# SecureLLM: Using Inference-time Compositionality to Build Secure Language Models

**Abdulrahman Alabdulkareem*, Christian M Arnold*†, Yerim Lee, Pieter M Feenstra, Conner Arnold, Boris Katz, Andrei Barbu, Brian Cheung,**
CSAIL and CBMM, MIT
* Equal Contribution, †US Space Force
{arkareem,cmarnold,boris,abarbu}@mit.edu

## Abstract

As Large Language Models (LLMs) increasingly support critical sectors such as healthcare, finance, and public governance, ensuring data confidentiality and robust access control is a pressing societal challenge. Traditional security mechanisms isolate sensitive resources from unauthorized users, yet existing LLM safety approaches often fail to enforce strict segregation of confidential data. In this work, we introduce SECURELLM, a novel compositional framework for building secure LLMs that integrates fine-tuning with traditional access security measures to protect private information. By fine-tuning LLMs on segregated, "siloed" training data and composing their outputs at inference time based solely on a user's verified credentials, SECURELLM not only prevents unauthorized data leakage but also enables accurate responses for complex queries spanning multiple data silos. Our method is demonstrated on a challenging natural-language-to-SQL translation task and is designed with real-world applications in mind, where protecting sensitive information is critical.

## 1 Motivation

In today's data-driven society, the deployment of LLMs in domains such as healthcare, finance, and public administration promises transformative benefits but also introduces significant risks. While LLMs excel at natural language understanding and generation, their vulnerability to prompt injection and data leakage can compromise sensitive information, a risk that is unacceptable in applications subject to strict privacy and regulatory requirements.

LLMs can be induced to reveal sensitive information, as demonstrated by prompt-hacking and jailbreak techniques for malicious content generation such as Do Anything Now (DAN) (Shen et al., 2024). To mitigate such attacks, many systems rely on "guardrail" models, which are auxiliary classifiers or rule-based filters designed to detect and block unsafe inputs or outputs before they reach the user. These guardrails operate as post hoc defenses, screening prompts or generated text for policy violations, prohibited content, or indicators of misuse. While guardrail models have been effective for sanitizing outputs in online deployments, they remain fundamentally reactive. For every guardrail developed, new attack strategies rapidly emerge to bypass it (Mangaokar et al., 2024; Banerjee et al., 2024; Dutta et al., 2024; Andriushchenko et al., 2024).

For enterprises where security must be guaranteed by law and regulation, including finance, healthcare, and national security, guardrails are therefore not legally sufficient to prevent the leakage of sensitive information. No prior work provides a method that guarantees data security for information silos that must remain isolated and enforce credential-based access controls. This limitation severely restricts the adoption of LLMs in security-critical domains. We present the first method for building secure LLM systems by leveraging compositionality to enforce access control guarantees comparable to those of credential-based security mechanisms.

Our work rethinks LLM security by embedding access control directly into the inference pathway rather than relying on post hoc filtering. We introduce SECURELLM, a framework that exploits the compositional structure of traditional security: permissions are defined over a set of resources (e.g., documents, databases, or silos), and secure systems must remain correct under composition operations such as adding a permitted resource, removing a forbidden one, and answering queries that require combining multiple permitted resources. In other words, access control is *set-valued* and *closed under composition*: a user's effective authority is the union of the silos they are authorized to access,



*Proceedings of the Seventh Workshop on Privacy in Natural Language Processing*, pages 26–52

and every response must be a function only of that authorized set. SECURELLM instantiates this principle by fine-tuning separate LLMs on isolated data silos and composing their behaviors at inference time conditioned on verified user credentials. At runtime, SECURELLM constructs a user-specific model $M_T$ from only those silo models indexed by the user's permission set $T$, ensuring that the generated output cannot depend on any silo outside $T$ and therefore cannot leak unauthorized information. At the same time, composition enables *cross-silo* queries: when a question requires the joint use of multiple authorized silos, SECURELLM can combine the relevant specialized competencies even though no single silo model can resolve the query in isolation. By reducing LLM confidentiality to the same compositional invariants enforced by a credential-based system, SECURELLM provides a principled path toward enhanced data security in regulated deployments while retaining the utility needed for real workflows.

We consider the scenario where an organization has $N$ separate and confidential data silos that must be kept separate for legal purposes, and there are users who legitimately have access to some arbitrary subset of $N$. We make the following assertions of properties that must be present to call a model secure:

1. The model must accurately respond to prompts on data for which the user has verified credentials;
2. The model must accurately respond to prompts that require the union of segregated data silos; and
3. The model will *never* reveal data trained from an unauthorized information silo to an unauthorized user.

We emphasize that this notion of security does not claim to prevent all possible inferences about unauthorized information. As in traditional access-controlled systems, it may be the case that information in a forbidden silo is logically implied by, correlated with, or reconstructible from the combination of data in multiple authorized silos. Preventing such semantic inference is, in general, impossible without severely restricting utility. Instead, SECURELLM enforces a *non-contribution* guarantee: the model does not introduce, encode, or explicitly expose information from unauthorized silos through its parameters or outputs. In contrast to a monolithic model trained jointly on all silos, which may internalize and surface sensitive

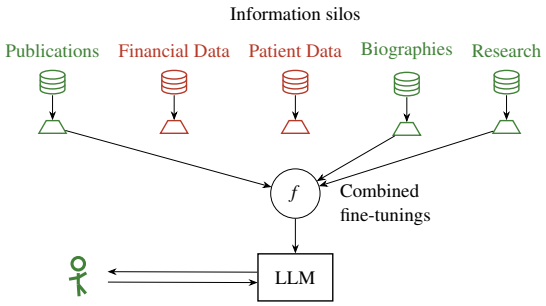



Figure 1: Assuming a perfect compositional function $f$ that runs at inference time, we propose a method that guarantees information security. Each model is fine-tuned on a previously segregated information silo. The user's credentials are validated using traditional security methods, and inference is only run on models for which the user has verified access. The outputs of each fine-tuned model are composed at inference time with the function $f$ and that single composition is passed to the user. Thus, SECURELLM reduces the problem from LLM security to that of existing information security systems. Existing compositional fine-tuning methods fail in this challenging environment. SECURELLM presents a new method that better approximates the function $f$.

information even when queried by unauthorized users, SECURELLM ensures that any generated response is a function only of the data sources for which the user holds verified credentials. In this sense, SECURELLM does not eliminate all potential downstream misuse or inference, but it avoids adding new leakage channels beyond those already inherent in the authorized data itself.

Trivially, one could fine-tune many models on the power set of $N$; but this is obviously infeasible. Using this trivial method, the number of models required to satisfy our Secure Model Properties is $\mathcal{O}(2^N)$ which quickly becomes impractical. Instead, we show how to achieve the same goals with a linear number of LLM fine-tunings (Figure 1). While using only one fine-tuned model per silo, we can configure and compose a model specific to the user's permissions at runtime.

While other methods have demonstrated compositionality for similar tasks, there are none that have been designed for situations where information silos are entirely disjoint. Disjointness makes composition substantially harder because there is no shared vocabulary, schema overlap, or transferable structure across silos that a model can exploit; a composed model must effectively "activate" the correct silo-specific parameters and then

stitch them together without any cross-silo training signal. To illustrate the compositional properties of SECURELLM, we consider the task of using natural-language-to-SQL translation. In this task, each SQL schema is entirely disjoint and prompts do not contain the exact table or column name, thus requiring the model to have perfect parameterized knowledge of the schema. For example, answering a query that simultaneously references instructors and course levels from one silo and appliance availability and store locations from another requires the model to correctly compose two unrelated SQL schemas without any shared identifiers or schema information in the prompt. SQL translation offers an extreme test of compositionality, because information is explicitly disjoint, and the task serves to demonstrate in an easily verifiable manner the efficacy of SECURELLM compared to other compositional methods. For practical SQL translation of the same task, it is simply easier to pass the siloed database schemas as part of the prompt to achieve the same result.[1]

LLM composition for security is reminiscent of recent work like LoraHub (Huang et al., 2023) which also composes fine-tunings. Given a target task, LoraHub selects a set of fine-tunings, Low Rank Adapters (LoRAs) (Hu et al., 2021), that are added together. However, those works are mostly designed for soft tasks where silos share structure and content. LLMs already perform on these soft tasks and composition only increases performance by a few percent.

The domain we consider here is different; we seek compositionality for unrelated silos of information. Questions that require access to silos A and B, by definition, cannot be answered with access to only A or only B. The underlying performance of individual models is nearly zero. LoraHub is not well-suited to such tasks and performs poorly. Our compositional method in SECURELLM solves this issue and can generate accurate responses when the union of silos A and B is required.

[1]Explicitly providing the database schemas in the prompt removes the core compositional challenge where the model can simply rely on in-context learning to provide an accurate query. In this setting, cross-silo queries no longer test whether the model can compose disjoint internal representations, but instead whether it can follow explicit instructions over a shared textual context. While this approach is effective for practical NL2SQL systems, it sidesteps the security and compositionality questions that SECURELLM is designed to address.

### 1.1 Contributions

Our contributions are:

1. We formulate a new compositional task that LLMs have difficulty with – natural-language-to-SQL where not only a model is trained on queries of individual databases but accurate responses require cross-database joins;
2. We formulate the notion of access security in terms of this task;
3. We demonstrate that existing fine-tuning methods fail in this compositional environment;
4. We introduce new compositional methods for this problem.

While we concern ourselves only with the task of understanding queries by translating them to SQL, our methods are generic and can be applied to numerous other domains like translating commands to API calls and answering questions from large collections of documents.

## 2 Related Work

**Model Composition**. Our framework relies on composing LLM fine-tunings at inference time which follows a set of previous works that use model composition. A recent method combines pre-trained LLM prompts each tuned for separate tasks to achieve generalization on downstream tasks (Sun et al., 2023) but requires training prior to inference. AdapterSoup composes fine-tunings by linearly averaging fine-tuned weights depending on criterion to determine which fine-tunings are relevant to the new domain (Chronopoulou et al., 2023). PEM Addition is a training-free composition method that combines fine-tunings via arithmetic operations on model weights, and is therefore directly applicable to compositional security testing without modification (Zhang et al., 2023). LoraHub is a recent simple framework that also composes different LoRA fine-tunings (Hu et al., 2021) at inference time where each fine-tuning is trained on a different task (Huang et al., 2023); we include comparisons using LoraHub and PEM Addition.

**Privacy Attacks**. Many recent works discuss a range of different privacy attacks against large language models and Deep Learning models in general. Membership inference is a type of privacy attack which tries to determine if a piece of text was contained in the training data of a model (Hisamoto et al., 2020; Nasr et al., 2019; Hu et al., 2022). An even larger security risk is posed by training data extraction attacks where large language models

leak text in their training data verbatim (Carlini et al., 2019) including personally identifiable information (Inan et al., 2021). This attack is shown to be successful even when such data was only mentioned in a single document and this behavior worsens with an increase in model size (Carlini et al., 2021). Similarly, training data extract attacks were effective on models fine-tuned on a smaller dataset (Zanella-Béguelin et al., 2020). Recent work tackles these privacy issues for Retrieval-Augmented Generation using multi-party communication (Zyskind et al., 2023).

**Differential Privacy**. A popular algorithmic technique to train machine learning models with certain privacy guarantees is differential privacy (Abadi et al., 2016) which has also been applied to large recurrent language models (McMahan et al., 2017). Multiple recent works manage to use differentially private learning on large language models with hundreds of millions of parameters to achieve efficient differentially private fine-tuning with slight degradation in performance (Li et al., 2021; Yu et al., 2021). Many other methods borrow inspiration from differential privacy like Confidentially Redacted Training which provably prevents memorization of the training data (Zhao et al., 2022). However, there are differences between Differential Privacy and our approach. In differential privacy, there exists a non-zero privacy loss parameterized by the privacy budget ($\epsilon$ and $\delta$) from the resulting model because the privatization step minimizes but does not entirely prevent the updated model parameters from leaking private information.

Much prior privacy-preserving work for LLMs, including differential privacy, treats individual records as the privacy unit by aiming to prevent memorization or leakage of any single example while still permitting the model to learn aggregate trends. In contrast, we treat an entire private silo as the confidentiality unit such that any information attributable to that silo, whether record-level or holistic, must remain private. Accordingly, rather than probabilistic leakage bounds, we provide a strict access-control guarantee where a user interacts only with model weights trained exclusively on data they are authorized to access.

## 3 Framework

SECURELLM takes several fine-tunings each trained on a distinct information silo, and composes them at inference time. The goal of the composed model is to answer questions about both individual silos and questions that span silos. For example, in our case, a natural-language to SQL LLM would need to be able to generate joins across the databases of multiple silos to answer complex questions that have never been seen at training time. This task is trivial for humans, but one that challenges LLMs. We go a step further: not only must such an LLM work, it must operate through a combination of fine-tunings, i.e., not only has it never seen combinations of silos at training time, its fine-tunings have only ever seen a single silo each. This challenges, and defeats, current fine-tuning methods. The upshot of this difficult task is that it highlights several key security problems for LLMs.

### 3.1 Compositional Security Guarantee

SECURELLM enforces confidentiality in the standard access-control sense: a user should not be able to obtain any information whose provenance depends on silos they are not authorized to access. Concretely, given confidential silos $\{S_1, \ldots, S_N\}$, silo-specific fine-tunings (e.g., LoRAs) $\{A_1, \ldots, A_N\}$ where each $A_i$ is trained *only* on $S_i$, and a credential set $T \subseteq \{1, \ldots, N\}$ produced by a conventional authentication/authorization system, SECURELLM instantiates at inference time a user-specific model

$$M_T \;=\; \mathrm{Compose}(M_0, \{A_i\}_{i \in T})\,,$$

and answers all queries using only $M_T$. Critically, the inference pathway for a user with credentials $T$ never loads, queries, or composes any adaptation $A_j$ for $j \notin T$.

