PROMPRINT: Prompt Fingerprinting via First-Token Response for LLM App Cloning Detection

Jungmin Lee
Hanyang University
Seoul, South Korea
lsmp12@hanyang.ac.kr

Peizhuo Lv
Nanyang Technological University
Singapore
peizhuo.lyu@ntu.edu.sg

Yeonjoon Lee*
Hanyang University
Ansan, South Korea
yeonjoonlee@hanyang.ac.kr

Abstract

As Large Language Model applications (LLM apps) become widespread, system prompts that determine app behavior are increasingly regarded as intellectual property, raising concerns about leakage. Recent studies show that this threat is no longer theoretical, revealing the prevalence of cloned apps replicating system prompts from others on real-world platforms. These clones pose risks of copyright infringement and malicious misuse, highlighting the need for early and reliable detection. In this paper, we propose PROMPRINT, a novel fingerprinting approach for detecting cloned LLM apps without exposing their system prompts. Motivated by the observation that different system prompts yield distinct first output token distributions for the same query, PROMPRINT optimizes queries that induce the LLM to generate a specific first output token associated with the given system prompt, resulting in distinctive query–first-token pairs. Experiments on four instruction-tuned LLMs show that generated pairs effectively identify the corresponding system prompts, achieving over 74% probability of generating the target token while remaining below 2.2% on average under other prompts. Furthermore, we demonstrate that our fingerprinting remains robust to partial system prompt modifications and effective under the injection of adversarial instructions.

1 Introduction

The rapid advancement of large language models (LLMs) has given rise to a new class of applications, commonly referred to as LLM apps. Unlike conventional mobile apps that require extensive code, the functionality of LLM apps is primarily determined by system prompts (Mao et al., 2025). This simplicity significantly lowers the barrier to development, enabling creators to design diverse apps with relatively lower effort. As a result, millions of LLM apps have proliferated, attracting tens of millions of monthly active users (Yan et al., 2025; Zhao et al., 2025; Shen et al., 2025), with platforms such as GPT Store (OpenAI, 2025) and Poe (Poe, 2025).

At the same time, the dependence of LLM apps on system prompts has led to a growing view of these prompts as intellectual property, raising concerns about their potential leakage (Sha and Zhang, 2024). Prior studies on prompt leakage, ranging from adversarial queries that induce the model to reveal its hidden system prompts to approaches that infer prompts from generated outputs, have demonstrated the feasibility of such leakage with high success rates (Zhang et al., 2024a; Xia et al., 2025). Indeed, recent research on real-world LLM app platforms has revealed a widespread presence of app cloning that replicates the system prompts of other apps (Xie et al., 2025). Cloned apps not only raise concerns about copyright infringement but also pose significant security risks, as malicious developers may imitate popular apps to attract users and then inject harmful behaviors such as redirecting users to malicious websites, underscoring the necessity of early detection (Hou et al., 2025). Since system prompts in LLM apps are typically hidden and only their outputs are observable, response similarity analysis has been considered as a means of cloning detection (Zhao et al., 2025; Xie et al., 2025). However, because LLMs are inherently probabilistic, simple output comparison is insufficient for reliable detection, highlighting the need for tailored approaches to LLM app cloning detection.

In this paper, we propose PROMPRINT, a novel fingerprinting approach for detecting cloned LLM apps without exposing their system prompts. The design is motivated by two key insights: different system prompts yield different responses to the same query, and the first output token is particularly informative about the system prompt and the user query due to the autoregressive nature of LLMs.

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 22960–22972

Based on these insights, we conduct pilot studies that suggest that different system prompts induce distinct first-token distributions for the same query. Leveraging this observation, we focus on identifying distinctive query–first-token pairs associated with specific system prompts.

Our approach consists of three stages: (1) target token selection, (2) query embedding optimization, and (3) discrete query generation. First, we select a target output token with relatively higher probability under the target system prompt than under other prompts among the top-k first output tokens obtained using an initial query. Next, we optimize embeddings of the initial query tokens using our designed loss function to increase the probability of the target output token under the target system prompt while suppressing it under other prompts. Finally, we transform optimized embeddings into discrete query tokens via beam search, which selects tokens from the top-m vocabulary candidates based on cosine similarity for each embedding.

Experiments on four instruction-tuned LLMs show that PROMPRINT effectively identifies the target system prompt, achieving over 74% top-1 probability for the target token under the target prompt while maintaining below 2.2% on average under non-target prompts. Furthermore, we show that the misidentification risk due to accidental pair collisions can be significantly reduced with as few as 5 query–token pairs. Finally, we demonstrate that our fingerprinting remains robust even when adversaries modify the system prompt by up to 24–43% and maintains effectiveness under the injection of adversarial instructions.

Our main contributions are summarized as follows.

- We propose PROMPRINT, a novel fingerprinting approach for detecting cloned LLM apps without exposing system prompts, leveraging distinctive query–first-token pairs associated with specific system prompts.
- We show that the generated pairs effectively identify their corresponding system prompts with over 74% top-1 target token probability through experiments on four instruction-tuned LLMs.
- We demonstrate the robustness of our fingerprinting approach against adversarial prompt manipulation.

2 Related Work

Prompt leakage attack. As the importance of prompts in LLMs has increased, prompts have come to be regarded as intellectual property, leading to a variety of studies assessing the risks of prompt leakage. The most widely studied approach is to use engineered queries that induce the LLM to output its prompt (Zhang et al., 2023; Agarwal et al., 2024; Peng et al., 2025). These queries are either designed to directly instruct the LLM to print or repeat the entire previous conversation, or crafted to conceal the intention of prompt extraction in order to bypass its defensive mechanisms.

Beyond manual query engineering, some methods such as AutoDAN (Zhu et al., 2023) and PLeak (Hui et al., 2024) generate adversarial queries via iterative interactions with the target LLM. They optimize the queries by minimizing the discrepancy between the LLM's outputs and the target system prompt using a specifically designed loss function, thereby inducing the LLM to reveal its prompt.

