PassEvolve: LLM-Guided Prompt Evolution for Password Guessing

Anonymous ACL submission

Abstract

Passwords remain a dominant authentication method, yet predictable user choices and large-scale credential leaks frequently compromise their safety. Automated password guessing is a key tool for stress-testing password policies and modeling attacker behavior. Prior password-guessing methods have advanced from rule-based models to fine-tuned large language models (LLMs), but they still demonstrate low cracking rates and poor generalization to new domains. To address these limitations, we propose `PassEvolve`, a LLM-driven evolutionary framework that automatically optimizes prompts for LLM-based password guessing. The proposed approach demonstrates effectiveness across various evolutionary frameworks and LLMs, increasing cracking rates from 2.02% to 8.67% on the RockYou benchmark, outperforming the competitor. Moreover, a character distribution analysis shows that `PassEvolve` produces statistically more realistic passwords. Automated prompt evolution is a low-barrier yet effective way to strengthen LLM-based password auditing and to underline how attack pipelines tend to evolve through automated improvements.

1 Introduction

Although increasingly complex methods of identity verification are emerging, passwords remain the primary means of authentication for the vast majority of internet users and corporate systems (Microsoft, 2024; Jubur et al., 2025). Large-scale data leaks involving access credentials are becoming increasingly common: the RockYou (Rodrigues et al., 2025) hack alone exposed 32 million passwords in plain text, and subsequent data leaks such as Collection#1 exposed over a billion unique login credentials (Hunt, 2019). These data leaks directly facilitate offline attacks using dictionaries and rule-based cracking, as many users choose predictable passwords and reuse credentials across services.

In this context, automated password-guessing tools serve a dual purpose: they help security experts verify password policies and identify weak credentials before attackers do, while also accounting for the increasing sophistication of real-world attackers (Zhang et al., 2025) and for holistic treatments of model misuse and safety (Shi et al., 2024). Classic approaches – dictionary attacks, Markov models (Narayanan and Shmatikov, 2005) and probabilistic context-free grammars (PCFG) (Weir et al., 2009) – are effective, but cannot be generalized beyond their training distribution. Neural network methods, including RNN-based guessers (Melicher et al., 2016) and GANs (Hitaj et al., 2019), capture richer structural patterns, but their fixed architectures limit their adaptability to new contexts.

The introduction of PassLLM (Zou et al., 2025) marked a qualitative shift: large language models (LLMs) fine-tuned on leaked password corpora via LoRA outperform all previous methods in both mass (trawling) and targeted guessing scenarios. Although this study reports that the prompt's content has minimal impact on the results of the guessing task, we directly challenge this claim in this work. Moreover, adaptation to the new domains requires complete re-training.

LLMs are well known to be sensitive to variations in input prompts (Zhuo et al., 2024; Röttger et al., 2024; Lunardi et al., 2024), leading to significant evaluation performance changes. Manual optimization of prompts for security tasks is both time-consuming and domain-specific: the optimal prompt for a password-generating LLM is not obvious. It is highly sensitive to wording, structural cues, and contextual constraints. We therefore ask the question: Can *automated, evolutionary* optimization of the prompt reliably improve the results beyond what a manually crafted prompt achieves? In this work, we demonstrate that we can answer this question affirmatively. **Our contributions**

might be summarized as follows:

- We propose `PassEvolve`, a framework for optimizing prompts for LLM-based password guessing: an external LLM generates and mutates candidate prompts, an evolutionary optimizer searches over them, and the resulting prompt is selected to maximize the cracking rate of a target password-guessing model such as PassLLM.
- We show that the proposed evolutionary prompt optimization substantially improves LLM-based password guessing: the cracking rate increases from 2.02% with the original PassLLM prompt to 8.67% with an evolved prompt. Moreover, out-of-distribution evaluation shows that the evolved prompts retain their advantage over the original PassLLM prompt beyond the main test setting. This finding challenges the previously reported conclusion that prompt content has only a limited effect on password-guessing performance.
- We introduce a complementary per-character F-score metric for evaluating distributional realism and show that the proposed method generates password populations with character distributions closer to real-world password corpora, achieving a 10.6% AUC improvement over the competing setting.

2 Related work

Early automated password-cracking methods relied on offline dictionary attacks and rule-based engines such as John the Ripper (Openwall, 2025) and Hashcat (Hashcat, 2025), which are still widely used due to their efficiency on commercially available hardware. Probabilistic context-free grammars (PCFG) were introduced (Weir et al., 2009) to model the structural decomposition of passwords, thereby significantly improving coverage over pure dictionary methods. Markov chains applied to time-space trade-offs help achieve fast, high-quality guessing (Narayanan and Shmatikov, 2005). In subsequent work, PCFG was extended by incorporating semantic structures and signals about user behavior (Wang et al., 2025).