This yields a *non-contribution* (non-interference) guarantee: for any fixed credential set $T$, if the contents of unauthorized silos $\{S_j\}_{j \notin T}$ were changed arbitrarily (including adversarially), the distribution of outputs produced to that user would remain unchanged, because the parameters used at inference are a function only of the public base model $M_0$ and the authorized adaptations $\{A_i\}_{i \in T}$. Equivalently, there is no computational path by which unauthorized silos can influence the model instance used to generate the response.

**Assumptions and scope.** The guarantee holds under two standard systems assumptions: (i) each silo adaptation $A_i$ is trained only from its silo $S_i$ (and any public data already in $M_0$), and (ii) the inference system has no side channels that query

Q: What's the average age of all teachers that are older than 72 or that taught art classes for 9th graders in the school. Answer:

```sql
SELECT AVG(instructors.teacher_age)
FROM instructors INNER JOIN classes
   ON instructors.teacher_id = classes.teacher_id
WHERE instructors.teacher_age >= 72
   OR classes.class_subject = 'art' AND
        classes.level = 9
```

(a) Sample from Silo 1 ($S_1$)

Q: What's the minimum height of all appliances in the inventory that are currently unavailable in stores located in NY, CA, or MA and with a rating higher than or equal to 2 stars. Answer:

```sql
SELECT MIN(inventory.height)
FROM inventory INNER JOIN store ON
   store.store_id = inventory.store_id
WHERE inventory.available = 0
   AND (store.location = 'NY'
    OR store.location = 'CA'
    OR store.location = 'MA')
   AND store.star_rating >= 2
```

(b) Sample from Silo 2 ($S_2$)

Q: Provide the names of all managers located in TX and the names of all teachers that are younger than 86 and that taught English, Sociology, or Art classes that achieved a grade higher than 89 in the database. Answer:

```sql
SELECT store.name
FROM classes
INNER JOIN instructors ON instructors.teacher_id
     = classes.teacher_id
INNER JOIN store ON store.name = instructors.name
WHERE store.location = 'TX'
 AND instructors.teacher_age <= 86
 AND (classes.class_subject = 'english' OR
      classes.class_subject = 'sociology' OR
      classes.class_subject = 'art')
 AND classes.grade >= 89
```

(c) Sample from Union Silo 1,2 ($S_{1\cup 2}$)

Figure 2: Examples of input/output pairs of a question paired with the target SQL query which are unconditional samples from a Context-Free Grammar.

unauthorized resources (e.g., external databases, other models, or cached logs). This is intentionally stronger than post hoc "guardrails": it is not a promise to *detect* attacks, but a statement about *information flow*—unauthorized silos are excluded from the instantiated model and therefore cannot contribute to the output. The guarantee does not claim to prevent semantic inference from authorized information, nor does it address other safety dimensions (toxicity, hallucinations, instruction-following failures). A formal definition and proof are provided in the Appendix.

### 3.2 Composing fine-tunings

There are existing methods that can be used for compositional security, but these methods were not intended for this purpose, so it comes as no surprise that they fail to achieve sufficient compositions. The two methods we tested against our method are described below, with additional methods we considered described in the appendix.

**LoraHub** The LoraHub method for composition introduced by (Huang et al., 2023) is a two-step process involving element-wise summation of LoRA fine-tunings (*COMPOSE*), and then learning weight optimizations via gradient-free methods to apply to each fine-tuning (*ADAPT*). For this study, we do not implement the *ADAPT* stage because weights for every possible combination would need to be learned, and we could not say that this process is completed at inference time.

We observe that LoraHub performs poorly on the secure composition task, as we show later in Section 6. The authors warn that combining too many fine-tunings can lead to poor performance; however, this cannot be the source of poor performance as we compose no more than three LoRAs.

**PEM Addition** The summation method introduced by (Zhang et al., 2023) is similar to LoraHub; however, instead of summing the embeddings of the encoder and decoder prior to receiving the input $x$, one executes each fine-tuning independently at the attention-layer level, and then adds the result. This version of summed composition shows improved performance over LoraHub.

### 3.3 Our Methods

**Maximum Difference** Given LoRA deltas $\{L_i(x)\}_{i=1}^n$, we form an element-wise signed maximum-absolute aggregate

$$h_{\max}(x)[e] = L_{i^*}(x)[e],\ i^* \in \operatorname*{argmax}_i |L_i(x)[e]|$$

The composed output is $h = \mathit{Attention}(x) + h_{\max}(x)$ (Appendix G and Algorithm 1).

**Logit Composition** Given fine-tunings to compose $M_1, \cdots, M_n$ and input $x$, we define logit composition as performing the complete forward pass for each fine-tuning independently to obtain logit probabilities. We select the maximum value of each logit. This can simply be interpreted as each fine-tuning independently predicting the next token and only picking the token from the most confident fine-tuning. (Appendix G and Algorithm 2)

We do not claim that this method is a superior compositional approach in all settings. The requirements of compositional security differ fundamentally from those of other compositional tasks. In security-driven composition, most component

models are typically misaligned with a given query because they were trained on disjoint and unrelated information silos. As a result, only a small subset of composed components produces high-magnitude, task-relevant activations, while the remaining components contribute low-signal or noisy responses. Our compositional methods are explicitly designed to amplify these dominant, task-relevant signals while suppressing spurious contributions from unrelated components, which explains both their effectiveness in this setting.

## 4 Data generation

Our goal is to automatically create a challenging dataset for compositions of silos (SecureSQL). While there are countless other NL2SQL datasets, none specifically focus on SQL queries for disjoint and unrelated database silos. Secure-NL2SQL contains three silos of disjoint schemas pertaining to different subjects, as well as the superset of unions between those three silos for a total of seven permutations $(S_1, S_2, S_3, S_{1\cup2}, S_{1\cup3}, S_{2\cup3}, S_{1\cup2\cup3})$. The dataset contains automatically generated questions and corresponding SQL queries across each silo and the union-silos focus on questions requiring knowledge from multiple silos.

We automatically generate SQL databases, one per silo, with 2-3 tables per database, that share columns which can be joined together both within and between databases. However, the databases are otherwise disjoint and contain different topics. Two methods are used to generate these pairs: a CFG (see Figure 2) and GPT-4 (see Figure 5). We provide additional details on the data generation process in Appendix E. For evaluating the generated outputs compared to the ground-truth query, we use ROUGE-L (Lin, 2004). However, we note that this is not optimal for capturing the distance between SQL queries, and we provide results of four other metrics along with a more proper SQL tree-edit distance metric (TED) (Zhang et al., 1996). More details discussed in Appendices E and F.

## 5 Experiments

To evaluate inference time model composition, we consider four base LLMs: Llama 3 (8B and 11B), Qwen 2 (2B), and Qwen 3 (8B). Then we obtain silo-specific fine-tunings by independently fine-tuning the base model using a Low-Rank Adaptation (LoRA) on each data silo $S_i$. These silo-specific LoRAs can be composed using several composition schemes, with the strict requirement that all composition occurs at inference time and requires no additional training.

We also train two insecure baseline models that serve as upper bounds on performance. The first, referred to as the *generalized baseline*, is trained jointly on all individual silos $(S_1, S_2, S_3)$. The second, referred to as the *exponential baseline*, is trained on all silos $(S_1, S_2, S_3, S_{1\cup2}, S_{1\cup3}, S_{2\cup3}, S_{1\cup2\cup3})$. Although this model achieves maximal performance, it violates privacy guarantees by construction. The term "exponential" reflects the fact that preserving privacy using this approach would require training $\mathcal{O}(2^N)$ separate models for $N$ silos, which is infeasible in practice.

Both baselines are considered insecure as they embed knowledge from all silos into the same parameters, with no mechanism for removing silo-specific information at inference time based on user credentials. In contrast, SECURELLM is capable of composing the silo-specific subset of fine-tunings which the user is authorized to access, enabling fine-grained control over which information is available during inference.

We report the results of the two insecure baseline models along with the secure composition ($M_1 \oplus M_2 \oplus M_3$ where $M_i$ was trained on Silo$_i$) using multiple compositional methods (as described in Framework) including our best method (Framework: Logit Composition) which is equivalent to the scenario where a user has credentials to access $T = \{1, 2, 3\}$. We note that neither the baseline generalized model nor the secure compositions have seen the union Silos ($S_{1\cup2}$, $S_{1\cup3}$, $S_{2\cup3}$, and $S_{1\cup2\cup3}$) which were only seen by the exponential baseline. The performance of the composed fine-tunings on the individual Silos would give an indication as to whether the resulting composition is able to retain the knowledge of each of its individual fine-tunings despite being $M_1 \oplus M_2 \oplus M_3$; This performance is expected to be traded off for privacy while the better compositional methods mitigate the extent of this trade-off and maintain maximal privacy. The performance on the union Silos indicates whether the composed fine-tunings are able to successfully generalize knowledge from the individual fine-tunings which is an essential component in answering questions that no individual fine-tuning or silo can answer.

We emphasize again that no additional training or gradient updates are required for our secure com-

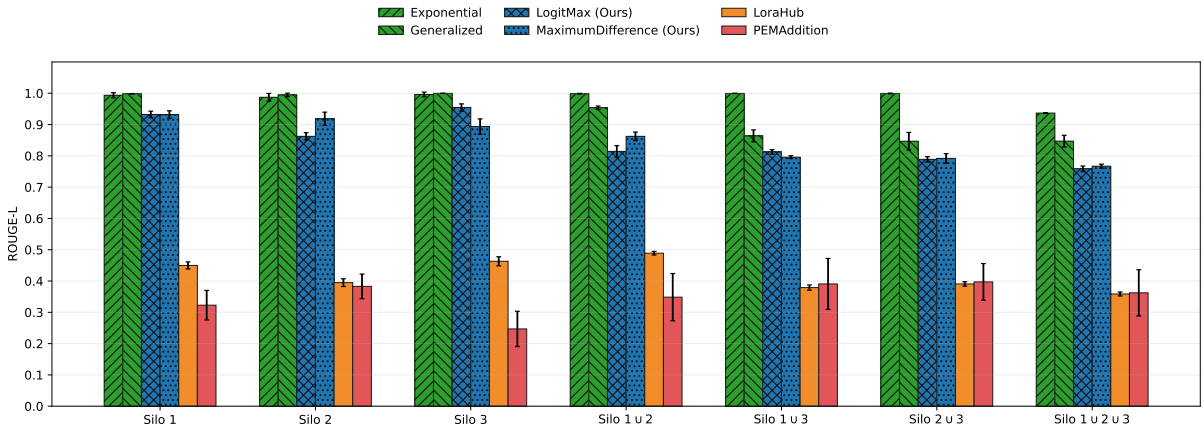



Figure 3: **ROUGE-L** metric for Llama 3 11B on CFG-generated question–SQL pairs. We report the mean and standard deviation over five independent train/evaluation runs for each method to minimize randomness in our results. The two *insecure* baselines are the *exponential* and *generalized* models as noted above. All other methods are used to construct SECURELLM. Overall, our methods substantially outperform prior work in this setting while preserving generalization: although the generalized baseline has access to all silos simultaneously (and thus should nominally have an advantage over per-silo fine-tuning), our approach attains performance close to the generalized baseline.

position which is the critical factor that prevents the exponential training runs required to achieve a proper composition for every possible subset of the available silos given the users credentials.

We run two additional ablation studies to assess the potential impact of our CFG-based data generation.

First, we rephrase CFG-generated questions with GPT to remove the repetitive patterns typical of CFGs. Training is unchanged and follows the exact same procedure as in the main experiment (i.e., we still train on CFG questions), but at test time we evaluate on LLM-rephrased questions. Additional details are provided in Appendix I.1.

Second, we obfuscate every database column using a randomized but consistent mapping. This allows us to more directly measure whether composition at test time retrieves parameterized knowledge acquired during training, rather than relying on surface-form column cues. Additional details are provided in Appendix I.2.

Across both ablations, results remain consistent with those obtained on the standard CFG-generated dataset.

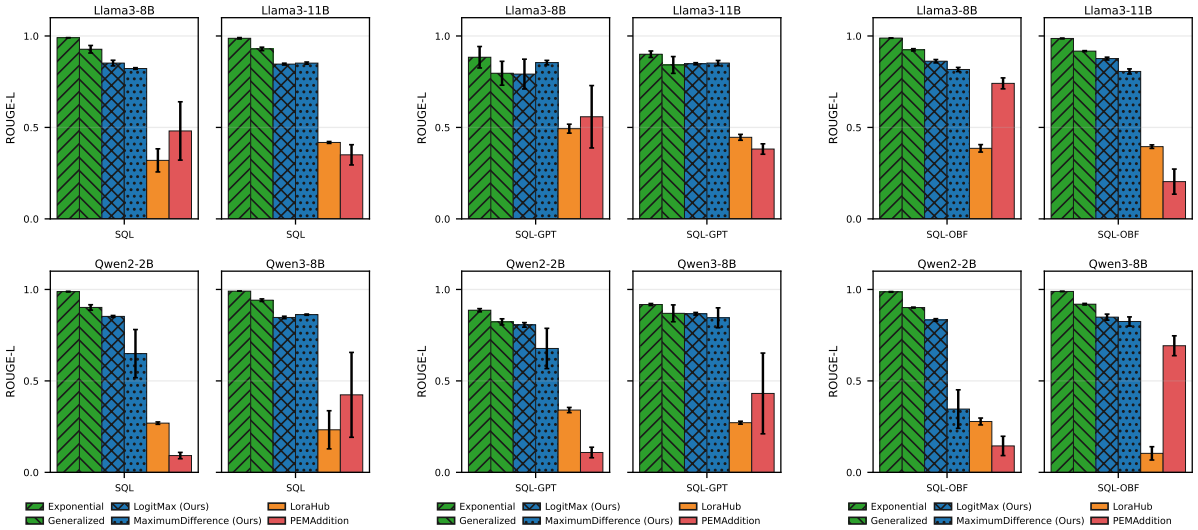
## 6 Results

The highest achievable performance occurs when the LLM is trained on the powerset of silos, which is the insecure baseline exponential model described above. Realistically, compositional techniques are bounded from above by a variant of the model that sees all of the silos at training time, but sees no combination of silos. Both of these are insecure, in that they have access to all of the data. Our results show that our composition technique is able to combine individual silo fine-tunings to achieve performance relatively close to that of the baseline generalized model.

