# Preserving Privacy and Utility in LLM-Based Product Recommendations

Tina Khezresmaeilzadeh
University of Southern California
Los Angeles, California, USA
khezresm@usc.edu

Jiang Zhang
University of Southern California
Los Angeles, California, USA
jiangzha@usc.edu

Dimitrios Andreadis
University of Southern California
Los Angeles, California, USA
dgandrea@usc.edu

Konstantinos Psounis
University of Southern California
Los Angeles, California, USA
kpsounis@usc.edu

## Abstract

Large Language Model (LLM)-based recommendation systems leverage powerful language models to generate personalized suggestions by processing user interactions and preferences. Unlike traditional recommendation systems that rely on structured data and collaborative filtering, LLM-based models process textual and contextual information, often using cloud-based infrastructure. This raises privacy concerns, as user data is transmitted to remote servers, increasing the risk of exposure and reducing control over personal information. To address this, we propose a hybrid privacy-preserving recommendation framework which separates sensitive from non-sensitive data and only shares the latter with the cloud to harness LLM-powered recommendations. To restore lost recommendations related to obfuscated sensitive data, we design a de-obfuscation module that reconstructs sensitive recommendations locally. Experiments on real-world e-commerce datasets show that our framework achieves almost the same recommendation utility with a system which shares all data with an LLM, while preserving privacy to a large extend. Compared to obfuscation-only techniques, our approach improves HR@10 scores and category distribution alignment, offering a better balance between privacy and recommendation quality. Furthermore, our method runs efficiently on consumer-grade hardware, making privacy-aware LLM-based recommendation systems practical for real-world use.

## Keywords

recommendation system, privacy, obfuscation, LLMs

## 1 Introduction

Recommendation systems are widely used across e-commerce, streaming platforms, and digital services to deliver personalized content and improve user experience. Traditional approaches, such as collaborative filtering [28], content-based methods [64], have been effective but often struggle with challenges like sparse data, cold-start problems, and a limited understanding of user preferences. Recent works have demonstrated that applying Machine

Learning (ML) models in diverse applications leads to improved accuracy and utility [1, 7, 12, 35, 45]. Recently, Large Language Models (LLMs) have shown strong capabilities in understanding and generating natural language, making them a promising addition to recommendation systems. By leveraging textual data—such as product descriptions, reviews, and search queries—LLMs offer a way to improve recommendations beyond structured interaction data to even understand user preferences explicitly [30]. This has motivated researchers to explore their integration into recommendation frameworks, aiming to address the limitations of conventional techniques [56].

Despite this promise, there are two primary challenges in deploying LLM-driven recommendation systems in real-world settings. First, As LLMs advance in capability their parameter sizes also grow substantially. State-of-the-art LLMs such as OpenAI's ChatGPT [44], Google's Gemini [15], and Anthropic's Claude [6] often involve billions of parameters, making them computationally demanding and impractical for local deployment on consumer devices. Instead, these models are typically accessed via Application Programming Interfaces (APIs) or proprietary websites. As a result, end users either lack the requisite hardware to deploy such models locally or face vendor restrictions that disallow local downloading and execution. Second, the reliance on remote servers raises critical privacy concerns. Because these models are "opaque" in their operation, users cannot fully ascertain how their personal data, such as purchase histories or conversation logs, are processed or stored in black box devices. Moreover, platforms offering access to these models typically require users to sign in, thus linking their activity to identifiable personal accounts and potentially archiving sensitive information. Recent research has also demonstrated that LLMs can infer private attributes about users from their interactions, raising further privacy concerns [48]. For example, purchase histories can uncover sensitive insights about an individual's health status, financial situation, marital status, or even personal details like skin and hair type, all of which are protected under strict privacy regulations (e.g., HIPAA in healthcare contexts) [53].