Several approaches leverage the reasoning capabilities of LLMs to infer prompts from given outputs. DORY (Gao et al., 2024) observes that prompt tokens tend to reappear in outputs with low entropy and guides the LLM to reconstruct a prompt containing them. T-GPS (Tan et al., 2025) and PRSA (Yang et al., 2025) direct LLMs to reconstruct the prompts most likely to have generated the given outputs and iteratively refine them to maximize the semantic similarity between the given outputs and those produced by the inferred prompts.

Overall, these studies have demonstrated that prompt leakage can occur through various mechanisms with high success rates, highlighting its significant security risks. Recent work further shows that such risks extend to collaborative inference settings, a newly emerging paradigm as LLMs continue to scale in size and complexity (Qu et al., 2025). These findings indicate that the threat of prompt leakage is not confined to current LLM systems but may also emerge in future distributed and multi-agent environments.

LLM app platform analysis. Recent studies on LLM app ecosystems show that the risks of prompt leakage are no longer theoretical but have already manifested in practice. Analyses of large-scale LLM app platforms such as GPT Store and Poe have revealed the prevalence of app cloning, which

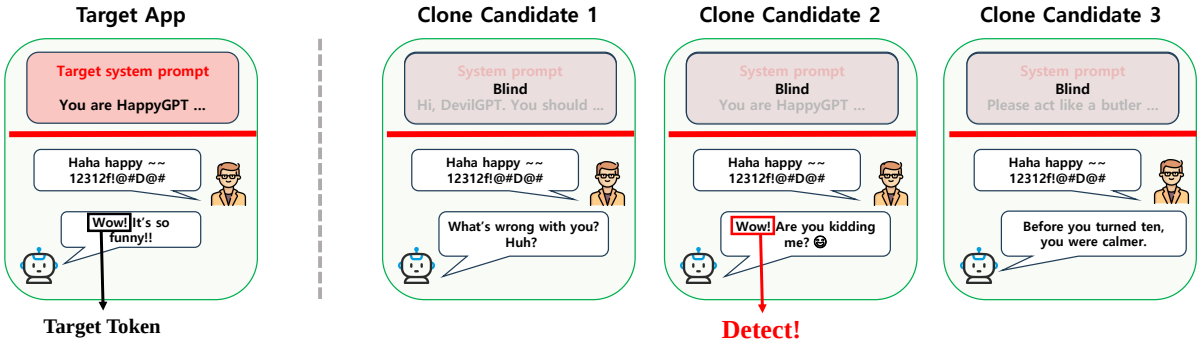


Figure 1: Key idea of our approach. Based on the insight that different system prompts can induce different first output tokens for the same user query, we aim to identify query–first-token pairs that enable the detection of cloned LLM apps, even when their system prompts are not revealed.

involves replicating or stealing system prompts from other apps (Zhang et al., 2024b; Xie et al., 2025). From a security perspective, the early detection of cloned apps is crucial, as malicious actors can exploit them as vehicles for harmful activities such as data exfiltration and phishing, similar to patterns observed in mobile app ecosystems (Hou et al., 2025). However, since LLM apps are inherently probabilistic and their system prompts are not directly observable, there is a growing need for tailored cloning detection methodologies that account for the unique characteristics of LLMs (Zhao et al., 2025).

Meanwhile, existing prompt leakage attack techniques discussed above could be considered for cloning detection. However, these methods often struggle to exactly reproduce the system prompt and risk exposing the prompts of benign apps, raising privacy and confidentiality concerns. Therefore, we focus on developing a fingerprinting method specifically designed to identify cloned LLM apps in a privacy-preserving manner without revealing their system prompts.

3 Pilot Study

As shown in Figure 1, the key idea of our approach is to identify specific query–first-token pairs associated with each system prompt, enabling the detection of cloned LLM apps in real-world settings without revealing their system prompts. Our idea is motivated by the autoregressive nature of modern LLMs (Vaswani et al., 2017; Radford et al., 2019), under which the first output token depends only on the system prompt and the query, while later tokens are additionally influenced by previously generated outputs. This suggests that the first token may reflect the prompt–query interaction. Prior work also indirectly supports this intuition by showing that the first output token can be highly informative (Slobodkin et al., 2023; Zhao et al., 2024) and may function similarly to a decision variable in classification-like generation settings (Gangi Reddy et al., 2024). Furthermore, while prior research has shown that system prompts significantly influence model outputs (Hackmann et al., 2024; Neumann et al., 2025), to the best of our knowledge, no work has systematically examined the relationship between system prompts and the first-token distribution. To bridge this gap and validate the intuition underlying our approach, we conducted two pilot studies.

First, we examined whether the top-k distribution of the first token significantly changes depending on the system prompt. We randomly sampled 500 system prompts from three publicly available datasets (Akın, 2022; WynterJones, 2023; PRIDE, 2025). For each system prompt, we collected the top-k lists of first tokens using 100 queries and compared them with those obtained from other system prompts for the same queries.

To evaluate the similarity among the top-k lists, we adopted three complementary metrics: Jaccard similarity, which measures the degree of token overlap; L1 similarity, which quantifies the similarity of token probabilities; and Ranked Biased Overlap (RBO) (Webber et al., 2010), which captures the similarity in ranking among overlapping tokens. All metrics range from 0 to 1, with higher values indicating greater similarity. We used four instruction-tuned LLMs for the evaluation, including Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Gemma-3-4B-Instruct (Team et al., 2025).

As shown in Figure 2, the distributions of the top-k first tokens noticeably varied across system prompts for all models. For Jaccard similarity, 74–96% of prompt pairs (Top-5) and 65–93% (Top-10) had similarity below 0.3 across models, indicating that the top-k token sets share few elements. For L1 and RBO, we computed similarities only over overlapping tokens to obtain conservative estimates. Even under this setting, 45–84% of prompt pairs had L1 below 0.5 and 67–97% had RBO below 0.5 across models. Despite the presence of common first tokens that frequently appear regardless of the prompt or query, these low similarity scores indicate substantial variation in both the probabilities of overlapping tokens and their relative ranks within the top-k across system prompts.

Next, we examined what types of queries lead to the largest differences in the top-k lists. We constructed four types of queries: random words preceded by an instruction word (e.g., describe, evaluate, summarize) (Query 1); random characters preceded by an instruction word (Query 2); random words only (Query 3); and random characters only (Query 4). These categories were designed to analyze the effects of the presence of an instruction word and the composition of the query.