We notice that the ROUGE-L metric for Llama 3 11B in Figure 3 (and four models on BLEU, F1, and TED metrics in Appendix H) show that our methods are consistently better than the tested previous method. We report the mean and standard deviation over five independent train/evaluation runs for each method to minimize randomness in our results.

In addition, we report the metrics for four models repeated five times in Figure 4a. The results are aggregated across the seven silos for compactness, and we include the full results in Appendix H. The results are also consistent across the two ablation experiments for the four models (Figures 4b and 4c) and the full results with addition details in Appendices I.1 and I.2.

We consistently find that across the four models (Llama 3 8B, Llama 3 11B, Qwen 2 2B, Qwen 3 8B) and four metrics (ROUGE-L, BLEU, F-1, TED) our compositional techniques consistently outperform other well-known methods under the framework of composition of silos at inference time.




(a) CFG-generated question-SQL. (b) GPT-rephrased ablation. (c) Obfuscated question-SQL ablation.

Figure 4: **ROUGE-L** for four models averaged across seven silos (Figure 4a). Rephrased ablation results (Figure 4b). Obfuscated ablation results (Figure 4c).

## 7 Conclusion

Ensuring the secure deployment of large language models is critical for their use in sensitive and societally important applications. In this work, we introduced SECURELLM, a novel compositional approach that integrates traditional access security with fine-tuning techniques to construct secure LLMs. Our framework effectively mitigates data leakage and prompt injection risks while retaining the model's ability to generalize as demonstrated on a challenging natural-language-to-SQL translation task.

By embedding security directly into the LLM's architecture, SECURELLM offers a scalable solution for sectors such as healthcare, finance, and public governance, where the protection of sensitive data is non-negotiable. Beyond its technical contributions, our work aligns with global initiatives aimed at safeguarding individual privacy and promoting ethical AI.

Looking forward, future work will extend SECURELLM to additional application domains, such as secure document question answering and multimodal data processing, and will explore closer collaboration with governmental and nonprofit organizations. One promising direction is the inverse problem: given a user query, automatically determining the minimal set of silos required to answer it. Such a capability could enable continuous monitoring of model interactions to ensure that responses remain confined to authorized information throughout a conversation. Another avenue for future research involves *negative silos*, which explicitly exclude particular topics or sources. By preventing certain information from influencing generation, negative silos could further reduce the risk of accidental disclosure. More broadly, the landscape of traditional access-control systems presents a rich set of challenges and opportunities for LLMs, and SECURELLM provides a foundation for systematically addressing them. By offering security guarantees, our work takes a key step toward enabling the safe deployment of LLMs in security-critical environments and bridging the gap between advanced AI research and real-world societal impact.

Finally, we emphasize that our experimental setting represents an extreme and deliberately challenging case, in which no prior linguistic or structural knowledge is useful for inferring silo contents, particularly when schema identifiers are obfuscated. Despite these constraints, our compositional approaches effectively leverage the model's generalization capabilities, achieving SQL edit distances comparable to those of a baseline model trained to generalize across silos. Put differently, composing silo-specific fine-tunings at inference time performs as well as jointly fine-tuning on all silos, which is the strongest outcome one can reasonably expect under strict security constraints. In practical deployments, using a more capable underlying LLM would further improve absolute execution accuracy while preserving these security guarantees.

## References


Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. 2016. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024. Jailbreaking leading safety-aligned llms with simple adaptive attacks. *arXiv preprint arXiv:2404.02151*.

Somnath Banerjee, Sayan Layek, Rima Hazra, and Animesh Mukherjee. 2024. How (un) ethical are instruction-centric responses of llms? unveiling the vulnerabilities of safety guardrails to harmful queries. *arXiv preprint arXiv:2402.15302*.

Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. 2019. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650.

Alexandra Chronopoulou, Matthew E Peters, Alexander Fraser, and Jesse Dodge. 2023. Adaptersoup: Weight averaging to improve generalization of pretrained language models. *arXiv preprint arXiv:2302.07027*.

Shumo Chu, Chenglong Wang, Konstantin Weitz, and Alvin Cheung. 2017. Cosette: An automated prover for sql. In *CIDR*.

C. J. Date, Hugh Darwen, and Nikos A. Lorentzos. 2003. *Temporal Data and the Relational Model A detailed investigation into the application of interval and relation theory to the problem of temporal database management*. Morgan Kaufmann Publishers.

Yilun Du, Shuang Li, and Igor Mordatch. 2020. Compositional visual generation and inference with energy based models. *arXiv preprint arXiv:2004.06030*.

Arka Dutta, Adel Khorramrouz, Sujan Dutta, and Ashiqur R KhudaBukhsh. 2024. Down the toxicity rabbit hole: A framework to bias audit large language models with key emphasis on racism, antisemitism, and misogyny. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence AI for Good*, pages 7242–50.

Sorami Hisamoto, Matt Post, and Kevin Duh. 2020. Membership inference attacks on sequence-to-sequence models: Is my data in your machine translation system? *Transactions of the Association for Computational Linguistics*, 8:49–63.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. 2022. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. 2023. Lorahub: Efficient cross-task generalization via dynamic lora composition. *arXiv preprint arXiv:2307.13269*.

Huseyin A Inan, Osman Ramadan, Lukas Wutschitz, Daniel Jones, Victor Rühle, James Withers, and Robert Sim. 2021. Training data leakage analysis in language models. *arXiv preprint arXiv:2101.05405*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Xuechen Li, Florian Tramer, Percy Liang, and Tatsunori Hashimoto. 2021. Large language models can be strong differentially private learners. *arXiv preprint arXiv:2110.05679*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Neal Mangaokar, Ashish Hooda, Jihye Choi, Shreyas Chandrashekaran, Kassem Fawaz, Somesh Jha, and Atul Prakash. 2024. Prp: Propagating universal perturbations to attack large language model guard-rails. *arXiv preprint arXiv:2402.15911*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. 2017. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*.

Milad Nasr, Reza Shokri, and Amir Houmansadr. 2019. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1671–1685.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Tianxiang Sun, Zhengfu He, Qin Zhu, Xipeng Qiu, and Xuanjing Huang. 2023. Multitask pre-training of modular prompt for Chinese few-shot learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11156–11172, Toronto, Canada. Association for Computational Linguistics.

Da Yu, Saurabh Naik, Arturs Backurs, Sivakanth Gopi, Huseyin A Inan, Gautam Kamath, Janardhan Kulkarni, Yin Tat Lee, Andre Manoel, Lukas Wutschitz, and 1 others. 2021. Differentially private fine-tuning of language models. *arXiv preprint arXiv:2110.06500*.

Santiago Zanella-Béguelin, Lukas Wutschitz, Shruti Tople, Victor Rühle, Andrew Paverd, Olga Ohrimenko, Boris Köpf, and Marc Brockschmidt. 2020. Analyzing information leakage of updates to natural language models. In *Proceedings of the 2020 ACM SIGSAC conference on computer and communications security*, pages 363–375.

Jinghan Zhang, Shiqi Chen, Junteng Liu, and Junxian He. 2023. Composing parameter-efficient modules with arithmetic operations. *arXiv preprint arXiv:2306.14870*.

Kaizhong Zhang, Jason TL Wang, and Dennis Shasha. 1996. On the editing distance between undirected acyclic graphs. *International Journal of Foundations of Computer Science*, 7(01):43–57.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. 2022. Provably confidential language modelling. *arXiv preprint arXiv:2205.01863*.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-sql with distilled test suites. *arXiv preprint arXiv:2010.02840*.

Guy Zyskind, Tobin South, and Alex Pentland. 2023. Don't forget private retrieval: distributed private similarity search for large language models. *arXiv preprint arXiv:2311.12955*.

# Appendix

## A Limitations

In this work, we disentangle and address a very specific slice of LLM safety, one that is often commingled with a larger story about safety. SecureLLM only concerns itself with security in the traditional sense: quantized access permissions to data. It relies on traditional security techniques to manage access permissions. There is a widely held belief that LLM security is a totally disjoint new field, but as we show, with the SecureLLM approach we can reduce many of those security problems back to traditional access permission issues. Many security problems of LLM are manageable through traditional means when one can assume that only vetted actors have access, just as they are with current document storage systems. The same systems that ensure patient privacy, financial privacy, and that manage secret information today can be used to manage collections of fine-tunings, and the same supervision methods can trace access to the LLM.

From this perspective SecureLLM solves data leakage and prompt injection attacks, in the same sense that traditional security solves data leakage: those without permissions cannot access this information, and those with permissions have full access with training and supervision. In the future one might imagine extensions that provide more fine-grained permissions. Organizations are already set up for this form of security, both for managing the user permissions and for the associated documents, making the deployment of SecureLLM straightforward. In settings without such policy and governance, stronger guarantees would require first introducing comparable access-control structure.

Explicitly out of scope are other notions of safety and security. For example, the LLM may still fabricate information, produce toxic or biased results, follow guidance that it should not, etc. The only mitigation that we offer is that only a user who has permissions to that data will be impacted directly; the user will still need appropriate training about the limitations and dangers of LLMs.

## B Impact Statement

SECURELLM enables *practical, credential-enforced confidentiality for LLMs* in regulated settings by replacing brittle "guardrail" filtering with an inference-time composition framework: separate silo-specific fine-tunings are only instantiated for the silos a user is authorized to access, yielding a non-contribution/non-interference guarantee that unauthorized silos cannot influence outputs while still supporting cross-silo queries for authorized users. Demonstrated on an intentionally hard NL2SQL benchmark with disjoint schemas (where successful answers require composing knowledge from multiple silos), this approach can broaden safe adoption of LLMs in healthcare, finance, and public administration by aligning model behavior with established access-control mechanisms and audit practices while also highlighting the need for strong governance, since improved secure analytics could be dual-use (e.g., enabling higher-throughput monitoring) if deployed without appropriate oversight.

## C Compositional Security Guarantee

Given $N$ data silos $\{S_1, S_2, \cdots, S_N\}$ and $N$ fine-tuned LLMs $\{M_1, M_2, \cdots, M_N\}$ where $M_i$ has been fine-tuned on the data silo $S_i$, and given a set of target indices $T \subseteq \{1, 2, \cdots, N\}$, the goal is to obtain a composed model $M_T := M_{T_1} \oplus \cdots \oplus M_{T_{|T|}}$ at inference time with no additional training such that $M_T$ is able to correctly answer any question about the information contained in the target silos $S_i, \forall i \in T$ and should fail to answer any question about information not contained in the target silos $S_j, \forall j \notin T$ as to not leak any information that the desired model $M_T$ is not intended to have. Additionally, the target model $M_T$ should be able to answer new *union questions* $q_{union,ij} \in S_{i \cup j}$ where $i \in T \wedge j \in T$ where the question relies on information contained in both $S_i$ and $S_j$. We note that the union questions $q_{union,ij}$ are not answerable by any individual data silos, thus none of the individual models $M_i$ are able to answer any union questions while a successfully composed model should be able to answer such questions without the need of any training.

It is critical that the composed model $M_T$ has no knowledge of any information silo that the user is not authorized to access, i.e. data silos $S_i, i \notin T$. Without this condition, a trivial solution is to train a single model $M_{All}$ on all data silos $\{1, \cdots, N\}$. But this approach is susceptible to leaking confidential information as the model would have knowledge of information contained in silos that users are not authorized to view. This approach is also problematic for scenarios that employ security through

contradiction, in that some silos may directly contradict information in another silo in order to protect sensitive information (SecureLLM could potentially solve this by applying weights to silos of higher confidentiality). We refer to $M_{All}$ as the Exponential Model that has seen every combination and such a model is used as an insecure upper bound to performance in our experiments.

We formalize SecureLLM's security claim in the standard access-control sense: an unauthorized user should not be able to obtain any information whose provenance depends on silos they are not permitted to access. SecureLLM achieves this by ensuring that, at inference time, the user interacts with a model instance whose parameters are a function only of (i) a public base model and (ii) the subset of silo-specific fine-tunings for which the user has verified credentials.

**Setup and notation.** Let $\mathcal{S} = \{S_1, \ldots, S_N\}$ be a collection of confidential data silos. Let $M_0$ denote a public base model with parameters $\theta_0$ (trained on non-confidential or otherwise globally permitted data). For each silo $S_i$, let $A_i$ denote a silo-specific adaptation (e.g., LoRA parameters, adapter weights, or a fine-tuned delta) trained *only* on $S_i$; we write its parameters as $\phi_i$. A user is associated with a credential set $T \subseteq \{1, \ldots, N\}$ obtained by an external, traditional access-control system. SecureLLM exposes an inference interface that, given $T$, instantiates a composed model

$$M_T \;=\; \mathrm{Compose}(M_0, \{A_i\}_{i \in T}),$$

and answers queries using only $M_T$. Critically, the mechanism never loads, executes, or queries $A_j$ for any $j \notin T$.