Recent research has explored privacy-preserving techniques in recommendation systems, see related work section for more details. Non-cryptography-based privacy-preserving techniques, such as differential privacy (DP) [20], anonymization [51] and federated learning [41], have been applied to protect user data. Additionally,

several cryptographic techniques, such as homomorphic encryption (HE) [29], secret sharing [63], and secure multi-party computation (SMPC) [47], have been applied to recommendation systems. However, differentially private language models, while protecting privacy, lose personalization [9]. Federated learning-based methods, while effective in securing data during training, focus on model training privacy [57, 58, 62] rather than addressing privacy concerns at the inference stage, which is the main concern when using cloud-hosted LLMs [39]. Also, cryptographic approaches tend to require significant computational resources and often render data unintelligible, making them incompatible with systems that rely on natural language prompts [10, 31]. More general, most of these techniques were not developed for recommendation systems that rely on conversation-style text inputs. Thus, there is a need for privacy-preserving methods that respect the constraints of text-based inference while not sacrificing the recommendation quality.

To address these challenges, we propose a novel hybrid privacy-preserving recommendation framework that balances user privacy and recommendation utility while being able to be run on-device efficiently. Specifically, our framework first employs a context-aware classification model, fine-tuned from BERT [17], to detect which products or user inputs contain sensitive information. As the result of this obfuscation step, only non-sensitive data is offloaded to remote servers for further processing, thus mitigating the privacy risks associated with sharing detailed personal history. Meanwhile, sensitive recommendations are handled locally using a highly compact LLM—Llama 3.2 (1B parameters) [2]. The lightweight Llama 3.2 model can run efficiently on mobile or edge devices, offering multilingual text generation capabilities with minimal latency. By processing these sensitive queries locally, the system avoids transmitting private information to external servers, thereby preserving user privacy and ensuring compliance with legal frameworks [53].

Empirical results on real-world e-commerce datasets indicate that our hybrid approach maintains competitive recommendation accuracy (e.g., HR@10) and preserves critical measures like recommendation diversity. Importantly, these performance metrics are achieved alongside meaningful privacy protection, making our method a suitable alternative for personalized recommendation services that need to protect sensitive user information.

In summary, our main contributions are:

- An end-to-end privacy-preserving e-commerce recommendation pipeline that separates sensitive and non-sensitive data to protect user privacy while maintaining recommendation quality.
- An obfuscator module that utilizes a context-aware product classifier to identify sensitive products, sending only non-sensitive purchase history to the server-based external LLM recommender system.
- Utilizing a lightweight local deobfuscator to process sensitive products locally, providing sensitive recommendations.
- An assessment of the efficiency of our approach through experiments on standard e-commerce datasets, evaluating privacy leakage, recommendation accuracy (HR@10), and category distribution consistency to compare the diversity of recommendations with the baseline.
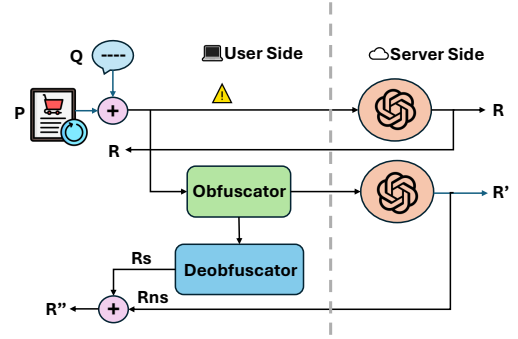


**Figure 1: High-level overview of the system.**

The rest of the paper is structured as follows: Section 2 defines concepts, goals, and assumptions. Section 3 presents our methodology, including the proposed approach and performance metrics. Section 4 details the system design, covering the obfuscator, server-based recommendation, deobfuscator, and combined recommendation list. Section 5 describes the experimental setup, followed by Section 6, which evaluates the system. Section 7 discusses findings and Section 8 presents limitations, followed by Section 9, which reviews related work on LLM-based recommendations and privacy preservation. Finally, Section 10 concludes and outlines future research directions.

