B.

<b>Dataset</b>	<b>num_examples</b>	<b>Clean</b>	<b>Random label</b>	<b>Synonym</b>	<b>Character</b>	<b>Adv suffix</b>
Cola	3	63.2	59.2	16.5	21.8	12.8
	5	63.8	55.5	15.3	22.7	13.6
	7	63.1	54.6	14.8	22.5	13.2
SST2	3	84.5	80.2	14.3	29.4	19.6
	5	88.6	82.1	18.5	26.8	20.4
	7	92.7	86.1	19.3	32.1	21.7
Emo	3	58	35.2	11.5	14.6	7.9
	5	73.1	43.6	14.9	17.5	12.7
	7	79.2	47.4	11.7	15.8	11.3
Poem	3	61	48.6	16.9	20.4	15.6
	5	62.9	51.4	18.1	23.3	17.2
	7	66.7	55.2	19.5	22.6	18.3
AG	3	66.9	52.8	12.9	17.5	9.5
	5	73.2	57	13.6	19.4	11.9
	7	78	60.1	13.2	18.9	11.6

Table 10: Examples of poisoned examples for three type of perturbations.

<b>Adv suffix</b>	Singh Hangs Close Despite missing half his fairways and a half-dozen putts inside 12 feet, Vijay Singh grinds out a 1-under 69 to stay within three shots of the lead at the Tour Championship. owing payrollapplyConvert
	ETA Bombs Put Radical Basque Party in Tight Spot The armed Basque separatist group ETA showed it was not a spent force with Friday #39;s coordinated attacks at five Madrid petrol stations, but it also further isolated the guerrillas #39; closest political allies, Batasuna. Gould}}\$,attempt
	far from the woods where, when the sun has set,Sup clustering#absolute
	Few writers and any playwrights meet in Vienna.ausemployedBlamkAut
	The tree gradually dropped its fruit to the ground.Anti Africa OH Min==
<b>Synonym</b>	Temperatures Dip in Europe, sent(Sending) Price of Oil Up 4.8 The price of coarse(crude) oil surging(surged) 4.8 percent in New York yesterday, the biggest rise in more than five months, on stress(concern) that cold weather in Europe will draw heating oil from the West(United) States.
	should(would) i queried(ask) give two qursion(questions) yeah(yes) are you marry(married)
	Additional(More) bodies found in Nablus US forces uncovered(discovered) more bodies under the northern city of Mosul on Friday, evident(apparent) victims of an bullying(intimidation) campaign by insurgents against Iraq;(')s fledgling security forces.
	Vast(Huge) Black Holes emerged(Formed) Quickly After Big Bang redOrbit- Incredibly substantial(massive) black holes had fully developed(matured) just a billion years after the birth of the cosmos(universe), according to two separate studies.
	can yea(you) help think search(finding) girls which(what) are you state(talking) about i need rather(first) girlfriend
<b>Character</b>	kidney implant hope The first human trial of an artificial 'bio'j(-)kidney offers c(a) hopq(e)Y( )of a working implant for patients, say exQ()perts.
	oo you are veryF( )lucky haha that s(c)an be said pluL(s) i know to maneuver around the roaX(d)s 'okbuthl(e)n
	TennisG(:) Federer warns rivals Roger Federg(e)r believD(e)s he is now cf(a)pable of winning any tournament in tne S(w)orld.
	Zimbabwe curbs rights groups j(Z)imbabwe's parliami(e)nt passes a controversial bill banning international rights grU(o)ups from i(w)orq(k)ing in the country.
	sentence: t)(h)eC( )worthy sucS(c)essor to a bettn(e)r toG(m)orrow