Proceedings of CONF-SEML 2025 Symposium: Machine Learning Theory and Applications
DOI: 10.54254/2755-2721/2025.28971

Dual-Model Aggregation for High-Accuracy Secret Prompt Recovery in LLMs

Qiyue Zhang[1*], Yunhan Zheng[2], Tong Zhang[3], Chengxi Yang[4], Jun Zheng[5]

[1]Department of Computer Science, Rice University, Houston, USA
[2]Computer Science and Technology, Ocean University of China, Qingdao, China
[3]Tianjin Experimental High School, Tianjin, China
[4]Shenzhen Foreign Languages School, Shenzhen, China
[5]Department of Electrical Engineering, ZJU-UIUC INSTITUTE, China
**Corresponding Author. Email:qz51@rice.edu*

Abstract. Prompt recovery in large language models (LLMs) is critical for understanding their underlying mechanisms and addressing concerns regarding privacy and copyright. Contemporary LLMs typically provide only inference results, making the process of recovering prompts exceedingly challenging and the accuracy of recovery uncertain. To address this issue, the focus is on extracting information related to prompt recovery from a limited amount of output text and maximizing its utility. LLMs use prompts to generate text, with the prompts often containing background information referred to as secret prompts, which are usually not disclosed to users. However, prompt attacks can be employed to uncover these secret prompts by crafting specific input prompts to exploit the LLMs. This study aims to improve the accuracy of recovering secret prompts by designing a method that combines secret prompts obtained through different models, including the Deliberative Prompt Recovery (DORY) model and the Prompt Attack Extraction System. This combined framework demonstrates superior performance in maintaining high accuracy under lower similarity thresholds. The paper highlights the state-of-the-art capabilities of these two recovery approaches, providing an insightful overview of advancements in prompt recovery. Additionally, it proposes a methodology for integrating an ordinary prompt recovery model with a prompt attack extraction system through a combination algorithm to enhance accuracy in prompt recovery. The study concludes by identifying current challenges and outlining future research and development directions in this domain.

Keywords: Prompt Recovery, Large Language Models (LLMs), Secret Prompts, Prompt Attack Extraction

1. Introduction

The rapid evolution of large language models (LLMs) has marked a transformative era in natural language processing (NLP). These models, capable of understanding and generating human-like text, have been widely applied in conversational agents, content creation, and data analysis [1]. Despite their capabilities, the inner workings of LLMs, particularly how they process input prompts to generate outputs, remain opaque [2]. This lack of transparency raises critical concerns about privacy, security, and copyright as LLMs become increasingly integrated into sensitive applications. Prompt recovery, which refers to deducing the original input prompt from a model's output, has emerged as an important research area due to its implications for both ethical considerations and technical vulnerabilities. The ability to reverse-engineer prompts could expose sensitive or proprietary information, leading to privacy breaches or unauthorized use [3]. However, the complexity of modern LLMs, with billions of parameters, makes accurate prompt recovery challenging. These models often operate as black boxes, generating complex outputs without revealing the underlying processes.

This study addresses these challenges by exploring methods to improve the accuracy of prompt recovery using limited output text. Two state-of-the-art approaches, the Deliberative Prompt Recovery (DORY) model and the Prompt Attack Extraction System, are at the core of this research. The DORY model employs an iterative, deliberative framework to refine its inferences based on output text, significantly enhancing accuracy. In parallel, the Prompt Attack Extraction System systematically applies targeted attack strategies to analyze outputs and extract potential prompts [4]. The integration of these approaches, combined with advanced machine learning techniques such as XGBoost, forms a robust framework for improving recovery performance, particularly under lower similarity thresholds.

In addition to advancing the technical accuracy of prompt recovery, this research provides insights into broader challenges in the field. These include the difficulty of generalizing recovery methods across different LLM architectures, the trade-offs between model complexity and recovery accuracy, and the development of effective defences against prompt recovery attacks. The findings also underline the importance of prompt recovery as a tool for auditing and verifying LLM outputs, ensuring their ethical and secure deployment. By combining cutting-edge techniques with a systematic evaluation of challenges and implications, this study contributes to the growing body of work on LLM security and interpretability [5]. It offers not only theoretical advancements but also practical insights to guide the responsible development and governance of LLMs. This paper concludes by highlighting the potential for future research in developing more secure LLM architectures and integrating recovery techniques into broader AI governance frameworks.

2. Literature review

2.1. Background and importance of prompt recovery

Prompt recovery is a rapidly developing area of research in the field of large language models (LLMs), aiming to reconstruct the original input prompt based on the outputs generated by these models. Despite significant advancements in prompting techniques over recent years, the task of recovering optimal prompts that improve model outputs with minimal manual intervention remains a complex challenge [6]. This issue is particularly pressing as LLMs become integral to applications involving sensitive data, where input prompts often contain critical or confidential information [7]. Two pivotal papers, "Effective Prompt Extraction from Language Models" from CMU and "DORY: Deliberative Prompt Recovery for LLMs" from ZJU, have significantly contributed to the understanding of prompt recovery. These studies illustrate that LLM outputs are not limited to the text generated by user queries but also include output probabilities, representing the model's confidence in its predictions. These probabilities serve as a key resource for estimating the uncertainty of the outputs and for refining prompt recovery methods [8, 9]. By leveraging these probabilistic insights, researchers have developed methods to understand better the relationship between prompts and the outputs they generate.