## 2 Problem Statement

In this section, we define system components (Section 2.1) and introduce the objectives of this work (Section 2.2) and the threat model (Section 2.3). Figure 1 provides an overview of the interaction between the user and the server-based recommendation system, demonstrating the communication process and the associated privacy risks.

Recent advancements in large language models (LLMs) have expanded the capabilities of recommendation engines, allowing them to generate context-rich suggestions [26, 37, 55]. However, these models typically run on remote servers, requiring users to send their data, which raises serious privacy concerns [14, 54].

A user interacts with the system by submitting a query along with a complete purchase history, which comprises both sensitive and non-sensitive products in order to receive personalized recommendations. Sharing the complete purchase history can lead to privacy leakage, as disclosure of sensitive products enables the server to infer information that the user might prefer to keep private. The objective is to develop a framework that allows users to leverage server-based recommendation systems while preventing the exposure of sensitive product information, without degrading the quality of recommendations.

### 2.1 Definitions

*Purchase History.* A purchase history is the collection of products previously bought by a user, represented as structured metadata. Each product in the purchase history contains multiple fields, including but not limited to:

- *Category:* Broad classification of the product (e.g., "Health & Personal Care").
- *Title:* The product name or short description.
- *Features:* Key attributes or specifications.
- *Description:* Detailed textual information about the product.
- *Details:* Additional metadata or optional information.

This structured format captures descriptive and categorical details about the products. It is often utilized in recommendation systems to analyze user preferences and provide personalized suggestions [46]. This is the input of our recommendation system.

*Server-based Recommendation System.* Server-based recommendation systems are a type of recommendation system where data processing and recommendation generation are performed on a centralized server. In this work, we focus specifically on systems that leverage large language models (LLMs), accessed via APIs. These systems operate using two primary inputs:

- *System Prompt:* A predefined instruction that defines the task for the recommendation system, such as directing the system to generate personalized product recommendations given user purchase history as the input.
- *User Prompt:* User-specific data, such as purchase history, that the system utilizes to generate recommendations.

The output is a list of *n* product descriptions that match the user's purchasing patterns and interests. These recommendations aim to provide products that the user may find useful or appealing.

*Sensitive Products.* Sensitive products include products in the user's purchase history that reveal private or personal information. While our framework does not depend on the particular split between sensitive versus non-sensitive products, to make things specific sensitive products are defined as data that may indicate:

- *Health Status:* Products designed for individuals with specific health concerns, such as sensitive or dry skin, hair loss, or dietary restrictions.
- *Specific Illnesses or Health Conditions:* Products explicitly linked to medical conditions, prescription medications, or medical diagnostic devices.

This definition of sensitive data is aligned with privacy principles that prioritize the protection of user health information, which is often considered personally identifiable information (PII) under regulations like HIPAA [53]. Products classified as sensitive must remain private and be processed locally, as sharing them with a centralized server could expose personal information. This definition is used throughout this work to determine the sensitivity of the products and guide their classification.

*Nonsensitive Products.* Nonsensitive products are intended for general consumers and do not provide any personally identifiable health information, even if known by others. These products include common consumer goods such as fashion, personal care, household items, electronics, and general wellness products that are not tied to a medical condition. While some nonsensitive products may relate to health or well-being, they are widely used by individuals without specific medical concerns and do not indicate any private health-related information.

## 2.2 Goals

The proposed system aims to address the trade-off between privacy preservation and recommendation utility in centralized recommendation systems. The key goals are:

1. **Privacy Preservation:** Sensitive products must remain private and be processed locally on the user's device.

2. **Utility:** Despite withholding sensitive data, the system must deliver high-quality recommendations, including recommendations related to sensitive products, evaluated through:

- *Prediction Accuracy (Hit Rate)*, which measures how well the recommendations align with user preferences.
- *Category Distribution Alignment*, which measures how well the distribution of product categories in the recommendations matches that of a baseline system using the full purchase history.

To address these challenges, the system must first distinguish between sensitive and nonsensitive products in the user's purchase history. Nonsensitive products can be processed externally by a powerful server-based large language model (LLM) to leverage its advanced contextual capabilities, while sensitive products are handled locally to preserve privacy.