We model inference as a deterministic (or randomized) program. Let $q$ be the user query, $r$ be any internal randomness (e.g., sampling noise), and let the system output be

$$y \;=\; \mathcal{I}(q, r; \theta_0, \{\phi_i\}_{i \in T}),$$

where $\mathcal{I}$ is the inference procedure for the composed model (including tokenization, forward passes, decoding, etc.). In particular, $\mathcal{I}$ is *parameterized only* by $(\theta_0, \{\phi_i\}_{i \in T})$.

**Security definition (non-interference with respect to unauthorized silos).** Fix a user credential set $T$. We say SecureLLM is *compositional-access secure* if, for any two worlds that differ only in the contents of unauthorized silos, the distribution of outputs available to the user is identical. Formally, let $\mathcal{S}$ and $\mathcal{S}'$ be two silo collections such that

$$S_i = S'_i \quad \forall i \in T,$$
and
$$S_j \neq S'_j \quad \text{may differ for } j \notin T.$$

Let $\phi_i$ and $\phi'_i$ be the resulting learned adaptations (i.e., the outputs of the training algorithm) from $\mathcal{S}$ and $\mathcal{S}'$, respectively. Then compositional-access security requires that for all queries $q$ and measurable output events $E$,

$$\begin{aligned}&\Pr[\mathcal{I}(q, r; \theta_0, \{\phi_i\}_{i \in T}) \in E] \\ &= \Pr\big[\mathcal{I}(q, r; \theta_0, \{\phi'_i\}_{i \in T}) \in E\big],\end{aligned}$$

where the probability is taken over $r$ (and over any randomness used in decoding).

**Theorem (SecureLLM prevents leakage from unauthorized silos by construction).** Assume: (A1) each adaptation $A_i$ is trained only from $S_i$ (and possibly public data already in $M_0$), and (A2) the inference system instantiates $M_T$ using only $\theta_0$ and $\{\phi_i\}_{i \in T}$ and has no side channels (e.g., it does not query external databases or log-probe other models). Then SecureLLM satisfies compositional-access security.

**Proof.** Fix $T$ and consider two silo collections $\mathcal{S}, \mathcal{S}'$ that agree on all $S_i$ for $i \in T$ but may differ for $j \notin T$. By assumption (A1), the training algorithm that produces $\phi_i$ depends only on $S_i$ (and public data), hence $\phi_i = \phi'_i$ for all $i \in T$. By assumption (A2), the inference procedure $\mathcal{I}$ for user credentials $T$ depends only on $(\theta_0, \{\phi_i\}_{i \in T})$. Therefore the internal computation graph, logits, and decoded output distribution are identical under $\mathcal{S}$ and $\mathcal{S}'$, since the parameterization is identical. Concretely, for every $q$ and randomness $r$,

$$\mathcal{I}(q, r; \theta_0, \{\phi_i\}_{i \in T}) = \mathcal{I}(q, r; \theta_0, \{\phi'_i\}_{i \in T}).$$

Taking probabilities over $r$ yields equality of output distributions for all events $E$, which is exactly the security definition.

**Interpretation.** This guarantee is stronger than "guardrails" or post hoc filters: it is not a statement about detecting attacks, but about *information flow*. For a user with credentials $T$, the system never instantiates parameters trained on $\{S_j\}_{j \notin T}$, so there is no computational path by which those silos can influence the output. In this precise sense,

SecureLLM reduces the confidentiality problem to the correctness of the access-control mechanism that selects $T$, just as conventional credential-based security reduces data disclosure to which resources are mounted and queried.

**Scope and limits of the guarantee.** The theorem guarantees non-leakage *from unauthorized silos* under the stated assumptions. It does not claim that $M_0$ (or any authorized adaptation) cannot itself leak information it already contains, nor does it address other safety dimensions such as toxicity, hallucination, or instruction-following failures. SecureLLM's claim is that confidential information contained exclusively in silos outside $T$ cannot be revealed, because the model instance used for inference is constructed without any parameters learned from those silos.

# D Further Discussion on SecureLLM Framework

## D.1 Security as a relaxation

What distinguishes security-oriented composition from other fine-tuning combination scenarios is that it admits a simple and analytically tractable idealization. Consider an abstract setting in which each information silo has a disjoint alphabet, the contents of each silo are represented by a state machine, and the valid outputs of the system are strings accepted by the union of these per-silo machines. In this formulation, security is straightforward: an output is correct if and only if it is generated entirely from the state machines corresponding to the silos the user is authorized to access. Crucially, the optimal behavior in this setting requires no blending or entanglement of silo-specific information. At each step, the output depends on exactly one silo, and unauthorized silos are simply excluded from the computation. The compositional methods we introduce are optimal in this idealized regime, which closely mirrors the structure of SQL generation.

In natural-language-to-SQL tasks, the model's primary responsibility is to select table names, column names, and predicates that belong to individual schemas, while adhering to a global SQL grammar that is independent of any specific silo. Real-world security scenarios can be viewed as *relaxations* of this idealization, in the sense that they weaken one or more simplifying assumptions of the ideal model (such as strict separability of symbol usage, independence of per-silo contributions, or the absence of cross-silo reasoning) while still requiring that unauthorized silos exert no influence on the output. For instance, cross-silo question answering remains faithful to the idealized model when portions of the response are derived independently from authorized silos, but becomes a relaxation when correct answers require jointly reasoning over multiple silos and integrating their information. As we demonstrate empirically, existing fine-tuning composition methods perform poorly even in the idealized case, and degrade further under these relaxations. By formalizing security in this way, we obtain a principled foundation for designing and evaluating new composition operators that are explicitly aligned with access-control guarantees. Through this work, we introduce two new composition operators, Logit Composition and Maximum Difference, and evaluate several alternative strategies that perform worse than existing methods adapted to the compositional security task.

# E Data Generation

Our goal is to automatically create a challenging dataset for compositions of silos. While there are countless other NL2SQL datasets, none specifically focus on SQL queries for disjoint and unrelated databases silos. Secure-NL2SQL contains three silos of disjoint schemas pertaining to different subjects, as well as the superset of unions between those three silos for a total of seven permutations ($S_1, S_2, S_3, S_{1\cup2}, S_{1\cup3}, S_{2\cup3}, S_{1\cup2\cup3}$). The dataset contains automatically generated questions and corresponding SQL queries across each silo.

Each dataset is normalized according to 6NF (Date et al., 2003) and table and column name are chosen to be non-trivial such that a general model would not be able to guess the table name of column name based on context given by the question. The scope of generated SQL statements is limited. All statements generated from our CFG, are SELECT statements that contain only the SQL keywords FROM, NATURAL JOIN, and WHERE. The majority of the complexity is in the WHERE clause which requires specialized knowledge about the schema along with language comprehension to properly generate based on the input question. The task for the LLM is to generate the WHERE clause of an SQL statement which answers the input question.

There are countless other natural-language-to-SQL datasets out there, which we do not aim to

replace. As such, we focus specifically on within and between silo questions. We also cover only a portion of SQL. We do not aim to exhaustively test how well models understand SQL, we aim to understand how well models generalize their knowledge from questions about individual silos to questions that span silos. As such, we consider only a subset of SQL which is otherwise an imposingly complex language.

We automatically generate SQL databases, one per silo, with 2-3 tables per database, that share columns which can be joined together both within and between databases. However, the databases are otherwise disjoint and contain different topics. For each database we generate natural language questions along their equivalent SQL. Then, we generate questions and SQL pairs that span pairs and triples of databases. Two methods are used to generate these pairs: a CFG (see Figure 2) and GPT-4 (see Figure 5). The CFG generates both the SQL and the question in parallel. We do this at large scale, with 100,000 pairs per silo or combination of silos. To ensure that our results scale to more realistic queries we also generate 300 pairs per silo or combination of silos.

We limit the scope of generated SQL statements. All statements generated from our CFG are SELECT statements that only contain the SQL keywords FROM, NATURAL JOIN, and WHERE. The majority of the complexity is in the WHERE clause which requires specialized knowledge about the schema along with language comprehension to properly generate based on the input question. The task for the LLM is to generate the WHERE clause of an SQL statement that answers the input question.

We utilize a normalization, which is a variation of 6NF(Date et al., 2003), for which we provide ablations in the results section. Our normalization ensures table and columns are not "guessable" by a non-fine-tuned model based on context provided in the prompt. In general, this transformation could help all SQL LLMs, and is bidirectional. As described in the results section, our composition methods are far superior irrespective of this normalization, but we believe it is a valuable observation that is likely to lead to many more LLM-specific normalizations as they become serious consumers of SQL.

For each SQL schema, we randomly generate a concrete database instance. Each evaluation sample consists of a natural-language question, its corresponding ground-truth SQL query, and the database instance on which that query is defined. For every sample, the model's predicted SQL is assembled into an executable statement and run against the same database instance. A prediction is considered correct if and only if the resulting output exactly matches the result of executing the ground-truth SQL query. To avoid trivial false positives, we ensure that every query in both the training and validation sets returns at least one record. Because natural-language questions often map to long and structurally complex SQL statements, this evaluation criterion is stringent: even a single incorrect token typically leads to an incorrect or empty result and is therefore scored as incorrect.

Determining whether a generated query is correct is a non-trivial problem as the problem of determining if two SQL statements are semantically equal is undecidable (Chu et al., 2017). We computed accuracy by empirically executing the generated query on a database and comparing the results to the ground-truth query. Using a single database increased the susceptibility of false positives, thus we tried employing a method inspired by *test suite accuracy* (Zhong et al., 2020) by instantiating several databases with diversified data, which reduced the probability of false positives. However, there are issues with using accuracy as a metric as a single incorrect token would render the entire generated query as incorrect.

Thus, we employ a score that parses the generated query conditions into a tree, then calculates the tree-edit distance (Zhang et al., 1996) between the ground-truth query and the generated query. This is the number of edit operations required to transition between the two, which are normalized by the number of nodes in the ground tree. These are averaged across all queries for a silo or collection of silos. This edit distance score provides a far more fine-grained view into the performance of models, fine-tunings, and compositions of fine-tunings.

## F Metrics

For the base metric we opted to use ROUGE-L (Lin, 2004) as it is a standard metric for comparing a generated response to a ground response when each individual token matters. However, for typical NL2SQL datasets, Exact Match (EM) accuracy is recorded, we found this metric to be insufficient to show differences in method performance. Instead, we calculate the tree-edit distance (Zhang et al., 1996) between the ground-truth query and the gen-

erated query. By computing the number of edit operations required to transition between the two, we can show how close syntactically a given generated query is to the correct query, whereas using only EM is a binary representation of correctness. This metric is sufficient for our analysis because the edit operations directly reflect deviations in table selection, column references, predicates, and logical structure, all of which are precisely the failure modes of interest in evaluating compositional generalization across disjoint schemas.

The metrics are averaged across all queries for a silo or collection of silos. This edit distance score provides a far more fine-grained view into the performance of models, fine-tunings, and compositions of fine-tunings. For our full results in Appendix H, we include four metrics: ROUGE-L, normalized tree-edit distance, as well as BLEU, and an F1 score (which is calculated using a trivial bag-of-words).

## G Additional Detail of Composing Fine-tunings

For the base fine-tunings, we fine-tune all models with one epoch until saturation (achieving near 100% accuracy on the CFG validation set) using a frozen base model with a trainable LoRA fine-tuning (Hu et al., 2021) using LoRA parameters $r = 8, \alpha = 32$ and a dropout (Srivastava et al., 2014) of $0.1$, an Adam optimizer (Kingma and Ba, 2014) with a learning rate of $0.0002$, a batch size of 32, and a weight decay of $0.002$.

**LoraHub** The LoraHub method for composition introduced by (Huang et al., 2023) is a two-step process involving element-wise summation of LoRA fine-tunings (*COMPOSE*), and then learning weight optimizations via gradient-free methods to apply to each fine-tuning (*ADAPT*). For this study, we do not implement the *ADAPT* stage because weights for every possible combination would need to be learned, and we could not say that this process is completed at inference time.

In our evaluation, we do not perform *ADAPT* for two reasons. It does not modify the performance of the system because our evaluation set is balanced, i.e., ADAPT would make the model better at one silo rather than another, but not change the composition of silos. Additionally, ADAPT would be expensive in practice and would need to be run per user unless one wanted to save an exponential number of ADAPT combinations for the entire powerset of silos.

The *COMPOSE* method consists of adding the encoder/decoder components of each Lora to act as one encoder/decoder pair. Given $m_i = A_i B_i$, the combined fine-tuning, $\hat{m}$ and $\hat{m}(x)$ on input $x$ is obtained by

$$\hat{m} = (B_1 + \cdots + B_n)(A_1 + \cdots + A_n), \quad (1)$$

$$\hat{m}(x) = (B_1 + \cdots + B_n)(A_1 + \cdots + A_n)(x). \quad (2)$$

We observe that LoraHub performs poorly on the secure composition task, as we show later in Section 6. The authors warn that combining too many fine-tunings can lead to poor performance, however this cannot be the source of poor performance as we compose no more than three LoRAs.

**Average of Adapter Weights** Computing the simple average of each Lora fine-tuning response, as suggested by (Chronopoulou et al., 2023), $\sum_{i \to n} \frac{L_i}{n}$, produced compositions that were 50% less effective than PEM Addition in initial informal tests, thus the method is excluded from our formal experiments.

**Variations of LogSumExp of Adapter Weights** (Du et al., 2020) proposes a disjunctive composition process based on Energy Based Modeling, $-logsumexp(-E_1(x), -E_2(x), \cdots)$. Every variation tried performed significantly worse than PEM Addition, and upon closer inspection, this process substantially distorts encoder and decoder embeddings.

**Adapter Concatenation** The (Mangrulkar et al., 2022) library implements weight concatenation, however we found that concatenating LoRA encoder/decoder fine-tunings performed significantly worse than PEM Addition.

**Strongest Output Embedding Response** Only using the LoRA Layer, $L_i$, that generated the strongest output embedding response at a given attention layer produced compositions that were 50% less effective than PEM Addition in initial informal tests.