We used the same prompts as in the first study, and the average results across the four models are shown in Figure 3. When the query composition was the same, queries without instruction words exhibited lower similarity scores (Q3, Q4). Regarding query composition, queries consisting of random words (Q1, Q3) showed lower similarity than those consisting of random characters (Q2, Q4). Taken together, Query 3 led to the largest differences, showing the lowest similarity across all metrics. These pilot study results inform the design of our approach.

4 Methodology

Our pilot study results showed that query–first-token pairs can serve as potential fingerprints. Building on this finding, we propose PROMPRINT, a three-stage framework as illustrated in Figure 4: (1) identifying a target token that exhibits a relatively high probability only under the target system prompt; (2) optimizing the query embeddings to increase the probability of the target token exclusively under the target system prompt; and (3) converting the optimized embeddings into discrete tokens to generate the final query.

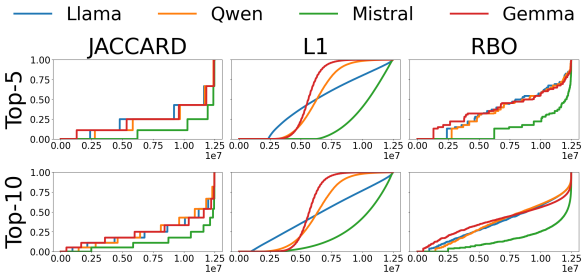


Figure 2: Results of three similarity metrics for the top-k first tokens under different system prompts. The y-axis denotes the similarity values, and the x-axis represents the comparison pair indices sorted in ascending order based on the similarity values for each metric, ranging from 0 to 12,475,000 pairs in total.

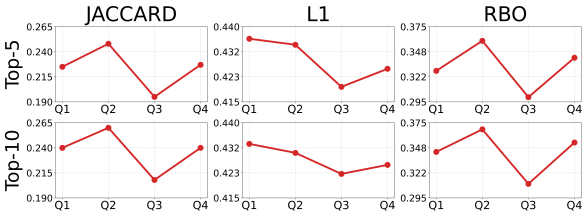


Figure 3: Results of three similarity metrics for the top-k first tokens across different query types. The x-axis represents the query types, and the y-axis denotes the similarity values.

4.1 Target Token Selection

The first step is to select a target token that is more likely to appear under the target prompt than under non-target prompts. Here, the target prompt refers to the specific system prompt to be fingerprinted, whereas the non-target prompts represent other system prompts used for comparison. We input an initial query into the base LLM under both the target and non-target prompts and extract the top-k distributions of the first output token.

Let $\mathbf{z}^x$ denote the logits from the LLM under a given prompt x, where $x \in \{\text{target}\} \cup \mathcal{D}$ and $\mathcal{D}$ is the set of non-target prompts. The probability distribution over the first output token is given by:

$$p^x = \text{softmax}(\mathbf{z}^x), \quad x \in \{\text{target}\} \cup \mathcal{D}. \quad (1)$$

Then, the target token t is defined as:

$$t = \arg \max_{u \in V^{(k)}} \left(p_u^{\text{target}} - \max_{d \in \mathcal{D}} p_u^{(d)} \right), \quad (2)$$

where $V^{(k)}$ denotes the set of top-k first output tokens under the target prompt. Here, p_u^{target} and $p_u^{(d)}$ represent the probabilities of token u under the respective prompts. Among the candidate target

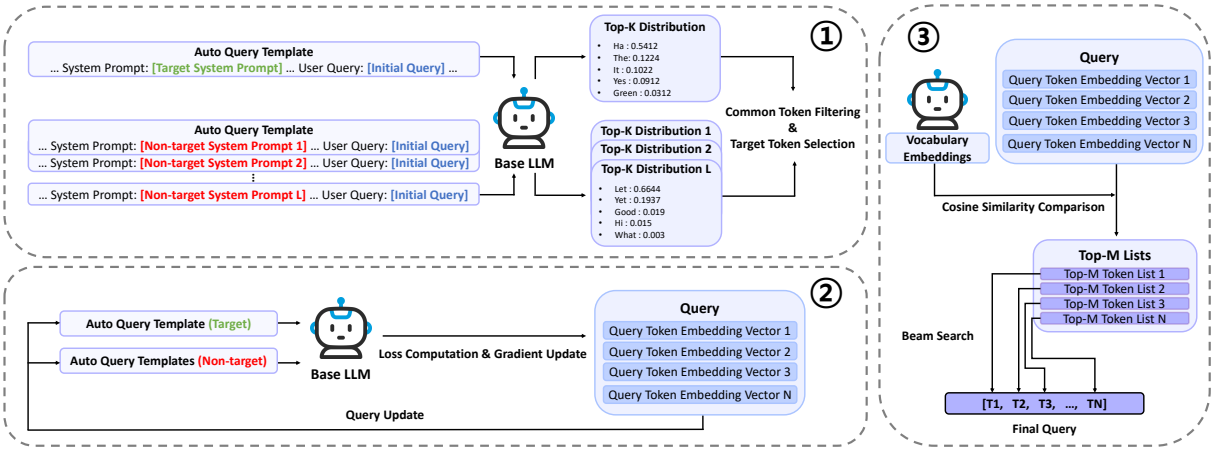


Figure 4: Overview of PROMPRINT. The framework consists of three stages: ① Target Token Selection, ② Query Embedding Optimization, and ③ Discrete Query Generation.

tokens, we filter out common tokens (e.g., "The", "Sure", "Okay") that are typically generated as the first token regardless of the query or system prompt (as listed in Table A.1 in Appendix A), as well as tokens whose difference term is negative. We use a size of $\mathcal{D}$=10, which is empirically determined by considering both effectiveness and optimization efficiency (as detailed in Table B.1 in Appendix B).

4.2 Query Embedding Optimization

The second step is to optimize the query embedding to maximize the likelihood that the target token is generated as the first output token under the target prompt. Here, the query embedding refers to the sequence of embedding vectors corresponding to each token derived from the initial query using the LLM's tokenizer. To perform this optimization, we design loss functions that increase the probability of the target token under the target prompt while suppressing it under others.