The ultimate goal is to deliver a unified recommendation list that balances privacy, utility, and quality, ensuring compliance with privacy constraints without sacrificing the user experience.

## 2.3 Threat Model

As illustrated in Figure 1, the obfuscator identifies sensitive products in the purchase history of the user and removes them before sending the data to the server-based recommendation system. In parallel, the deobfuscator processes these sensitive products locally, generates relevant sensitive recommendations, and merges them with the nonsensitive recommendations returned by the server. This approach enables personalized suggestions while keeping the sensitive products list private.

*User.* The user's objective is to share a purchase history with the server-based recommendation system and receive personalized product suggestions without revealing sensitive products to the server. To achieve this, the user employs the local obfuscation-deobfuscation framework mentioned above. This configuration restricts the server's access to private data by confining sensitive products processing to the user's device. The threat model assumes that the user's device is trusted, secure and not compromised by malware or adversarial manipulation.

*Server-based Recommendation System.* The goal of the server-based recommendation system is to generate product suggestions based on the user's provided purchase history. Unlike real-world e-commerce platforms such as Amazon.com [5] or Google Shopping [27], which retrieve product information from live databases, this system does not have real-time access to product listings, inventory, or user transaction logs. Instead, it relies on pre-trained data and provides recommendations based on learned knowledge rather than platform-specific product suggestions.

The system processes only the purchase history explicitly included in each input prompt. Unlike platforms such as Amazon.com, where transaction records are permanently logged and cannot be removed

by the user [4], we assume a recommendation system where the system allows users to exclude specific purchase data before sending it to the server, limiting what information is exposed.

The server operates under a semi-honest model, meaning it follows the defined protocol but may attempt to infer user personal data from the received input prompt. However, it does not have access to external sources, cross-site tracking mechanisms, or any data stored locally on the user's device. The server processes only the data explicitly provided in each prompt and does not supplement it with external sources. Furthermore, the system has sufficient computational resources to handle recommendation generation efficiently.

## 3 Methodology

### 3.1 Proposed Method

The system processes a user's purchase history to generate personalized recommendations while preserving privacy. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ denote a user's complete purchase history, where each product $p_i$ is associated with textual metadata including its main category, title, features, and description. The system splits $\mathcal{P}$ into two disjoint subsets, $P_s = \{\text{Sensitive products}\}$ and $P_{ns} = \{\text{Nonsensitive products}\}$:

- $P_s$: Products whose metadata potentially reveals private or sensitive information (e.g., specific health conditions).
- $P_{ns}$: The subset of products for general users considered safe for external processing.

The sizes of these sets are denoted as $|P_s|$ and $|P_{ns}|$, representing the number of sensitive and nonsensitive products, respectively. Based on this partition, the system generates two categories of recommendations:

- $R_s$: Recommendations derived from $P_s$ (sensitive products), generated locally on the user's device to preserve privacy.
- $R_{ns}$: Recommendations derived from $P_{ns}$ (nonsensitive products), generated using a server-based recommendation system.

Let $n_s$ and $n_{ns}$ represent the number of recommendations derived from $P_s$ and $P_{ns}$ respectively, that is, $n_s = |R_s|$ and $n_{ns} = |R_{ns}|$. The final recommendation list $R''$ is produced by merging $R_s$ and $R_{ns}$ and it is of size $n = n_s + n_{ns}$.

Figure 2 illustrates an overview of the entire architecture, which comprises the following components:

*Obfuscator:* Processes the user's purchase history to separate sensitive ($P_s$) and nonsensitive ($P_{ns}$) products. This classification is handled using a fine-tuned BERT model that employs a weighted focal loss to prioritize the detection of sensitive information while reducing false negatives. The model relies on textual metadata to make predictions.