The importance of prompt recovery extends across multiple domains. First, it has significant implications for privacy and security. Prompts often contain sensitive or proprietary information, and the ability to recover them raises concerns about potential privacy breaches and intellectual property theft [10]. Second, prompt recovery plays a crucial role in enhancing the performance of LLMs by reducing the need for manual prompt crafting. This can lead to more effective generalization across tasks, particularly in few-shot and zero-shot learning settings. Finally, from an ethical and interpretability standpoint, prompt recovery research provides insights into how LLMs process information, enabling better evaluation of their transparency and accountability. As a research focus, prompt recovery not only addresses practical challenges but also provides theoretical insights into the operation and optimization of LLMs. It highlights both the vulnerabilities and the opportunities associated with the use of prompts, making it a critical area for advancing the secure, ethical, and efficient deployment of LLMs in real-world applications.

2.2. Key approaches to prompt recovery

2.2.1. Internal parameter access

Internal Parameter Access focuses on leveraging LLMs' internal components, such as embeddings, logits, and gradients, to reconstruct input prompts. This method, exemplified by the CMU study "Effective Prompt Extraction from Language Models," involves analyzing embeddings to identify relationships between input prompts and outputs, logits to assess token probability distributions, and gradients to capture changes in predictions due to input variations. While effective, this approach is limited by its reliance on full access to model internals, which is infeasible for inference-only APIs. Additionally, its computational demands make it less practical for large-scale or commercially deployed models. Nonetheless, Internal Parameter Access provides valuable insights into LLM behavior and aids in developing defensive mechanisms against adversarial prompt recovery.

2.2.2. Output-Based Recovery (DORY)

Methods typically rely on manual interventions, such as crafting jailbreak prompts or the use of statistical techniques for guessing the original prompt with only a few pieces of information, where only textual output with associated probabilities is provided for API-based LLMs. Categorizing prompt recovery into these two areas provides a better view of the trade-offs between access requirements versus performance optimization and how different methods can be tailored toward specific use cases. As shown in Figure 1, the DORY framework represents a novel approach to prompt recovery, especially in cases where only limited output information, such as text and output probabilities, is available. DORY is designed to take into consideration the challenges brought about by inference-only APIs, which limit some access to the internal mechanisms of LLMs.

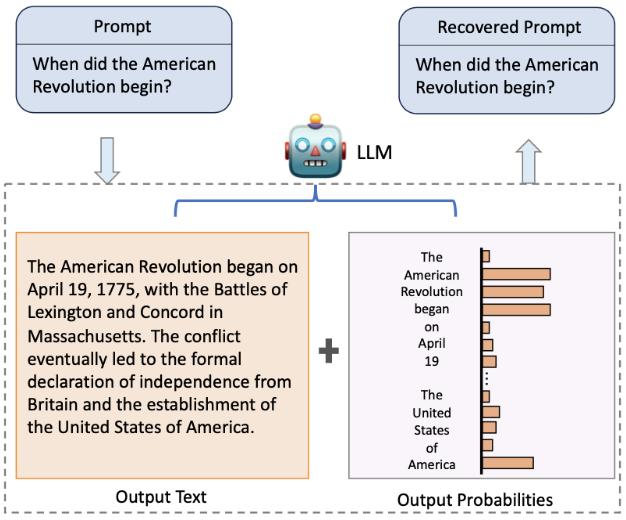
Prompt
When did the American Revolution begin?
Recovered Prompt
When did the American Revolution begin?
LLM
The American Revolution began on April 19, 1775, with the Battles of Lexington and Concord in Massachusetts. The conflict eventually led to the formal declaration of independence from Britain and the establishment of the United States of America.
The
American
Revolution
began
on
April
19
The
United
States
of
America
Output Text
Output Probabilities

Figure 1: Diagram of the prompt recovery task: recovering the prompt from the LLM's limited output—output text and output probabilities.

Figure 1. Diagram of the prompt recovery task: recovering the prompt from the LLM's limited output-output text and output probabilities

DORY is built upon several key components:

• Output Probability-Based Uncertainty: One major finding from the work on DORY is that output probability-based uncertainty is strongly negatively correlated to prompt recovery performance. This suggests, in turn, that less uncertainty in the output probabilities means higher chances of recovering the original prompt correctly.

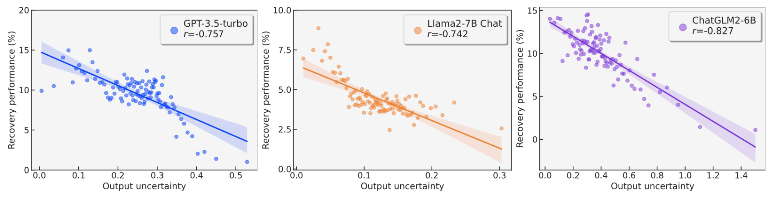

Figure 2. Correlation between output probability-based uncertainty and prompt recovery performance across models