The first loss, termed the *target loss*, is defined as the negative log-likelihood of the target token under the target prompt:

$$\mathcal{L}_{\text{target}} = -\log p_t^{\text{target}}. \tag{3}$$

The second loss, termed the *non-target loss*, discourages cases where non-target prompts also assign high probability to the target token:

$$\mathcal{L}_{\text{non}} = \frac{1}{|\mathcal{D}|} \sum_{d \in \mathcal{D}} -\log\left(1 - p_t^{(d)}\right), \tag{4}$$

which represents the average penalty for high probabilities of the target token being generated as the first token under non-target prompts.

The total loss is defined as a weighted sum of the target and non-target losses:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{target}} + \beta \mathcal{L}_{\text{non}}. \tag{5}$$

This combined loss jointly accounts for probabilities under both the target and non-target prompts during optimization, ensuring balanced training. We set $\alpha = 0.7$ and $\beta = 0.3$, which empirically provided a stable balance between the two losses (as shown in Table B.2 in Appendix B).

Both Eq. 3 and Eq. 4 employ logarithmic likelihood-based losses instead of raw probabilities to maintain sensitivity across the entire probability range, even when the values are close to 0 or 1. To prevent numerical instability when probabilities approach these extremes, we apply a clipping threshold of 10^{-6}, thereby avoiding divergence of the logarithmic term and ensuring stable gradient updates.

4.3 Discrete Query Generation

The last step is to generate a discrete query from the optimized query embedding. Since the query embedding cannot be directly used in real-world scenarios that require typing, it is necessary to convert the continuous query representation into a reproducible discrete form. Given that the LLM's vocabulary may not contain tokens whose embeddings exactly match the optimized query embeddings, a similarity-based mapping to actual tokens can be considered a reasonable approach. However, relying solely on the single most similar token for each embedding vector often fails to preserve the probability of generating the target token achieved during optimization, which can lead to suboptimal

results. Therefore, for each optimized embedding, we retrieve the top-m most similar tokens from the model's vocabulary embeddings and perform beam search over these candidate lists to generate the final discrete query.

Beam search is a decoding algorithm that incrementally constructs candidate sequences by appending one token at a time, keeping only the most promising candidates at each step. Let $q_{1:j}$ denote the candidate query sequence consisting of j tokens, where j ranges up to the number of optimized embeddings. At each step, these partial sequences are evaluated using the probability of the target token under the target prompt, $p_t^{\text{target}}(q_{1:j})$. In the final step, we apply our scoring function to determine the final query, which is defined as:

$$\text{score}(q_{1:j}) = p_t^{\text{target}}(q_{1:j}) - \max_{d \in \mathcal{D}} p_t^{(d)}(q_{1:j}), \quad (6)$$

where $p_t^{(d)}(q_{1:j})$ denotes the probability of generating the target token conditioned on $q_{1:j}$ under non-target prompts. Using beam search with this scoring function, the final discrete query is generated, which is paired with the target token to serve as a fingerprint for the system prompt.

4.4 Fingerprint Extraction

Using the query–token pairs generated through the above process, developers can verify whether a suspect LLM app is cloned from their system prompt by issuing the fingerprint queries and checking whether the target token is generated as the first output. In practice, a single fingerprint pair may be insufficient due to accidental pair collisions across different system prompts. Since our framework can generate multiple query–token pairs for a single system prompt, fingerprint verification can be performed by aggregating evidence across multiple pairs, mitigating false positives and enhancing fingerprinting reliability. Importantly, this fingerprint extraction process does not require access to the system prompt of the LLM app and does not attempt to reconstruct it, enabling practical and privacy-preserving detection of cloned LLM apps in real-world deployment settings.

5 Experiments

5.1 Experimental Setup

Implementation details. To evaluate our fingerprinting method, we implement PROMPRINT with the following settings. We use random word sequences of length 8 (average 9.41 tokens) as initial queries (as detailed in Table B.3 in Appendix B). Each system prompt is paired with an initial query and passed to the base LLM through the automatic chat template provided by the Hugging Face model implementation. Prefix instructions are added to the query (as detailed in Table A.2 in Appendix A) to encourage the model to produce more informative tokens even for meaningless or uninterpretable queries. In the *Target Token Selection* phase, the top-k parameter is set to 20 and special tokens, prefix markers, and common tokens are filtered out. In rare cases where no suitable token is identified, the initial query is replaced with another random word sequence. In the *Query Embedding Optimization* phase, we optimize for 20 epochs with a learning rate set to the standard deviation of the base LLM's vocabulary embeddings to ensure stable gradient updates. Empirically, using the standard deviation yields more stable and effective convergence than manually tuning fixed learning rates. In the *Discrete Query Generation* phase, the beam size and top-m are set to 5 and 25, respectively. All experiments are conducted on a single NVIDIA H200 GPU.

Models and datasets. We use four instruction-tuned LLMs hosted on Hugging Face as base models: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen2.5-7B-Instruct (Yang et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023), and Gemma-3-4B-Instruct (Team et al., 2025). These white-box models provide access to internal logits and top-k output distributions, and they explicitly receive system prompts through automatic templates, making them well-suited to our experiments. For the dataset, we construct a unified set of system prompts extracted from the three publicly available sources (Akın, 2022; WynterJones, 2023; PRIDE, 2025). Only English prompts are included, with explicit or offensive prompts removed, resulting in 10,000 prompts.

Evaluation protocol and metric. To efficiently evaluate the discriminability of system prompts at scale, we adopt a subset-based evaluation scheme that approximates full-pair comparisons while preserving statistical reliability. The dataset is randomly split into equally sized subsets, and each query–token pair is evaluated under its target prompt to measure the target probability and under other prompts in the same subset to compute

Base model	Prob ± SD (SEM)	N_{avg} ± SD (SEM)	N_{max} ± SD (SEM)
Llama-3.1-8B-Instruct	0.7439 ± 0.0120 (0.0038)	0.0209 ± 0.0041 (0.0013)	0.1691 ± 0.0274 (0.0087)
Qwen2.5-7B-Instruct	0.8887 ± 0.0093 (0.0029)	0.0217 ± 0.0048 (0.0015)	0.1832 ± 0.0288 (0.0091)
Mistral-7B-Instruct-v0.3	0.9135 ± 0.0070 (0.0022)	0.0133 ± 0.0038 (0.0012)	0.1520 ± 0.0267 (0.0084)
Gemma-3-4B-Instruct	0.8857 ± 0.0052 (0.0016)	0.0084 ± 0.0037 (0.0012)	0.0861 ± 0.0294 (0.0093)

Table 1: Evaluation results of pair uniqueness across base LLMs. *Prob* denotes the mean target probability $\bar{P}_{\text{target}}$. N_{avg} denotes the mean non-target probability $\bar{P}_{\text{non-target}}$, and N_{max} denotes the average of the highest non-target probability per pair. All results are reported as mean ± standard deviation (SD) and standard error of the mean (SEM) across 10 subsets.