*Server-Based Recommendation System:* The nonsensitive product subset ($P_{ns}$) is processed by a powerful server-based LLM to generate nonsensitive recommendations ($R_{ns}$). The model uses a zero-shot system prompt alongside user query ($Q$) and product metadata from $P_{ns}$ as the user prompt to produce product recommendations. Only nonsensitive data is sent to the server, keeping private information on the user's device.

*Local Deobfuscator:* The sensitive product subset ($P_s$) is processed locally on the user's device using a lightweight local LLM to generate sensitive product recommendations ($R_s$). Note that the final recommendation list ($R''$) is constructed by combining $R_{ns}$ and $R_s$ after resolving conflicts and ranking adjustments.

Looking at Figure 2, the overall data flow is as follows: the obfuscator classifies products as either sensitive ($P_s$) or nonsensitive ($P_{ns}$), with $P_{ns}$ being sent to the server-based recommendation system and $P_s$ processed locally by the deobfuscator, before merging the respective recommendations ($R_{ns}$, $R_s$) into the final recommendation set ($R''$).

To assess the recommendation quality, we use a local vector-based retriever built with ChromaDB [16]. Each generated recommendation is compared to an indexed product database to retrieve the most semantically similar product, allowing us to evaluate how well recommendations align with actual product options.

### 3.2 Performance Metrics

To assess the system's performance comprehensively, we evaluate both utility (the quality and relevance of recommendations) and privacy preservation (the impact of obfuscation on sensitive data removal). Our evaluation uses the following utility and privacy metrics:

*3.2.1 Utility.* HR@10 is a widely recognized metric in recommendation systems, used to evaluate the relevance and accuracy of recommendations. It measures whether the system successfully includes the actual target product within the top-10 recommended items. In our system, HR@10 assesses whether the recommendation system retrieves the $(n + 1)_{th}$ product in a user's purchase sequence, given the first $n$ products in their purchase history.

However, due to the scale and diversity of our dataset (see Section 5.1), HR@10 is often not informative, as the probability of retrieving the exact target product is extremely low. To address this, we introduce two alternative metrics: *Category-Based HR@10* and *Semantic Similarity HR@10*, which better capture the relevance of recommendations in a practical setting.

**1. Category-Based HR@10:** Category-Based HR@10 measures the system's ability to recommend items within the same main category as the target product. A hit is recorded if the main category of the target product matches the main category of any of the top-10 recommendations. This approach ensures that the system captures category-level relevance, reflecting its ability to suggest products within the same domain as the user's historical preferences. The main categories are derived from product metadata, and matching is performed using a vector-based database retriever that maps recommended product descriptions to their respective categories.

**2. Semantic Similarity HR@10:** To evaluate the system's ability to recommend conceptually related products, we compute the cosine similarity between the target product's ($n+1$) description and each of the top-10 recommended product descriptions [49]. Cosine similarity measures the alignment between two vector embeddings in a high-dimensional space and is calculated as:

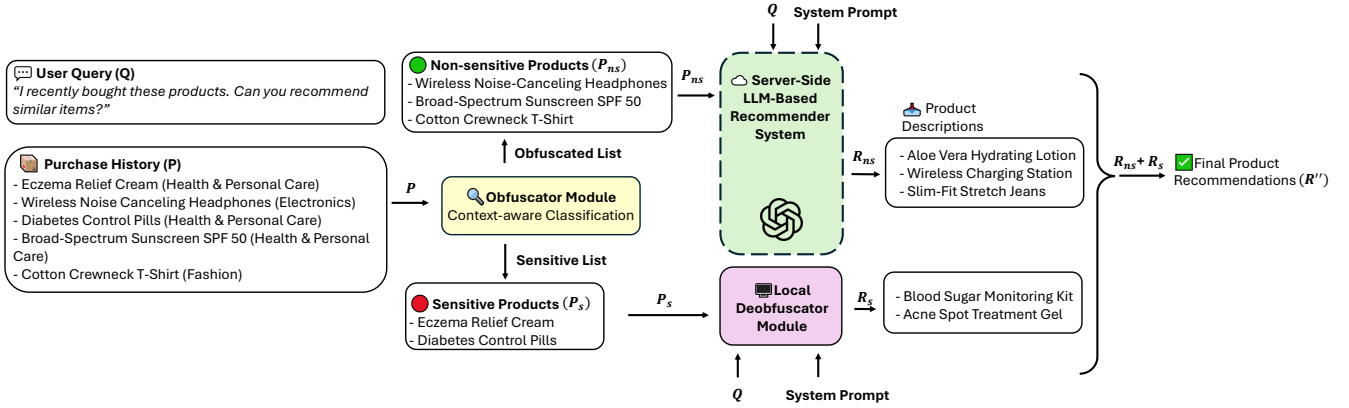$$\text{Cosine Similarity} = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}, \tag{1}$$

