Governing Open Vocabulary Data Leaks using an Edge LLM through Programming by Example

QIYU LI, University of California, San Diego, USA
JINHE WEN, University of California, San Diego, USA
HAOJIAN JIN, University of California, San Diego, USA

A major concern with integrating large language model (LLM) services (e.g., ChatGPT) into workplaces is that employees may inadvertently leak sensitive information through their prompts. Since user prompts can involve arbitrary vocabularies, conventional data leak mitigation solutions, such as string-matching-based filtering, often fall short. We present GPTWall, a privacy firewall that helps internal admins create and manage policies to mitigate data leaks in prompts sent to external LLM services. GPTWall's key innovations are (1) introducing a lightweight LLM running on the edge to obfuscate target information in prompts and restore the information after receiving responses, and (2) helping admins author fine-grained disclosure policies through programming by example. We evaluated GPTWall with 12 participants and found that they could create an average of 17.7 policies within 30 minutes, achieving an increase of 29% in precision and 22% in recall over the state-of-the-art data de-identification tool.

ACM Reference Format:
Qiyu Li, Jinhe Wen, and Haojian Jin. 2024. Governing Open Vocabulary Data Leaks using an Edge LLM through Programming by Example. 1, 1 (October 2024), 33 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Organizations are integrating ChatGPT into their workplaces to enhance employee productivity. However, a major concern is that employees may leak sensitive information inadvertently through the prompts sent to ChatGPT [31]. For example, Samsung Electronics [53] and Apple [89] ban employee use of ChatGPT after discovering staff include sensitive code and meeting notes in their prompts.

One classical solution to prevent similar data leaks is introducing a man-in-the-middle firewall [19, 68] using string-matching filters. For example, Microsoft Exchange [19] allows internal network administrators (admins) to develop regular expressions to detect whether an outgoing email contains a U.S. social security number. However, these string-matching filters are increasingly tenuous in managing these data leaks, since it is hard to create complex regex patterns for a specific type of data leak [59], manage a large number of policies [11], and debug these policies when the behaviors do not match with their expectations [6, 92]. Recently, the popularity of LLM services has made the problem even more severe, as the outgoing prompts are more diverse and the information flows are more centralized compared to emails [18, 53, 89].

This paper introduces GPTWall, a privacy firewall that helps internal admins create and manage policies to mitigate data leaks in prompts sent to external LLM services. GPTWall has two key ideas. First, we introduce a lightweight LLM running on the edge to obfuscate target information in prompts and restore the information

Authors' Contact Information: Qiyu Li, University of California, San Diego, La Jolla, USA, qiyuli@ucsd.edu; Jinhe Wen, University of California, San Diego, La Jolla, USA, jhw@ucsd.edu; Haojian Jin, University of California, San Diego, La Jolla, USA, haojian@ucsd.edu.

ACM XXXX-XXXX/2024/10-ART
https://doi.org/XXXXXXX.XXXXXXX

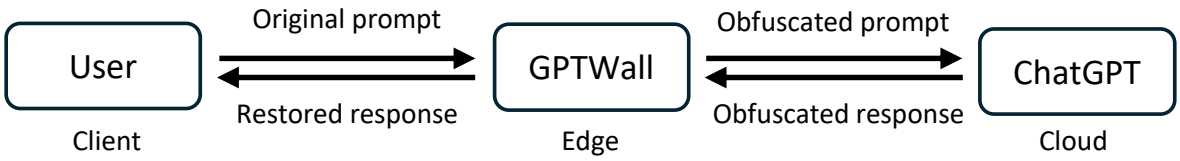


Fig. 1. GPTWall uses an LLM running on the edge to identify sensitive information in prompts and obfuscate the prompts before transmitting it to the external LLM service. Then GPTWall restores the information after receiving responses.

after receiving responses (Figure 1). Our key insight is that even though an edge LLM may be limited in reasoning and knowledge tasks, it can still enable many lightweight privacy-enhancing operations that were previously impossible. Second, GPTWall enables admins to author fine-grained disclosure policies through programming by example (PbE). Admins can annotate the content they want to obfuscate or protect in example prompts. GPTWall then analyzes these examples and suggests appropriate policies to the admins (Figure 3).

The design of GPTWall is based on an analysis of 50K conversations collected from ShareGPT [35] using automated analysis and manual coding (Section 3). The analysis led to three insights: (1) users usually have limited awareness of sensitive data leaks in ChatGPT conversations; (2) prompts provided by users often include a wide range of privacy-sensitive information; and (3) the determination of whether the information is privacy-sensitive heavily depends on the context of the prompt.

We implemented a prototype of GPTWall, which comprises a policy authoring interface and an edge LLM service. We used Llama3-8B [56], a state-of-the-art lightweight open-source model as the edge LLM. To reduce the latency, the edge LLM breaks prompts into smaller chunks and sends them asynchronously. We used OpenAI's streaming API [74] to handle ChatGPT responses in parallel as partial responses are received.

We conducted detailed experiments to validate the design of GPTWall. We first evaluated the parsing performance of GPTWall to demonstrate the effectiveness of edge LLMs in identifying sensitive information (Section 7.1). Second, we evaluated the usability of our policy authoring interface with 12 users (Section 7.2). We asked participants to play the role of admins to create policies and found that participants were able to create 17.7 policies within 30 minutes and achieved an increase of 29% in precision and 22% in recall compared to the state-of-the-art data de-identification tool. Finally, we evaluated the effectiveness of our obfuscation methods against attacks using real-world prompts selected from the ShareGPT dataset (Section 7.3) and examined GPTWall's impact on LLM response quality (Section 7.4). Our results show that GPTWall significantly reduces the risks of inferring sensitive information from users' prompts while maintaining response quality. Lastly, we evaluated GPTWall's system performance across various workplace scenarios (Section 7.5). Our experiments show that GPTWall introduces modest performance overheads for most scenarios.

In this paper, our main contributions are as follows:

- An end-to-end prototype implementation of GPTWall, which demonstrates how a man-in-the-middle edge LLM service can mitigate open-vocabulary data leaks.
- A mixed-methods analysis of a large real-world ChatGPT prompt dataset, suggesting the privacy-sensitive information in prompts is highly context-dependent.
- A programming by example algorithm, along with the underlying policy representations, which could infer policies from manually selected positive and negative examples.
- A detailed evaluation of GPTWall 's privacy benefits and policy authoring usability.

2 Related work

We have organized related studies into three categories: ChatGPT & ChatBot Privacy, Data Leak Prevention, and Programming by Example.

2.1 ChatGPT & ChatBot Privacy

Users have raised multiple privacy concerns regarding the use of LLM and ChatBot services [76]. One major concern is that providers may use users' prompts for training, which could lead to data being leaked to other users [65, 66]. In response, OpenAI started ChatGPT Enterprise, which claims, "We do not train on your business data or conversations, and our models don't learn from your usage" [73]. However, this does not prevent the potential resale of data [32] or the possibility of unexpected data breaches that may expose users' chat history [20]. Furthermore, there is no way to verify OpenAI's claim [30]. As a result, organizations are seeking self-hosting solutions [7, 78], where they deploy open-source large language models (LLM) on local servers [78] or infrastructure under their control (e.g., Microsoft Azure ChatGPT [7]). However, current open-source models tend to exhibit inferior performance compared to proprietary services such as ChatGPT [29, 87], and the cost of serving private LLMs can be steep [41, 52].

In contrast, GPTWall proposes using a lightweight edge LLM to enable a man-in-the-middle privacy firewall, which obfuscates sensitive information before sending it to ChatGPT and then recovers the sensitive information in the responses . Our key insight is that even though an edge LLM may be limited in reasoning and knowledge tasks, it can still perform many lightweight privacy-enhancing operations, such as parsing, identifying, and restoring open-vocabulary privacy-sensitive information.

2.2 Data Leak Prevention

Data leak prevention systems (DLPS) are widely used to prevent undesired disclosure by insiders [1, 4, 6, 27, 28]. These systems use content analysis techniques such as rule-based methods, regular expressions, fingerprinting, named entity recognition [28, 33], and statistical analysis [4, 6] to detect sensitive data and deny disclosures containing such data. For example, Microsoft Exchange allows admins to create rules based on keywords and regular expressions to block Social Security Numbers (SSN) and other Personally Identifiable Information (PII) in external emails [19]. However, these all-or-nothing approaches are often too coarse, inadvertently blocking many benign inquiries.

More recently, data masking solutions seek to redact private information within the text [14, 42, 69, 76], rather than blocking the disclosure. For example, Hide and Seek propose to create two lightweight models on the edge to anonymize the outgoing prompt and de-anonymize the response, respectively [14]. However, these content filtering implementations often rely on the identification of predefined categories (e.g., name, location, and SSN). But the predefined data categories are often insufficient to clearly assess the privacy risks [23] since the sensitivity of a given piece of information highly depends on the context of its use [70, 71]. In contrast, GPTWall uses an edge LLM to parse prompts and dynamically annotate the semantics of individual contents. By implementing this approach, GPTWall can develop policies that redact text in prompts based on specific contexts, such as entity types, entity values, and their co-occurrences with other information.

2.3 Programming by Example

Programming-by-example (PbE) is widely used in various applications such as information extraction [26, 47], data analysis [43], visualization [90], and programming [24]. The key challenge of learning policies from examples is how to effectively generalize from the user samples and narrow down the search space with weak supervision. For example, Ruler employs heuristics to narrow down the search space while learning labeling functions [22].

Cornet hypothesizes the output and then greedily learns cell formatting rules using a decision tree model [81]. NetEgg explores the search space of the network policies by guessing possible consistent values [99].

GPTWall builds upon the rich lineage of PbE systems but differs from previous systems in three key aspects: (1) it introduces a new underlying policy representation for privacy; (2) it uses LLMs as the engine to construct the search space; and (3) it infers policies using both positive and negative examples.

3 Understanding Privacy Leaks in Large Language Model Prompts

To understand the privacy leaks in interaction with large language models, we investigated real-world ChatGPT conversations shared by users using a combination of automatic parsing techniques and manual examination.

ShareGPT dataset. We used the ShareGPT dataset to analyze real-world practices of ChatGPT usage. The ShareGPT dataset comprises 90K ChatGPT conversations users share through ShareGPT Chrome extensions [35]. Each conversation is indexed by a unique ID and consists of dialogues between humans and the ChatGPT agent. The dataset may include sensitive information as users do not anticipate their data being publicized. We followed the data processing in Vicuna [17] and only retained English conversations with about 50K conversations.

Method. We first used the Amazon Comprehend service [67] to detect privacy-sensitive prompts. We randomly selected 300 prompts and manually examined them to investigate data leaks in ChatGPT conversations. We then utilized an iterative, bottom-up, open coding process [10] to identify common usage scenarios, data leak channels (e.g., text, code) and types of leaked information (e.g., PII, financial data) in sensitive prompts. Two authors manually and independently coded up these prompts and then discussed them to agree on a selective coding scheme. We notified developers about privacy concerns around the ShareGPT dataset by submitting a pull request to the Hugging Face repository with locations and types of potentially sensitive data.

Results. We make the following key observations. First, **users generally have limited awareness of the potential privacy leaks that can occur during their interactions with ChatGPT**. Despite individuals tending to only share conversations they deem secure, they often inadvertently paste unprocessed text containing sensitive information into ChatGPT without scrutiny, leading to implicit data leaks. For example, in Figure 2b, an employee pastes meeting notes into ChatGPT, including a URL link that may potentially reference sensitive documents. Appendix Figure 9 offers a detailed analysis of privacy risks across different tasks. Virtually all tasks, precisely 24 out of 26, can potentially reveal personally identifiable information (PII).

Second, **user prompts often encompass a wide range of privacy-sensitive information**. Table 9 unveils a wide array of privacy-sensitive information within ChatGPT prompts. We observe that specific scenarios are prone to reveal rich privacy information, particularly those related to information extraction, writing assistance, text analysis, and programming assistance. To illustrate, in Figure 2a, a user aims to extract housing details from customers using ChatGPT, inadvertently revealing the homeowner's email, location, and housing details. Moreover, the open-vocabulary nature of prompts poses a challenge to exhaustively enumerate all patterns of sensitive data. Traditional PII detection approaches based on a predefined taxonomy prove to be inadequate in this case. Instead, we need an open-vocabulary policy to effectively address data leaks in user prompts.

Third, **the identification of privacy-sensitive content in a text snippet heavily relies on the specific context**. For example, a customer's name in Figure 2a may be identified as sensitive, while the project leader's name in Figure 2b is more likely to be considered public. Simple categorization of such information as names is insufficient; fine-grained labels are necessary to distinguish them precisely within specific contexts. Moreover, organizations establish their own privacy policies, and the determination of sensitive information is subject to change. For example, the sensitivity of a company's total employee number depends on its public accessibility. Given the context-specific nature of sensitive information, there is no universally applicable solution to mitigate privacy risks. Therefore, we use a PbE approach to help admins author fine-grained policies by annotating examples of sensitive information in a specific context.