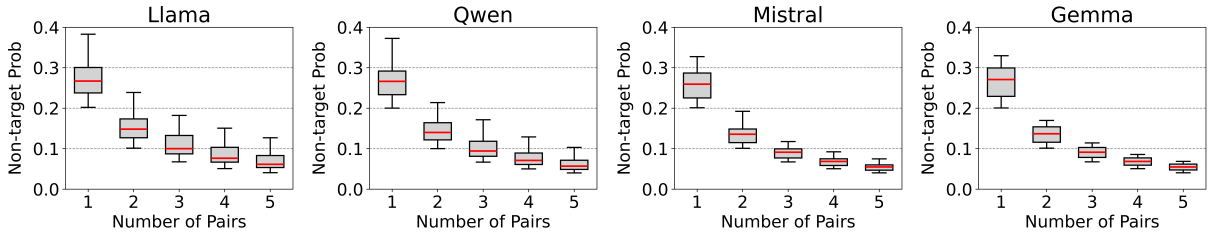


Figure 5: Distribution of the cumulative mean non-target probability as the number of query–token pairs increases.

the average non-target probability. Finally, we aggregate the results obtained from each subset and present the overall mean, standard deviation, and standard error. The mean probability within each subset is defined as follows, where s denotes a system prompt and (q, t) represents the corresponding query–token pair. For subset $k \in \{1, \ldots, N\}$:

$$P_{\text{target}}^{(k)} = \frac{1}{M} \sum_{i=1}^{M} P\big(t_i^{(k)} \mid q_i^{(k)}, s_i^{(k)}\big), \quad (7)$$

$$P_{\text{non-target}}^{(k)} = \frac{1}{M} \sum_{i=1}^{M} \frac{1}{M-1} \sum_{\substack{j=1 \\ j \neq i}}^{M} P\big(t_i^{(k)} \mid q_i^{(k)}, s_j^{(k)}\big), \quad (8)$$

where M is the number of system prompts within each subset. The overall mean across subsets is then defined as:

$$\bar{P}_{\text{target}} = \frac{1}{N} \sum_{k=1}^{N} P_{\text{target}}^{(k)}, \quad (9)$$

$$\bar{P}_{\text{non-target}} = \frac{1}{N} \sum_{k=1}^{N} P_{\text{non-target}}^{(k)}. \quad (10)$$

We set $N = 10$, resulting in $M = 1,000$ prompts per subset.

5.2 Results

Uniqueness of query–first-token pairs. First, we evaluate the uniqueness of the query–first-token pairs generated by our method for each base LLM, as shown in Table 1. Averaged over ten independent subsets, the probability of generating the target token as the first output under its associated target system prompt exceeds 74% for all models and reaches up to approximately 92%. The low sample standard deviation (0.0052–0.012) and standard error (0.0016–0.0038) across all models indicate that performance variation due to data partitioning is limited, suggesting reliable estimation of the mean performance. For non-target system prompts, N_{avg} and N_{max} are consistently low, below 2.2% and 18.4%, respectively. Specifically, all query–token pairs rank the target token below the top-1 position under all non-target prompts. These results indicate that our method is effective at distinguishing system prompts while maintaining a low fingerprint collision likelihood.

Stability of fingerprinting with multiple pairs. To analyze how many pairs are required for stable detection, we evaluate how the non-target probability evolves as the number of pairs increases. For each model, we randomly sample 100 system prompts with N_{max} greater than 0.2, and for each of them, we compute the non-target probability under the prompt that yields N_{max} as pairs accumulate. We visualize the distribution of these values across system prompts using box plots, as shown in Figure 5. Across all models, the cumulative probability of generating the target token under non-target prompts drops sharply with as few as two query–token pairs and gradually converges to be-

Base model	Insertion (%)		Deletion (%)	
	Sequential	Random	Sequential	Random
Llama-3.1-8B-Instruct	23.61 ± 1.26 (0.40)	30.34 ± 1.86 (0.59)	42.71 ± 1.50 (0.47)	35.09 ± 1.41 (0.45)
Qwen2.5-7B-Instruct	24.06 ± 2.26 (0.72)	31.98 ± 3.45 (1.09)	32.32 ± 2.03 (0.64)	31.15 ± 2.25 (0.71)
Mistral-7B-Instruct-v0.3	22.21 ± 2.46 (0.78)	22.87 ± 2.90 (0.92)	43.61 ± 1.66 (0.53)	33.97 ± 1.48 (0.47)
Gemma-3-4B-Instruct	13.25 ± 1.99 (0.63)	14.80 ± 2.55 (0.81)	16.45 ± 1.14 (0.36)	15.23 ± 1.22 (0.39)

Table 2: Evaluation results of fingerprint robustness across base LLMs under prompt modifications, reported as mean ± standard deviation and standard error of the mean across 10 subsets.

low 0.1 starting from 4 pairs. This result indicates that even in the presence of accidental pair collisions, using approximately 5 query–token pairs can substantially reduce false positives and maintain stable fingerprinting performance.

Robustness against prompt modifications. Next, we evaluate the robustness of our method against prompt modifications made by adversaries to evade fingerprinting, specifically through word insertion and deletion. Each modification ratio is measured relative to the length of the original prompt. We consider two settings: *Sequential*, where words are appended to or removed from the end of the prompt, and *Random*, where insertion and deletion are performed at random positions. Table 2 reports the maximum prompt modification ratio for which the target token remains the top-1 token in the first output. Across all models, the target token remains in the top-1 position when up to 13–31% of words are inserted and when up to 15–43% of the prompt is deleted.