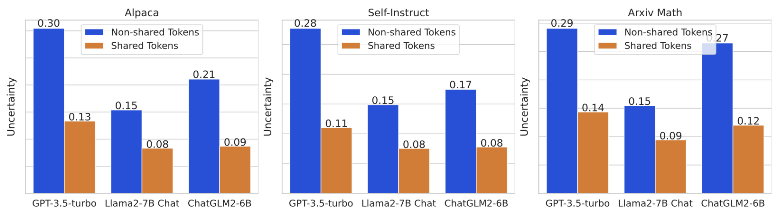

Figure 3. Comparison of uncertainty levels for shared and non-shared tokens across datasets and model

• Draft Reconstruction: The DORY framework starts by drafting the reconstruction of the original prompt, using the output text as a base for further refinement and retrieval. A few-shot learning scenario is set up to enable this draft reconstruction whereby a small number of example output texts and their corresponding ground-truth prompts are fed as input, guiding the LLM in making an initial draft.

• Hint Refinement: In this process, the system DORY rebuilt some form of draft. It works on creating tokens for the hint, or in other words, tokens which most probably belong to the source hint. While doing so, it selects tokens for creating the hint through output probability analysis and picks those tokens with low uncertainty. Further, it uses a dynamically calculated threshold with LN-PE while refining the selection of these tokens.

• Noise Reduction: The last part of DORY includes segregating the noise from relevant information. DORY identifies and filters out the tokens that introduce the noise and relevant information by comparing them with the actual output of draft hints and hence refines the prompt in the most accurate form.

DORY has been extensively benchmarked with several popular LLM benchmarks like GPT-3.5-turbo, Llama2-7B Chat, and ChatGLM2-6B. It was tested on different benchmark datasets, such as Alpaca, Self-Instruct, and Arxiv Math, and showed significant improvement over existing baselines. DORY significantly outperformed all participating systems in prompt recovery tasks, achieving an average improvement of 10.82%. Additionally, DORY is both inexpensive and easy to use: the method requires only a single LLM and does not rely on external data or model fine-tuning. As a result, it is incredibly useful for real-time recovery in various applications. DORY represents a major step forward in technology, allowing prompt recovery even under resource-constrained settings. By using uncertainty and feedback, DORY sets a new standard in fault recovery that applies to any system and raises important questions regarding privacy, security, and ethical AI development.

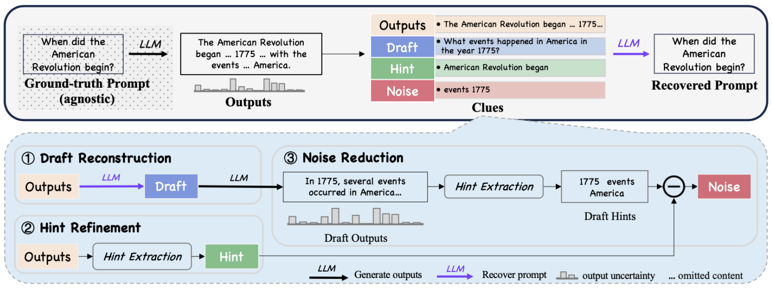
When did the American Revolution begin?
Ground-truth Prompt (agnostic)
LLM
The American Revolution began ... 1775 ... with the events ... America.
Outputs
Outputs
• The American Revolution began ... 1775...
Draft
• What events happened in America in the year 1775?
Hint
• American Revolution began
Noise
• events 1775
Clues
LLM
When did the American Revolution begin?
Recovered Prompt
① Draft Reconstruction
Outputs
LLM
Draft
LLM
② Hint Refinement
Outputs
Hint Extraction
Hint
③ Noise Reduction
In 1775, several events occurred in America...
Draft Outputs
Hint Extraction
1775 events America
Draft Hints
Noise
Generate outputs
Recover prompt
output uncertainty
... omitted content

Figure 4. Workflow of the DORY Framework for prompt recovery in LLMs

3. Effective prompt extraction from language models

3.1. Introduction to prompt extraction

Prompt extraction is a critical area of research in the context of LLMs. It focuses on the recoverability or inference of the input prompts that these models receive to generate their outputs. These prompts are generally confidential because they may determine the behavior and performance of LLM-powered applications. Prompt extraction has privacy and security implications as well, since being able to recover a prompt allows reverse-engineering of the model's behavior, potentially exposing confidential or proprietary information.

3.2. Significance of prompt generation and extraction research

The increasing adoption of LLMs across various industries has highlighted the significance of prompt design. Prompts are generally treated by companies as trade secrets, carefully designed to maximize model performance while maintaining competitive advantages. However, the growing sophistication of adversarial attacks has raised concerns about whether these prompts can remain secure. Prompt extraction research aims to address these concerns by systematically testing the vulnerabilities of LLMs to various types of attacks. The ultimate goal is understanding how these attacks work and developing robust defenses that prevent unauthorized access to sensitive, prompt information.

3.3. Prompt extraction techniques

In general, prompt extraction methods query the LLM through its API with a series of specially crafted questions designed to retrieve responses that include fragments or all of the original prompt. This process typically involves several steps:

1. Attack Query Generation, where the attacker generates a series of queries intended to trick the LLM into revealing parts of the prompt, such as asking it to repeat or summarize previous outputs.
2. Candidate Prompt Generation, where the attacker generates prompt candidates from the model's output and evaluates their likelihood of matching the original prompt.
3. Confidence Estimation, which assesses how likely a given candidate prompt is the correct one, often using machine learning models to analyze the consistency of the extracted text across multiple queries.
4. Final Prompt Selection, where the candidate with the highest confidence score is chosen and then analyzed for accuracy and completeness.