**Figure 2: Overview of different components of privacy-preserving recommendation system. The user query ($Q$) and purchase history ($\mathcal{P}$) are the inputs to the system. $\mathcal{P}$ is processed by the obfuscator, which classifies products into sensitive ($P_s$) and nonsensitive ($P_{ns}$) categories. $Q$ and $P_{ns}$ are sent to the server-side LLM recommender, while $P_s$ is processed locally using deobfuscator module. The final recommendation set is obtained by combining the locally processed sensitive recommendations ($R_s$) with the server-generated nonsensitive recommendations $R_{ns}$.**

where $u$ and $v$ are the embeddings of the target product and a recommended product, respectively. A similarity score of 1 indicates perfect semantic alignment, while 0 implies no similarity.

Product descriptions are transformed into embeddings using a pre-trained Sentence Transformer model, such as "all-MiniLM-L6-v2," [22] which maps text into a 384-dimensional dense vector space optimized for clustering and semantic search. These embeddings are used in the cosine similarity calculations to quantify the contextual and semantic alignment between the target product and the recommendations.

For each user's purchase history, the maximum cosine similarity among the top-10 recommendations is recorded as the semantic similarity HR@10.

Note that Semantic Similarity HR@10 complements Category-Based HR@10 by providing an additional perspective on the system's recommendation accuracy. While category matching captures broader relevance by aligning recommendations within the same category, Semantic Similarity HR@10 evaluates the relationships between products based on their contextual and semantic meaning. This ensures that the system can identify meaningful connections even when products are categorized differently or vary significantly within the same category.

**Category Distribution Metrics.** Preserving the diversity of recommendations is critical for ensuring that a privacy-preserving recommendation system continues to reflect user preferences without skewing toward nonsensitive categories. To evaluate this, we use category distribution vectors and compute deviations from the baseline system using L1 and L2 distances. These metrics provide a robust framework to assess system-wide diversity and balance across product categories.

For a recommendation list of $n$ items and $C$ categories, the category distribution vector is defined as $v = [v_1, v_2, ..., v_C]$, where $v_i$

represents the proportion of recommendations belonging to category $i$, that is, $v_i = \frac{\text{count of items in category } i}{n}$, for $i = 1 \ldots C$, and $\sum_{i=1}^{C} v_i = 1$.

*Distance Metrics for Evaluation.* To evaluate how well the obfuscated systems replicate the category diversity of the baseline, we compute the L1 distance and L2 distance between their respective category distribution vectors.

**L1 Distance:** The L1 distance measures the total absolute difference between the baseline ($v_{baseline}$) and obfuscated system ($v_{system}$) distributions:

$$L_1 = \sum_{i=1}^{C} |v_{\text{baseline},i} - v_{\text{system},i}|. \tag{2}$$

This metric measures overall imbalances in category distributions. A smaller L1 distance indicates better preservation of category balance, reflecting minimal disruption to the original diversity of recommendations.

**L2 Distance:** The L2 distance, or Euclidean distance, quantifies the magnitude of deviations:

$$L_2 = \sqrt{\sum_{i=1}^{C} (v_{\text{baseline},i} - v_{\text{system},i})^2}. \tag{3}$$

The L2 distance emphasizes larger deviations, making it sensitive to noticeable changes in category proportions. A smaller L2 value indicates a closer alignment with the baseline distribution.