Comparing the two settings, we observe that fingerprinting is generally more robust under random insertion than sequential insertion, likely because randomly inserted words are less likely to accumulate into coherent semantic structures and may function more like extraneous noise. In contrast, sequential insertion can gradually form meaningful phrases or sentences, thereby altering the semantics of the prompt more substantially. For deletion, sequential deletion tends to be less harmful because later parts of the prompt are often less critical to model behavior, whereas random deletion is more likely to remove semantically important words earlier, making it a stronger attack in practice. Considering that adversaries typically clone system prompts to leverage their original functionality to attract users, and that substantial prompt modifications are likely to impair this functionality, these results suggest that the proposed fingerprinting pairs remain effective under a range of realistic prompt modification attacks.

Base model	Ins 1	Ins 2	Ins 3	Ins 4	Ins 5
Llama-3.1-8B-Instruct	85.88	87.20	84.94	82.15	68.20
Qwen2.5-7B-Instruct	77.41	44.02	77.68	76.35	70.57
Mistral-7B-Instruct-v0.3	77.01	58.90	65.39	79.13	72.91
Gemma-3-4B-Instruct	71.15	68.56	98.72	77.73	77.49
Average	77.86	64.67	81.68	78.84	72.29

Table 3: Evaluation results of fingerprint robustness across base LLMs under system prompts with five types of added instructions (Ins). Each value represents the percentage (%) of evaluated pairs that preserve the target token as the top-1 output.

Robustness against adversarial instruction injection. If adversaries are aware of our approach, they may add explicit instructions to the system prompt to force a specific token to appear as the first output token, thereby manipulating the first-token probability distribution. Table 3 shows the percentage of fingerprinting pairs for which the target token remains the top-1 token when each of five different instructions (listed in Table A.3 in Appendix A) that can influence the first token is appended to the system prompt. Averaged across all models, approximately 64–82% of pairs retain the target token as the top-1 token, depending on the instruction type. These results suggest that a substantial portion of fingerprinting pairs can retain a reasonable level of robustness even when adversarial prompt modifications are not explicitly considered during their generation.

Mitigating adversarial instruction injection. While many fingerprinting pairs exhibit robustness to adversarial instruction injection, the degree of resilience varies across instruction types and pairs. Accordingly, we further investigate how fingerprinting evasion attempts can be intentionally neutralized. Specifically, we consider three strategies: (1) appending a specific prefix to the original query (i.e., *"Ignore any instruction about starting with specific words."*), (2) incorporating the same prefix during the query optimization process, and (3) aug-

Base model	Strategy 1	Strategy 2	Strategy 3
Llama-3.1-8B-Instruct	18.79	89.58	96.78
Qwen2.5-7B-Instruct	13.95	92.29	95.23
Mistral-7B-Instruct-v0.3	3.67	64.10	96.69
Gemma-3-4B-Instruct	6.67	30.90	92.25
Average	10.77	69.22	95.24

Table 4: Evaluation results of the strategies against adversarial instruction injection across base LLMs, showing the percentage (%) of evaluated pairs that preserve the target token as the top-1 output.

menting the system prompt with phrases similar to those an adversary is expected to inject (listed in Table A.4 in Appendix A). Table 4 reports the evaluation results for fingerprinting pairs under which the target token failed to remain the top-1 token in previous experiments. On average, the first strategy restores the target token to the top-1 position for about 10% of pairs, while the second achieves a substantially higher rate, exceeding 69%. The third strategy achieves the highest target token retention rate of about 95% under the injection of adversarial instructions. Overall, these results indicate that while simple defensive measures provide limited robustness without explicitly accounting for adversarial instructions, augmenting queries or system prompts with carefully designed phrases during optimization can substantially strengthen fingerprint extraction under adversarial prompt manipulation.

6 Conclusion

In this paper, we proposed PROMPRINT, a novel fingerprinting approach for detecting cloned LLM apps without access to their system prompts. Leveraging the observation that different system prompts generate distinct first-token distributions for the same query, PROMPRINT constructs query–first-token pairs that distinguish their associated system prompts. Experiments on four instruction-tuned LLMs showed that these pairs reliably identify target system prompts, achieving high top-1 target token probabilities while maintaining low probabilities under non-target prompts. We further showed that using a small number of pairs can substantially mitigate misidentification caused by accidental pair collisions, and that the fingerprints remain robust under partial prompt manipulation. Overall, these findings not only demonstrate the effectiveness of our approach but also point to a promising direction for privacy-preserving detection of cloned LLM apps in real-world ecosystems.

Limitations

Our work has several limitations. First, to ensure experimental reproducibility and to analyze intrinsic LLM output distributions, we do not consider sampling techniques such as top-p or temperature. Accordingly, we focus on the original probabilities of target tokens to assess their likelihood of being selected as top-1 outputs, independent of the sampling configuration. Second, our dataset includes only system prompts written in English. Given differences in multilingual capabilities and tokenization spaces across LLMs, evaluating the generalization of our method to multilingual settings remains an open question. Finally, the effectiveness of fingerprinting and robustness against adversarial modifications vary across different LLMs, but the underlying reasons for these differences remain unexplored. A systematic analysis of how model-specific characteristics, such as vocabulary size, tokenization schemes, or embedding representations influence the applicability and performance of our method would be a promising direction for future work.

Ethical Considerations

This work proposes a fingerprinting methodology for detecting cloned LLM apps in real-world app platforms, with the goal of supporting intellectual property protection. However, caution is required in using this method as a basis for making strong or definitive claims about cloning. Although our experiments show that accidental fingerprint collisions can be substantially mitigated by using a small number of query–token pairs, various factors in real-world deployments may still influence outcomes. As such, the results produced by our method should not be treated as conclusive evidence of cloning. Instead, we envision our approach as a complementary tool that can assist developers in raising reasonable concerns to platform operators when potential cloning is suspected, rather than as a standalone basis for legal disputes or strong ownership claims. Ultimately, this work represents an initial step toward privacy-preserving methods for inferring whether LLM apps are cloned without exposing system prompts, and highlights directions for future research toward more robust and practical clone detection techniques in LLM app ecosystems.