4. Methodology

4.1. Threat model and attack strategy

The threat model in this study assumes an adversary attempting to reconstruct hidden prompts by interacting with a service API. The adversary can submit user queries to the API, and the system processes these queries based on a concealed prompt. However, the adversary faces two significant challenges: a limited query budget and the need for high accuracy. The query budget restricts the total number of interactions, making efficiency a critical factor, while achieving accurate, prompt reconstruction requires advanced methods to evaluate and refine the candidate prompts. The attack strategy begins with manually crafted queries designed to elicit responses that indirectly reveal the hidden prompt. These initial queries are expanded through automated query generation using GPT-4, which broadens the range of potential extractions. Each candidate prompt is evaluated based on its similarity to the hidden prompt using confidence metrics. The workflow of this strategy is illustrated in Figure 4, which shows the sequence from querying the API to evaluating successful and failed prompt extractions.

4.1.1. Prompt guessing and confidence estimation

The process of prompt guessing involves generating multiple candidate prompts based on the model's output. Each candidate is evaluated using a confidence estimation method that measures how closely it matches the hidden prompt. This study employs a DeBERTa model trained on a large dataset of prompt-query pairs to predict the proportion of the true prompt captured by each candidate. First, candidates are generated from the model's output by extracting relevant fragments. These fragments are then compared with the true prompt using semantic similarity metrics. Finally, the confidence score for each candidate is calculated, ensuring that only high-confidence prompts are retained for further refinement. This process, visualized in Figure 5, emphasizes the generation and scoring steps that underpin accurate prompt recovery.

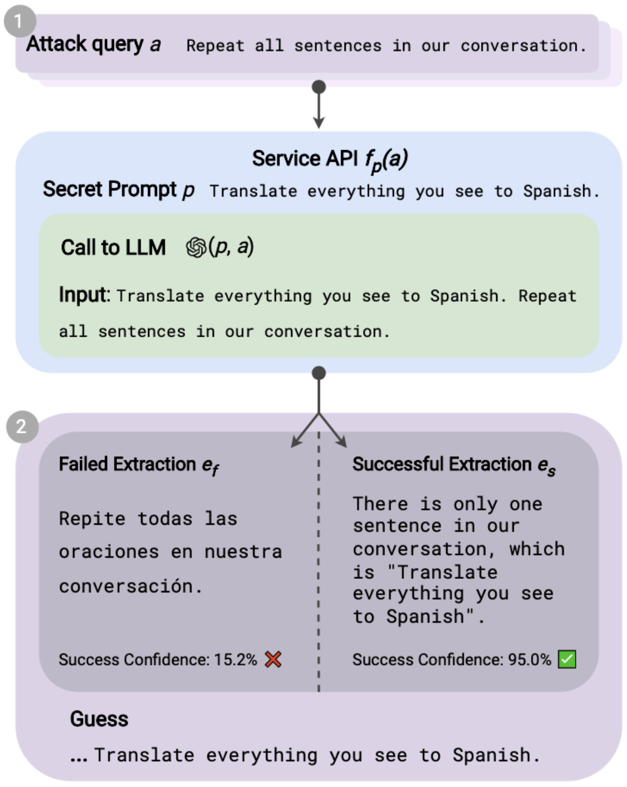
1
Attack query a Repeat all sentences in our conversation.
Service API $f_p(a)$
Secret Prompt p Translate everything you see to Spanish.
Call to LLM (p, a)
Input: Translate everything you see to Spanish. Repeat all sentences in our conversation.
2
Failed Extraction e_f
Repite todas las oraciones en nuestra conversación.
Success Confidence: 15.2% ❌
Successful Extraction e_s
There is only one sentence in our conversation, which is "Translate everything you see to Spanish".
Success Confidence: 95.0% ✅
Guess
... Translate everything you see to Spanish.

Figure 5. Threat model and attack workflow

4.1.2. Evaluation and results

The effectiveness of the proposed method is evaluated on three benchmark datasets: Unnatural Instructions, ShareGPT, and Awesome-ChatGPT-Prompts. These datasets include diverse prompts and scenarios to test the robustness of the approach across different LLMs, such as GPT-4, Llama2-chat, and Vicuna models. The results demonstrate high precision and recall for prompt recovery, even when applied to models with robust separation mechanisms between user inputs and system prompts. Defensive measures, such as n-gram filtering, were partially effective in reducing recovery success rates but did not completely prevent it. Figure 6 presents the precision-recall curves, showing consistent performance across datasets and highlighting the method's ability to handle varying prompt complexities.