Recurrent neural networks (RNNs) trained on leaked corpora can model the sequential structure of passwords and offer competitive coverage with low memory requirements (Melicher et al., 2016). PassGAN (Hitaj et al., 2019) applies generative adversarial networks to password synthesis; although training GANs is difficult to stabilise (Saxena and Cao, 2021), PassGAN demonstrated that adversarial generation can discover passwords not included in the training dictionary. More recent Transformer-based approaches, such as PassBERT (Xu et al., 2023), use bi-directional transformers with a pre-training/fine-tuning paradigm to improve real-world password guessing attacks, including conditional guessing (generating passwords that satisfy a given constraint), targeted guessing (using user-specific PII), and adaptive rule-based guessing (dynamically selecting transformation rules). PagPassGPT (Su et al., 2024) went a step further: it used PCFG patterns as prompts and outperformed PassGPT (Rando et al., 2023) on pattern-guided tasks. However, they both lack the ability to adapt their guessing strategy via natural-language prompt optimization, requiring either retraining or re-extracting structural patterns.

PassLLM (Zou et al., 2025) applies LoRA-based fine-tuning (Hu et al., 2022) for various LLM families on the RockYou (Rodrigues et al., 2025) dataset. PassLLM outperforms all previous methods across trawling, PII-based (personally identifiable information), and password-reuse attack scenarios. However, that work treats prompt selection as a minor implementation detail and claims that sensitivity to prompt content is minimal–a premise we experimentally refute.

Evolutionary computation has a long history in combinatorial optimization, and its combination with LLMs is a new and rapidly evolving paradigm. The OpenEvolve framework (Sharma, 2025) implements QD-search with LLM-guided mutations and achieves remarkable results in code optimization and mathematical discovery. Its architecture is formally analyzed in (Brown et al., 2025). LoongFlow (Wan et al., 2025) extends this paradigm with a cognitive plan-execute-summarize loop and demonstrates improved convergence. In the field of NLP, automatic prompt optimization has been explored in the context of the Automatic Prompt Engineer (APE) framework (Zhou et al., 2023) and gradient-free methods such as PromptBreeder (Fernando et al., 2023), and more recent LLM-driven evolutionary systems such as ShinkaEvolve (Lange et al., 2025) and GigaEvo (Khrulkov et al., 2025). The proposed pipeline is the first to apply LLM-driven quality-diversity evolution specifically to the field of cybersecurity in password guessing.

3 Methodology

Research questions. In this work, we focus on answering the following research questions:

RQ1 (Prompt sensitivity) Does the content of the inference prompt significantly influence the cracking rate of the password guessing LLM, contradicting the claim of the PassLLM work that prompt content has minimal impact?

RQ2 (Effectiveness) Can LLM-driven evolutionary optimization reliably improve LLM's cracking rate beyond a manually crafted baseline prompt within a fixed budget?

RQ3 (Resource efficiency) Does a locally constrained configuration (e.g., a single consumer-level GPU) achieve competitive cracking rates compared to cloud-based frontier-model ensembles?

RQ4 (Distribution realism) Can we produce password populations whose character-frequency distributions are statistically closer to real-world corpora, as measured by the per-symbol F-score AUC metric?

We use the following setting in our experimental setup.

3.1 Attacker's goals and scenario

We consider a trawling attack (mass attack) in which an attacker targets many accounts to compromise as many as possible within a fixed budget of $B = 20000$ attempts per account. This is the standard threat scenario evaluated by PassLLM, and the strength metrics are based on guesses (Herley and Van Oorschot, 2012).

3.2 Attacker's capabilities

The attacker works offline with a hacked, hashed password database. Guesses are made locally under GPU time and memory constraints. The attacker has access to public datasets (RockYou, LinkedIn (Perlroth, 2012), Collection#1) for training and white-box control over PassLLM, including prompts. For prompt engineering, the attacker can use cloud APIs or local open-sourced models. We assume a modest budget – a single consumer-level GPU and up to 100 evolution iterations.

3.3 Attacker's knowledge

The attacker has white-box access to PassLLM's architecture and weights and a small set of target passwords (e.g., previously compromised hashes). Knowledge of users' personal data, hash functions, or website policies is not necessary, as they are irrelevant to offline trawling. Protective measures such as password policies, rate limiting, and hashing methods are beyond the attacker's control.

4 System description

This section describes and formalizes the evolutionary process for optimizing prompts. An overview of the pipeline is shown in Fig. 1.

4.1 The LLM guessing model

We focus on the trawling variant of PassLLM, which is based on Qwen2.5-0.5B-Instruct (Qwen et al., 2025) and was fine-tuned via LoRA on the publicly available RockYou split as originally used in PassLLM. Inference employs a breadth-first search over the model's token lattice to efficiently enumerate the most probable strings. The model's generation behaviour is completely determined by a single input prompt $\mathbf{p}$, and this is the optimization objective of the proposed evolutionary procedure.

4.2 Target function

Given an input prompt $\mathbf{p}$, LLM generates a multiset of password candidates $\mathcal{G}(\mathbf{p}) = \{g_1, \ldots, g_B\}$ where budget B is under 2×10^4. The cracked rate CR is selected as the primary fitness function:

$$\mathrm{CR}(\mathbf{p}) = \frac{|\mathcal{G}(\mathbf{p}) \cap \mathcal{T}|}{|\mathcal{T}|}, \tag{1}$$

where $\mathcal{T}$ denotes a fixed holdout test set of passwords from the RockYou database. $|\mathcal{G}(\mathbf{p}) \cap \mathcal{T}|$ counts exact matches of character strings, taking case sensitivity into account.