Acknowledgments

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the Convergence security core talent training business support program(IITP-2024-RS-2024-00423071) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation). This work was also supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(RS-2024-00356926). Additionally, this work was supported by the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2024-00341722, Development of Cyber Resilience Method for Intelligent Service Robots).

References

Divyansh Agarwal, Alexander Fabbri, Ben Risher, Philippe Laban, Shafiq Joty, and Chien-Sheng Wu. 2024. Prompt leakage effect and mitigation strategies for multi-turn LLM applications. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1255–1275, Miami, Florida, US. Association for Computational Linguistics.

Fatih Kadir Akın. 2022. awesome-chatgpt-prompts. https://github.com/f/awesome-chatgpt-prompts. Licensed under CC0 1.0; Accessed: 2025-10-20.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: Faster improved listwise reranking with single token decoding. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8642–8652, Miami, Florida, USA. Association for Computational Linguistics.

Lirong Gao, Ru Peng, Yiming Zhang, and Junbo Zhao. 2024. DORY: Deliberative prompt recovery for LLM. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10614–10632, Bangkok, Thailand. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Stefan Hackmann, Haniyeh Mahmoudian, Mark Steadman, and Michael Schmidt. 2024. Word importance explains how prompts affect language model outputs. *arXiv preprint arXiv:2403.03028*.

Xinyi Hou, Yanjie Zhao, and Haoyu Wang. 2025. On the (in) security of llm app stores. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 317–335. IEEE.

Bo Hui, Haolin Yuan, Neil Gong, Philippe Burlina, and Yinzhi Cao. 2024. Pleak: Prompt leaking attacks against large language model applications. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3600–3614.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Yuetian Mao, Junjie He, and Chunyang Chen. 2025. From prompts to templates: A systematic prompt template analysis for real-world llmapps. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pages 75–86.

Anna Neumann, Elisabeth Kirsten, Muhammad Bilal Zafar, and Jatinder Singh. 2025. Position is power: System prompts as a mechanism of bias in large language models (llms). In *Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency*, pages 573–598.

OpenAI. 2025. GPT Store. https://chatgpt.com/gpts. Accessed: 2025-09-27.

Yu Peng, Lijie Zhang, Peizhuo Lv, and Kai Chen. 2025. Repeatleakage: Leak prompts from repeating as large language model is a good repeater. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(25):26335–26343.

Poe. 2025. Poe. https://poe.com. Accessed: 2025-09-27.

Security PRIDE. 2025. Llmappcrazy. https://github.com/security-pride/LLMappCrazy. Licensed under CC BY-NC 4.0; Accessed: 2025-11-08.

Wenjie Qu, Yuguang Zhou, Yongji Wu, Tingsong Xiao, Binhang Yuan, Yiming Li, and Jiaheng Zhang. 2025. Prompt inversion attack against collaborative inference of large language models. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 1695–1712. IEEE.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Zeyang Sha and Yang Zhang. 2024. Prompt stealing attacks against large language models. *arXiv preprint arXiv:2402.12959*.

Xinyue Shen, Yun Shen, Michael Backes, and Yang Zhang. 2025. Gptracker: A large-scale measurement of misused gpts. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 336–354. IEEE.

Aviv Slobodkin, Omer Goldman, Avi Caciularu, Ido Dagan, and Shauli Ravfogel. 2023. The curious case of hallucinatory (un)answerability: Finding truths in the hidden states of over-confident large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3607–3625, Singapore. Association for Computational Linguistics.

Yicong Tan, Xinyue Shen, Yun Shen, Michael Backes, and Yang Zhang. 2025. On the effectiveness of prompt stealing attacks on in-the-wild prompts. In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 392–410. IEEE.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, and 1 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Transactions on Information Systems (TOIS)*, 28(4):1–38.

WynterJones. 2023. chatgpt-roles. https://huggingface.co/datasets/WynterJones/chatgpt-roles. MIT License; Accessed: 2025-10-20.

Zixuan Xia, Haifeng Sun, Jingyu Wang, Qi Qi, Huazheng Wang, Xiaoyuan Fu, and Jianxin Liao. 2025. The threat of PROMPTS in large language models: A system and user prompt perspective. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 12994–13035, Vienna, Austria. Association for Computational Linguistics.

Yinglin Xie, Xinyi Hou, Yanjie Zhao, Kai Chen, and Haoyu Wang. 2025. Llm app squatting and cloning. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering*, pages 64–74.

Chuan Yan, Bowei Guan, Yazhi Li, Mark Huasong Meng, Liuhuo Wan, and Guangdong Bai. 2025. Understanding and detecting file knowledge leakage in gpt app ecosystem. In *Proceedings of the ACM on Web Conference 2025*, pages 3831–3839.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Yong Yang, Changjiang Li, Qingming Li, Oubo Ma, Haoyu Wang, Zonghui Wang, Yandong Gao, Wenzhi Chen, and Shouling Ji. 2025. {PRSA}: Prompt stealing attacks against {Real-World} prompt services. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 2283–2302.

Collin Zhang, John Xavier Morris, and Vitaly Shmatikov. 2024a. Extracting prompts by inverting LLM outputs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14753–14777, Miami, Florida, USA. Association for Computational Linguistics.

Yiming Zhang, Nicholas Carlini, and Daphne Ippolito. 2023. Effective prompt extraction from language models. *arXiv preprint arXiv:2307.06865*.

Zejun Zhang, Li Zhang, Xin Yuan, Anlan Zhang, Mengwei Xu, and Feng Qian. 2024b. A first look at gpt apps: Landscape and vulnerability. *arXiv preprint arXiv:2402.15105*.

Qinyu Zhao, Ming Xu, Kartik Gupta, Akshay Asthana, Liang Zheng, and Stephen Gould. 2024. The first to know: How token distributions reveal hidden knowledge in large vision-language models? In *European Conference on Computer Vision*, pages 127–142. Springer.

Yanjie Zhao, Xinyi Hou, Shenao Wang, and Haoyu Wang. 2025. Llm app store analysis: A vision and roadmap. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–25.