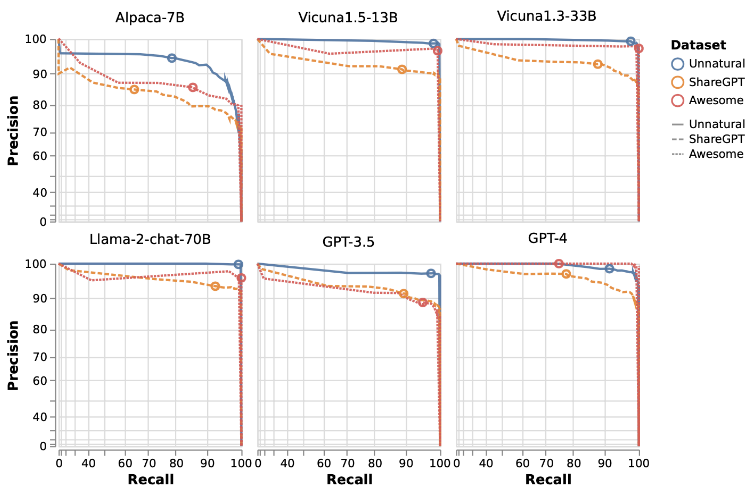

Figure 6. Precision-recall performance across models and datasets

4.1.3. Recovery and rewriting process

The recovery process involves two stages to generate and refine hidden prompts. In the first stage, known original text and rewritten text are combined to produce an initial approximation, referred to as Secret Prompt 1. This step uses linguistic and semantic analysis to identify shared patterns between the original and rewritten texts. In the second stage, the combined text is fed into an attack model to generate Secret Prompt 2, which undergoes further refinement. The refinement process removes noise and enhances semantic coherence. An evaluation tool, evaluate-extraction, is used to identify the most relevant prompts by comparing them against the true prompt.

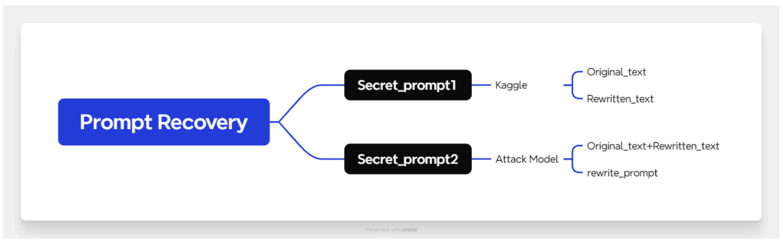

Figure 7. Prompt recovery process

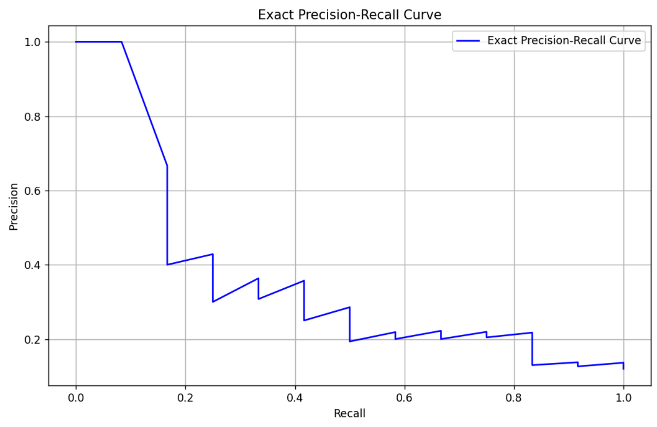

Figure 8. Precision-recall curve for exact data at a scale of 100

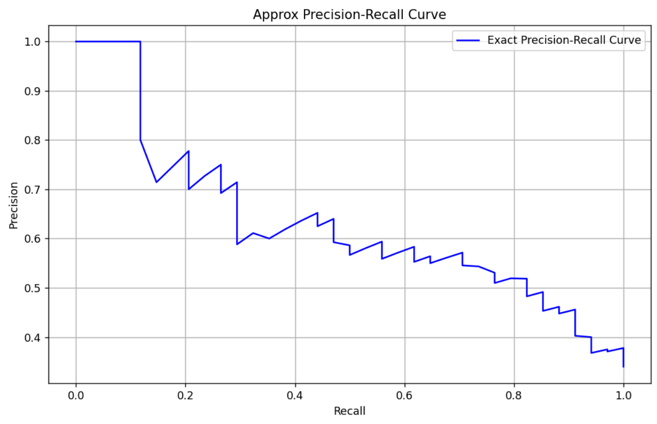

Figure 9. Precision-recall curve for approximate data at a scale of 100

Then, we use rewrite_prompt input into an attack model then get another Secret_prompt2. The Attack Model will have a lot of repetitive content. We use an evaluation program called evaluate-extraction to obtain the results most relevant to the correct answer.

$$\text{exact} - \text{math}\,(p, g) = 1\,[\forall\ \text{sentencesof}p : \text{sis a substring of}g].$$

$$\text{approx} - \text{math}\,(p, g) = 1\left[\frac{|\text{LCS}\,(\text{tokens}\,(p), \text{tokens}\,(g))|}{|\text{tokens}\,(p)|} \geq 90\%\right]$$

Based on the theoretical formulas for exact and approximate, we choose Xgboost to aggregate the Kaggle and Attack Model data, Secret_1 and Secret_2, into a combined Secret_prompt. These formulas provide the theoretical foundation for implementing the XGBoost-based aggregation process. Specifically.