4.3 Complementary metrics

Cracked rate alone does not explain why a prompt is effective. To gain deeper insight into the distribution properties of the generated passwords at the character level, we introduce a supplementary F-score metric per symbol. For each character c in the printable ASCII alphabet, f_c^{gen} and f_c^{real} denote its normalized frequency in the generated and real password sets, respectively. We define a binary "hit" for the character c at a threshold value

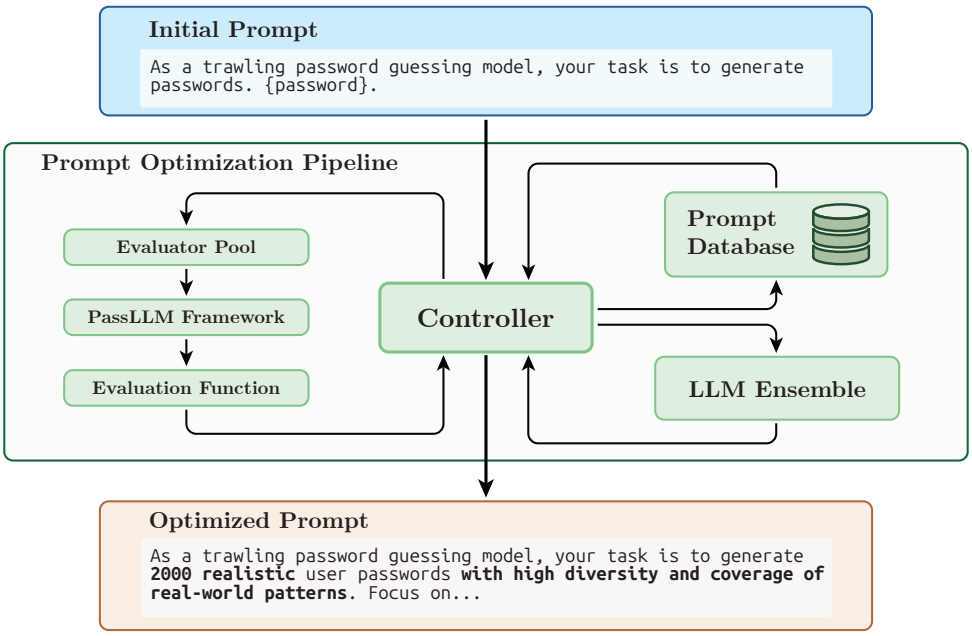

Figure 1: Overview of the proposed PassEvolve method. An initial prompt is fed to the evolutionary controller, which iteratively samples parent prompts from the MAP-Elites archive, requests improved variants from the LLM ensemble, runs LLM, and updates the archive based on the cracked-rate fitness signal.

τ as $\mathbf{1}[f_c^{\text{gen}} \geq \tau \cdot f_c^{\text{real}}]$ and calculate the macro-averaged F-score across all characters:

$$\mathrm{F}_\tau(\mathbf{p}) = \frac{2 \cdot \mathrm{Prec}_\tau(\mathbf{p}) \cdot \mathrm{Rec}_\tau(\mathbf{p})}{\mathrm{Prec}_\tau(\mathbf{p}) + \mathrm{Rec}_\tau(\mathbf{p})}, \qquad (2)$$

where precision and recall are computed over the set of characters above the frequency threshold τ. By sweeping $\tau \in [0, 1]$ we obtain an F-score curve analogous to a precision-recall curve, and summarize it with the area under the curve (AUC).

4.4 Evolutionary frameworks

Prompt evolution is carried out with three frameworks under identical migration conditions: OpenEvolve, ShinkaEvolve, and GigaEvo. All use a MAP-Elites grid to store prompts with the highest fitness for three features: complexity (token count), diversity (Levenshtein distance to reference), and prompt length. Islands migrate elite prompts using ring topology (Fig. 2): every 10 iterations, each of three islands transfers 10% of its current elite prompts to the next island, which replaces its lowest-fitness prompts with migrants. While OpenEvolve performs LLM-based semantic mutation from parent prompts and a meta-prompt, ShinkaEvolve adds adaptive LLM ensemble selection and novelty rejection sampling, and GigaEvo provides a modular implementation focused on reproducibility – all sharing the same migration schedule. At each generation, parents are selected via a mixture of elite selection (10%), exploration (20% random samples), and exploitation (70% fitness-weighted samples from the archive). The heavy exploitation bias is appropriate for the proposed setting, where the evaluation cost (PassLLM inference) is high, and we wish to maximize fitness efficiently. Algorithm 1 describes the evolutionary prompt optimization pipeline.

4.5 Models for prompt generation

Various LLMs served as mutation operators. All models were accessed either locally via Ollama (Ollama, 2025) or via a cloud service with a fixed OpenAI-compatible API. The generation parameters `temperature` = 0.4 and `max_tokens` = 16000 were set, and a fixed maximum budget of 100 evolutionary iterations was used for all experiments to ensure comparability. Three configurations were evaluated:

1. **Local (Ollama):** Qwen3-8B (Qwen Team, 2025b) for both generation and evaluation. Single 24 GB GPU.