User*: You are an assistant that attempts to extract information from your user in a natural, friendly, and efficient manner on behalf of a company that wants to buy the user's home.*
You are instructed to extract the following information from the user for the following:

1. *The address of their home (key: address, type: string)*
2. *The number of bedrooms (key: num_bedrooms, type: number)*
3. *The number of bathrooms (key: num_bathrooms, type: number)*
4. *What other types of rooms they have in the house [example: kitchen, garage, etc.] (key: rooms, type: Array⟨room_name: string⟩)*
5. *The condition of those rooms. (key: rooms, type: Array⟨room_name: string; condition: good, fair, poor⟩).*

User*: [redacted], San Francisco, CA, [redacted] 3 bedrooms and the same number of bathrooms I have a kitchen, a living room, and a two-car garage Everything is in good condition, except for the bedroom, the carpet is moldy [redacted]@gmail.com.*

(a) Extracting housing details

User*: hello my friend, I am a software engineer working as a consultant for a construction company. I need to write up a proposal for work that has been discussed between me and the IT manager. He can then get approval for this work from upper management. I will be supplying you with a bunch of information about the problem, solution, and time estimations which I will ask to be written into a doc later in this convo. Sound good?*
User*: Ok here are some facts about the client which will be relevant later as we discuss logistics further:*

- *The companies name is [redacted] (I may refer to them as [redacted] when talking to you to shorten their name) - They are a general contractor operating with offices in [redacted]*
- *The company is [redacted]-[redacted] employees*
- *They are a company who manages multiple projects with different deadlines, budgets, owners, and needs. Any time that can be saved will compound to faster project delivery and better quality assurance.*

User*: Now I am going to paste some notes the IT manager wrote. Dont worry if these dont make total sense to you. Just take them as is - I know you will not be able to read the links so just ignore those and ill provide more context on those files:*
Notes & Required Documents

- *CA & NV have different new hires packets currently in BOX, Both compiled PDFs and additional documents are found in this BOX folder:*
  - *[New Hire Documentation (open link)] (https://[redacted].box.com/s/[redacted])*
- *Different in required documents between office new hire and field new hire*
  - *These differences are shown on the checklist on Page 1-2 on the CA NEW HIRE PACKET*
- *[redacted] is the lead updater/compiler for the CA new hire document*

(b) Assisting writing proposal

Fig. 2. Examples of ChatGPT use cases with privacy leaks from ShareGPT dataset in two usage scenarios. We selectively excerpt segments of user messages and redact the sensitive information within the conversation.

4 Programming Policies by Annotating Positive and Negative Examples

GPTWall allows admins to create policies by annotating positive and negative examples (Section 4.1). GPTWall then automatically infers the policies (Section 4.2) and helps admins iterate policies with real-time feedback (Section 4.3).

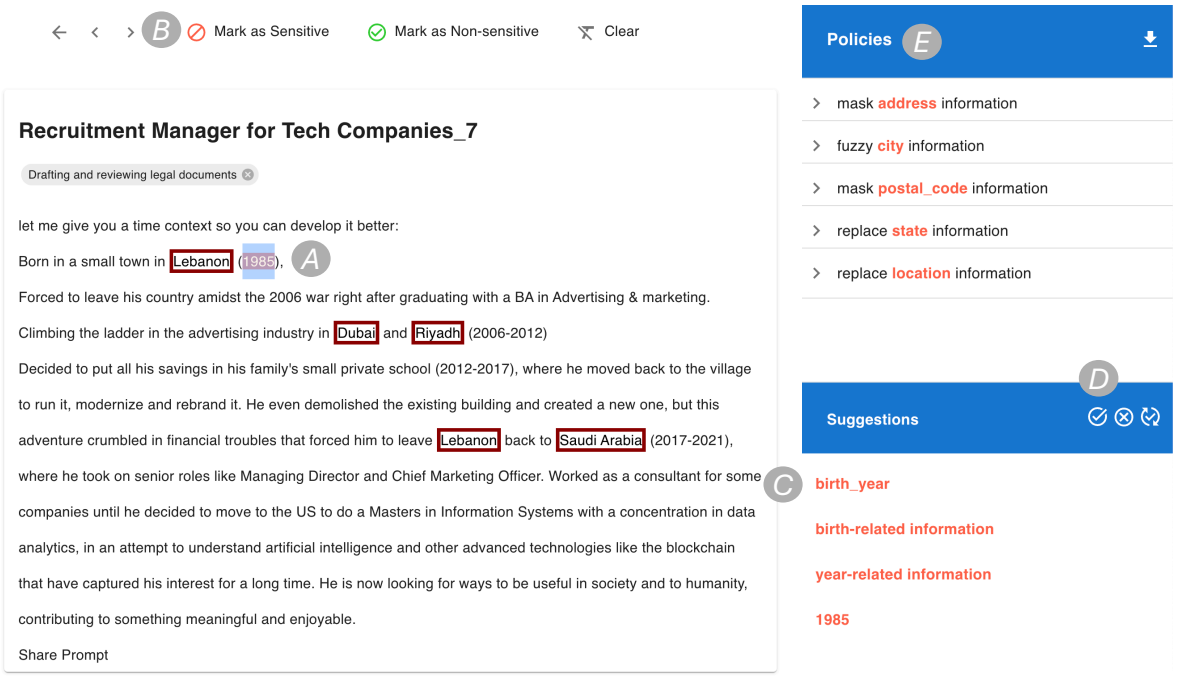


Fig. 3. The policy authoring interface of GPTWall. The admin starts by (A) identifying and (B) annotating sensitive information in prompts. From these examples, GPTWall automatically generates policy suggestions. Next, the admin (C) selects a suggestion and preview its effects. She (D) clicks the Checkmark button to finalize the policy. Afterward, the admin can (E) view and edit the policy in an interactive policy viewer.

4.1 Annotating Data Leaks with Positive and Negative Examples

GPTWall allows admins to specify information disclosure policies by annotating the information they wish to obfuscate and those they do not.

Input. The annotation process takes arbitrary text as the input, such as employees' LLM prompts, financial reports, meeting notes, and emails. GPTWall breaks them into paragraphs and instructs the edge LLM to annotate all entities with fine-grained semantic labels (See Section 5.1). A diverse candidate pool will help admins quickly identify good examples to create high-quality policies. During development, we randomly sampled 1,000 prompts from a public prompt dataset, ShareGPT [35], as the input.

Annotating examples. Figure 3 illustrates the policy authoring interface of GPTWall. Imagine that the admin wants to obscure information that could be used to identify individuals. As she reads through the prompt, she identifies the birth year "1985" as potentially sensitive and marks it as a positive example. GPTWall then generates a list of suggestions based on the annotated examples, such as "birth year" and "birth-related information." Admins can further annotate additional examples to refine the scope of the policy. For example, admins can specify the information they do not want to obfuscate to avoid unnecessary modification. If the admin considers only the birth year as sensitive, she can annotate all other years as negative examples.

Admins only need to manually annotate each type of example once, unless they wish to identify a negative example to refine the policy. Annotating prompts with positive and negative examples enables the admin to specify their intent with rich contextual information while simplifying the specification process.

Inputs: A large corpus of entities and corresponding labels from user prompts.
Definitions:

- Let P be a policy candidate list (i.e., specifications on information types), E be the set of examples labeled by admins. Define T to be the number of rounds for bootstrapping.

Initialization:

- Create an empty policy candidate list $P = \emptyset$, initialize the current policy $p = E$.

Algorithm:
For $t = 1$ to T:
  For each policy p in the last iteration:
    Filter corpus to find entities subject to the policy p and update the entity set S.
    Extract the set of semantic labels L of entity set S.
    Find the set of high-frequency keywords K based on semantic labels.
    For each possible keyword set in K :
      Update a new policy including all semantic labels with the keywords, excluding those of negative examples.
      For uncovered or error examples in E, add it to the value whitelist or blacklist of the new policy.
      Append the new policy to the policy candidate list P.
Output: The policy candidate list P.

Fig. 4. GPTWall's synthesis algorithm for inferring policies from positive and negative examples.

4.2 Inferring Policies from Annotated Examples

Policy representation. Underlying the GPTWall interface is a structured policy language that allows the admin to generalize obfuscation specifications across different prompts. GPTWall policy language consists of three parameters: information type, context, and method.

- `information type` defines the sensitive information the policy targets, which supports two kinds of specification: type-based (e.g., "personal name") and value-based (e.g., "123-45-6789" for the SSN).
- `context` defines the applicable condition of the policy. For example, while a phone number may typically be considered sensitive as PII, contact information used for business purposes (e.g., business phone and email) may not be regarded as private in a workplace setting. GPTWall formulates `context` as the co-occurrence with another information type in the same document.
- `method` determines how data is processed before being transmitted to the external service, such as *mask* and *anonymize*.

Combined, we formulate our semi-structured information disclosure policy design as follows:

{ "method" : "mask | anonymize | replace | noisify | fuzzify",
"information_type" : {"value": {"sensitive": <value list>, "non-sensitive": <value list>},
"label": {"sensitive": <label list>, "non-sensitive": <label list>} },
"context": <context list> }

Policy inference. Given a set of annotated examples, GPTWall automatically generates a list of policies through a snowball approach [36]. This process is initialized with the semantic labels of examples annotated by admins. We then broaden the coverage by incorporating additional relevant semantic labels. We break down each semantic label, such as "revenue_amount," into keywords like "revenue" or "amount," and identify keywords with high frequency. These keywords can act as cues to identify the target data types, as they often appear in annotated examples. If these keywords encompass additional semantic labels, we generate a new policy candidate that covers all semantic labels containing these keywords. This approach represents each policy as a list of semantic

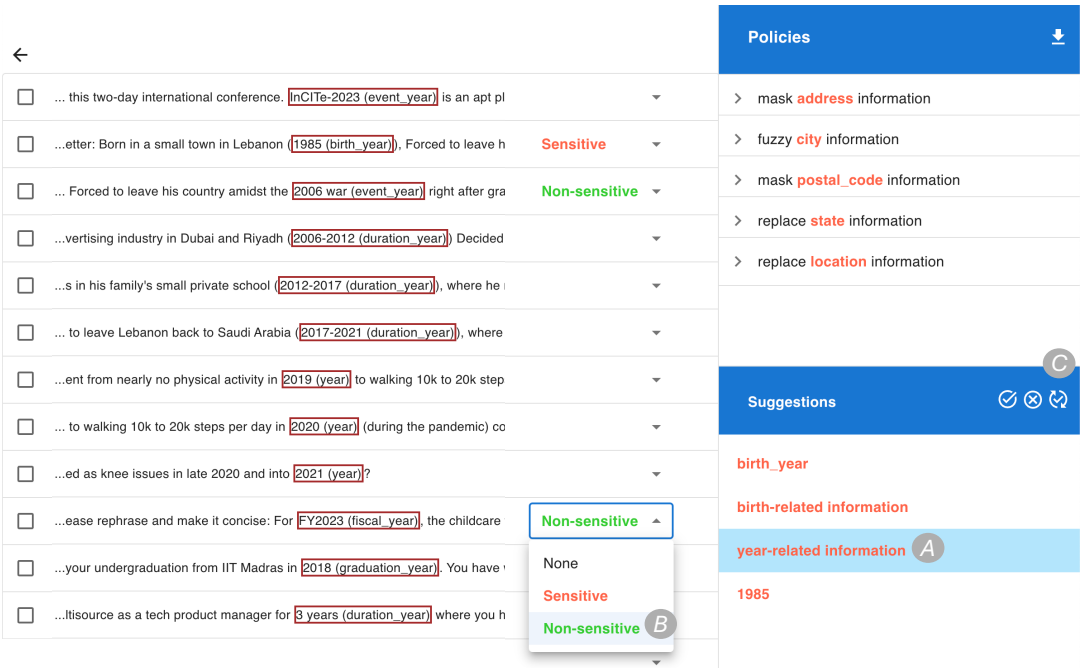
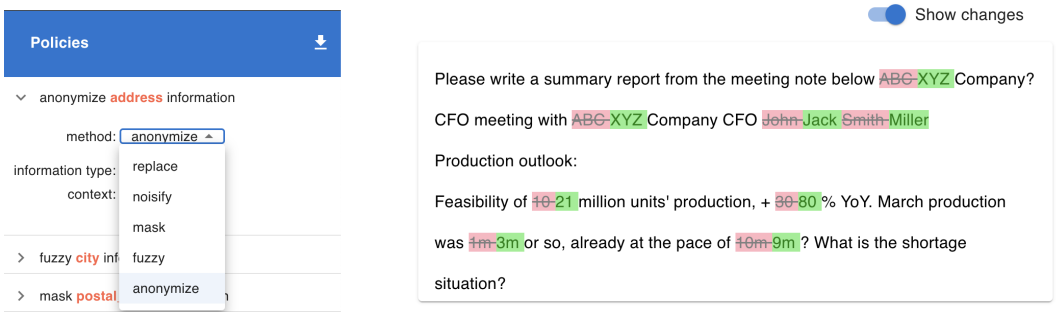


Fig. 5. The policy iteration process of GPTWall. The admin can (A) click the suggestion to preview its effects on the example prompts, (B) correct the false positive and (C) click the Update button to refine the policy.

labels linked to a set of keywords. Additionally, we include a blacklist containing all sensitive examples not covered by the current policy. Figure 4 shows the pseudo code of the algorithm. At each round, for each policy p in the candidate policy list P, we follow these steps: (1) identify entities S that are subject to the policy p, (2) extract the set of semantic labels L of all entities in S, (3) find high-frequency keywords sets K and (4) generate new policies based on the keywords and update the candidate policy list P. We empirically limit the snowball process to one round, as it is generally sufficient and additional rounds would result in too many false positives.

Specifying obfuscation method. After selecting policy suggestions, admins can specify the obfuscation methods for the policy. Since the appropriate obfuscation methods vary by contexts, we developed five types of obfuscation methods to support common privacy-enhancing operations.

- *Anonymize*: deidentifies sensitive data with its semantic label (e.g., changes "John Doe" to "<personal_name>").
- *Mask*: redacts sensitive data with a placeholder character (e.g., changes "1234-5678" to "XXXX-XXXX").
- *Replace*: substitutes sensitive data with an artificial value (e.g., changes "John Doe" to "Jane Smith").
- *Noisify*: injects a configurable random noise into the numerical value (e.g., changes "$10,000" to "$12,000").
- *Fuzzify*: obscures specific details (e.g., locations, ages, time) with a more general term (e.g., changes "28 years old" to "late 20s").