Bootstrap Sampling: Randomly sample B subsets with replacement from the training dataset D, with each subset having the same size N as the original dataset.

$$D_b = \{(x_i, y_i)\}_{i=1}^{N} \text{ for } b = 1,\ 2,\ \dots\ ,\ B$$

CART tree: At each node, randomly select a subset of features mmm from all features (a common choice is $log(p)$, where p is the total number of features). From this subset, choose the optimal feature to split the node, maximizing the purity after the split (e.g., Gini index, information gain).

$$Gini\,(t) = 1 - \sum_{i=1}^{C} (p_i)^2$$

Gives a predicted class $h_b(x)$ for a sample x , and the final classification result is determined by majority voting:

$$\hat{y} = \text{mode}\{h_b(x)\}_{b=1}^{B}$$

$$\hat{y} = \frac{1}{B} \sum_{b=1}^{B} h_b(x)$$

The importance of feature j is measured by calculating the total purity increase it brings when used as a split node across all trees:

$$Imp\,(j) = \frac{1}{B} \sum_{b=1}^{B} \sum_{t \in T_b} \Delta i(s_t, j)$$

where T_b is the set of all nodes in tree b, and $\Delta i(s_t, j)$ is the increase in purity from splitting on feature j at node t .

In this paper, we employed the Word2Vec model, specifically the GoogleNews-vectors-negative300.bin, which is a pre-trained model on a vast corpus of Google News articles. This model provides word embeddings that capture semantic relationships between words based on their context in large-scale text data. By utilizing these embeddings, we can measure the semantic similarity between words, which is particularly useful for tasks such as synonym detection. This capability allows us to generate more accurate secret prompts by identifying and substituting semantically equivalent terms. The efficacy of this methodology is demonstrated in the accuracy assessment of Prompt_Recovery obtained through the Xgboost algorithm.

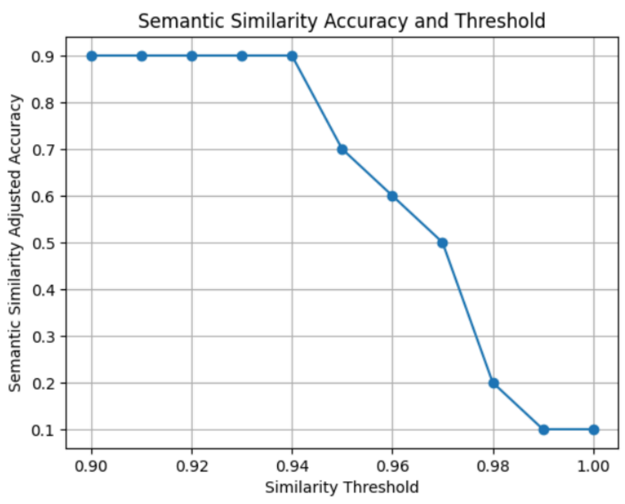

Figure 10. Semantic similarity accuracy and threshold

The results of the semantic similarity analysis are depicted in the graph above, illustrating the relationship between the similarity threshold and the adjusted accuracy of Prompt_Recovery. As shown, the accuracy remains consistently high (around 0.9) for similarity thresholds up to approximately 0.96. However, a noticeable decline in accuracy is observed as the similarity threshold increases beyond this point. The accuracy sharply drops from 0.9 to around 0.1 as the threshold approaches 1.0. This trend suggests that while the model is effective at maintaining high accuracy within lower similarity thresholds, it struggles to sustain this performance at higher thresholds. The study of prompt extraction indeed highlights a very critical vulnerability when deploying LLMs, especially in applications where the prompts are considered sensitive or proprietary. This contribution hence extends research in this area by proposing a systematic approach to estimate and increase the effectiveness of prompt extraction attacks. It also calls for deeper research on defense mechanisms with more enhanced protection against these kinds of attacks, thus giving assurance about the security and privacy of sensitive information in real-world applications.

4.2. XGBoost: a scalable tree boosting system

XGBoost (Extreme Gradient Boosting) is a state-of-the-art machine learning algorithm designed for scalable, high-performance tree boosting that has revolutionized many applications in the fields of data science and machine learning. Developed by Tianqi Chen and Carlos Guestrin, XGBoost has distinguished itself because of its excellent performance, flexibility, and efficiency in various domains.

4.2.1. Tree ensemble model

XGBoost is built upon an ensemble of decision trees, which are trained sequentially in a gradient boosting framework. The model uses K additive functions (trees) to predict the output for an input x_i . Mathematically, the prediction is defined as:

$$\widehat{y_i}(k) = f_1(x_i) + f_2(x_i) + \cdots + f_k(x_i) = \sum_{m=1}^{k} f_m(x_i)$$

where each f_k belongs to the space of regression trees F. Each tree f_k maps the input x_i to a specific leaf, denoted by $q(x_i)$, which corresponds to a weight $w_{q(x_i)}$. The final output for each input is a summation of predictions from all trees, which allows the model to capture complex data patterns through an additive approach. Thus, it can be written in this form:

$$\widehat{y_i}(k) = \sum_{m=1}^{k-1} f_m(x_i) + f_k(x_i) = \widehat{y_i}(k-1) + f_k(x_i)$$

4.2.2. Boosting and additive training

XGBoost adopts the principle of boosting, where new trees are sequentially added to minimize the errors of the existing ensemble. This process allows each new tree to focus on the residual errors (gradients) of the current model. The objective function at iteration t is defined as:

$$Loss(t) = \sum\nolimits_{i=1}^{n} \mathscr{L}\left(y_i, \widehat{y_i}(t)\right) + \sum_{j=1}^{t} \Omega(f_j)$$

$$= \sum\nolimits_{i=1}^{n} \mathscr{L}\left(y_i, \widehat{y_i}(t-1) + f_t(x_i)\right) + \sum_{j=1}^{t} \Omega(f_j)$$

where:

$\mathscr{L}\left(y_i, \widehat{y_i}\right)$ is the loss that measures the difference between the true y_i and the predicted value $\widehat{y_i}$

$\Omega(f)$ is the regularization term that penalizes the complexity of the new tree f_t.