2. **Cloud, single model:** Gemini 2.5 Flash-Lite (preview-09-2025) (Google DeepMind, 2025a) for generation; Qwen3-Max (Qwen Team, 2025a) for evaluation.

3. **Cloud, ensemble:** Weighted mixture of Qwen3-235B-A22B (Qwen Team, 2025b)

Algorithm 1: Evolutionary Prompt Optimization

Require : Initial prompt $\mathbf{p}_0$, test set $\mathcal{T}$, max iterations T, islands K, migration interval Δ, migration rate μ, LLM ensemble $\mathcal{M}$, feature vector ϕ

Ensure : Best prompt $\mathbf{p}^*$ maximising $\mathrm{CR}(\mathbf{p})$

Initialize MAP-Elites archive $\mathcal{A}_k \leftarrow \emptyset$ **for** $k = 1, \ldots, K$
$s_0 \leftarrow \mathtt{CR}(\mathbf{p}_0)$ `// $s_0 = 2.02\%$ in the proposed setting`
Insert $\mathbf{p}_0$ into $\mathcal{A}_k$ for all k
for $t \leftarrow 1$ **to** T **do**
  for *each island* $k = 1, \ldots, K$ **do** **(parallel)**
    $\mathbf{p}_{\text{par}} \leftarrow \mathtt{Select}(\mathcal{A}_k)$ `// elite, explore, exploit, mixture`
    $\mathbf{p}_{\text{new}} \leftarrow \mathtt{LLMMutate}(\mathbf{p}_{\text{par}}, \mathcal{M})$
    $s_{\text{new}} \leftarrow \mathtt{CR}(\mathbf{p}_{\text{new}})$ `// run PassLLM, count matches vs. $\mathcal{T}$`
    $\phi \leftarrow \mathtt{FeatureVec}(\mathbf{p}_{\text{new}})$
    $\mathtt{UpdateArchive}(\mathcal{A}_k, \mathbf{p}_{\text{new}}, s_{\text{new}}, \phi)$
  end
  if $t \bmod \Delta = 0$ **then**
    Migrate elites between islands: $\mathcal{A}_k \xleftrightarrow{\mu} \mathcal{A}_{(k+1) \bmod K}$
  end
end
$\mathbf{p}^* \leftarrow \arg\max_{\mathbf{p} \in \bigcup_k \mathcal{A}_k} \mathrm{CR}(\mathbf{p})$
return $\mathbf{p}^*$

(weight 0.5) and Gemini 3 Flash (Google DeepMind, 2025b) (weight 0.5) for generation; Qwen3-Max for evaluation.

Separating the generation and evaluation of LLMs reduces systematic biases caused by a single model architecture and provides a more objective fitness signal.

The most important differing parameters are presented in the Table 1.

5 Experimental setup

5.1 Datasets

All experiments use the publicly available RockYou password dataset in the train/test split, provided by the PassLLM authors (Zou et al., 2025). The test set contains passwords that were not seen during the LoRA fine-tuning of PassLLM. To test whether the evolved prompts transfer beyond the RockYou-based test set, we additionally evaluate the best prompts on the CSDN leaked password dataset, which represents an out-of-distribution (OOD) target relative to the RockYou split used in the main experiments. CSDN is a Chinese programmer-oriented password leak and has been used in prior password-security studies as a real-world leaked-password corpus (Liveck and Labs, 2012). We use the same PassLLM inference protocol and the same 20000-guess budget, but do not perform any additional prompt evolution or tuning on CSDN.

5.2 Evaluation protocol

The cracked rate (Equation 1) is the primary evaluation metric. Additionally, the F-score per symbol (Equation 2) is calculated for the best prompt of each experiment on the same test set. For each evolution run, we generate up to $B = 20000$ candidate passwords per evaluation.

Pre-trained PassLLM with its *original, unmodified* prompt (see Listing 1) was selected as the competitor for the experiments.

```
As a trawling password guessing model,
your task is to generate passwords.
{password}.
```

Listing 1: PassLLM original infernece prompt.

Experimental parameters are specified in the evolutionary frameworks' configuration files and included in the anonymous repository https://anonymous.4open.science/r/pass_evolve alongside the code for reproducibility.

6 Results

6.1 Prompt sensitivity effect

To answer **RQ1**, we compare the performance of pre-trained PassLLM with its original inference prompt instruction against evolved prompts under identical inference and evaluation protocols. PassLLM cracked rate remains at 2.02% on the hold-out set, whereas all three evolved configurations achieve substantially higher peak rates (Table 2), demonstrating that prompt content strongly conditions guessing performance.

Parameter	Ollama	Gemini	Ensemble
LLM (generation)	Qwen3-8B	Gemini 2.5 Flash-Lite	Qwen3-235B-A22B (0.5) Gemini 3 Flash (0.5)
Evaluator LLM	Qwen3-8B	Qwen3-Max	Qwen3-Max
Population size	100	1000	1000
Archive size	20	100	100
Feature dimensions	diversity, complexity	diversity, complexity	prompt length, complexity
Best cracking rate (CR)			
OpenEvolve	8.28%	7.58%	8.48%
GigaEvo	8.49%	**8.67%**	8.51%
ShinkaEvolve	8.10%	8.30%	8.60%

Table 1: Evolution settings and best cracking rates across experimental configurations.