**Element-wise Maximum** Using the standard max function in pytorch,

$$max\left([L_1(x), \cdots, L_n(x)], dim = 0\right),$$

produced compositions that were 50% less effective than PEM Addition in initial informal tests.

**Sum of Deltas** Given a base model output $h_0(x)$ and a set of active LoRA adapters $T$, each adapter

$i \in T$ defines a low-rank update

$$\Delta_i(x) \;=\; s_i\, B_i(A_i(D_i(x)))\,, \tag{3}$$

where $D_i$ is dropout, $A_i$ and $B_i$ are the LoRA $A/B$ projections, and $s_i$ is the LoRA scaling. We compose by adding all active deltas to the base output:

$$h(x) \;=\; h_0(x) \;+\; \sum_{i \in T} \Delta_i(x). \tag{4}$$

**Average of Deltas** Using the same $\Delta_i(x)$ definition, we instead average the active deltas:

$$h(x) \;=\; h_0(x) \;+\; \frac{1}{|T|} \sum_{i \in T} \Delta_i(x). \tag{5}$$

**Our Maximum Difference Method**

The intuition behind this method is to select the embeddings from each fine-tuning with the strongest response (either positive or negative) at each attention layer. In order to accomplish this, each LoRA fine-tuning is evaluated separately on input $x$. Then a mask of zeros with the same dimension as the output is created, $h_{max}$, to aggregate LoRA responses. For each LoRA fine-tuning response $L_i$, an element-wise comparison is made, and if the absolute values of the fine-tuning response are greater than the aggregated response, the signed response from that fine-tuning replaces the element in the aggregated response. This process is defined by Algorithm 1.

**Algorithm 1** Element-wise Maximum Difference

$h_{\text{max}} \leftarrow \text{zeros\_like}(\textit{Attention}(x))$
**for** $i = 0$ **to** $n - 1$ **do**
  $h_{\text{mask}} \leftarrow \text{zeros\_like}(h_{\text{max}})$
  $L_{\text{mask}} \leftarrow \text{zeros\_like}(h_{\text{max}})$
  **for all** elements $e$ in $h_{\text{max}}$ **do**
    **if** $|h_{\text{max}}[e]| > |L_i[e]|$ **then**
      $h_{\text{mask}}[e] \leftarrow 1$
    **else**
      $L_{\text{mask}}[e] \leftarrow 1$
    **end if**
  **end for**
  $h_{\text{max}} \leftarrow h_{\text{max}} \cdot h_{\text{mask}} + L_i \cdot L_{\text{mask}}$
**end for**
$h = \textit{Attention}(x) + h_{\text{max}}$

**Logit Composition** Given Base model $M_0$ and fine-tunings $\{A_i\}_{i \in T}$, we run each independently on $x$ and compose by taking the element-wise maximum of the *last-token* logits (Equation (8))

$$M_i = \text{Compose}(M_0, \{A_i\})(x) \tag{6}$$

$$\ell_i(x)_{-1} \;=\; \text{logits}(M_i)_{-1} \in \mathbb{R}^V. \tag{7}$$

$$\ell(x) \;=\; \max_{i \in T}\; \ell_i(x)_{-1}, \tag{8}$$

Given all our experiments, this technique on average consistently performed the best. The process is defined by Algorithm 2.

**Algorithm 2** LogitMax Composition (per-token decoding)

**Require:** Base model $M_0$, fine-tunings $\{A_i\}_{i \in T}$, fixed prompt $x_{1:m}$, generation length $L$
**for** $t = 1$ **to** $L$ **do**
  **for all** $i \in T$ **do**
    $M_i \leftarrow (\text{Compose}(M_0, \{A_i\})(x_{1:m+t-1}))$
    $\ell_i \leftarrow \text{logits}M_i$ ▷ $\ell_i \in \mathbb{R}^V$
  **end for**
  $\ell \leftarrow \max_{i \in T} \ell_i$ ▷ element-wise max over vocab
  Choose/sample $x_{m+t} \sim \text{softmax}(\ell)$
**end for**

Q: What is the average age of instructors who are aged 72 or older or teach art at level 9? Answer:

```
SELECT AVG(instructors.teacher_age)
FROM instructors INNER JOIN classes
   ON instructors.teacher_id = classes.teacher_id
WHERE instructors.teacher_age >= 72
   OR classes.class_subject = 'art' AND
         classes.level = 9
```

(a) Sample from Silo 1 ($S_1$) that was rephrased with GPT

Q: What's the minimum height of all appliances in the inventory that are currently unavailable in stores located in NY, CA, or MA and with a rating higher than or equal to 2 stars. Answer:

```
SELECT MIN(inventory.sloth)
FROM inventory INNER JOIN store ON
   store.bear = inventory.bear
WHERE inventory.pony = 0
   AND (store.alpaca = 'NY'
    OR store.alpaca = 'CA'
    OR store.alpaca = 'MA')
   AND store.raccoon >= 2
```

(b) Sample from Silo 2 ($S_2$) with obfuscated column names

Figure 5: Examples of input/output pairs of a question paired with the target SQL query which (a) are from the GPT rephrased silos and (b) use column names obfuscated by an arbitrary but consistent mapping.

## H Full Main Results

Repetitions of the experiment reported in Figure 3 across four base LLMs: Llama 3 8B (Figure 6a), Llama 3 11B (Figure 6b), Qwen 2 2B (Figure 7b), and Qwen 3 8B (Figure 7a). For each model and each we train exactly 5 LoRA finetunings for these reported results. The *exponential baseline*, trained on all silos ($S_1, S_2, S_3, S_{1\cup2}, S_{1\cup3}, S_{2\cup3}, S_{1\cup2\cup3}$). The *generalized baseline*, trained jointly on all individual silos ($S_1, S_2, S_3$). The three silo-specific fine-tunings $\{A_1, A_2, A_3\}$ where each $A_i$ is trained *only* on $S_i$. The three silo-specific fine-tunings are composed together using the various composition methods discussed and evaluated on each silo's test-set and the metrics are reported.

All reported results are from a held-out test set. We repeat the process of training and evaluating each model/method (and all other experiments in this paper) five times to achieve the standard deviation and minimize randomness in our results.

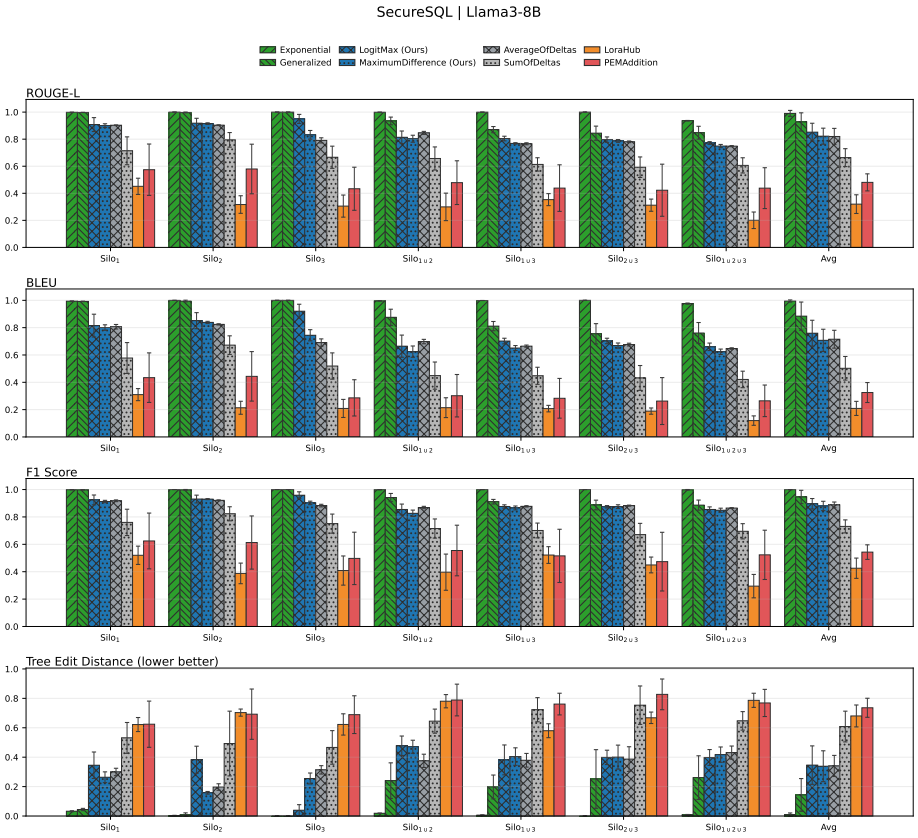
## I Ablation Experiments

### I.1 SecureSQL Rephrased Results

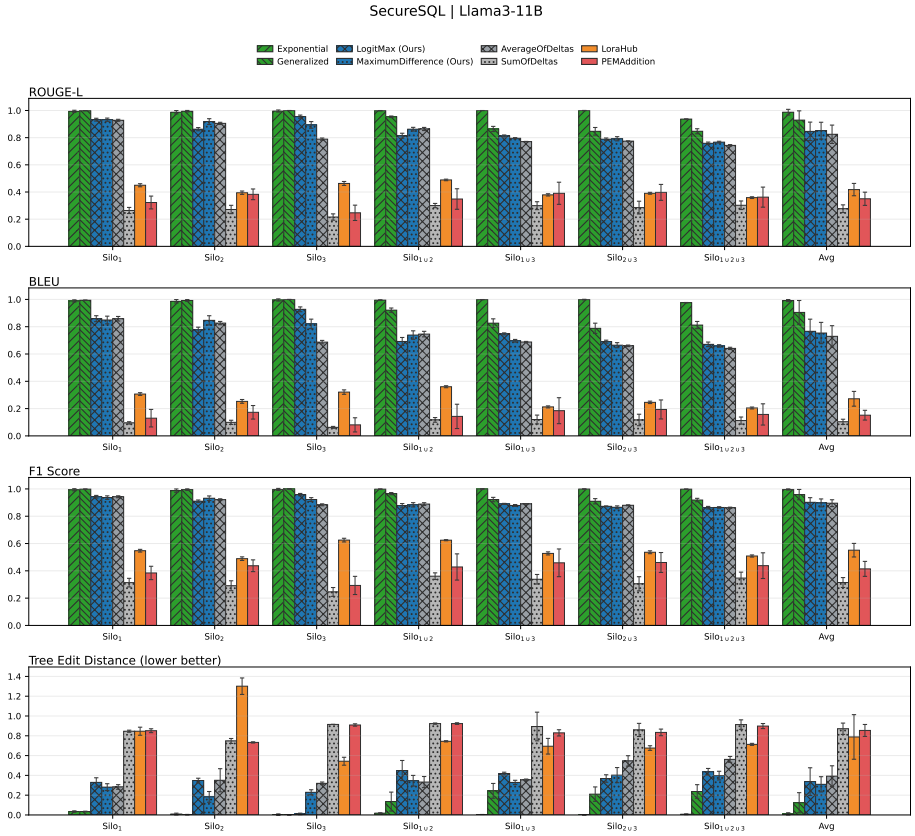
One might wonder if our gains are merely an artifact of the CFG-based data generation. We therefore replicate the experiment on a held-out test set in which questions are paraphrased by GPT (paraphrases are used *only* for testing, not for training). The averaged results are in Figure 4b. The full results are in Figures 8a, 8b, 9a and 9b support the same conclusions: prior composition methods (e.g., LoraHub and PEM Addition) substantially underperform our approach. This evaluation is intentionally challenging because training follows the CFG distribution, while evaluation is performed on out-of-distribution paraphrases.

### I.2 SecureSQL Obfuscation Results

To guard against a trivial solution in which the model guesses table/column names from semantic cues, we also introduce a column-name obfuscated variant. We are interested in the ability of models to retain compositional reasoning, rather than circumventing the task. In any case, in real world conditions column names are often rather complex. In this ablation experiment, each column is assigned an arbitrary but stable name (e.g., an animal) using a fixed isomorphic mapping. The averaged results are in Figure 4c. The full results are in Figures 10a, 10b, 11a and 11b show that our method loses little performance, meaning that it encourages compositional reasoning.

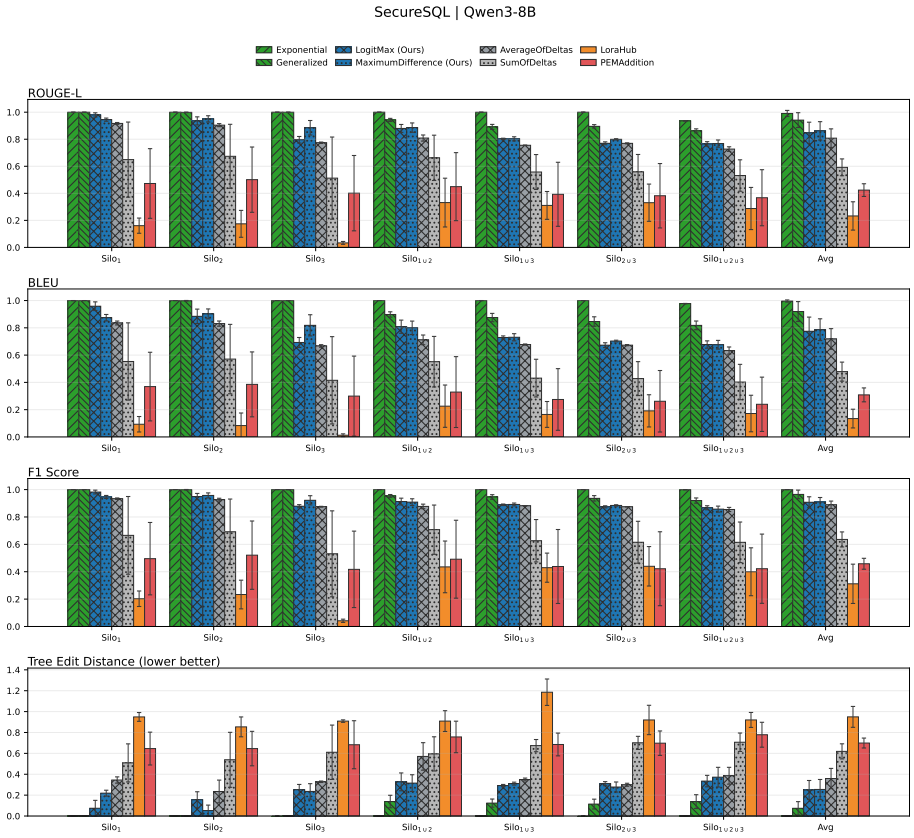



(a) Main Results Appendix H using Llama-3 8B model (model #1/4).