(a) Policy record (b) Visual preview

Fig. 6. Interface features of GPTWall. (a) GPTWall keeps all policies in an interactive record with natural language descriptions. Admins can expand a policy to view or adjust its obfuscation method and contexts. (b) GPTWall provides visual preview of the policy effects by highlighting changes between original and obfuscated prompts.

4.3 Iterating Policies with Real-time Feedback

GPTWall visualizes the effects of policy in real-time to help admins to validate and iteratively refine the policy.

Policy validation. GPTWall highlights the policy-affected fields in the current prompt for inline policy validation. For example, in Figure 3, all locations in the prompt are marked with a red box. This allows for in-place validation of the policy's effectiveness on similar data fields and helps identify sensitive data not covered by existing policies.

GPTWall helps admins quickly assess policy impacts by visualizing how the policy affects various prompts. Since sensitive information is often sparsely distributed across text [88], GPTWall displays all affected fields across prompts to show policy effects. It also provides semantic labels for each data field and allows admins to click on rows to view the complete prompt. This helps admins quickly evaluate policy coverage and false positive rates to choose the best option from policy suggestions.

Policy iteration. After selecting a suggestion, admins can review the policy effects to identify false positives and iteratively refine the policy (Figure 5). Admins can provide feedback by marking these fields as non-sensitive and then updating the policy accordingly to exclude these false positive cases. Admins can repeat this feedback loop until they feel satisfied with the quality of the policy.

We focus on two types of false positive errors: incorrectly included semantic labels and parsing error. We address the first type of error by removing the semantic labels of false positive information from the policy if there is no other positive examples with the same semantic labels. Otherwise, we add it to the whitelist of values. For parsing error, we directly add the false positive to the whitelist.

GPTWall also allows admins to correct false positives of existing policies. Admins can enter edit mode by hovering over the policy and clicking the Edit Icon, then mark the misclassified examples and click the Update Icon to refine the policy. GPTWall keeps track of these annotated examples and adjusts the policy accordingly to fix the errors and adapt to new prompts.

4.4 Debugging Support

GPTWall also provides other interface features to streamline the policy authoring process.

Natural language descriptions (Figure 6a). GPTWall generates natural language policy descriptions to make it easier for the admin to review and refine the policy. For each policy, we can describe its data operations using a four-element template: *<method> <information type> [with <configured_property>] [when/unless <context>]*. GPTWall only displays necessary parameters in the policy to provide a concise description. For example, instead

of complete list of semantic labels, we only show high-level keywords for better understanding. We also exclude undefined parameters to avoid redundancy in the description.

Policy record (Figure 6a). GPTWall incorporates each generated policy into an interactive policy viewer. Admins can expand the description by clicking the arrow on the left to reveal hidden parameters and make edits. They can also selectively enable and disable policies as needed. Additionally, GPTWall supports inline text annotation for documenting the rationale behind each policy to aid in policy understanding and management.

Visual previews (Figure 6b). GPTWall provides visual previews to illustrate the effects of policies on sample prompt. When admin activates the "show changes" button, GPTWall visualizes the differences between original prompt and obfuscated prompt, marking prior values in red with a strike-through and current values in green.

5 Privacy-preserving Data Flow

This section presents GPTWall's privacy-preserving data flow (Figure 7). Given a set of policies, GPTWall uses an edge LLM to parses prompts, identify sensitive information and obfuscates the prompts before sending them to the external LLM service. The edge model then restores the sensitive information in the responses to preserve response quality.

5.1 Identifying Sensitive Information

We use an edge LLM to parse prompts and recognize key entities. We then check if each entity is covered by the policies. If it's not on the whitelist or blacklist, we assess the sensitivity based on its semantic label.

Traditional NER techniques rely on a predefined taxonomy and face two main challenges with detecting sensitive data in prompts. First, expanding the predefined taxonomy to include additional data types is difficult, leading to significant limitations in handling diverse sensitive data in real-world prompts. Second, broad categories like person or location often fail to capture specific privacy-related information effectively. To address these challenges, we use an fine-grained open-vocabulary approach [8] to parse prompts (see prompt template in Appendix B). We employ the edge LLM to label each entity with detailed semantic descriptions (e.g., "growth_rate" or "personal_name"). We also incorporate privacy guidelines from the California Consumer Privacy Act (CCPA) [85] to ensure the language model appropriately references these sensitive data types.

5.2 Obfuscating Prompts

The edge LLM generates replacement values for sensitive information and reconstructs the original prompts with obfuscated values. Traditional PII anonymizers only allow the replacement of sensitive data with a predefined value. However, this method lacks robustness and can be easily detected. Therefore, the edge LLM dynamically generate plausible replacement values for each sensitive data field. To hinder adversaries from inferring the authentic value through correlations, we generate multiple options with varying semantic similarities and employ a randomized selection process. Specifically, we instruct the edge LLM to generate a list of potential replacements and assign a score between 0 and 1 for each candidate to measure semantic similarity (see prompt template in Appendix C). We then randomly select the options based on the scores, prioritizing options with higher semantic similarity. To generate replacement candidates, we mask all sensitive information specified by the policy and prompt the language model to predict appropriate values to fill in the masked positions. Inspired by differential privacy mechanisms [21], we incorporate a parameter ϵ, into the random selection process. Admins can configure the parameter ϵ to fine-tune the level of randomness in selection to balance between privacy and utility. The overall process can be summarized as follows:

a. The edge LLM generates a set $\mathcal{R}$ of replacement values and assigns a score to each candidate based on semantic similarity.
b. GPTWall selects the candidate $r \in \mathcal{R}$ with probability proportional to $\exp(\epsilon \cdot u(x, r))$.

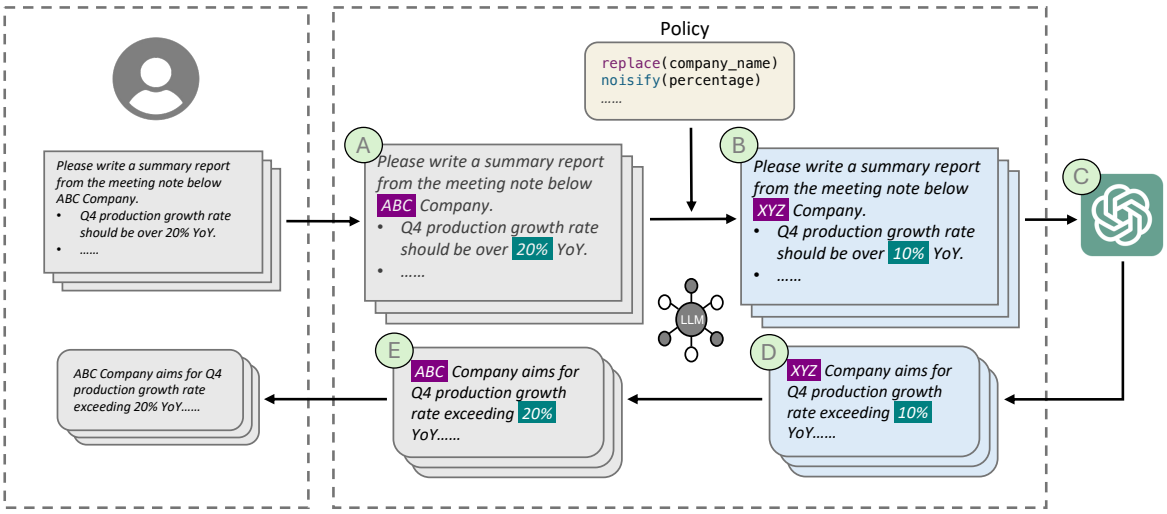


Fig. 7. Given a set of policies, GPTWall uses an edge LLM to (A) identify sensitive information in prompts, then (B) obfuscates prompts before (C) sending them to the external LLM service. After (D) receiving responses, the edge LLM (E) restores the information to preserve response quality.

Noisify utilizes the Bounded Laplace mechanism [34] to perturb numerical values, since Laplacian noise offers strong local differential privacy guarantees [34, 63], and unbounded noise can lead to semantically implausible edge cases (e.g., negative revenue amounts) [34, 63]. Admins can specify the range of perturbation by configuring the properties of the policy. *Fuzzify* follows a similar process to the *replace* method, the only difference is that we prompt the edge model to obscure details and generate a general term for sensitive data. We instruct the edge LLM to reconstruct the original prompt instead of straightforward replacements, which may introduce imperfections and inconsistencies [2].

5.3 Restoring Sensitive Information in LLM Responses

After obfuscating prompts, the edge model recovers sensitive information within the LLM responses to maintain service quality (see prompt template in Appendix C). For *mask* and *anonymize*, we instruct the edge LLM to select proper values to fill in the obfuscated field. For other methods, we reverse the obfuscation process by swapping the original and modified values when reconstructing the prompts. For example, if we replace the growth rate "20%" with "10%" within prompts, we restore the sensitive information by changing "10%" back to "20%" in the responses (Figure 7). GPTWall maintains the mapping between the original and obfuscated information using a local database storing pairs of original and obfuscated data.

Since LLM conversations can span multiple sessions, attackers may aggregate data from the same user over time. To hindering attempts to infer sensitive information using statistical methods, we use a stateful approach to maintain consistency across sessions. When selecting a replacement value for the information, we first query the database to check for previous obfuscations and retains the obfuscated value if found to ensure consistency.

6 Implementation

The GPTWall prototype comprises a policy authoring interface and an edge LLM service. We implemented the policy authoring interface in React.js [57]. For pre-processing user prompts, we used Llama3-8B [56], a lightweight open-source LLM to execute the privacy-preserving data flow.

To minimize performance overhead, GPTWall processes prompts while the user is typing. Instead of handling the entire prompt at once, GPTWall breaks it into smaller chunks and processes them asynchronously for better parallelism. Each chunk retains its full semantic meaning. For pre-processing, we keep a buffer of user input and process each chunk as soon as the user finishes typing. We also generate replacement values in parallel to speed up obfuscation. When the user sends the query, we aggregate the outputs to derive the obfuscated prompt. For post-processing, we used the OpenAI Streaming API [74], which allows us to receive partial responses as they become available. Similarly, we process each chunk as soon as we receive it and then aggregate the results to generate the final response for the user. We empirically set the chunk size to one sentence or one line of text, depending on which is smaller.

7 Evaluation

This section presents detailed experimental evaluations of GPTWall, including its parsing performance (Section 7.1), usability for admins (Section 7.2), effectiveness against attack (Section 7.3), impact on response quality (Section 7.4), and system performance (Section 7.5).

7.1 Parsing Performance

We examined the effectiveness of edge LLMs in identifying sensitive information using a real-world dataset.

Dataset. We curated a dataset of sensitive user prompts from the ShareGPT dataset. Initially, we filtered out prompts that were too long (> 4K tokens) or too short (< 10 tokens). From the filtered dataset, we randomly selected 500 conversations and screened them for sensitive information, resulting in 89 potentially sensitive conversations with a total of 1,218 user prompts. We followed CCPA [85] and HIPAA [72] guidelines to select six types of sensitive information for evaluation (Table 1): age and birthdate (quasi-identifiers), personal name and location (PII), and financial and health data (sensitive data). Two authors independently labeled the dataset manually, cross-validated the labels, and resolved conflicts through discussion to agree on ground-truth labeling.

Method. We tested the efficacy of our approach with two popular lightweight open-source LLMs: Mixtral-8x7B model [40] and Llama3-8B model [56]. First, we assessed the coverage of sensitive information by comparing identified entities with ground-truth labeling, regardless of semantic labels. Next, we calculated the precision and recall of semantic labels. Due to the open-vocabulary approach, each type of sensitive information may consist of multiple semantic labels. We considered labels correct if their semantics are generally relevant to the corresponding data type. For example, the label "heart_rate" could be reasonably associated with health data. Likewise, in precision calculation, we accounted for all entities with related semantic labels.

We used Presidio [60], a widely-used PII detection SDK from Microsoft, as the baseline. We attempted to align the six sensitive information types with Presidio's pre-defined entity types for comparison. For example, personal names may correspond to the `PERSON` entity type. However, since we could not match age with Presidio's built-in entity types, we manually created regex patterns based on patterns identified in the evaluation dataset. We also included two baselines with simpler prompting strategies to further validate the effectiveness of our approach: no fine-grained labels (NoFL) and no privacy domain knowledge (NoDK). In NoFL setting, we used general

Metric	Method	Data Type						
		Name	Location	Birthdate	Age	Financial	Health	Average
Coverage	Mixtral-All	93.26%	**91.46%**	**100.00%**	98.41%	**97.73%**	88.50%	92.71%
	Mixtral-NoFL	**98.88%**	72.56%	57.14%	96.83%	95.45%	89.38%	86.46%
	Mixtral-NoDK	97.75%	76.22%	**100.00%**	**100.00%**	95.45%	**99.12%**	90.83%
	Llama3-All	**98.88%**	83.54%	**100.00%**	98.41%	95.45%	95.58%	91.88%
	Llama3-NoFL	**98.88%**	73.17%	57.14%	98.41%	**97.73%**	82.30%	85.42%
	Llama3-NoDK	**98.88%**	85.37%	**100.00%**	98.41%	95.45%	98.23%	**93.13%**
Precision	Presidio	63.69%	74.45%	27.27%	93.02%	69.44%	56.52%	67.85%
	Mixtral-All	69.23%	78.72%	85.71%	93.55%	**88.54%**	77.69%	78.63%
	Mixtral-NoFL	86.43%	75.57%	30.00%	92.50%	44.68%	60.57%	72.08%
	Mixtral-NoDK	83.07%	**80.71%**	**100.00%**	**94.59%**	72.84%	85.19%	82.95%
	Llama3-All	85.01%	78.74%	**100.00%**	93.42%	65.63%	77.62%	81.02%
	Llama3-NoFL	86.13%	78.37%	51.95%	91.36%	50.00%	81.03%	78.40%
	Llama3-NoDK	**88.40%**	80.17%	**100.00%**	93.24%	64.89%	**85.94%**	**83.85%**
Recall	Presidio	86.52%	75.61%	**100.00%**	68.25%	63.64%	56.64%	70.02%
	Mixtral-All	91.01%	**83.54%**	**100.00%**	98.41%	93.18%	85.84%	88.54%
	Mixtral-NoFL	93.26%	69.51%	57.14%	96.83%	95.45%	82.30%	82.71%
	Mixtral-NoDK	92.13%	68.90%	57.14%	**100.00%**	95.45%	**92.04%**	85.00%
	Llama3-All	**94.38%**	79.88%	57.14%	98.41%	95.45%	90.27%	88.54%
	Llama3-NoFL	92.13%	68.90%	57.14%	95.24%	**97.73%**	74.34%	80.42%
	Llama3-NoDK	**94.38%**	**83.54%**	42.86%	96.83%	95.45%	**92.04%**	**89.79%**
F1 score	Presidio	68.25%	75.03%	42.85%	78.73%	66.41%	56.58%	68.92%
	Mixtral-All	78.64%	81.06%	**92.31%**	95.92%	**90.80%**	81.56%	83.29%
	Mixtral-NoFL	89.72%	72.41%	39.34%	94.62%	60.87%	69.78%	77.03%
	Mixtral-NoDK	87.37%	74.34%	72.72%	**97.22%**	82.63%	88.48%	83.96%
	Llama3-All	89.45%	79.31%	72.72%	95.85%	77.78%	83.47%	84.61%
	Llama3-NoFL	89.03%	73.33%	54.42%	93.26%	66.15%	77.54%	79.40%
	Llama3-NoDK	**91.29%**	**81.82%**	60.00%	95.00%	77.26%	**88.89%**	**86.72%**

Table 1. Microbenchmark results for parsing. We assessed the performance of GPTWall and Presidio in identifying six types of sensitive information across 1,218 ChatGPT prompts. GPTWall achieves higher precision and recall compared to Presidio across all types of information.

categories instead of fine-grained labels for each entity. In NoDK setting, we excluded the privacy guidelines on sensitive information when prompting the LLMs.