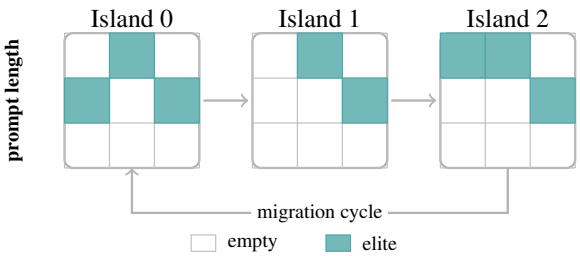

Figure 2: MAP-Elites archive and island model. Three islands each maintain an independent `prompt_length`× `complexity` grid (bins 0–9), where every cell stores one best prompt (elite). Filled green cells indicate occupied elites. Ring-topology migration occurs at fixed intervals.

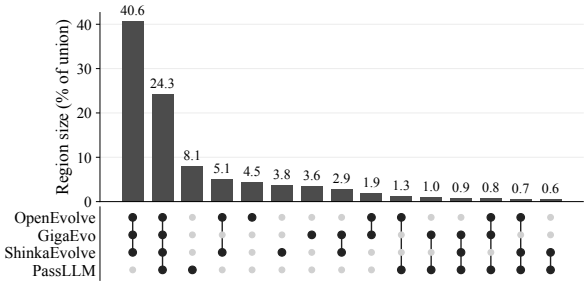

Figure 3: Exact UpSet plot of cracked password overlap across three evolutionary frameworks and PassLLM. Bars show intersection regions as percentages of the union of all cracked passwords; filled dots indicate which frameworks contribute to each overlap.

Metric	PassLLM	Ollama	Gemini	Ensemble
Mean CR	2.02%	4.83%	5.81%	6.96%
SD	–	2.05%	1.21%	1.25%
Min CR	–	0.91%	1.35%	1.43%
Best CR	2.02%	8.49%	**8.67%**	8.59%
ΔCR	–	×4.20	×4.29	×4.25

Table 2: Descriptive statistics for each experimental setting for the PassEvolve averaged across evolutionary frameworks. Mean and SD denote the mean and standard deviation of per-iteration cracking rates. Best CR denotes the peak cracking rate. ΔCR denotes improvement over PassLLM.

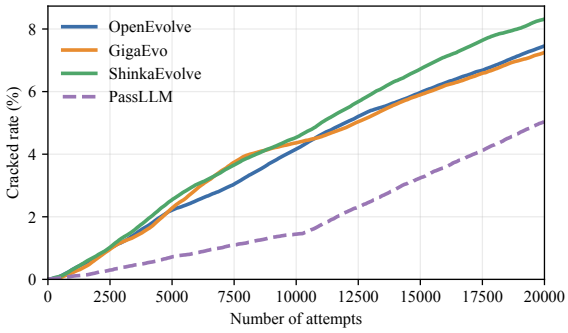

Figure 4: Cracked rate across the guessing budget.

6.2 Effectiveness of evolution

We run evolutionary frameworks and record the best CR in each run. Table 2 shows improvements of 3.75-4.3× over PassLLM. For instance, Fig. 5 shows that large gains can appear within the first iterations, with some additional gain achieved later in the run. Fig. 3 and Fig. 4 show that prompt evolution both increases the cracked rate across the guessing budget and expands coverage beyond the original PassLLM prompt, which answers the **RQ2** affirmatively.

6.3 Resource efficiency

We contrast the local single-GPU setup with cloud frontier ensembles. The local configuration reaches a best CR of 8.49%, compared with 8.60% for the ensemble. The gap is only 0.11 percentage points, which is smaller than the ensemble run's per-iteration standard deviation. This answers **RQ3**: competitive prompts can be discovered without relying exclusively on large cloud models.

Table 2 presents per-run descriptive statistics. The high standard deviation in all evolved runs ($\geq$ 1.2 percentage points) reflects the stochastic

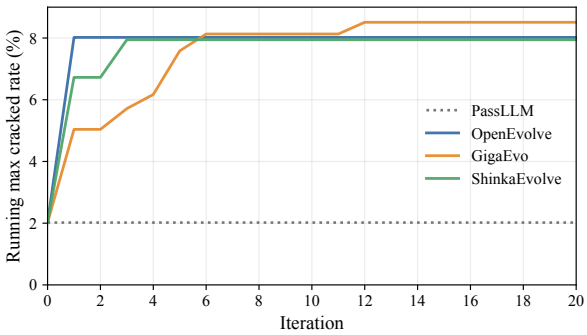

Figure 5: Running maximum cracked rate during first 20 evolution iterations. For each evolutionary framework, the curve averages three types of experiments. Further iterations are omitted because the running maxima remain unchanged.

nature of both LLM-driven mutation and PassLLM inference. Notably, the ensemble run maintains a mean cracked rate of 6.96% – already $3.4\times$ more than PassLLM – throughout the full 100-iteration run, indicating that even non-peak prompts are substantially superior to the original.