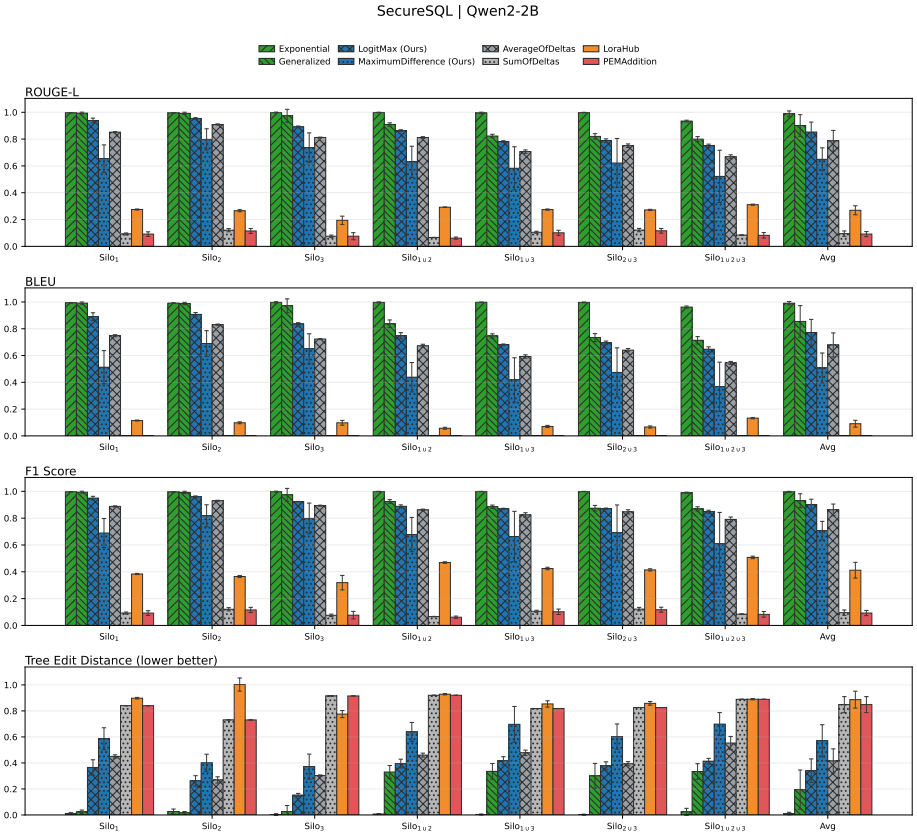



(b) Main Results Appendix H using Llama-3 11B model (model #2/4).

Figure 6: Main Results Appendix H (models #1–2 out of 4).

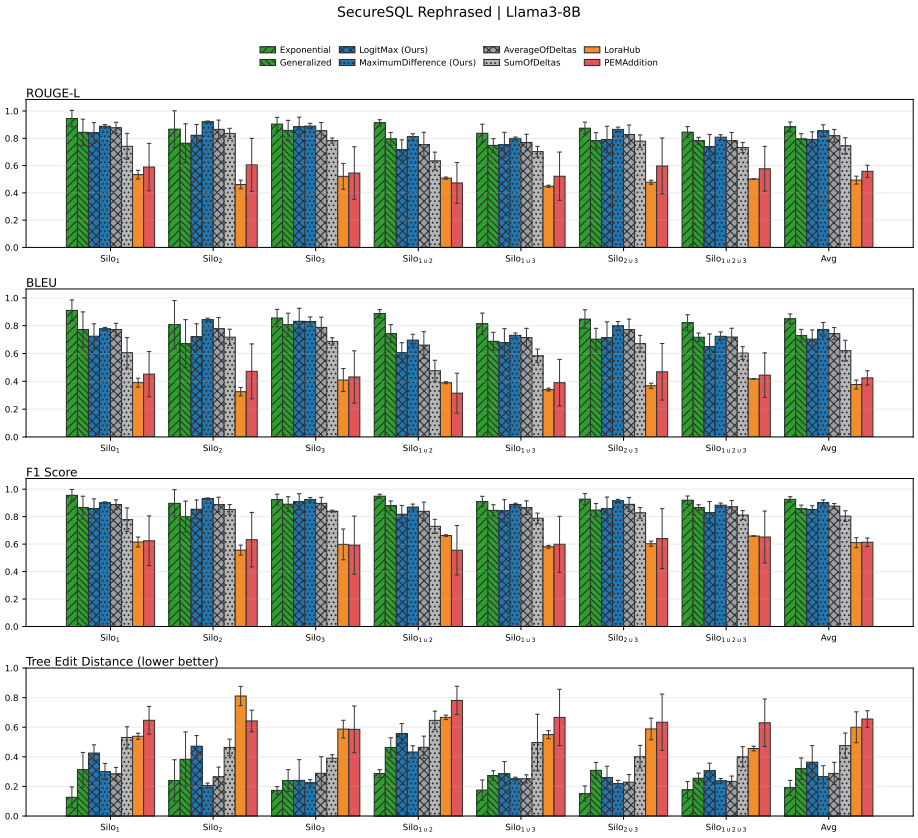



(a) Main Results Appendix H using Qwen3 8B model (model #3/4).

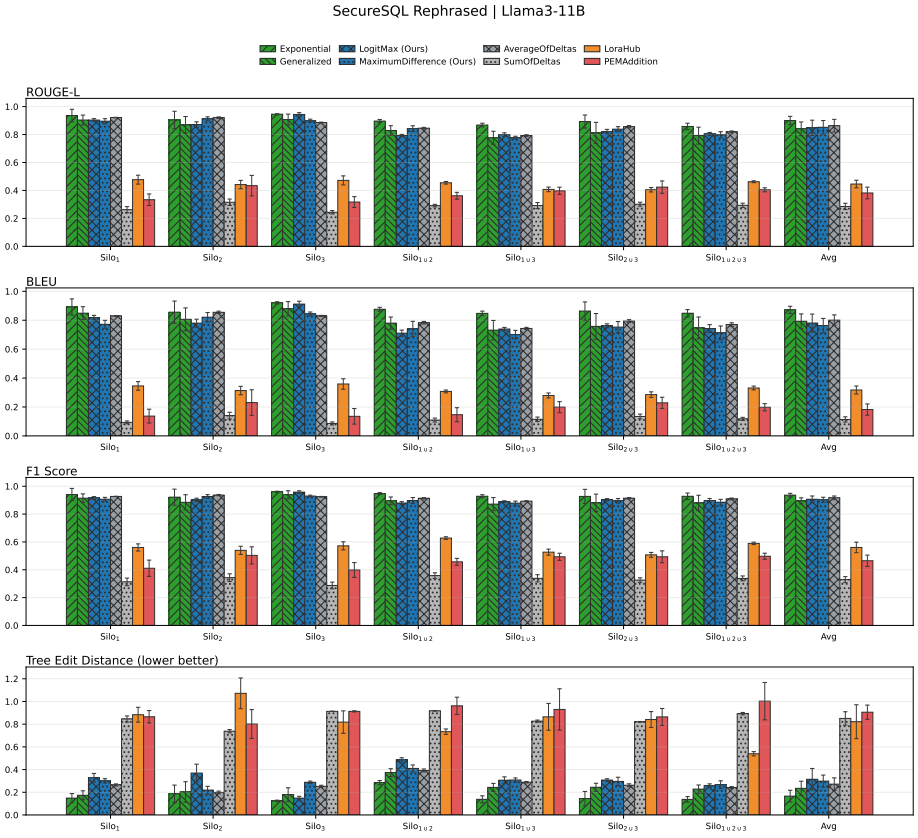



(b) Main Results Appendix H using Qwen2 2B model (model #4/4).

Figure 7: Main Results Appendix H (models #3–4 out of 4).

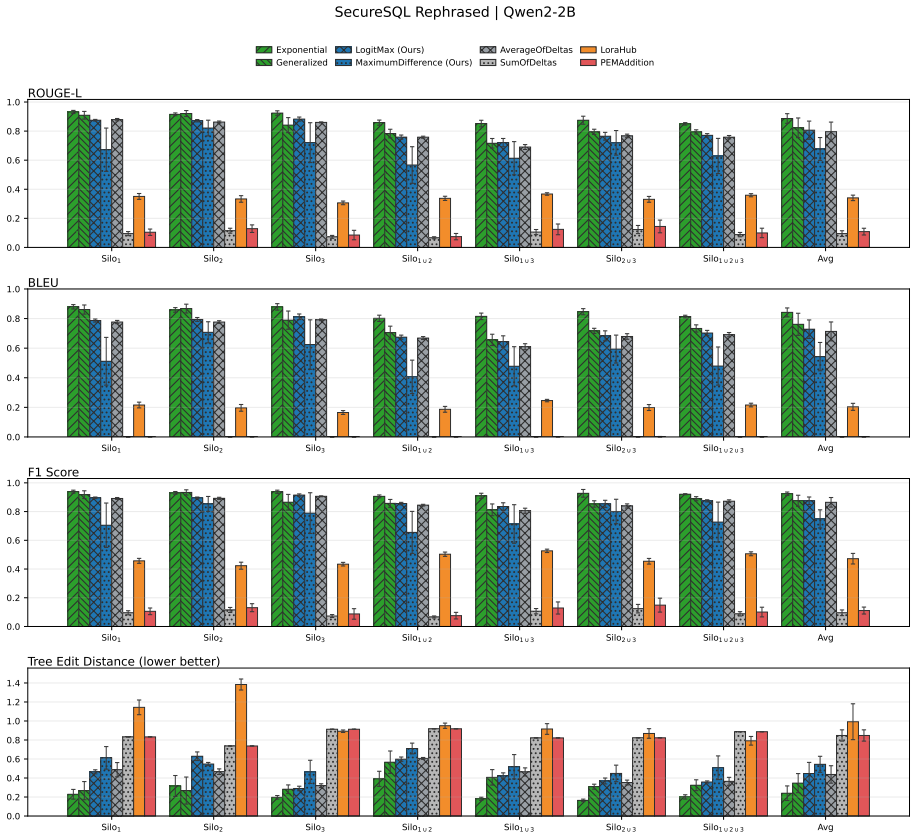



(a) Ablation (#1/2): Rephrased Results (Appendix I.1) using Llama-3 8B model (model #1/4).

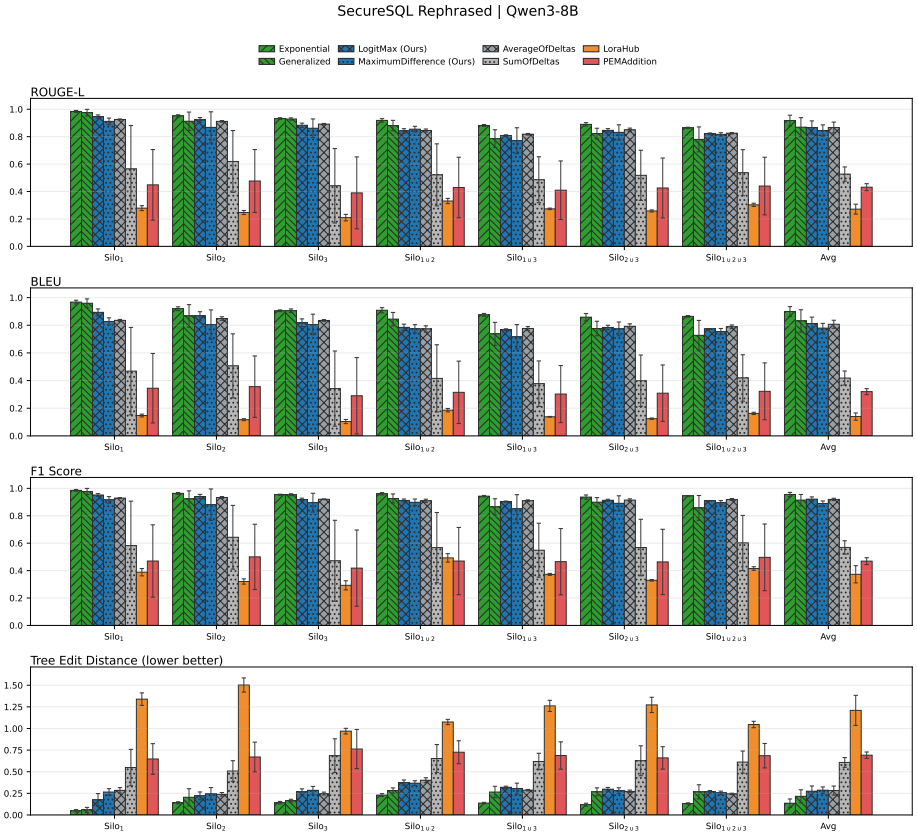



(b) Ablation (#1/2): Rephrased Results (Appendix I.1) using Llama-3 11B model (model #2/4).

Figure 8: Ablation (#1/2): Rephrased Results (Appendix I.1) (models #1–2 out of 4).