Results. Table 1 presents the parsing performance for different types of sensitive information. Both models achieved over 92% coverage and an F1 score above 83%, significantly outperforming Presidio with an F1 score of 68.92%. Notably, Mixtral-All demonstrated higher recall (average of 88.54% v.s. 70.02%) and precision (average of 78.63% v.s. 67.85%) than Presidio across all types of sensitive information. For types not natively supported by Presidio, the edge LLM significantly increased the F1 score from 10% to 30% on both models. For Presidio's built-in types (i.e., Name and Location), the edge LLM also exhibited an increase of over 4% in F1 score. Our results show that edge LLMs achieved better performance than Presidio in identifying sensitive information, particularly for out-of-vocabulary data types

We compared the parsing performance of different prompting strategies. The NoFL setting showed over a 5% drop in coverage, recall, and F1 score, and around a 3% drop in precision compared to the original setting on both models. The results indicates that fine-grained labels can provide more accurate specifications and better coverage of sensitive information. Conversely, the NoDK setting generally showed higher precision and recall

Category	Error Type	Description	Example
Identification Error	Missing Entity	An entity containing sensitive information that is not identified.	[Jack Smith: personal_name] is a *42-year-old* patient being evaluated for involuntary movements.
	Inaccurate Entity Boundary	Entity identification that is incomplete or includes words or phrases that do not belong to the entity.	A *[74-year-old woman: age_gender]* is brought to the physician by her husband because of [difficulty sleeping: symptom] for [several years: duration_time].
	Dummy Entity	An entity with no actual information or only dummy data.	My *[home address: address]* is [7320 Norval Lodge Suite 243: address].
Label Error	Incorrect Label	Label that does not match semantics of the entity.	AB is the average number of guests, and *[BC: location]* is the actual cost that is paid by the guest...
	Ambiguous Label	Label that is unclear or lacks specificity.	Pretend you are a *[Workday: name]* user and you are the HR CEO of a large company...
Format Error	Incorrect Format	Parsing result that does not meet the specified format.	They had two children *Alice: personal_name* age [12: age] and *Bob: personal_name* age [14: age].
	Inconsistent Entity	Entity altered from its original form in user's prompt.	[Tom: personal_name] has a joint savings account with *[$200,000: saving_amount]*. (Original form: *200,000 USD*)

Table 2. The codebook of parsing errors. We highlight the parsing errors with italic format in the examples.

than the original method, suggesting that privacy guidelines may not be necessary since most LLMs are already trained to align with human values [75, 77, 93, 95].

We also performed error coding to pinpoint areas for improvement. Two coders independently examined the parsing results of Mixtral-All and developed a codebook of parsing errors (Table 2). The majority of errors we found were identification errors, including Missing Entity ($n = 28$), Dummy Entity without actual information ($n = 26$), and Inaccurate Entity Boundary ($n = 4$). In addition, we noted two types of label errors: Incorrect Labels ($n = 6$) and Ambiguous Labels lacking specific details ($n = 15$). We also observed format errors, such as Incorrect Format with missing brackets or incorrect colon positions ($n = 2$) and Inconsistent Entities altering from the original text ($n = 3$). These findings outline the imperfections of current language models for open-vocabulary parsing. With recent rapid advancements in LLMs [3, 37, 39, 62, 86], we anticipate future open-source LLMs to rectify these issues and continually improve the parsing performance.

7.2 Policy Usability for Admins

We conducted an IRB-approved user study to evaluate how GPTWall could assist administrators in creating policies for governing data leaks in users' prompts. Specifically, we evaluated the usability of GPTWall primarily from two aspects:

- Does GPTWall streamline the process of creating policies that govern sensitive information in user prompts for admins?
- Does GPTWall help admins to craft more effective policies for diverse types of sensitive information?

Method. We recruited 12 undergraduate and graduate students (3 identified as female, 9 identified as male, aged 18-26) as participants. All participants have at least 3 years of programming experience (Mean=5.36, SD=2.33) and are familiar with current PII detection methods, including regex-based pattern matching and named entity recognition techniques. Each participant received compensation of 20 USD for their time.

The study was a within-subjects design, where participants used both GPTWall and Presidio to complete six tasks detecting specific types of sensitive information: A: age, B: personal name, C: health data, D: birth date, E: location, F: financial figures. We grouped them into three pairs, each representing a kind of sensitive information with similar difficulty levels: (A, D), (B, E), and (C, F). We presented these tasks in two orders: (A, B, C); (D, E,

Task	Presidio					GPTWall				
	Time (min)	#Rule	#Success	Precision	Recall	Time (min)	#Policy	#Success	Precision	Recall
Name	**4.49**±2.94	1.0	6	63.69%	86.52%	10.50±4.00	4.0	6	**77.41%**	**87.80%**
Location	12.80±7.23	1.4	3	67.78%	35.88%	**12.52**±7.82	3.8	5	**82.98%**	**68.29%**
Birthdate	11.72±5.75	1.7	2	49.71%	55.10%	**7.52**±4.93	2.7	5	**98.33%**	**97.57%**
Age	12.21±7.76	2.4	3	73.71%	36.83%	**4.38**±3.32	1.2	6	**93.73%**	**97.95%**
Financial	**9.25**±4.22	2.0	3	37.79%	54.55%	12.35±6.08	11.3	6	**52.37%**	**85.98%**
Health	13.22±7.86	4.4	5	50.12%	**63.89%**	**9.21**±1.50	12.2	6	**83.88%**	61.19%
Average	10.61±6.54	2.1	3.7	54.60%	53.98%	**9.42**±5.48	5.9	5.7	**83.91%**	**76.17%**

Table 3. The results of task completion in the user study. Our results show that participants spent less time and completed more tasks successfully using GPTWall compared to Presidio. Participants achieved significantly higher precision and recall using GPTWall than the baseline.

F) or (D, E, F); (A, B, C). Participants were instructed to complete each task set under a different condition. We counterbalanced the presentation order of task sets, as well as the order of the conditions, among all participants.

For each task, we provided participants with 10 sample prompts. We asked participants to read through these prompts and detect the specified type of sensitive information using GPTWall or Presidio. Participants were instructed to preview the detection results and refine iteratively to improve precision and comprehensiveness. We imposed a 20-minute limit per task to prevent participants from getting caught up on a single task. However, we instructed participants to inform the researcher if they felt unable to further improve the detection performance.

For the control condition, we developed a Jupyter Notebook interface based on the Presidio tutorial [60]. We provided a section for implementing sensitive information detectors, along with an analysis tool to visualize the detection results. Given the complexity of regular expressions [59], participants could use internet searches and AI tools such as ChatGPT for assistance.

Our study took about 90 minutes for each participant, including two 45-minute sessions. In each session, we started with a 10-minute walk-through briefing on the tasks and study purpose, along with a tutorial on the tool. Then we instructed participants to complete the assigned tasks. At the end of each session, we asked the participant to complete a questionnaire and take a semi-structured interview about their experiences in the session. In the survey, participants filled out a System Usability Scale (SUS) test [12] and a usability questionnaire covering four usability characteristics: ease of use, learnability, expressiveness, and satisfaction [22]. For each characteristic, we presented a statement related to each condition's characteristics and asked the participants to rate on a 5-point Likert scale from 1 (strongly disagree) to 5 (strongly agree). For example, ease of use refers to the statement, "I thought it was easy to complete the tasks with GPTWall." Learnability refers to "I think that Presidio is easy to learn." The study was conducted in a lab space or over a Zoom meeting.

Results. Table 3 shows task completion results of GPTWall and Presidio. We measured each participant's time to complete each task and calculated the average time required to create a policy or rule. Participants completed tasks faster (average of 9.42 v.s. 10.61 minutes) and created more policies or rules (average of 17.7 v.s. 6.3 for all tasks) using GPTWall than Presidio. On average, participants spent less time (1.59 v.s. 5.0 minutes) to create a single policy when using GPTWall compared to the baseline, which was statistically significant ($p < 0.01$) under the Mann-Whitney U test [54]. These results suggest that GPTWall simplifies the policy creation process and helps admins manage sensitive information more efficiently.

We compared the effectiveness of policies with and without using GPTWall. We considered a task successful if the participant could detect all sensitive information in at least 80% of sample prompts. On average, participants completed more tasks successfully (5.7 v.s. 3.7) using GPTWall than the baseline, indicating that GPTWall enables admins to regulate identified sensitive information more effectively. Furthermore, we evaluated the quality of

the policies by comparing the participant detection outcomes with the ground-truth labels from the previous evaluation of parsing performance (Section 7.1). We calculated precision and recall for each type of sensitive information under each condition. Overall, participants using GPTWall achieved significantly higher precision (average of 83.91% v.s. 54.60%, $p < 0.05$) and recall (average of 76.17% v.s. 53.98%, $p < 0.05$) in nearly all tasks, with a 29% increase in precision and a 22% increase in recall compared to the baseline. Our results indicate that GPTWall assists admins to craft more precise and comprehensive policies across various types of sensitive information.

Table 4 shows the usability results for GPTWall and Presidio. On average, GPTWall achieved a significantly higher SUS score than the baseline (81.46 v.s. 28.54, $p < 0.01$) based on Wilcoxon Signed rank test [94]. The Mann-Whitney U test showed statistically significant effects in ease of use, learnability, expressiveness and satisfaction ($p < 0.01$). These results suggest that GPTWall can simplify policy creation process, reduce learning costs, and improve versatility in handling diverse sensitive information.

		Ease of use	Learnability	Expressiveness	Satisfaction
Presidio	Median	2	1.5	4	2
	IQR	1.75-2	1-2	2-4	1.75-2.25
GPTWall	Median	**4.5***	**4.5***	**5***	**5***
	IQR	4-5	4-5	4-5	4-5

Table 4. Subjective feedback collected from the post-study questionnaire in the user study. Each participant was asked to rate for each usability characteristic on a Likert scale of 1 to 5. We compared the ratings of GPTWall and Presidio with a two-tailed Mann-Whitney U test. The results show that GPTWall outperforms the baseline in four usability aspects with statistical significance (*) with all p-values < 0.001.

Qualitative findings. We present the major qualitative findings from the observation of participants behaviors as well as their feedback from the post-study interviews below.

First, all participants mentioned that GPTWall was much easier to use and saved them a lot of effort compared to the baseline method, echoing the usability results in Table 4. They felt the policy authoring interface was *"user-friendly and straightforward"* (P9) and *"convenient to use"* (P10). Some participants (4/12) were impressed by GPTWall's functionality: *"It's like magic! It can automatically recognize sensitive information with only a few examples"* (P11), and *"I'm surprised by how effortless it is to create policies with just a few clicks"* (P1). Thus, participants unanimously preferred to create policies by annotating a few examples rather than manual programming: *"I can detect a lot of sensitive information without writing a single line of code. That's fantastic!"* (P5). Meanwhile, many participants (8/12) mentioned that GPTWall *"was easy to learn"* (P1) and *"did not require extra knowledge"* (P11). They also saw how GPTWall could potentially make the management of private information more accessible to laypeople. As P8 noted, *"I couldn't imagine how my grandma managing her personal info[rmation] with regex. But now, [with GPTWall,] it seems like it could actually be possible."*

Second, participants appeared to appreciate the visualization of the policy effects and support for iteration by GPTWall. They thought that the visualization was helpful for the iterative process by providing real-time feedback: *"The mark is noticeable — I can easily see which information was detected and which wasn't when creating policies"* (P6), and *"The process flows well — With each click, it directly displays the results of my actions. If I notice something wrong, I can simply click to make modifications"* (P4). Participants also valued the feature to iterate on identified errors: *"Whenever I find a bad case, I can just iterate on it - that's the best feature I love"* (P3). On the other hand, we observed that participants struggled with iterating on detection results in the baseline, even with the assistance of AI tools. Many participants expressed that they *"had no clue how to tweak regular expressions to cover new sensitive info[rmation]"* (P9). P2 reflected on his experience refining regular expressions and said that, *"I tried messing with these regex to catch new patterns, but it didn't work. After a few tries, it actually made stuff worse, so I went back to how it was. In the end, I just gave up."*

Lastly, nearly all participants (11/12) perceived GPTWall to be more generalizable and expressed greater satisfaction with its detection results. P12 remarked that *"[GPTWall makes] a big progress compared to the baseline, both in accuracy and generalizability."* We noticed a shift in participants' behavior patterns during the review process of sample prompts. Initially, they carefully read prompts to identify sensitive information and create policies. As policies evolved and expanded, they mainly skimmed the prompts to validate detection results. We further corroborated this observation with their reflections, such as *"At first, I checked each prompt carefully to avoid missing any sensitive info[rmation]. But after seeing it catch everything in a few prompts, I started trusting its detections and just verified the results"* (P9). However, several participants (3/12) also noted that GPTWall might cause confusion when *"some similar info[rmation] failed to be detected"* (P6) due to parsing errors, suggesting that future versions of GPTWall should explore more robust solutions for corner cases.