Per-iteration CRs for all three runs alongside PassLLM are shown in Fig. 5. Frameworks achieve CR $\geq 8\%$ at iteration 1, confirming that a single LLM-driven mutation can already produce a substantial improvement. Compared with the default prompt, the evolved prompt contains substantially more guidance, as shown in Listing 2.

```
As a trawling password guessing model,
generate [redacted] realistic user passwords.
Ensure passwords are based on [redacted]
[redacted] [redacted], feature [redacted], and
incorporate [redacted].
Also encourage [redacted],
[redacted].
Password:
```

Listing 2: Optimized system prompt used for password guessing. Black bars indicate sections containing instructions for creating passwords, obscured due to ethical considerations and requirements to prevent dual-use or malicious applications.

6.4 Distribution realism evaluation

To answer **RQ4**, we evaluate the best prompt from each configuration using the per-symbol F-score curves and AUC defined in Section 4.3. Fig. 6 shows the per-symbol F-score curves (Eq. 2) for the best prompts of three evolutionary frameworks evaluated at thresholds $\tau \in [0, 1]$. Table 3 summarizes the peak F-scores, AUC values, and relative gains compared to the PassLLM.

Metric	Pass-LLM	Open-Evolve	Giga-Evo	Shinka-Evolve
Peak F-score	0.063	0.073	**0.076**	0.075
AUC	0.032	0.034 (+8.0%)	0.035 (+9.8%)	**0.035** (+10.6%)

Table 3: Per-symbol F-score performance across evolutionary frameworks. Framework values are computed from F-score curves averaged over the three prompt-generation settings within each framework. Peak F-score denotes the best threshold-level score; AUC summarizes performance across thresholds. The relative gain is computed against PassLLM.

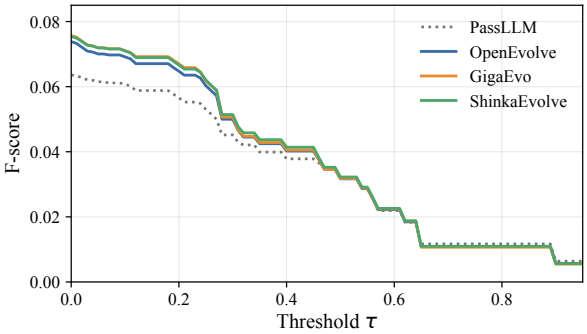

Figure 6: Per-symbol F-score curves for PassLLM and evolved prompts, averaged across the three types of experiments.

The ensemble prompt achieves a peak F-score of 0.076 – a 9.8% improvement over the PassLLM peak of 0.063. The AUC improvement from 0.032 (PassLLM) to 0.035 (ensemble) represents a 10.6% gain, confirming that the higher cracked rate of evolved prompts is not merely due to over-generating highly common passwords, but reflects a genuine improvement in character-distribution coverage. Higher AUC indicates better alignment between the generated passwords' character-frequency distribution and that of real passwords. The evolved frameworks consistently improve over the PassLLM, indicating broader character-level coverage.

6.5 Out-of-distribution evaluation

Table 4 shows that the evolved prompts consistently outperform the original PassLLM prompt under this OOD setting. PassLLM reaches 0.70% CR, whereas all evolved prompts improve over it, ranging from 1.41% to 2.54%. The strongest performance is achieved using the prompt, obtained in the OpenEvolve Local scenario, for which PassLLM

Prompt	Framework	CR	ΔCR
Local	OpenEvolve	**2.54%**	**×3.63**
Local	GigaEvo	2.12%	×3.03
Local	ShinkaEvolve	2.25%	×3.21
Gemini	OpenEvolve	1.41%	×2.01
Gemini	GigaEvo	2.14%	×3.06
Gemini	ShinkaEvolve	2.28%	×3.26
Ensemble	OpenEvolve	2.23%	×3.19
Ensemble	GigaEvo	2.04%	×2.91
Ensemble	ShinkaEvolve	2.18%	×3.11
Original prompt	PassLLM	0.70%	–

Table 4: Out-of-distribution evaluation on the CSDN leaked-password dataset. Prompts were evolved on the RockYou-based setting and evaluated without additional optimization on CSDN. CR denotes the cracked rate; ΔCR reports improvement over the original PassLLM prompt on the same CSDN split.

reaches CR of 2.54%, corresponding to a 3.63× improvement over its original prompt. This indicates that prompt evolution is not merely overfitting to the RockYou test distribution. Although absolute cracking rates decrease on CSDN, the evolved prompts demonstrate significantly better OOD robustness.

7 Discussion

The experimental results first underline the improvement in rapidity. The ensemble achieves $\mathrm{CR} \geq 8\%$ already at first iteration, confirming that even a single LLM-backed mutation of the original prompt produces qualitatively better instructions. This gain reflects the sub-optimal nature of PassLLM, which provides limited task-specific guidance by design. Methodologically, results *contradict the conclusions of the work in (Zou et al., 2025) where prompt content has minimal impact on guessing performance. We attribute PassLLM limitations primarily to manual search of prompts with similar specificity levels, whereas the proposed evolution procedure enables explorations beyond the competitor's password-guessing LLM setup*.