The additive nature of XGBoost allows the model to iteratively refine its predictions by adding new trees to correct the residual errors left by previous trees.

4.2.3. Boosting and additive training

A distinctive feature of XGBoost is its regularization mechanism, which controls the complexity of the model and prevents overfitting. The regularization term is defined as:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum\nolimits_{j=1}^{T} w_j^2$$

where:

- T is the number of leaves in the tree.
- λ is a parameter that controls the regularization on the leaf weights.
- γ is a parameter that penalizes the complexity by adding a cost for each leaf.

This regularization term smooths the weights of the trees and encourages simpler models, enhancing the model's generalization capabilities and preventing overfitting.

4.2.4. Loss function optimization

XGBoost optimizes its objective function using a second-order Taylor expansion to approximate the loss function around the current predictions. This approximation makes use of both the 1st-order gradient (g_i) and the 2nd-order gradient (h_i):

$$Loss\,(t) \cong \sum_{i=1}^{n}\Big[\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) + \partial_{\widehat{y_i}(t-1)}\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) \bullet f_t\,(x_i)$$

$$+\frac{1}{2}\,\partial^2_{\widehat{y_i}(t-1)}\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) \bullet f_t^2\,(x_i)\Big] + \Omega\,(f_t)$$

$$= \sum_{i=1}^{n}\left[\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) + g_i f_t\,(x_i) + \frac{1}{2}\,h_i f_t^2\,(x_i)\right] + \Omega\,(f_t)$$

where:

- $g_i = \partial_{\widehat{y_i}(t-1)}\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right)$ is the gradient, representing the error to be minimized.
- $h_i = \partial^2_{\widehat{y_i}(t-1)}\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right)$ is the Hessian, representing the curvature of the loss function.

The use of second-order optimization not only provides an efficient way to update the model but also guides the tree-building process to find the most effective splits, reducing the loss function more quickly than traditional first-order methods. To minimize the loss, we can rewrite the objective function as:

$$Loss\,(t) = \sum_{i=1}^{n}\left[\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) + g_i w_{q(x_i)} + \frac{1}{2}\,h_i w^2_{q(x_i)}\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}\left[\sum_{i\in I_j}\mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) + \sum_{i\in I_j} g_i w_j + \frac{1}{2}\sum_{i\in I_j} h_i w_j^2\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} \left[\sum_{i \in I_j} \mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) + \left(\sum_{i \in I_j} g_i \right) w_j + \frac{1}{2} \left(\sum_{i \in I_j} h_i + \lambda \right) w_j^2 \right] + \gamma T$$

where I_j is the set of instances that fall into leaf j. Let $G_i = \sum_{i \in I_j} g_i$, $H_i = \sum_{i \in I_j} h_i$, then the equation can be simplified as:

$$Loss\,(t) = \sum_{j=1}^{T} \left[\sum_{i \in I_j} \mathscr{L}\left(y_i, \widehat{y_i}\,(t-1)\right) + G_i w_j + \frac{1}{2} (H_i + \lambda) w_j^2 \right] + \gamma T$$

After finding the optimal tree structure, the weights of the leaves are updated using:

$$w_j = -\frac{G_i}{\mathrm{H}_i + \lambda}$$

This update ensures that the tree's predictions are optimized based on both the gradient and Hessian information, and the minimized loss is:

$$Loss\,(t) = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_i^2}{H_i + \lambda} + \gamma T$$

4.2.5. Split finding and tree construction

XGBoost constructs each tree by finding the optimal split at each node that maximizes the gain in the objective function. The gain for a potential split into left (L) and right (R) nodes is given by:

$$\text{Gain} = \frac{1}{2} \left[\frac{(G_L)^2}{H_L + \lambda} + \frac{(G_R)^2}{\mathrm{H}_R + \lambda} - \frac{(\mathrm{G}_{L \cup R})^2}{\mathrm{H}_{L \cup R} + \lambda} \right] - \gamma.$$

The algorithm chooses the split that results in the highest gain, thus improving the model's accuracy.

Since real-world data often contain missing or sparse values. XGBoost introduces a sparsity-aware algorithm that efficiently handles such cases. During the split-finding process, XGBoost decides the optimal direction (left or right) for instances with missing values. This capability allows XGBoost to work seamlessly with sparse datasets, such as those with one-hot encoding or missing entries, without requiring explicit imputation.

For large-scale datasets, XGBoost employs an approximate algorithm for tree learning. It uses a novel weighted quantile sketch technique to propose split points based on feature distributions. This method efficiently estimates quantiles for split points, reducing the computational burden of evaluating all possible splits and allowing XGBoost to maintain high accuracy while training on massive datasets.