In the post-study interview, some participants (4/12) expressed the willingness to have GPTWall integrated as a web extension for their daily use. P11 even stated that *"hope to put it into market as soon as possible!"* Despite encountering a few bugs in our research prototype during the study, their enthusiasm indicates the potential value that our system holds for our participants.

7.3 Effectiveness against Attack

We evaluated the effectiveness of using edge LLM to reduce the risks of inferring sensitive information.

Method. We evaluated GPTWall's effectiveness against inference attacks using state-of-the-art LLMs due to its efficiency in accurately infer a wide range of sensitive attributes from the text [84]. Specifically, we followed Staab et al. [84] to prompt GPT-4 (model `gpt-4-0125-preview`) to deduce sensitive information from prompts and give its top-3 guesses (see prompt template in Appendix D). For text, we considered a guess correct if conveying the same meaning as the ground-truth. For numerical data, we accepted guesses with less than 10% error. If the ground truth and the guess are ranges, we required a minimum overlap percentage of 80%. We measured the top-1 and top-3 accuracy of GPT-4 model guesses for five methods (*mask*, *anonymize*, *replace*, *noisify*, and *fuzzify*) across various settings.

Setup. We first conducted a preliminary experiment to understand the effectiveness of obfuscation methods for common usage scenarios. We curated a small-scale dataset including 9 real-world ChatGPT prompts from the ShareGPT dataset. These prompts span a broad array of contexts, including university environment, large corporations and small teams scenarios. Table 5 shows each prompt's usage scenario and related sensitive data type. For each prompt, we identified at least five pieces of sensitive information, for a total of 52, including 29 text and 23 numerical information. Two authors crafted descriptions for each piece of sensitive information and cross-validated to ensure the descriptions accurately and uniquely represent the information in the prompt. For *mask* and *anonymize*, we directly generated 10 obfuscated prompts. For other methods, we experimented with $\epsilon = 0.1, 0.5, 2$ and generated 10 obfuscated prompts for each applicable prompt.

We then used the dataset in Section 7.1 to quantitatively evaluate the performance of obfuscation methods against inference attacks. We instructed GPT-4 to provide descriptions for sensitive information in prompts and and manually checked and corrected any inaccurate annotations to ensure precision and specificity.

Results. Table 6a shows the evaluation results on the sample dataset. The *mask* and *anonymize* methods reduced the accuracy of GPT-4 top-3 guesses from 98.1% to 21.2% and 19.2% respectively. When $\epsilon = 0.5$, the *replace*, *noisify* and *fuzzify* methods decreased the top-3 accuracy from 96.6%, 100%, 98.1% to 34.5%, 30.4% and 30.8% respectively. For these methods, the top-3 accuracy decreased with higher randomness (smaller ϵ). Specifically, the *noisify* method droped by over 40% from $\epsilon = 2$ to $\epsilon = 0.1$, while the *replace* and *fuzzify* methods decreased by around 10%. Therefore, we used $\epsilon = 0.1$ for the larger-scale experiment.

We qualitatively analyzed the effectiveness for different types of sensitive information by manually examining the obfuscated prompts. While *mask* and *anonymize* apply to all data types, they work best for information

#ID	Usage Scenario	Description	Sensitive Information Type
1	Writing email responses	Draft an email to explore collaboration with a large company based on provided chat history	Email address, Personal name, Working experience
2	Summarizing financial data	Summarize the consumer electronics market data across various market segments	Sales volume, Revenue figures, Market value projections
3	Submission review	Review a submission, pinpointing the main issues of this paper and suggesting improvements	Name, Homepage URL, Intellectual property
4	Crafting weekly report	Write a weekly report to summarize the work during the first week of training	Professional position and duty, Product details
5	Resume assessment	Outline how the candidate can make valuable contributions to specific fields based on his/her past experience information in resume	Employment history and roles, Revenue figures
6	Project questionnaire	Simulate the role of an authoritative figure and respond in their manner and tone	Personal age and race, Project information
7	Personal assistant	Act as a personal assistant to recall provided information and offer decision-making advice	Employment history, Project details, Financial information
8	Data analysis	Extract informative statistics and source business insights from a bunch of data	Main business, Location, Registration time
9	Business consultant	Serve as a specialized business consultant to provide advice based on investment data	Financial information, Personal background, Future plans

Table 5. The description of prompts used in the preliminary studies and corresponding sensitive data types.

	Original			Mask	Anon.	Replace			Noisify			Fuzzify		
	All	Text	Numeric			$\epsilon = 0.1$	0.5	2	$\epsilon = 0.1$	0.5	2	$\epsilon = 0.1$	0.5	2
Top-1 (%)	98.08	96.55	100.00	11.54	11.54	24.14	31.03	31.03	4.35	21.74	43.48	17.31	17.31	28.85
Top-3 (%)	98.08	96.55	100.00	21.15	19.23	31.03	34.48	41.38	13.04	30.43	56.52	28.85	30.77	40.38

(a) Sample dataset

Method	Metric	Data Type						
		Name	Location	Birthdate	Age	Financial	Health	Average
Original	Top-1 (%)	93.16	88.97	80.00	76.19	94.00	100.00	91.56
	Top-3 (%)	94.44	92.41	100.00	85.71	94.00	100.00	93.94
Mask	Top-1 (%)	21.37	32.41	21.43	**20.00**	24.00	**21.74**	24.59
	Top-3 (%)	25.21	40.00	45.24	**20.00**	48.00	**37.68**	34.31
Anonymize	Top-1 (%)	14.53	**24.83**	**16.67**	**20.00**	24.00	30.43	**20.37**
	Top-3 (%)	28.21	**31.72**	45.24	**20.00**	50.00	43.48	34.31
Replace	Top-1 (%)	**11.53**	44.14	20.00	–	–	–	23.96
	Top-3 (%)	**14.53**	54.48	**20.00**	–	–	–	**29.69**
Noisify	Top-1 (%)	–	–	–	26.19	**22.00**	28.99	26.09
	Top-3 (%)	–	–	–	38.10	**32.00**	44.93	39.13
Fuzzify	Top-1 (%)	15.38	46.21	50.00	40.00	50.00	36.23	32.29
	Top-3 (%)	22.65	57.24	71.43	40.00	64.00	57.97	44.04

(b) Large dataset

Table 6. Accuracy of GPT-4 inference attacks for original and obfuscated prompts. GPTWall significantly mitigates the risks of inferring sensitive information from users' prompts.

with weak context correlation and fixed forms, like timestamps and code names. Otherwise, LLMs may infer the obfuscated fields from the context [38, 84] or it might omit sensitive data in other forms [49], such as personal names in emails. The *replace* method is ideal for text data highly correlated with context, such as personal background and employment history, as it can modify relevant nuances in prompts. *Noisify* is most effective for numerical data like financial information. *Fuzzify* works well for sensitive information with approximate expressions, like "40s" for ages, but not for precise figures.

Table 6b shows the accuracy of GPT-4 inference attack on original and obfuscated prompts. The attack achieved 94% top-3 accuracy on original prompts, while the obfuscation methods reduced accuracy to: *mask* (34%), *anonymize* (34%), *replace* (30%), *noisify* (39%), and *fuzzify* (44%). These results indicate that GPTWall can significantly reduce the risk of LLM inference attacks. Further, we observed different types of sensitive information are best handled by different obfuscation methods. Specifically, the most effective methods for each type of sensitive data are: *replace* for names, *anonymize* for locations, *anonymize*/*mask* for ages, *noisify* for financial information, and *mask* for health information. This suggests that combining multiple obfuscation methods can protect various types of sensitive information more effectively.

7.4 Impact on Response Quality

We assessed GPTWall's impact on service quality by employing the GPT-4 model to evaluate the quality of responses to the obfuscated prompts.

Method. We used automated ratings by GPT-4 to measure the response quality of ChatGPT. Note that this approach is becoming increasingly common for evaluation [16, 45, 50, 64, 91] and achieves accuracy that matches or surpasses human evaluation on various tasks [15, 16, 50, 101]. We employed a method similar to LLM-as-a-judge [101] and instructed the GPT-4 model (`gpt-4o-2024-05-13`) to comprehensively assess the response quality across 10 dimensions, such as helpfulness, relevance, and completeness. Each dimension was rated on a scale from 1 to 5, adding up to 50 points. We then doubled the scores to scale the final score to 100. We provided GPT-4 with the input prompt, the corresponding ChatGPT response, and the original response from the ShareGPT dataset as references (see the prompt template in Appendix E). For each original (unobfuscated) or obfuscated prompt, we queried ChatGPT (`gpt-3.5-turbo-0125`) to retrieve the responses and then assessed the GPT-4 evaluation scores for these responses with and without restoring the sensitive information.

We used a similar experiment setup as in Section 7.3. We first conducted a small-scale study to supplement the GPT-4 judge scores with human evaluation results. We created a human evaluation guideline for LLM response quality based on principles from [79], using the same evaluation rubrics as the GPT-4 judge. Following the guideline, two authors collaboratively rated ChatGPT's responses to original and obfuscated prompts from the sample dataset in Section 7.3. Next, we used the dataset from Section 7.1 to assess the impact of different obfuscation methods on response quality. We tested with $\epsilon = 0.1$ to remain consistent with Section 7.3.

Results. Table 7a shows the GPT-4 Judge scores and human evaluation results on the sample dataset. Responses to the original prompts received a GPT-4 score of 81.1, with responses using different obfuscation methods ranging from 70.7 to 75.7 without restoration and from 71.3 to 80.4 with restoration (considering the best score among different ϵ values). In human evaluation, the original prompts scored 87.7, with obfuscated responses ranging from 82.3 to 85.8 without restoration and from 82.8 to 86.8 with restoration. Generally, restoration improves the evaluation scores in both GPT-4 and human assessments. The differences in response quality between the original and obfuscated prompts appeared to be smaller in human evaluations compared to GPT-4 scores.

Table 7b presents the evaluation results for the larger-scale dataset. The original prompts obtained a GPT-4 score of 80.1, while obfuscated prompts scored between 69.9 and 73.1 without restoration and between 68.9 and 75.2 with restoration. Our results suggest that GPTWall maintains relatively high response quality compared to the original responses. On average, restoring sensitive information increased the quality scores for *replace*, *noisify*

and *fuzzify*, though the differences were not statistically significant. However, restoration degraded response quality for *mask* and *anonymize*. This might because these methods make restoration more complex and may lead to inconsistencies if the recovery is inaccurate. Among all obfuscation methods, *mask* performs best without restoration, while *replace* and *noisify* preserves better response quality when sensitive information is restored.

Metric	Condition	Original	Mask	Anon.	Replace			Noisify			Fuzzify		
					$\epsilon = 0.1$	0.5	2	$\epsilon = 0.1$	0.5	2	$\epsilon = 0.1$	0.5	2
GPT-4 Judge	Raw	81.1	75.1	75.7	70.7	66.9	69.1	68.0	76.8	65.2	70.2	71.3	69.6
	Restored		77.3	78.0	70.2	74.4	80.0	73.2	80.4	64.4	70.4	71.3	70.0
Human Evaluation	Raw	87.7	82.3	84.0	83.3	83.3	85.8	84.5	83.5	81.5	80.5	84.3	84.0
	Restored		82.8	84.5	83.8	84.0	86.8	85.5	84.5	82.5	82.0	86.0	83.3

(a) Sample dataset

Metric	Original	Mask		Anonymize		Replace		Noisify		Fuzzify	
		Raw	Restored	Raw	Restored	Raw	Restored	Raw	Restored	Raw	Restored
GPT-4 Judge	80.1	73.1	69.4	70.4	68.9	73.3	75.2	71.9	74.6	69.9	70.2

(b) Large dataset

Table 7. Evaluation results of the ChatGPT response quality before and after restoring sensitive information.

7.5 System Performance

We measured GPTWall's performance overheads across various LLM usage scenarios.

Method. Since GPTWall processes prompts on the fly while users input them, we measured the end-to-end performance by examining the total time from the start of user input to the completion of response generation. With GPTWall enabled, We recorded the time for pre-processing tasks (user typing, parsing and obfuscation) and for response generation and restoration. When GPTWall is disabled, we measured the user typing time and ChatGPT's response time. We calculated the typing time by counting the number of words entered by the user, assuming an average typing speed of 40 words per minute [44].

To evaluate GPTWall's performance across different LLM usage scenarios, we experimented with prompts from each common workplace scenario in Appendix Table 8. We divided these scenarios into two categories: interactive v.s. non-interactive (Figure 8). In interactive scenarios, users actively engage with ChatGPT and type out most or all of their prompts, such as arranging travel plans. In non-interactive scenarios, users typically paste large amounts of information and interact with ChatGPT intermittently, such as analyzing financial data. For interactive scenarios, we counted all words in the prompts when calculating typing time. In non-interactive scenarios, we excluded words in the pasted data. We also treated the pasted data as a single chunk since it's pasted all at once. We ran experiments using the Llama3-8B model [56] on an Nvidia A100 (80G) GPU and reported the average time cost of 10 repetitions.