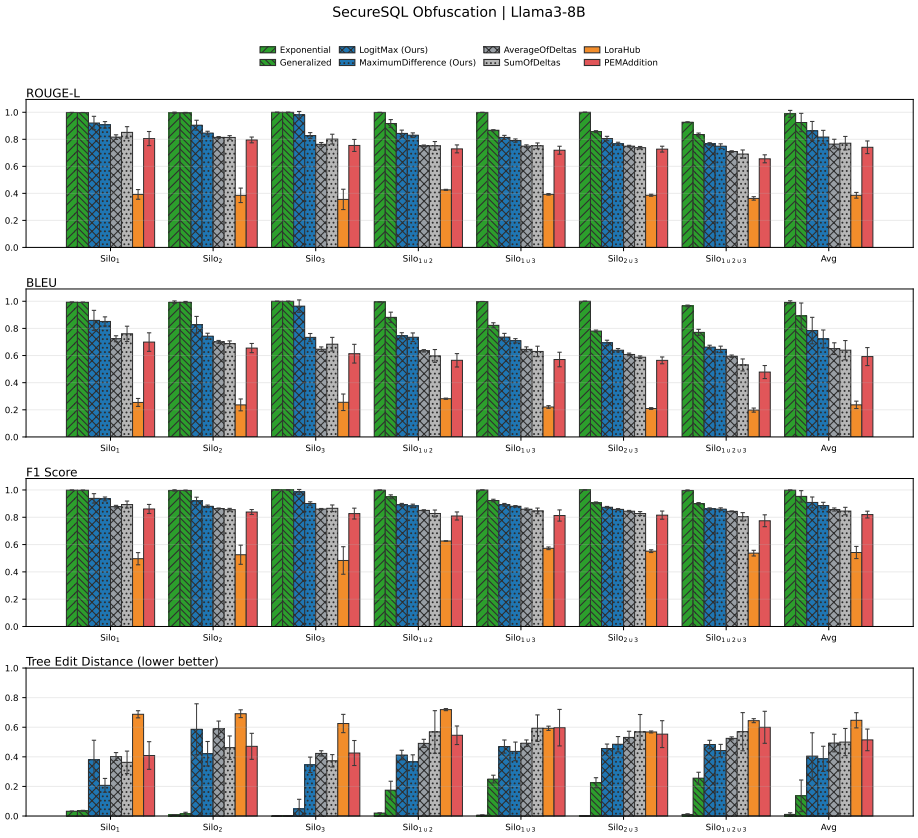



(a) Ablation (#1/2): Rephrased Results (Appendix I.1) using Qwen2 2B model (model #3/4).

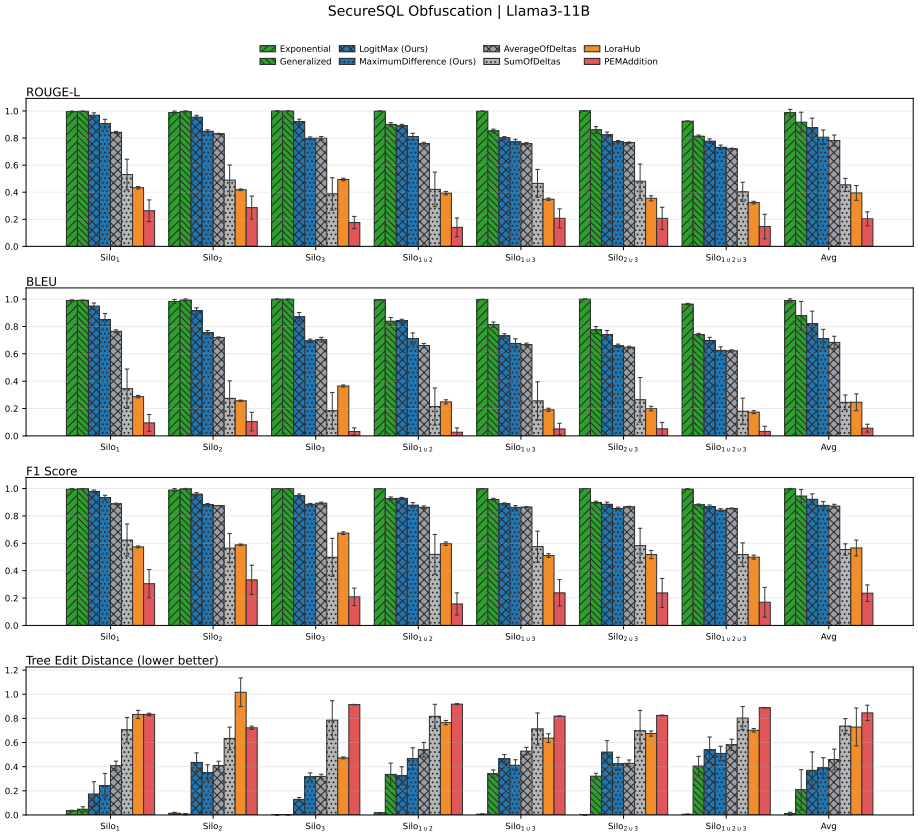



(b) Ablation (#1/2): Rephrased Results (Appendix I.1) using Qwen3 8B model (model #4/4).

Figure 9: Ablation (#1/2): Rephrased Results (Appendix I.1) (models #3–4 out of 4).

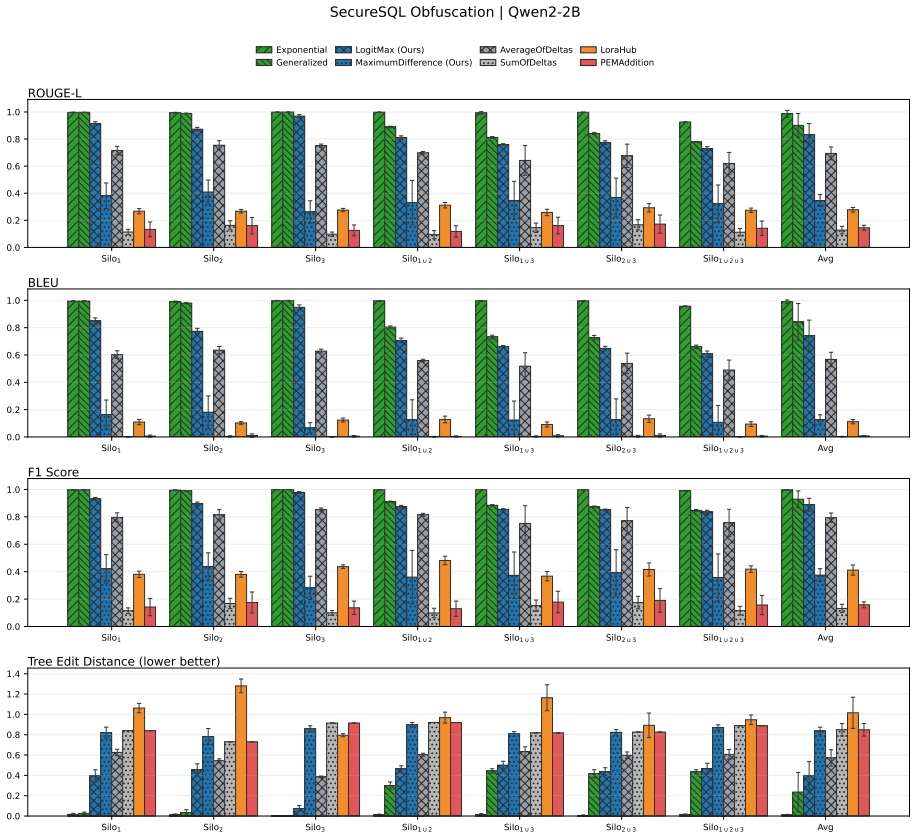



(a) Ablation (#2/2): Obfuscation Results (Appendix I.2) using Llama-3 8B model (model #1/4).

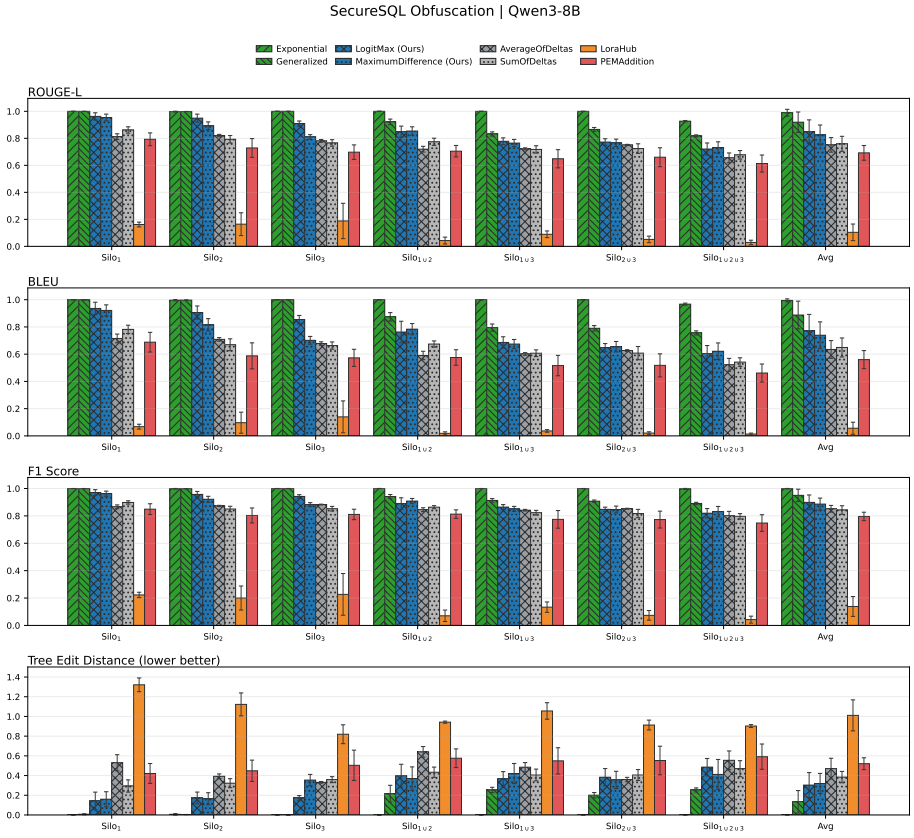



(b) Ablation (#2/2): Obfuscation Results (Appendix I.2) using Llama-3 11B (model #2/4).

Figure 10: Ablation (#2/2): Obfuscation Results (Appendix I.2) (models #1–2 out of 4).




(a) Ablation (#2/2): Obfuscation Results (Appendix I.2) using Qwen2 2B model (model #3/4).




(b) Ablation (#2/2): Obfuscation Results (Appendix I.2) using Qwen3 8B (model #4/4).

Figure 11: Ablation (#2/2): Obfuscation Results (Appendix I.2) (models #3–4 out of 4).

## J Other Datasets

A common theme in prior work is to compare compositional techniques on well-known public benchmarks. For completeness, we report results on a standard public dataset (MMLU) in Figures 12–15. We emphasize, however, that all publicly available datasets we are aware of exhibit the limitations discussed in our data-generation appendix (Appendix E). Firstly, because these datasets are public, a base LLM may already have been pretrained on their training data; in that case, our security guarantees no longer apply, and improvements from composition may not reflect newly acquired silo-specific knowledge (the base model could have already encode this information in its weights). In particular, apparent gains could be attributed to fine-tuning that *steers* or *unlocks* capabilities already present in the base model. By contrast, in SECURESQL the CFGs are hand-crafted and private, so correct generation requires access to the composed silo-specific knowledge beyond mere educated guessing. Moreover, our rephrasing ablation (Appendix I.1) further mitigates guessability by randomly shuffling the underlying mapping during dataset construction, making success without the appropriate composition exceedingly unlikely. Also, standard public benchmarks typically lack *union-silo* instances that require jointly leveraging information from multiple silos, which is a central objective of our setting.

For MMLU, we simply define our four silos by grouping some related MMLU subject together:

- $S_1$ (Math): `abstract_algebra`, `formal_logic`, `college_mathematics`, `elementary_mathematics`.
- $S_2$ (Physics): `conceptual_physics`, `HS_physics`, `college_physics`, `astronomy`.
- $S_3$ (Biology): `anatomy`, `HS_biology`, `college_biology`, `medical_genetics`, `virology`.
- $S_4$ (CS): `HS_computer_science`, `college_computer_science`, `machine_learning`, `computer_security`.

We avoid any free-response subjects and thus can simply report the accuracy for the questions tested. The 'Valid' metric simply measures how many LLM answers were a valid choice among the listed multiple choices. This is simply a sanity check and should be 1.0 (representing 100%) for any model capable of tackling multiple choice questions. Results for the four models are reported in Figures 12–15

## K Correlation with Model Size

A natural question to ask given the performance of the compositions on each silo across different models is whether performance is correlated with model size. We show the correlation between the performance and the models in each of the three experiments (Figure 16). We also provide a comparison with the number of parameters (in Billions) and simply considering each model separately.

We notice that for certain methods such as the two baselines and Logit Max, the performance is mostly constant across the model. While for the other methods, performance is somewhat positively correlated with the model size. No method is negatively correlated ($\leq -0.01$) with the model size.

## L Additional Discussion and Reproducibility Information

### L.1 Ethics

SecureLLM could pose some ethical issues. At the moment, surveillance is limited by the need to process a deluge of data. This results in mostly processing metadata. Enabling LLMs to work in secure environments could contribute to large-scale monitoring, detection, and tracking. Novel uses of LLMs in secure environments can advance numerous defense applications. As with many other dual-use technologies, we hope that this work will be used for positive ends.