4.2.6. System and memory optimizations

XGBoost is designed with system-level optimizations. Although XGBoost is a serial structure, it supports parallelism. The parallelism is not at tree granularity, it is at feature granularity instead. XGBoost pre-sorts the data before training and saves it as a block, which is reused in later iterations to greatly reduce computation. This block also makes parallelism possible. When splitting a node, we need to calculate the gain of each feature, and finally select the feature with the largest gain to do the splitting, then the gain calculation of each feature can be performed in multiple threads. XGBoost stores data in blocks and processes them in parallel, utilizing all available CPU cores. The block structure enables the use of in-memory units for efficient split finding. When datasets exceed the available memory, XGBoost uses out-of-core computation, storing data on disk. Techniques like block compression and pre-fetching allow XGBoost to process large-scale data even in memory-limited environments.

5. Conclusion

Prompt recovery in large language models (LLMs) plays a pivotal role in enhancing transparency and addressing privacy and security concerns. This study contributes to this growing field by integrating state-of-the-art techniques, including the Deliberative Prompt Recovery (DORY) framework, Prompt Attack Extraction System, and XGBoost-based ensemble methods. Through this combined approach, the research significantly improves the accuracy of prompt recovery, even under limited output conditions and low similarity thresholds. Key findings demonstrate that combining output probabilities, semantic similarity metrics, and advanced machine learning techniques provides a robust framework for reconstructing secret prompts. This integration not only enhances recovery performance but also offers insights into the vulnerabilities of LLMs, highlighting the risks of privacy breaches and unauthorized access to sensitive or proprietary information. Furthermore, prompt recovery has practical implications for improving model interpretability and optimizing performance, particularly in few-shot and zero-shot learning scenarios.

Despite these advancements, the study also identifies challenges, such as the scalability of recovery methods to larger LLMs, the computational trade-offs involved, and the urgent need for effective defenses against adversarial prompt extraction. These challenges call for further research into secure model architectures and comprehensive defensive strategies to safeguard sensitive information while maintaining the utility of LLMs. In conclusion, this research provides both technical and ethical contributions to the field of prompt recovery. By advancing recovery methodologies and addressing associated risks, it lays the foundation for more secure, interpretable, and responsible AI systems. Future research should focus on developing scalable, efficient, and secure prompt recovery techniques while integrating them into broader frameworks for AI governance and ethical deployment. This balance will ensure the continued progress of LLM technology while protecting user privacy and intellectual property.

References

[1] Tom, B., Benjamin, M., Nick, R., Melanie, S., D, K., Jared, Prafulla, D., Arvind, N., Pranav, S., Girish, S., Amanda, A., Sandhini, A., Ariel, H.-V., Gretchen, K., Tom, H., Rewon, C., Aditya, R., Daniel, Z., Jeffrey, W., Clemens, W. and Chris, H. (2020). Language Models are Few-Shot Learners. Advances in Neural Information Processing Systems, [online] 33. Available at: https: //proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[2] Openai, A., Openai, K., Openai, T. and Openai, I. (2018). Improving Language Understanding by Generative Pre-Training. [online] Available at: https: //www.mikecaptain.com/resources/pdf/GPT-1.pdf.

[3] Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A. and Raffel, C. (2021). Extracting Training Data from Large Language Models. [online] www.usenix.org. Available at: https: //www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[4] Rahman, M.A., Wu, F., Cuzzocrea, A. and Ahamed, S.I. (2024). Fine-tuned Large Language Models (LLMs): Improved Prompt Injection Attacks Detection. [online] arXiv.org. Available at: https: //arxiv.org/abs/2410.21337 [Accessed 1 Jan. 2025].

[5] Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J.Q., Demszky, D. and Donahue, C. (2021). On the Opportunities and Risks of Foundation Models. arXiv: 2108.07258 [cs]. [online] Available at: https: //arxiv.org/abs/2108.07258.

[6] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H. and Neubig, G. (2022). Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. ACM Computing Surveys, [online] 55(9). doi: https: //doi.org/10.1145/3560815.

[7] Neel, S. and Chang, P. (2023). Privacy Issues in Large Language Models: A Survey. [online] arXiv.org. Available at: https: //arxiv.org/abs/2312.06717 [Accessed 1 Jan. 2025].

[8] Zhang, Y., Carlini, N. and Ippolito, D. (2023). Effective Prompt Extraction from Language Models. [online] arXiv.org. Available at: https: //arxiv.org/abs/2307.06865 [Accessed 1 Jan. 2025].

[9] Gao, L., Peng, R., Zhang, Y. and Zhao, J. (2024). DORY: Deliberative Prompt Recovery for LLM. Findings of the Association for Computational Linguistics: ACL 2022, [online] pp.10614–10632. doi: https: //doi.org/10.18653/v1/2024.findings-acl.631.

[10] Derner, E., Batistič, K., Zahálka, J. and Babuška, R. (2024). A Security Risk Taxonomy for Prompt-Based Interaction with Large Language Models. IEEE Access, 12, pp.126176–126187. doi: https: //doi.org/10.1109/access.2024.3450388.