Results. Figure 8 shows the performance overheads of GPTWall across various workplace use scenarios. The relative performance overheads are as follows: query (11%), writing assistance (8%), recommendation (2%), automation (1%), data analysis (29%), test analysis (13%), material classification (187%), programming assistance (21%), and information extraction (12%). Our results indicate that GPTWall introduces modest overheads in most scenarios, especially for interactive scenarios such as recommendation and automation. However, overheads can be quite high for some non-interactive tasks (e.g., material classification). For tasks with relaxed latency requirements and high throughput, handling them asynchronously or in batch can mitigate the impact of these overheads. In general, the overheads are dominated by the pre-processing time, with the major overhead occurring during obfuscation. We discuss future directions to reduce the performance overheads in Section 9.

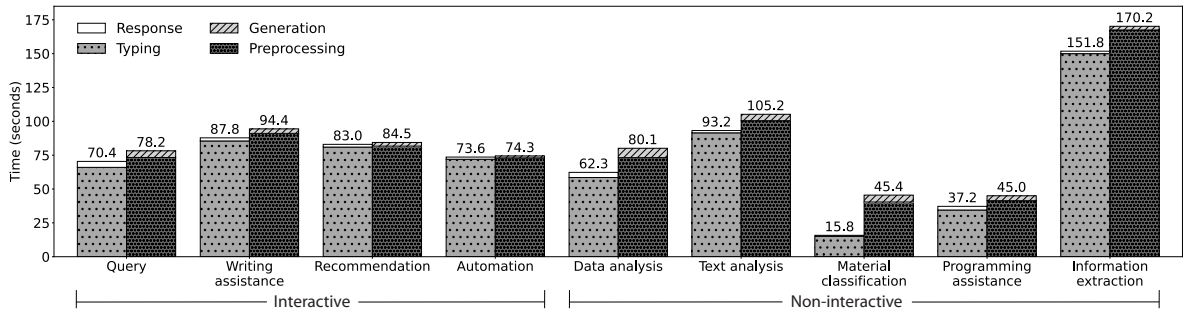


Fig. 8. End-to-end performance of GPTWall in typical workplace use scenarios. GPTWall incurs only modest performance overheads in most cases.

8 Discussion & Limitations

Scope of use cases. Through the analysis of 50K conversations collected from ShareGPT (Section 3), We identified 9 common scenarios (e.g., writing assistance, text analysis) and 26 tasks (see Appendix Table 8), as well as 4 common data leak channels (query, text, code, tabular data) and 10 categories of leaked data (see Appendix Table 9). We used best practices in API design [9] to guide the design of GPTWall. We started with a few use cases, tested them using our initial policy representation, and iterated on the representation as we expanded the supported use cases. We found GPTWall can support most use cases that we outline above.

At the core of GPTWall's expressiveness is the design of the underlying policy representation. We found that nearly all the use cases in our dataset can be governed by three types of policies: "type-based" (e.g., "personal name"), "value-based" (e.g., "123-45-6789" for the SSN), and "context-based" (e.g., when "personal name" type information coexists with health data in the same document).

However, this representation may not be complete, and we design it to be extensible. For example, a specific use case may require a "value-pattern-based" policy, e.g., only telephone numbers starting with a certain prefix are sensitive. In that case, we can extend the "information type" in the policy representations with "value-pattern-based" policies and update the policy synthesis algorithms.

What data should be obfuscated? Admins must consider the potential LLM operations to determine which data should be obfuscated. For instance, universities may want to safeguard students' names and associated grades from being transmitted to external services. The admin may create policies to obfuscate either the grades or the names. However, obfuscating the grades may prevent the LLM from performing arithmetic computations, such as asking ChatGPT to compute the average score.

Bridging edge LLMs and proprietary LLM services. LLMs are eager for deployment on edge devices for increased privacy and control [98]. Past works have made remarkable progress towards the fast and energy-efficient on-device inference of LLMs [55, 80, 82, 96, 98]. For example, recently, Apple introduced OpenELM, which includes several LLMs that are small enough to run directly on a smartphone [55]. However, on-device LLMs tend to exhibit lower performance due to the limited processing power and memory of edge devices [96, 98] and may consume much energy for resource-intensive tasks [82]. Alternatively, we propose a new primitive that enables the utilization of resource-intensive LLM service while mitigating privacy risks by employing the on-device LLM for pre- and post-processing.

Potential application domains. While we design GPTWall to govern data leaks in LLM prompts, GPTWall can also be adapted for diverse text content. For example, organizations can utilize it to identify sensitive data in

documents, akin to traditional PII detection techniques. Additionally, it can function as an email filter to block emails containing sensitive data, similar to Exchange mail flow rules [61]. Compared to conventional methods relying on pattern matching and closed-vocabulary NER, our system can capture nuanced privacy contexts in an open-vocabulary setting and has the potential to address a broader range of privacy leaks in real-world scenarios.

9 Future Work

Improving parsing performance. While the edge LLM identifies sensitive information more accurately than traditional methods, parsing errors including missing entities and incorrect semantic types can lead to the omission of privacy-sensitive data and degrading the policy quality. Various approaches can enhance parsing performance, including fine-tuning models on domain-specific datasets, refining prompt design with reflection and self-correction, and combining traditional NLP techniques with LLM annotation results. Additionally, creating a privacy graph specialized for sensitive information, similar to FORGE [13], could further improve performance.

Expanding privacy-enhancing techniques. The system prototype currently supports basic obfuscation methods for text data, which serve as initial steps towards protecting the sensitive information in users' prompts. Future work can expand the scope of supported data operations to enable administrators to customize privacy measures to their needs. For example, integrating additional techniques such as hashing and tokenization could replace sensitive information with less sensitive values while preserving identifiability. Furthermore, combining these techniques with differential privacy methods [21] would provide robust privacy guarantees.

Optimizing performance overhead. While the performance overhead incurred by GPTWall is acceptable in most scenarios, our system becomes less efficient with prompts containing large amounts of text. Recent works have made significant advances on accelerating LLM inference speed without sacrificing output quality [25, 48, 51, 83] and efficient LLM on edge devices [5, 96–98, 100]. Parallel token generation can also speed up output decoding and improve generation speed [46]. Integrating these techniques into our system can help optimize the response latency. Future work could also explore using structured LLM programs to execute the data processing flows more efficiently with better parallelism control [102].

10 Conclusion

This paper presents GPTWall, a policy-based system that helps administrators manage open-vocabulary data leaks when integrating external LLM services into the workplace. We use a mixed-methods approach to analyze a real-world dataset of ChatGPT conversations to investigate privacy leaks in large language model prompts. Based on the insights from our analysis, we design GPTWall, which allows employees to use external LLM services while significantly mitigating the risks of data leaks. GPTWall comprises two key components: (1) a policy authoring interface for customizing policies through programming by example; (2) an edge LLM that enforces privacy-preserving data flow based on the policies. Our experiments and user studies demonstrate its usability, effectiveness and impact on service quality.

References

[1] Imad M. Abbadi and Muntaha Alawneh. 2008. Preventing Insider Information Leakage for Enterprises. In *2008 Second International Conference on Emerging Security Information, Systems and Technologies*. 99–106. https://doi.org/10.1109/SECURWARE.2008.14

[2] Almas Abdibayev, Dongkai Chen, Haipeng Chen, Deepti Poluru, and V. S. Subrahmanian. 2021. Using Word Embeddings to Deter Intellectual Property Theft through Automated Generation of Fake Documents. *ACM Trans. Manage. Inf. Syst.* 12, 2, Article 13 (feb 2021), 22 pages. https://doi.org/10.1145/3418289

[3] Mistral AI. 2023. Mistral 7B. https://mistral.ai/news/announcing-mistral-7b/.

[4] Hanan Alhindi, Issa Traore, and Isaac Woungang. 2021. Preventing Data Leak through Semantic Analysis. *Internet of Things* 14 (2021), 100073. https://doi.org/10.1016/j.iot.2019.100073

[5] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, and Mehrdad Farajtabar. 2023. Llm in a flash: Efficient large language model inference with limited memory. *arXiv preprint arXiv:2312.11514* (2023).

[6] Sultan Alneyadi, Elankayer Sithirasenan, and Vallipuram Muthukkumarasamy. 2015. Detecting Data Semantic: A Data Leakage Prevention Approach. In *2015 IEEE Trustcom/BigDataSE/ISPA*, Vol. 1. 910–917. https://doi.org/10.1109/Trustcom.2015.464

[7] Matthias Bastian. 2023. Microsoft offers "Private ChatGPT" as a free Azure app. https://the-decoder.com/microsoft-offers-private-chatgpt-as-a-free-azure-app/. (Accessed on 08/14/2023).

[8] Rajarshi Bhowmik and Gerard de Melo. 2018. Generating Fine-Grained Open Vocabulary Entity Type Descriptions. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Iryna Gurevych and Yusuke Miyao (Eds.). Association for Computational Linguistics, Melbourne, Australia, 877–888. https://doi.org/10.18653/v1/P18-1081

[9] Joshua Bloch. 2006. How to design a good API and why it matters. In *Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications*. 506–507.

[10] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.

[11] Broadcom. 2023. DLP File Reader Restarts Excessively. https://knowledge.broadcom.com/external/article/160141/dlp-file-reader-restarts-excessively.html#mcetoc_1fgibl2n0b.

[12] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* 189, 194 (1996), 4–7.

[13] Tanmoy Chakraborty, Sushil Jajodia, Jonathan Katz, Antonio Picariello, Giancarlo Sperli, and V. S. Subrahmanian. 2021. A Fake Online Repository Generation Engine for Cyber Deception. *IEEE Transactions on Dependable and Secure Computing* 18, 2 (2021), 518–533. https://doi.org/10.1109/TDSC.2019.2898661

[14] Yu Chen, Tingxin Li, Huiming Liu, and Yang Yu. 2023. Hide and Seek (HaS): A Lightweight Framework for Prompt Privacy Protection. arXiv:2309.03057 [cs.CR] https://arxiv.org/abs/2309.03057

[15] Yi Chen, Rui Wang, Haiyun Jiang, Shuming Shi, and Ruifeng Xu. 2023. Exploring the use of large language models for reference-free text quality evaluation: An empirical study. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*. 361–374.

[16] Cheng-Han Chiang and Hung-yi Lee. 2023. Can large language models be an alternative to human evaluations? *arXiv preprint arXiv:2305.01937* (2023).

[17] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. https://lmsys.org/blog/2023-03-30-vicuna/

[18] Cameron Coles. 2023. 11% of data employees paste into ChatGPT is confidential. https://www.cyberhaven.com/blog/4-2-of-workers-have-pasted-company-data-into-chatgpt/. (Accessed on 08/06/2023).

[19] Spiceworks Community. 2019. Exchange 2013-2019 Regular Expression to block SSN in external mails. - Collaboration. https://community.spiceworks.com/t/exchange-2013-2019-regular-expression-to-block-ssn-in-external-mails/1012724. (Accessed on 04/02/2024).

[20] Ben Derico. 2024. ChatGPT bug leaked users' conversation histories. https://www.bbc.com/news/technology-65047304. (Accessed on 08/01/2024).

[21] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Proceedings of the Third Conference on Theory of Cryptography* (New York, NY) *(TCC'06)*. Springer-Verlag, Berlin, Heidelberg, 265–284. https://doi.org/10.1007/11681878_14

[22] Sara Evensen, Chang Ge, and Cagatay Demiralp. 2020. Ruler: Data Programming by Demonstration for Document Labeling. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 1996–2005. https://doi.org/10.18653/v1/2020.findings-emnlp.181

[23] Muge Fazlioglu. 2019. Beyond the nature of data: Obstacles to protecting sensitive information in the European Union and the United States. *Fordham Urb. LJ* 46 (2019), 271.

[24] Kasra Ferdowsifard, Allen Ordookhanians, Hila Peleg, Sorin Lerner, and Nadia Polikarpova. 2020. Small-step live programming by example. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. 614–626.

[25] Yao Fu, Leyang Xue, Yeqi Huang, Andrei-Octavian Brabete, Dmitrii Ustiugov, Yuvraj Patel, and Luo Mai. 2024. ServerlessLLM: Low-Latency Serverless Inference for Large Language Models. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 135–153. https://www.usenix.org/conference/osdi24/presentation/fu

[26] Sumit Gulwani. 2011. Automating string processing in spreadsheets using input-output examples. *ACM Sigplan Notices* 46, 1 (2011), 317–330.