## M Reproducibility

We use NVIDIA Titan RTX 24GB VRAM GPUs for all our experiments. For all of our PEFT parameters, less than one GPU-hour per PEFT was required for training. Approximately one GPU-hour was required for composition growing with respect to the number of compositions. For each run, approximately 24-48 GB of VRAM is needed depending on the base model as we use half precision for all training and inference. Each PEFT can be trained. We estimate a total of 10 GPU-hours is required to replicate results for training, and 20 GPU-hours is required to perform the same

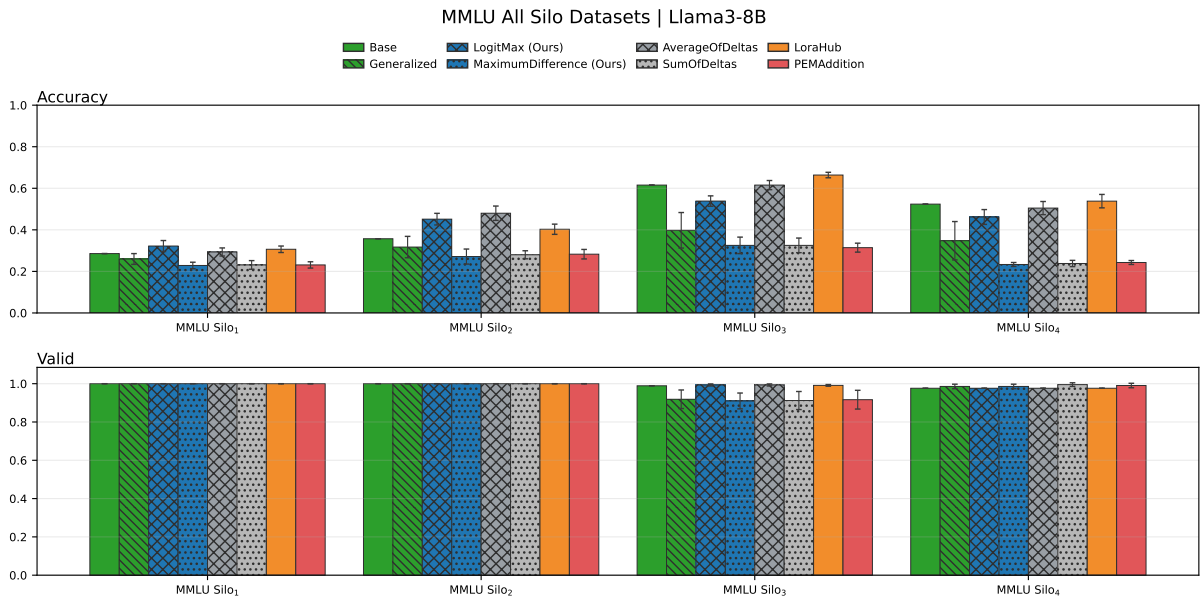



Figure 12: MMLU Results (Appendix J) using Llama-3 8B model (model #1/4).

experiments described in this manuscript (including only a single repetition, 100 GPU-hours for 5 repetitions).

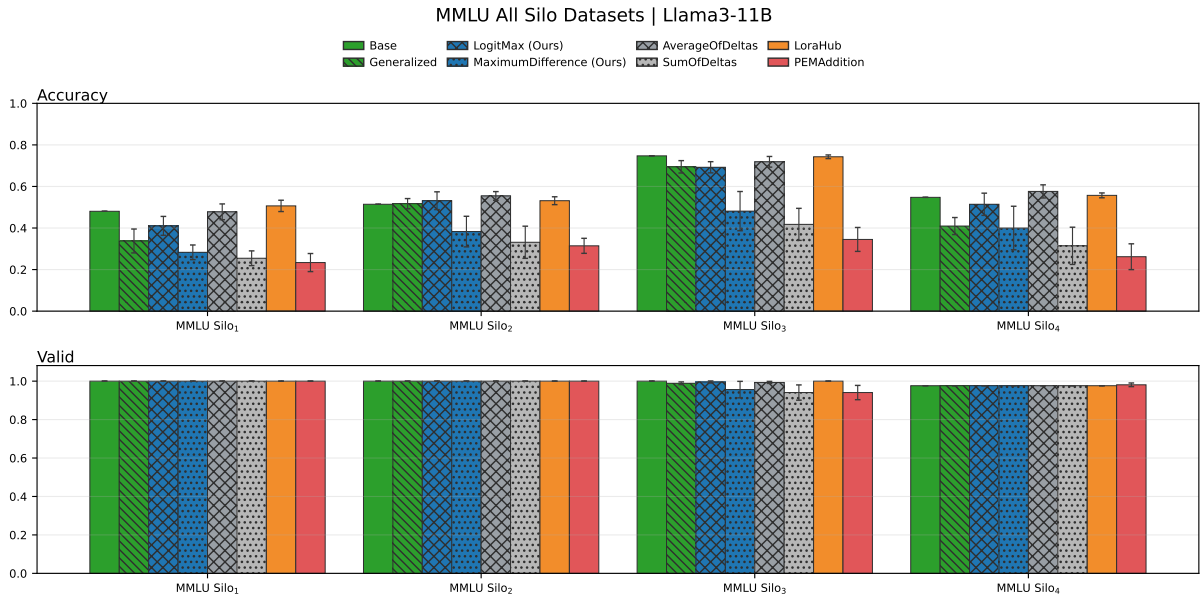



Figure 13: MMLU Results (Appendix J) using Llama-3 11B (model #2/4).

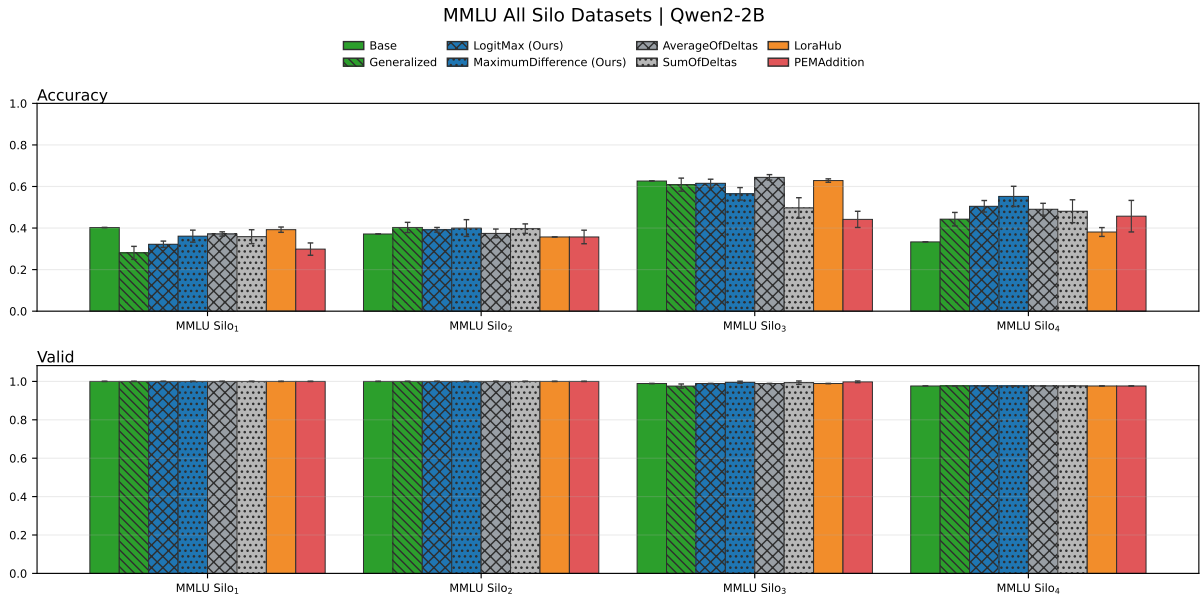



Figure 14: MMLU Results (Appendix J) using Qwen2 2B model (model #3/4).

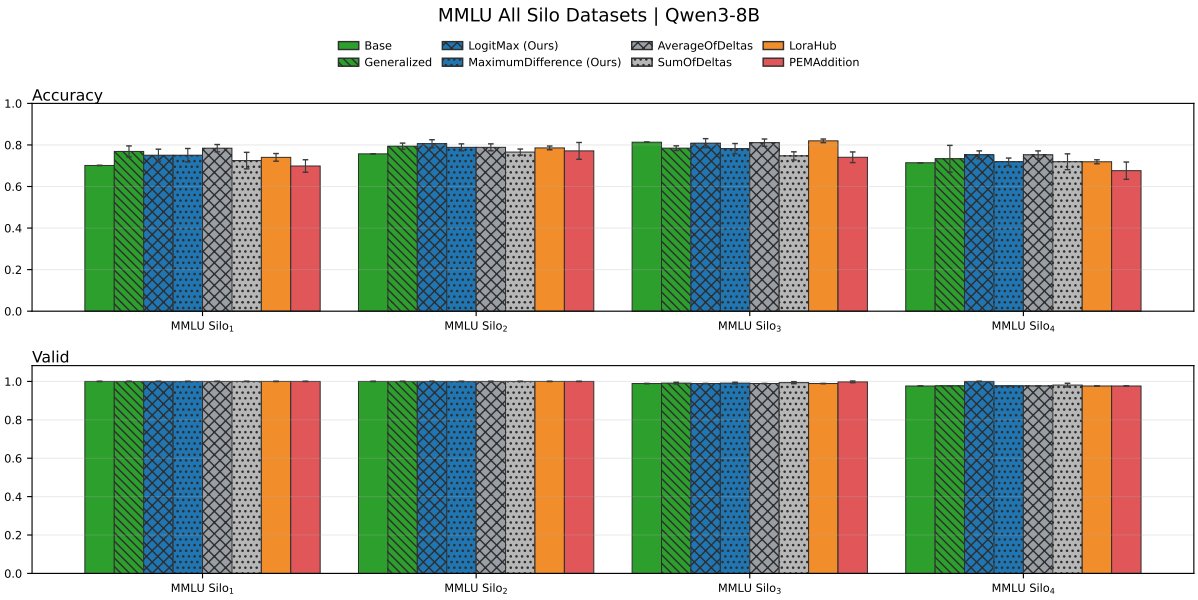
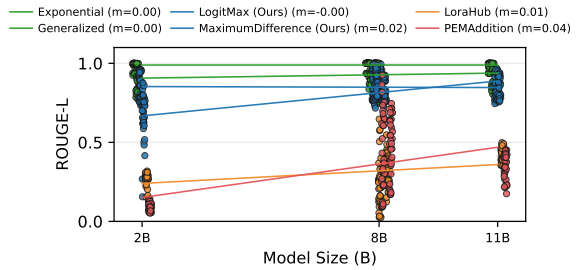
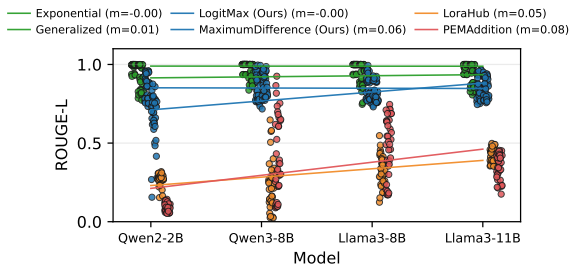


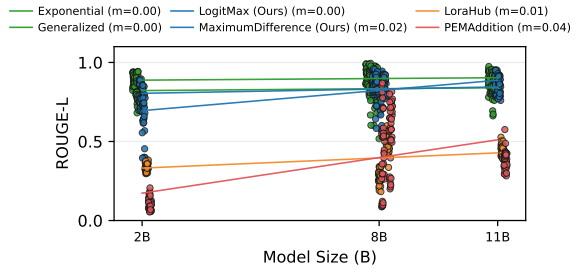

Figure 15: MMLU Results (Appendix J) using Qwen3 8B (model #4/4).



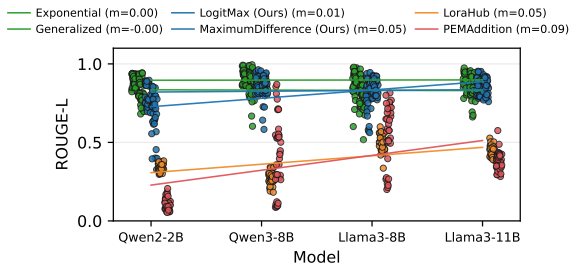

(a) ROUGE-L vs Model Size (B) on the main experiment



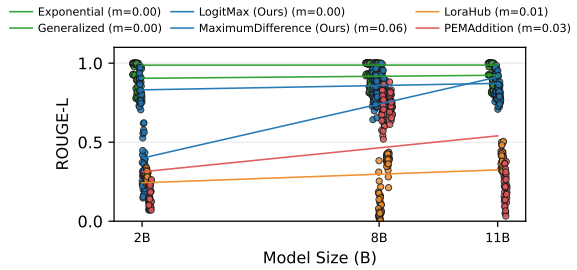

(b) ROUGE-L vs Base Model on the main experiment



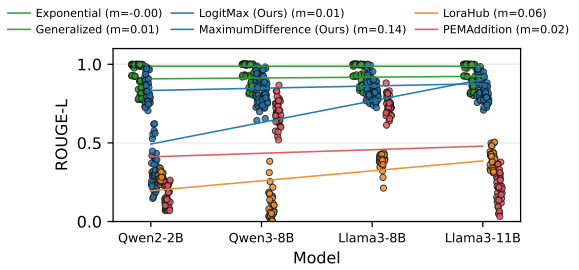

(c) ROUGE-L vs Model Size (B) on the rephrased ablation experiment




(d) ROUGE-L vs Base Model on the rephrased ablation experiment




(e) ROUGE-L vs Model Size (B) on the obfuscation ablation experiment




(f) ROUGE-L vs Base Model on the obfuscation ablation experiment

Figure 16: Comparisons of performance on Model Size for each composition method. Appendix K