Developing a Private Data Blocker for Websites Using LLM Analysis

Shuning Gu

Github Link: https://github.com/shuninggu/private-data-blocker.git

1. Introduction

1.1 Background

As AI becomes more integrated into daily activities, many users now rely on AI to assist with tasks like code generation. A common practice is to visit a chatbot website and ask it to generate code. However, numerous low-quality websites may secretly store users' input and output, posing significant privacy risks. For instance, when a user requests a Language Model (LLM) to generate a curl command to send a POST request with sensitive data, such as username="annie" and password="123456", the generated code could unintentionally expose this private information. While well-known platforms like ChatGPT are generally committed to privacy protection, lesser-known sites may not adhere to the same standards, putting user data at risk.

Another potential risk arises when users ask AI to generate code for tasks like querying a database. When inputting sensitive information such as usernames, passwords, and database addresses, users may inadvertently expose critical data if the website storing the code does not properly secure it. This could lead to attackers gaining access to private data and performing malicious actions. Similarly, personal tasks like writing emails, scheduling appointments, or printing labels can also lead to privacy breaches if such data is mishandled or stored without proper safeguards.

1.2 Objectives

The primary objective of this project was to develop a Google Chrome extension designed to protect users' private information while interacting with AI websites. The extension functions as a personal assistant, storing sensitive data locally to prevent it from being transmitted to third-party companies. The system works by replacing private information in the user's input with hashed values, allowing users to submit encrypted data to the AI model. The AI

processes the information and returns results based on the anonymized input. Users can then select the relevant output, and the extension will restore the original private data, ensuring that their personal information remains secure throughout the process.

2. Methodology

2.1 Data Handling

I developed a system to ensure user privacy and security while interacting with AI websites. The system works through a front-end JavaScript extension and a back-end Node.js server to process user input and selected text efficiently.

When the user visits an AI website and types a question into an input field, the extension's "Replace" button is clicked. This action triggers the system to automatically capture the sentence from the input field. The original text is then displayed in the input box on the extension's interface for review. Once displayed, the text is sent to the back-end server, where sensitive information is identified and replaced with a hashed value, ensuring that no private data is exposed. After processing, the server sends the modified text back to the front end, and the replaced text is shown in the output display box.

Additionally, when the user highlights a portion of the text on the page and clicks the "Restore" button of the extension, the selected text is extracted and sent to the back-end server. In this case, the server decrypts the information and returns the original content. The restored text is then displayed in the output box on the extension's interface.

This approach ensures that all potentially sensitive information entered by the user or selected from the AI website is processed securely. By replacing private data with hashed values and providing a method to restore the original text when needed, the system aims to protect user privacy while maintaining the functionality of the AI website.

2.2 Utilization of Local LLM

The system leverages Ollama to run Llama2 locally, ensuring that all data processing remains within the user's environment. This setup effectively protects user privacy, as no data is transmitted to external servers. The prompt used in the system is carefully crafted to specify the desired output format, ensuring the generated responses maintain proper sentence

structure and consistent use of double quotes. This approach allows for efficient and private handling of sensitive information while providing the required results.

2.3 Privacy Processing Pipeline

The system processes user input in a structured manner through several stages to ensure privacy protection. First, the input text is sent to the local LLM (Large Language Model), which processes the data. Next, the LLM identifies any sensitive information, such as personal details or credentials, and formats it for further processing. After identifying sensitive data, the system parses and validates the results to ensure accuracy and completeness. Once validated, the original sensitive values are replaced with randomized strings, ensuring that no personal information is exposed. Finally, mappings of the original values and their corresponding randomized replacements are securely stored, enabling future restoration if necessary. This pipeline ensures that user privacy is maintained throughout the process, and sensitive information is appropriately handled.

3. Evaluation

3.1 Test Cases

The following test cases cover a variety of privacy-disclosing questions that users may enter in the chat box of the AI website.

(1) Please write a greeting card for Billie when she is 18 years old.

(2) Please help me write the terminal code to connect to the MySQL database remotely. The database name is anniedb.com. The username is Annie and password is 659876. Select the database and select all users in the users table.

(3) Write a curl command to send a POST request to the abc.com/user endpoint with the parameters username="ning" and password="123456", and return the user's information.

(4) Please write a birthday invitation for Sarah who lives at New York. Her email is sarah@email.com and phoneNumber is 212-555-0123

(5) Help me book a doctor's appointment. My name is Michael Johnson, insurance ID is MED123456, and my birthday is 03/15.

(6) Schedule a medical appointment for patient James, health insurance number HC654321.

His primary doctor’s name is Smith.

(7) Create a shipping label for delivery to Houston. phone number: 312-999-3456.

3.2 Evaluation Results

This system can generally identify and replace privacy-related information in these issues. Some of the testing process and results are shown in the demo.

To enhance its performance in the future, I plan to leverage advanced prompt engineering techniques to refine the prompts and optimize their compatibility with different models. Additionally, I aim to deploy a more powerful local LLM on a high-performance computer to improve accuracy and speed.

4. Discussion

4.1 Conclusions

The developed system successfully meets its core objective of protecting users' private information while interacting with AI websites. Specifically, the extension prevents sensitive data, such as usernames, passwords, and other personal details, from being exposed to or misused by the websites. By securely storing this information locally, it helps mitigate the risk of data theft or unauthorized access, thus enhancing user privacy.

The extension was designed to work seamlessly across different AI websites without requiring major configuration changes. Its compatibility with various platforms ensures that it can safeguard user data regardless of the AI service being used. This feature makes it a practical, easy-to-use solution that can be employed widely, protecting user privacy while interacting with AI tools.

Overall, the extension provides a simple yet effective means of protecting personal information when using AI tools on the web, offering a solution that is both flexible and user-friendly.

4.2 Future Work

While the extension offers a significant step toward protecting privacy, there are still several

challenges that need to be addressed. One limitation is that, due to modern web security and how many web frameworks manage user inputs, the extension cannot permanently control or modify the actual value in an input field if the website's JavaScript is actively managing that input's state (as seen in many modern web applications, such as Perplexity). This is an intentional security feature of web browsers, preventing malicious extensions from permanently tampering with website functionality.

However, this also presents a usability issue for users, as they are required to manually click a button to replace private data each time they use the extension. This is a necessary step due to the privacy mechanisms in place, but it can be inconvenient for users. Future improvements could explore ways to reduce this friction, either by refining the extension's functionality or by considering alternative solutions that do not require user interaction.

Furthermore, the performance of local large language models (LLMs) remains an area for improvement. Local models, such as Llama 2, are still quite unstable and often fail to meet the requirements of the given prompts. For example, while GPT-3.5 can reliably provide perfect responses, Llama 2 frequently produces inaccurate or irrelevant results, even when the task is clearly explained. Future work could focus on refining prompt engineering techniques to improve the performance and reliability of these models.

In conclusion, while the current extension successfully addresses key privacy concerns, further development is needed to enhance its functionality, improve user experience, and tackle the challenges posed by local LLMs.