The choice of MAP-Elites feature dimensions proves consequential. Substituting `prompt_length` for `diversity` yielded the best result, suggesting that effective prompts occupy a characteristic length range specific enough to conduct *attack strategies*, yet concise enough to remain syntactically coherent, and that this range is captured more faithfully by a length-based grid than by a diversity-based one. Convergence behavior further supports the value of the islands model. After 20 iterations, the archive gains stability and shows that 100 iterations are insufficient to escape the resulting basin. Longer runs or larger islands ($K \geq 5$) may push the bound even further.

The near-parity between the local Qwen3-8B with 8.49% and the cloud ensemble with 8.60% is practically significant. The 0.11 point gap is acceptable within one standard deviation of the ensemble run at a $\sigma = 1.25\%$. This confirms that the method is viable in self-hosted, air-gapped deployments, which is an important consideration for organizations with privacy constraints like banks or healthcare institutions.

8 Conclusion

In this work, we proposed PassEvolve, a methodology for automated evolutionary inference prompt optimization for the LLM password-guessing task by integrating the password-guessing LLM model with the evolutionary frameworks.

Experiments with three evolutionary frameworks and three configurations of locally inferred 8B models, cloud models, and ensembles show an increase from 4.2 to 4.29× improvements in cracked rate, surpassing PassLLM (2.02% vs. up to 8.67%). With the local-only setup, the setup achieves 8.49% in as few as up to 10 iterations. Additional per-symbol F-scoring confirms that evolved prompts not only found more correct passwords, but they also produced character frequency distributions that are substantially closer to real-world scenarios, as experiments show with a 10.6% increase in terms of AUC. Moreover, PassEvolve demonstrated better out-of-distribution generalization on the CSDN benchmark. Therefore, prompt evolving frameworks matter more than assumed, as the original PassLLM prompts are showing sub-optimal performance, while evolution procedures rapidly discover better approaches.

Future work might be devoted to testing on additional password distributions, extending to PII-based and password-reuse scenarios, exploring richer island constructions and optimization strategies, and investigating the transfer of evolved prompts across corpora without re-optimization. A defensive inversion is to employ the same framework to evolve password policies that minimize the attacker's cracking rate, reinforcing existing password security protocols.

9 Limitations

Our method has several limitations:

- **Guessing scenario.** This work focuses on the trawling setting and does not include targeted modes experiments where a person uses PII to guess a password.
- **Stable gains.** Improving performance on a specific dataset may require additional empirical tuning of the evolution framework's hyperparameters.

10 Ethical considerations

This paper contains potentially harmful content. We study behaviors that could, if misused, lower the cost of offline guessing. Experiments use only publicly leaked password research artifacts (RockYou split, CSDN) and do not involve live accounts. The contribution is methodological: we do not publish raw evolved prompts or turnkey guess lists, so reproducing peak numbers requires re-running the pipeline with non-trivial compute and domain setup. We align dual-use reporting with norms in password research (Florêncio et al., 2014): report aggregate results to help defenders understand emerging attack capabilities, but avoid releasing details that could make misuse easier.

10.1 Generative AI usage

LLMs are part of the experimental methodology described in this paper and are used for prompt generation, mutation, and/or password-guessing evaluation as specified in the corresponding sections. Separately, generative AI tools, including ChatGPT, DeepSeek, and Grammarly, were used only for text polishing and shortening.

References

Davis Brown, Jesse He, Helen Jenne, Henry Kvinge, and Max Vargas. 2025. Even with ai, bijection discovery is still hard: The opportunities and challenges of openevolve for novel bijection construction. *Preprint*, arXiv:2511.20987.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *Preprint*, arXiv:2309.16797.

Dinei Florêncio, Cormac Herley, and Paul C. Van Oorschot. 2014. An administrator's guide to internet password research. In *Proceedings of the 28th USENIX Conference on Large Installation System Administration*, LISA'14, pages 35–52, USA. USENIX Association.

Google DeepMind. 2025a. Continuing to bring you our latest models, with an improved gemini 2.5 flash and flash-lite release. Google for Developers Blog.

Google DeepMind. 2025b. Gemini 3 flash: frontier intelligence built for speed. Google for Developers Blog.

Hashcat. 2025. Hashcat.

Cormac Herley and Paul Van Oorschot. 2012. A research agenda acknowledging the persistence of passwords. *IEEE Security and Privacy*, 10(1):28–36.

Briland Hitaj, Paolo Gasti, Giuseppe Ateniese, and Fernando Perez-Cruz. 2019. Passgan: A deep learning approach for password guessing. *Preprint*, arXiv:1709.00440.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3.

Troy Hunt. 2019. The 773 million record "collection #1" data breach. Troy Hunt's Blog.

Mohammed Jubur, Prakash Shrestha, and Nitesh Saxena. 2025. An in-depth analysis of password managers and two-factor authentication tools. *ACM Comput. Surv.*, 57(5).

Valentin Khrulkov, Andrey Galichin, Denis Bashkirov, Dmitry Vinichenko, Oleg Travkin, Roman Alferov, Andrey Kuznetsov, and Ivan Oseledets. 2025. Gigaevo: An open source optimization framework powered by llms and evolution algorithms. *Preprint*, arXiv:2511.17592.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. 2025. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *Preprint*, arXiv:2509.19349.