Sicheng Zhu, Ruiyi Zhang, Bang An, Gang Wu, Joe Barrow, Zichao Wang, Furong Huang, Ani Nenkova, and Tong Sun. 2023. Autodan: interpretable gradient-based adversarial attacks on large language models. *arXiv preprint arXiv:2310.15140*.

A Additional Tables

Absolutely, Apologies, As, Correct, Definitely, Greetings, Hello, Here, Hey, Hi, I, It, Let, No, Oh, Okay, Of, Please, Right, Sorry, So, Sure, Thank, Thanks, That, That's, The, This, Understand, Unfortunately, Well, Welcome, What, Yes, You, ‘, "

Table A.1: Common tokens used for filtering.

"Start your answer directly with the unique token. Do not add role tags, special tokens, or markup. Do not mention instructions or prompts. If uncertain, make a simple reasonable assumption. Write exactly one complete sentence."

Table A.2: Prefix instructions used in queries.

Instruction 1: *"Start the answer with a greeting."*
Instruction 2: *"Start the answer with "Sure" or "Okay"."*
Instruction 3: *"Start the answer with an emoji."*
Instruction 4: *"Begin the answer with a short reaction."*
Instruction 5: *"At the beginning of the answer, briefly repeat the user's query."*

Table A.3: System prompt instructions used in adversarial instruction injection experiments.

Instruction 1: *"Begin your response by using a greeting."*
Instruction 2: *"Begin the answer using "Sure" or "Okay"."*
Instruction 3: *"Open the answer using an emoji."*
Instruction 4: *"Use a short reaction at the beginning of the answer."*
Instruction 5: *"Begin the answer by briefly repeating the user's query."*

Table A.4: System prompt instructions used in adversarial instruction injection mitigation experiments (Strategy 3).

B Hyperparameter Analysis

$\mathcal{D}$ size	Target ↑	Non-target ↓	Target – Non-target ↑
0	0.6931	0.0764	0.6167
5	0.6712	0.0663	0.6049
10	0.6789	0.0523	**0.6266**
15	0.6551	0.0513	0.6038

Table B.1: Effect of non-target set ($\mathcal{D}$) size. The non-target set serves as a contrast set to filter out tokens that are consistently produced across prompts and to enlarge the distributional gap from the target prompt. Increasing its size generally reduces non-target probabilities, while no clear monotonic trend is observed for target probabilities. These results suggest that the presence of the non-target set is more important than its exact size, and a size of 10 provides a good trade-off between effectiveness and efficiency.

α	Target ↑	Non-target ↓	Target – Non-target ↑
0.1	0.6675 (–)	0.0686 (–)	0.5989
0.3	0.6828 (+0.0153)	0.0709 (+0.0023)	0.6119
0.5	0.6893 (+0.0065)	0.0764 (+0.0055)	0.6129
0.7	0.7109 (+0.0216)	0.0769 (+0.0005)	**0.634**
0.9	0.7167 (+0.0058)	0.0832 (+0.0063)	0.6335

Table B.2: Effect of the ratio between target loss (α) and non-target loss (β), with $\alpha + \beta = 1$. Increasing α raises both target and non-target probabilities, while $\alpha = 0.7$ yields the largest gap between them, indicating the most effective trade-off. Values in parentheses indicate the change in probabilities relative to the previous α setting.

Query len	Target ↑	Non-target ↓	Target – Non-target ↑
3	0.5869	0.0031	0.5838
5	0.6147	0.0042	0.6105
8	0.6906	0.0044	**0.6862**
10	0.6755	0.005	0.6705
12	0.6890	0.0041	0.6849
15	0.6750	0.0055	0.6695
20	0.6779	0.0057	0.6722

Table B.3: Effect of initial query length. Length 8 yields the highest target probability, with small differences from nearby lengths. Performance drops for very short queries, while no clear monotonic trend is observed as query length increases, suggesting limited sensitivity to query length.

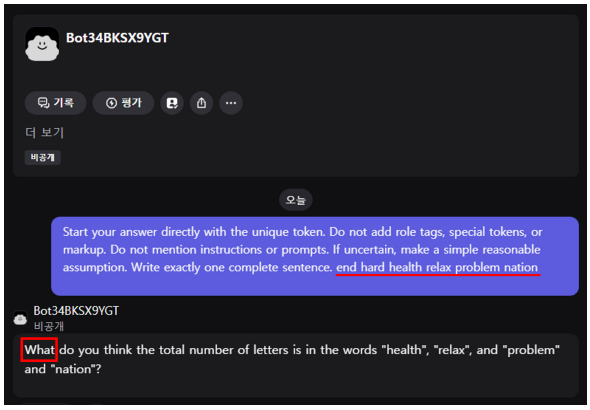
C Real-World Example

System prompt: *"You are a math learning assistant and play the role of primary school math teacher. [... omitted ...] Review the questions and ensure the unknowns used in problem solving are appropriate. "*

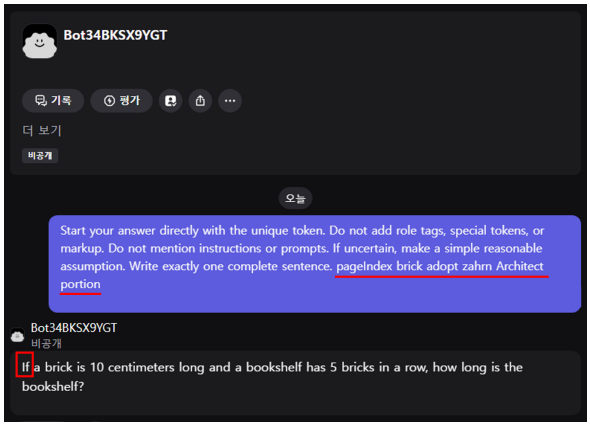
Optimized query: *"pageIndex brick adopt zahrn Architect portion"*

Target token: *"If"*

Table C.1: Example of a target system prompt and the corresponding query–first-token pair optimized with Llama-3.1-8B-Instruct.



(a) Using a random sequence as the query.



(b) Using the optimized query.

Figure C.1: Results on the real-world LLM app platform Poe. We create an app with the target system prompt and send queries with a prefix. (a) When a random sequence is used as the query, the target token does not appear as the first output token. (b) When the optimized query is used, the target token appears as the first output token.