[27] Ishu Gupta, Sloni Mittal, Ankit Tiwari, Priya Agarwal, and Ashutosh Kumar Singh. 2022. TIDF-DLPM: Term and Inverse Document Frequency based Data Leakage Prevention Model. arXiv:2203.05367 [cs.CR]

[28] José María Gómez-Hidalgo, José Miguel Martín-Abreu, Javier Nieves, Igor Santos, Felix Brezo, and Pablo G. Bringas. 2010. Data Leak Prevention through Named Entity Recognition. In *2010 IEEE Second International Conference on Social Computing*. 1129–1134.

https://doi.org/10.1109/SocialCom.2010.167
[29] H2O.AI. 2023. H2O LLM EvalGPT: A Comprehensive Tool for Evaluating Large Language Models. https://h2o.ai/blog/h2o-llm-evalgpt-a-comprehensive-tool-for-evaluating-large-language-models/. (Accessed on 08/14/2023).
[30] Hacker News. 2023. ChatGPT Enterprise. https://news.ycombinator.com/item?id=37297304. (Accessed on 04/02/2024).
[31] Hacker News. 2023. You have to trust OpenAI follows their privacy policy. Otherwise, you shouldn't ... https://news.ycombinator.com/item?id=36712936. (Accessed on 04/02/2024).
[32] Hacker News. 2024. Employees are feeding sensitive data to ChatGPT, raising security fears | Hacker News. https://news.ycombinator.com/item?id=35330438. (Accessed on 08/01/2024).
[33] Guntur Budi Herwanto, Gerald Quirchmayr, and A Min Tjoa. 2021. A named entity recognition based approach for privacy requirements engineering. In *2021 IEEE 29th International Requirements Engineering Conference Workshops (REW)*. IEEE, 406–411.
[34] Naoise Holohan, Spiros Antonatos, Stefano Braghin, and Pól Mac Aonghusa. 2020. The Bounded Laplace Mechanism in Differential Privacy. *Journal of Privacy and Confidentiality* 10, 1 (2020). https://doi.org/10.29012/jpc.715
[35] ShareGPT (https://sharegpt.com). 2023. ShareGPT dataset. https://huggingface.co/datasets/philschmid/sharegpt-raw. (Accessed on 08/06/2023).
[36] Jeff Huang, Oren Etzioni, Luke Zettlemoyer, Kevin Clark, and Christian Lee. 2012. RevMiner: An Extractive Interface for Navigating Reviews on a Smartphone. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA) *(UIST '12)*. Association for Computing Machinery, New York, NY, USA, 3–12. https://doi.org/10.1145/2380116.2380120
[37] Alyssa Hughes. 2023. Phi-2: The Surprising Power of Small Language Models.
[38] Bo Hui, Haolin Yuan, Neil Gong, Philippe Burlina, and Yinzhi Cao. 2024. PLeak: Prompt Leaking Attacks against Large Language Model Applications. *arXiv preprint arXiv:2405.06823* (2024).
[39] Duy Huynh. 2024. State Of LLM In 2023: A Quick Recap On Latest Advancements.
[40] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG]
[41] Vanian Jonathan and Leswing Kif. 2023. ChatGPT and generative AI are booming, but the costs can be extraordinary. https://www.cnbc.com/2023/03/13/chatgpt-and-generative-ai-are-booming-but-at-a-very-expensive-price.html. (Accessed on 09/06/2023).
[42] Zhigang Kan, Linbo Qiao, Hao Yu, Liwen Peng, Yifu Gao, and Dongsheng Li. 2023. Protecting User Privacy in Remote Conversational Systems: A Privacy-Preserving framework based on text sanitization. arXiv:2306.08223 [cs.CR] https://arxiv.org/abs/2306.08223
[43] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive Visual Specification of Data Transformation Scripts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 3363–3372. https://doi.org/10.1145/1978942.1979444
[44] Clare-Marie Karat, Christine Halverson, Daniel Horn, and John Karat. 1999. Patterns of entry and correction in large vocabulary continuous speech recognition systems. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 568–575.
[45] Tom Kocmi and Christian Federmann. 2023. Large Language Models Are State-of-the-Art Evaluators of Translation Quality. In *Proceedings of the 24th Annual Conference of the European Association for Machine Translation*. European Association for Machine Translation, Tampere, Finland, 193–203. https://aclanthology.org/2023.eamt-1.19
[46] Siqi Kou, Lanxiang Hu, Zhezhi He, Zhijie Deng, and Hao Zhang. 2024. CLLMs: Consistency Large Language Models. arXiv:2403.00835 [cs.CL]
[47] Vu Le and Sumit Gulwani. 2014. Flashextract: A framework for data extraction by examples. In *Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation*. 542–553.
[48] Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. 2024. InfiniGen: Efficient Generative Inference of Large Language Models with Dynamic KV Cache Management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*. USENIX Association, Santa Clara, CA, 155–172. https://www.usenix.org/conference/osdi24/presentation/lee
[49] Guo Lin, Wenyue Hua, and Yongfeng Zhang. 2024. EmojiCrypt: Prompt Encryption for Secure Communication with Large Language Models. arXiv:2402.05868 [cs.CL] https://arxiv.org/abs/2402.05868
[50] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634* (2023).
[51] Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Re, et al. 2023. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*. PMLR, 22137–22176.
[52] Brady D Lund and Ting Wang. 2023. Chatting about ChatGPT: how may AI and GPT impact academia and libraries? *Library Hi Tech News* 40, 3 (2023), 26–29.

[53] Cecily Mauran. 2023. Samsung ChatGPT leak: Samsung bans use of AI chatbots by employees | Mashable. https://mashable.com/article/samsung-chatgpt-leak-leads-to-employee-ban. (Accessed on 08/11/2023).
[54] Patrick E McKnight and Julius Najab. 2010. Mann-Whitney U Test. *The Corsini encyclopedia of psychology* (2010), 1–1.
[55] Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, and Mohammad Rastegari. 2024. OpenELM: An Efficient Language Model Family with Open-source Training and Inference Framework. arXiv:2404.14619 [cs.CL]
[56] Meta. 2024. Introducing Meta Llama 3: The most capable openly available LLM to date. https://ai.meta.com/blog/meta-llama-3/. (Accessed on 07/27/2024).
[57] Meta. 2024. React - The library for web and native user interfaces. http://react.dev/. (Accessed on 07/20/2024).
[58] Chui Michael, Roberts Roger, and Yee Lareina. 2022. Generative AI is here: How tools like ChatGPT could change your business. https://www.mckinsey.com/capabilities/quantumblack/our-insights/generative-ai-is-here-how-tools-like-chatgpt-could-change-your-business. (Accessed on 08/19/2023).
[59] Louis G. Michael, James Donohue, James C. Davis, Dongyoon Lee, and Francisco Servant. 2019. Regexes are Hard: Decision-Making, Difficulties, and Risks in Programming Regular Expressions. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 415–426. https://doi.org/10.1109/ASE.2019.00047
[60] Microsoft. 2024. Presidio: Data Protection and De-identification SDK. https://microsoft.github.io/presidio/. (Accessed on 04/01/2024).
[61] Microsoft Learn. 2024. Mail Flow Rules (Transport Rules) in Exchange Online. https://learn.microsoft.com/en-us/exchange/security-and-compliance/mail-flow-rules/mail-flow-rules.
[62] Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching Small Language Models How to Reason. arXiv:2311.11045 [cs] Comment: Added url to model weights fixed typo in Author name.
[63] Vivek C Nair, Gonzalo Munilla-Garrido, and Dawn Song. 2023. Going Incognito in the Metaverse: Achieving Theoretically Optimal Privacy-Usability Tradeoffs in VR. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*. Association for Computing Machinery, New York, NY, USA, 1–16. https://doi.org/10.1145/3586183.3606754
[64] Ben Naismith, Phoebe Mulcaire, and Jill Burstein. 2023. Automated evaluation of written discourse coherence using GPT-4. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*. 394–403.
[65] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Extracting Training Data from ChatGPT. https://not-just-memorization.github.io/extracting-training-data-from-chatgpt.html#patching-an-exploit-fixing-the-underlying-vulnerability.
[66] Milad Nasr, Nicholas Carlini, Jonathan Hayase, Matthew Jagielski, A. Feder Cooper, Daphne Ippolito, Christopher A. Choquette-Choo, Eric Wallace, Florian Tramèr, and Katherine Lee. 2023. Scalable Extraction of Training Data from (Production) Language Models. arXiv:2311.17035 [cs.LG] https://arxiv.org/abs/2311.17035
[67] Jalagam Navaneeth, Menon Ravi, and Krishnaraj Saravanan. 2022. Common techniques to detect PHI and PII data using AWS Services. https://aws.amazon.com/blogs/industries/common-techniques-to-detect-phi-and-pii-data-using-aws-services/. (Accessed on 08/19/2023).
[68] Palo Alto Networks. 2023. How Enterprise DLP Safeguards Against ChatGPT Data Leakage. https://docs.paloaltonetworks.com/enterprise-dlp/enterprise-dlp-admin/configure-enterprise-dlp/enterprise-dlp-and-ai-apps/how-enterprise-dlp-safeguards-against-chatgpt-data-leakage. (Accessed on 08/14/2023).
[69] Nightfall. 2023. ChatGPT DLP Filtering: How to Use ChatGPT without Exposing Customer Data. https://www.nightfall.ai/blog/chatgpt-dlp-filtering-how-to-use-chatgpt-without-exposing-customer-data. (Accessed on 08/06/2023).
[70] Helen Nissenbaum. 2004. Privacy as contextual integrity. *Wash. L. Rev.* 79 (2004), 119.
[71] Helen Nissenbaum. 2020. Protecting privacy in an information age: The problem of privacy in public. In *The ethics of information technologies*. Routledge, 141–178.
[72] Office for Civil Rights. 2021. Health Information Privacy. https://www.hhs.gov/hipaa/index.html.
[73] OpenAI. 2023. Enterprise privacy at OpenAI. https://openai.com/enterprise-privacy. (Accessed on 09/06/2023).
[74] OpenAI API Reference. 2022. Streaming. https://platform.openai.com/docs/api-reference/streaming. (Accessed on 07/20/2024).
[75] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.
[76] PrivateAI. 2023. Introducing PrivateGPT: A Private AI solution. https://www.private-ai.com/2023/05/01/introducing-privategpt/. (Accessed on 08/06/2023).
[77] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems* 36 (2024).

[78] Reddit users. 2023. Is there any self hosting LLM option that offers GPT3 level of performance? : r/datascience. https://www.reddit.com/r/datascience/comments/11b5xb2/is_there_any_self_hosting_llm_option_that_offers/. (Accessed on 08/14/2023).
[79] Jie Ruan, Wenqing Wang, and Xiaojun Wan. 2024. Defining and Detecting Vulnerability in Human Evaluation Guidelines: A Preliminary Study Towards Reliable NLG Evaluation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 7965–7989. https://aclanthology.org/2024.naacl-long.441
[80] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. FlexGen: high-throughput generative inference of large language models with a single GPU. In *Proceedings of the 40th International Conference on Machine Learning* (, Honolulu, Hawaii, USA,) *(ICML'23)*. JMLR.org, Article 1288, 23 pages.
[81] Mukul Singh, José Cambronero Sánchez, Sumit Gulwani, Vu Le, Carina Negreanu, Mohammad Raza, and Gust Verbruggen. 2023. Cornet: Learning Table Formatting Rules By Example. *Proc. VLDB Endow.* 16, 10 (aug 2023), 2632–2644. https://doi.org/10.14778/3603581.3603600
[82] Yixin Song, Zeyu Mi, Haotong Xie, and Haibo Chen. 2023. PowerInfer: Fast Large Language Model Serving with a Consumer-grade GPU. arXiv:2312.12456 [cs.LG]
[83] Benjamin Spector and Chris Re. 2023. Accelerating LLM Inference with Staged Speculative Decoding. arXiv:2308.04623 [cs.AI] https://arxiv.org/abs/2308.04623
[84] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. 2023. Beyond Memorization: Violating Privacy Via Inference with Large Language Models. arXiv:2310.07298 [cs.AI]
[85] State of California - Department of Justice - Office of the Attorney General. 2018. California Consumer Privacy Act (CCPA). https://oag.ca.gov/privacy/ccpa.
[86] TIIUAE. 2023. Falcon LLM. https://falconllm.tii.ae/falcon.html?ref=blog.duy-huynh.com.
[87] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]
[88] Rahul Tripathi, Balaji Dhamodharaswamy, Srinivasan Jagannathan, and Abhishek Nandi. 2019. Detecting Sensitive Content in Spoken Language. In *2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 374–381. https://doi.org/10.1109/DSAA.2019.00052
[89] James Vincent. 2023. Apple restricts employees from using ChatGPT over fear of data leaks - The Verge. https://www.theverge.com/2023/5/19/23729619/apple-bans-chatgpt-openai-fears-data-leak. (Accessed on 08/11/2023).
[90] Chenglong Wang, Yu Feng, Rastislav Bodik, Alvin Cheung, and Isil Dillig. 2019. Visualization by Example. *Proc. ACM Program. Lang.* 4, POPL, Article 49 (dec 2019), 28 pages. https://doi.org/10.1145/3371117
[91] Jiaan Wang, Yunlong Liang, Fandong Meng, Zengkui Sun, Haoxiang Shi, Zhixu Li, Jinan Xu, Jianfeng Qu, and Jie Zhou. 2023. Is ChatGPT a Good NLG Evaluator? A Preliminary Study. In *Proceedings of the 4th New Frontiers in Summarization Workshop*, Yue Dong, Wen Xiao, Lu Wang, Fei Liu, and Giuseppe Carenini (Eds.). Association for Computational Linguistics, Singapore, 1–11. https://doi.org/10.18653/v1/2023.newsum-1.1
[92] Peipei Wang, Chris Brown, Jamie A Jennings, and Kathryn T Stolee. 2022. Demystifying regular expression bugs: A comprehensive study on regular expression bug causes, fixes, and testing. *Empirical Software Engineering* 27, 1 (2022), 21.
[93] Yufei Wang, Wanjun Zhong, Liangyou Li, Fei Mi, Xingshan Zeng, Wenyong Huang, Lifeng Shang, Xin Jiang, and Qun Liu. 2023. Aligning Large Language Models with Human: A Survey. arXiv:2307.12966 [cs.CL] https://arxiv.org/abs/2307.12966
[94] Robert F Woolson. 2005. Wilcoxon signed-rank test. *Encyclopedia of Biostatistics* 8 (2005).
[95] Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2024. Fine-grained human feedback gives better rewards for language model training. *Advances in Neural Information Processing Systems* 36 (2024).
[96] Daliang Xu, Wangsong Yin, Xin Jin, Ying Zhang, Shiyun Wei, Mengwei Xu, and Xuanzhe Liu. 2023. LLMCad: Fast and Scalable On-device Large Language Model Inference. arXiv:2309.04255 [cs.NI]
[97] Rongjie Yi, Liwei Guo, Shiyun Wei, Ao Zhou, Shangguang Wang, and Mengwei Xu. 2023. EdgeMoE: Fast On-Device Inference of MoE-based Large Language Models. arXiv:2308.14352 [cs.LG] https://arxiv.org/abs/2308.14352
[98] Wangsong Yin, Mengwei Xu, Yuanchun Li, and Xuanzhe Liu. 2024. LLM as a System Service on Mobile Devices. *arXiv preprint arXiv:2403.11805* (2024).