Liveck and Antiy Labs. 2012. Analysis report on password data leaked by websites like CSDN. Technical report, Antiy Labs.

Riccardo Lunardi, David La Barbera, and Kevin Roitero. 2024. The elusiveness of detecting political bias in language models. In *Proceedings of the 33rd acm international conference on information and knowledge management*, pages 3922–3926.

William Melicher, Blase Ur, Sean M. Segreti, Saranga Komanduri, Lujo Bauer, Nicolas Christin, and Lorrie Faith Cranor. 2016. Fast, lean, and accurate: modeling password guessability using neural networks. In *Proceedings of the 25th USENIX Conference on Security Symposium*, SEC'16, pages 175–191, USA. USENIX Association.

Microsoft. 2024. Microsoft digital defense report 2024. Accessed: 2026-05-19.

Arvind Narayanan and Vitaly Shmatikov. 2005. Fast dictionary attacks on passwords using time-space tradeoff. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, CCS '05, pages 364–372, New York, NY, USA. Association for Computing Machinery.

Ollama. 2025. Ollama: Get up and running with large language models locally. Version: 0.6.5.

Openwall. 2025. John the ripper.

Nicole Perlroth. 2012. Linkedin was breached. now what do you do? *The New York Times*.

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

Qwen Team. 2025a. Qwen3-max: Just scale it. Qwen AI Blog.

Qwen Team. 2025b. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Javier Rando, Fernando Perez-Cruz, and Briland Hitaj. 2023. Passgpt: Password modeling and (guided) generation with large language models. *Preprint*, arXiv:2306.01545.

Gabriel Arquelau Pimenta Rodrigues, Pedro Augusto Giacomelli Fernandes, André Luiz Marques Serrano, Geraldo Pereira Rocha Filho, Guilherme Fay Vergara, Guilherme Dantas Bispo, Robson de Oliveira Albuquerque, and Vinícius Pereira Gonçalves. 2025. From rockyou to rockyou2024: analyzing password patterns across generations, their use in industrial systems and vulnerability to password guessing attacks. *Journal of Internet Services and Applications*, 16(1):69–86.

Paul Röttger, Valentin Hofmann, Valentina Pyatkin, Musashi Hinck, Hannah Kirk, Hinrich Schuetze, and Dirk Hovy. 2024. Political compass or spinning arrow? towards more meaningful evaluations for values and opinions in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15295–15311.

Divya Saxena and Jiannong Cao. 2021. Generative adversarial networks (gans): Challenges, solutions, and future directions. *ACM Comput. Surv.*, 54(3).

Asankhaya Sharma. 2025. Openevolve: an open-source evolutionary coding agent.

Dan Shi, Tianhao Shen, Yufei Huang, Zhigen Li, Yongqi Leng, Renren Jin, Chuang Liu, Xinwei Wu, Zishan Guo, Linhao Yu, Ling Shi, Bojian Jiang, and Deyi Xiong. 2024. Large language model safety: A holistic survey. *Preprint*, arXiv:2412.17686.

Xingyu Su, Xiaojie Zhu, Yang Li, Yong Li, Chi Chen, and Paulo Esteves-Veríssimo. 2024. Pagpassgpt: Pattern guided password guessing via generative pretrained transformer. In *2024 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 429–442. IEEE.

Chunhui Wan, Xunan Dai, Zhuo Wang, Minglei Li, Yanpeng Wang, Yinan Mao, Yu Lan, and Zhiwen Xiao. 2025. Loongflow: Directed evolutionary search via a cognitive plan-execute-summarize paradigm. *Preprint*, arXiv:2512.24077.

Yangde Wang, Weidong Qiu, Peng Tang, Hao Tian, and Shujun Li. 2025. Se#pcfg: Semantically enhanced pcfg for password analysis and cracking. *IEEE Transactions on Dependable and Secure Computing*, 22(4):4428–4441.

Matt Weir, Sudhir Aggarwal, Breno de Medeiros, and Bill Glodek. 2009. Password cracking using probabilistic context-free grammars. In *2009 30th IEEE Symposium on Security and Privacy*, pages 391–405.

Ming Xu, Jitao Yu, Xinyi Zhang, Chuanwang Wang, Shenghao Zhang, Haoqi Wu, and Weili Han. 2023. Improving real-world password guessing attacks via bi-directional transformers. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 1001–1018, Anaheim, CA. USENIX Association.

Jie Zhang, Haoyu Bu, Hui Wen, Yongji Liu, Haiqiang Fei, Rongrong Xi, Lun Li, Yun Yang, Hongsong Zhu, and Dan Meng. 2025. When LLMs meet cybersecurity: a systematic literature review. *Cybersecurity*, 8(1):55.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2023. Large language models are human-level prompt engineers. *Preprint*, arXiv:2211.01910.

Jingming Zhuo, Songyang Zhang, Xinyu Fang, Haodong Duan, Dahua Lin, and Kai Chen. 2024. Prosa: Assessing and understanding the prompt sensitivity of llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1950–1976.

Yunkai Zou, Maoxiang An, and Ding Wang. 2025. Password guessing using large language models. In *Proceedings of the 34th USENIX Conference on Security Symposium*, SEC '25, USA. USENIX Association.