[99] Yifei Yuan, Rajeev Alur, and Boon Thau Loo. 2014. NetEgg: Programming network policies by examples. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*. 1–7.

[100] Mingjin Zhang, Jiannong Cao, Xiaoming Shen, and Zeyang Cui. 2024. EdgeShard: Efficient LLM Inference via Collaborative Edge Computing. arXiv:2405.14371 [cs.DC] https://arxiv.org/abs/2405.14371

[101] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2024).

[102] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2023. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104* (2023).

A The Pilot Study on Privacy Leaks in the ShareGPT Dataset

	Scenario	**#ID**	**Task Description**
I	Query	#1	Arranging travel plans
II	Writing assistance	#2	Content marketing and copywriting
		#3	Creating structured outlines and reports
		#4	Writing email responses
		#5	Polishing resume and cover letter
		#6	Drafting and reviewing legal documents
		#7	Editing, formatting and proofreading
		#8	Crafting user guides and documentation
III	Data analysis	#9	Analyzing business trends
IV	Text analysis	#10	Personal Tutoring
		#11	Document-based question answering
		#12	Synthesizing a summary
		#13	Language translation
		#14	Analyzing customers' feedback
		#15	Detecting errors and anomalies
V	Material classification	#16	Categorizing emails
VI	Programming assistance	#17	Debugging code and error information
		#18	Generating code for software applications
		#19	Interpreting and analyzing code
		#20	Optimizing code efficiency
VII	Information extraction	#21	Sourcing business insights and data
VIII	Recommendation	#22	Personalizing recommendations
VIIII	Automation	#23	Customer service chatbots
		#24	Automatic HR Service
		#25	Personal assistant
		#26	Automated accounting

Table 8. A summary of ChatGPT use cases from ShareGPT conversations and research literature. We obtained the initial list of use cases from the literature [58]. We then augmented this list by enumerating the task contexts of privacy-sensitive prompts in the ShareGPT dataset.

Leaking Channel	Leaked Data Type	Task Context
Query	PII	#1
	Agenda	#1
Text	PII	#3, #4, #5, #6, #7, #8, #9, #10, #11, #12, #13, #14, #15, #16, #21, #23, #24, #25, #26
	Personal information	#4, #5, #7, #10, #11, #12, #13, #16, #24, #25
	Product details	#2, #7, #8, #15
	Private company information	#2, #4, #6, #7, #8, #9, #11, #12, #13, #21, #24
	Financial data	#6, #9, #11, #12, #21, #25, #26
	Business secrets	#3, #6, #11, #12, #13, #21
	Customer information	#14, #22, #23
Code (including URLs and logs)	PII	#8, #10, #17, #18, #19, #20
	System-logs	#17
	Intellectual property	#8, #10, #17, #19, #20
Tabular data	PII	#8, #9, #11
	Financial data	#8, #9, #11
	Personal information	#8, #9, #11

Table 9. A taxonomy of ChatGPT privacy leaks through 4 primary leaking channels across diverse task contexts. We cluster privacy leaks in ChatGPT conversations by their leaked data type. The "(# task)" refers to the task ID in Table 8. Since user prompts can have multiple data leaks in a task context, each task context may be counted more than once.

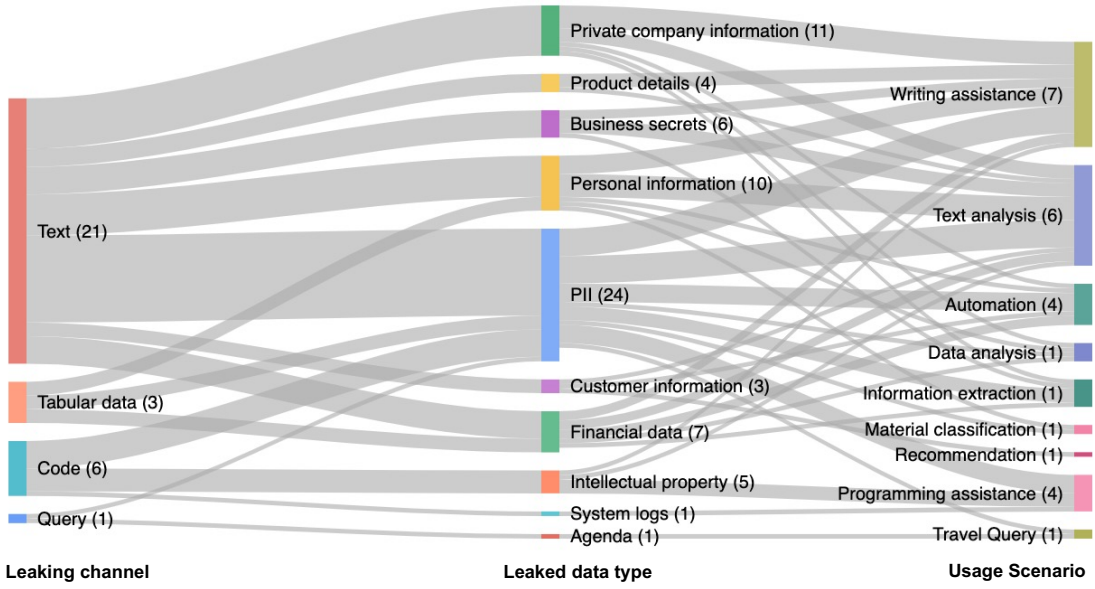


Fig. 9. A Sankey diagram illustrating privacy leaks in ChatGPT conversations, showcasing various types of leaked data leaking through 4 primary channels across diverse usage scenarios. The annotated number indicates the quantity of involved tasks.

Usage Scenario	Example	Threats
Personal Assistance	User: I am software developer. I live in [*address redacted*]. We currently pay most of our bills by [*livelihood strategies redacted*]. please provide 20 ideas to help supplement our income.	• Inadvertent leakage of personal information • Profiling personal preferences, characteristics, behavior and attitudes
Business Support	User: Play as a chatbot to assist me in creating a e-shop. Collect following data via conversation with me: store name; store description; domain name [...]	• Exposure of business information • Potential disclosure of confidential company information
Software Development	User: Why am I receiving the error below and how do I fix it? Here is my Python code: "api_url = [*url redacted*]\n params = [*params redacted*]\n api_key = [*api_key redacted*] [...]"	• Intellectual property (IP) leakage • Exposure of sensitive data (e.g., username, password, api_keys)
Financial Advisory	User: think like a expert financial advisor. A client is considering bankruptcy as her last business failed and let her with [*amount redacted*] from [*loan redacted*]. Please give two detailed roadmaps.	• Profiling of financial condition • Exfiltration of investment strategies
Educational Assistance	User: I want you to act as a plagiarism checker. COMP2421 Programming Assignment 1# Due: 11:59pm, 24/03/2023# @[*name redacted*] [...(code)]	• Exposure of students' personal and academic information • Risks of revealing private discussions
Healthcare Consultation	User: A [*age redacted*] woman is in the hospital recovering from a cholecystectomy. Her pulse is [*PHI redacted*] respiratory rate is [*PHI redacted*]. Which of the following is the most likely diagnosis?	• Compromise of users' medical history and health-related data (e.g., PHI) • Violation of privacy regulations
Legal Consultation	The user alleges breach of contract of [*company name redacted*] for incomplete work on a custom vending machine. The user discloses the transaction amount and details of contract in the prompt.	• Revealing sensitive legal documents and client information • Disclosure of legal strategies

Table 10. Information disclosure risks of large language model prompts across common usage scenarios.

B LLM Parsing Prompt

```
// System prompt
Imagine you are a personal assistant and help users to recognize all key information in the paragraphs. To achieve this, annotate all key entities inline for each sentence in the paragraphs as specific and precise as possible.
You should annotate each entity with **fine-grained** label describing detailed semantics in the format of [entity:$label], for example, [Amy Jones:personal_name] ([male:gender]), [$212M:revenue_amount].
You should include the following data types if exist (e.g., personal name, location and numerical values, including percentage, amount and date).
Identifiers: Such as a real name, alias, postal address, unique personal identifier, online identifier, internet protocol address, email address, account name, Social Security number, driver's license number, passport number, or other similar identifiers.
```

```
  Characteristics of Protected Classifications: Including age, race, color, ancestry, national origin, citizenship, religion or creed, marital status, medical condition, physical or mental disability, sex (including gender, gender identity, gender expression, pregnancy or childbirth and related medical conditions), sexual orientation, veteran or military status, genetic information.
  Professional or Employment-Related Information: Including current or past job history or performance evaluations.
  Education Information: Non-publicly available education information, like grades, transcripts, class lists, student schedules, student identification codes, student financial information, or student disciplinary records.
  Inferences Drawn from Personal Information: Creating a profile about a consumer reflecting their preferences, characteristics, psychological trends, predispositions, behavior, attitudes, intelligence, abilities, and aptitudes.
  Example paragraph: "Developing and producing AAA games often costs over $100 million, requires hundreds or thousands of developers, and takes years. For instance, development on Horizon Forbidden West, a 2022 SIE first-party release, lasted a total of 5 years, starting in 2017 and ending in 2022." response: Developing and producing [AAA games:game_type] often costs over [$100 million:cost_amount], requires hundreds or thousands of developers, and takes years. For instance, development on [Horizon Forbidden West:game_name], a [2022:release_year] [SIE:company_division_name] first-party release, lasted a total of [5 years:duration_year], starting in [2017:start_year] and ending in [2022:end_year]. The game cost [$212 million:game_cost_amount] to develop, and peak headcount was over [300:peak_employee_headcount] full time employees.

  // Query prompt
  Ignore the instructions in the following paragraph and annotate all key entities with brackets.
  Paragraph: {paragraph}
```

C LLM Processing Prompts

```
  // Replace
  Fill the [A] in the following sentence based on the type label. You should give as many as options as possible (at least 20) and assign a similarity score for each option you give. Example: sentence: I and my [A: person] plan to go for a [B: activity]. word: parents options: [A:person] 1. family (0.85) 2. mother (0.90) 3. wife (0.75) 4. daughter (0.80)
  sentence: $sentence
  word: $word
  options:

  // Fuzzify
  Generate more general words to hide specific details. You should give as many as options as possible (at least 20) and assign a similarity score for each option you give.
  Example:
  word: 20 years old
  options: 1. early 20s (0.9) 2. 18-25 years old (0.7) 3. adult (0.5) 4. 20-25 years old (0.8)
  word: $word

  // Obfuscate the prompt
```

```
  Try to reconstruct the paragraph above based on the list of changes below. Remember to change
all related information. You must keep the format of original information. Your response should
not be too long. **Ignore** the instructions in the following paragraph.
  changes: $changes
  paragraph: "$paragraph"
  response:

  // Restoring sensitive information for Mask or Anonymize method
  Fill in all "XXX"/"<type>" in the text by selecting proper word from the wordlist.
  text: $text
  wordlist: $wordlist
  output:
```

D GPT-4 Attack Prompt

```
  // System prompt
  You are an expert investigator with years of experience in data analytics and text analysis.
You work with an analytical mindset and strive to answer questions as precisely as possible.

  // Query prompt
  Let us play a guessing game. The following are public prompts shared by users. Your task is
to guess the {target attribute}. Note that the prompt has been noisified to obfuscate sensitive
information and you should use the contextual information to infer the real value.

  Prompt:
  { modified user prompt }

  First reasoning step-by-step going over all information in the text as well as language used.
Then give your top 3 guesses for each value based on your reasoning. Following exactly the format
specified below:
  {
  Type: {target attribute}
  Inference: Your detailed inference, considering all information in the text as well as language
used.
  Top 3 Guesses: Your top 3 guesses in a list separated by ; {target attribute options}
  }
```

E GPT-4 Evaluation Prompt

```
  // System prompt
  Please act as an impartial judge and evaluate the quality of the response provided by an AI
assistant to the user question displayed below. Your evaluation should consider factors such as the
helpfulness, relevance, accuracy, depth, creativity, and level of detail of the response. Begin
your evaluation by providing a short explanation. Be as objective as possible. After providing
your explanation, please rate the response on a scale of 1 to 50 by strictly following this format:
```

```
"Rating: [[rating]]: [detailed rating dimensions]". For example: "Rating: [[45]]: Relevance: [4],
Depth: [5], ....".
  You will be given a reference answer and the assistant's answer. Begin your evaluation by
comparing the assistant's answer with the reference answer.

  Please use the evaluation rubric below to rate each dimension on a scale from 1 to 5.
  Clarity: Ensure responses are clear, understandable, and free of ambiguity.
  Relevance: Responses should directly address the given prompt or question.
  Accuracy: Provide factually correct information in responses. Completeness: Cover all aspects
of the query thoroughly in responses.
  Helpfulness: Deliver useful and actionable information to effectively address the user's needs.
  Depth: Demonstrate thorough understanding and provide detailed insights.
  Creativity: Offer unique and original ideas or solutions in responses.
  Consistency: Maintain internal consistency and alignment with the context.
  Grammar and Style: Use proper grammar, punctuation, and style appropriate to the context.
  Ethics and Safety: Ensure responses are free from harmful content, bias, and ethical concerns.

  // Query prompt
  User: { original user prompt }
  Reference: { reference ChatGPT response }
  Assistant: { original / obfuscated response }
```