Users expect recommendations to capture their diverse preferences. However, noticeable deviations in category distributions can result in imbalanced recommendations, either by overrepresenting non-sensitive categories or omitting sensitive ones. By using L1 and L2 distances, we can measure these deviations to ensure that obfuscation and deobfuscation mechanisms maintain the diversity present in a user's purchase history. Unlike HR@10, which focuses

on individual-level recommendation relevance, category distribution metrics provide a system-wide perspective. They evaluate how obfuscation and deobfuscation affect the overall balance of recommendations across all categories, addressing both the accuracy for individual users and fairness across the system. The HR@10 and distance metrics together capture both the relevance of recommendations for individual users and the preservation of category diversity across the system.

*3.2.2 Privacy.* We measure privacy exposure using two metrics: *Binary Privacy Leakage* ($PL_b$) and *Semantic Privacy Leakage* ($PL_s$). The first metric quantifies the fraction of leaked sensitive products, while the second accounts for their varying sensitivity levels.

*Binary Privacy Leakage ($PL_b$).* $PL_b$ measures the average fraction of sensitive products that have been leaked. It is defined as:

$$PL_b = \frac{1}{n_s} \sum_{i=1}^{n_s} x_i, \qquad (4)$$

where $n_s$ is the total number of sensitive products in the purchase history, and $x_i = 1$ if the $i$th sensitive product is leaked and 0 otherwise.

*Semantic Privacy Leakage ($PL_s$).* $PL_s$ extends $PL_b$ by incorporating sensitivity scores of the products, assigning higher importance to more sensitive items. It is defined as:

$$PL_s = \frac{\sum_{i=1}^{n_s} x_i s_i}{\sum_{i=1}^{n_s} s_i}, \qquad (5)$$

where $s_i$ represents the sensitivity score of the $i$th sensitive product. The numerator accumulates the sensitivity scores of leaked products, while the denominator normalizes by the total sensitivity of all sensitive products.

Both metrics assume that sensitivity scores are predefined and reflect the relative importance of different products. If sensitivity scores are not available, $PL_b$ serves as a simpler alternative. However, $PL_s$ provides a more refined measure by accounting for varying degrees of sensitivity among leaked products.

To measure sensitivity scores, we use few-shot prompting with ChatGPT-4-o [43] to assign a sensitivity score between 0 and 1 to each product based on its perceived sensitivity. The exact few-shot prompt used for this evaluation is provided in Appendix A.4. For the products used in our analysis (see Section 6.1), the mean sensitivity of the sensitive products (i.e., those labeled with a ground-truth label as sensitive) is 0.5422, and the standard deviation is 0.2317.

## 4 System Design

### 4.1 Obfuscator

The Obfuscator module enhances user privacy by separating products in the purchase history ($\mathcal{P}$) into sensitive ($P_s$) and nonsensitive ($P_{ns}$) subsets. Sensitive products ($P_s$) may reveal private information about the user (e.g., health-related conditions), while nonsensitive products ($P_{ns}$) are safe for external processing. This classification is achieved through two methods: the Categorical Obfuscator and the BERT-based Obfuscator.

*4.1.1 Data Preparation.* Each product in $\mathcal{P}$ is represented as a textual string derived from the following fields:

- *Main Category:* Broad product categorization (e.g., "Health & Household," "Beauty & Personal Care").
- *Title:* The product's short name or headline.
- *Features:* Key attributes of the product, often presented as bullet points.
- *Description:* Detailed text providing an overview of the product.
- *Details:* Supplementary metadata, when available.

This representation presents the input for the obfuscation methods, enabling a complete view of the product's attributes and context. While this detailed representation is used for the BERT-based approach, the Categorical Obfuscator relies only on predefined category mappings.

*4.1.2 Categorical Obfuscator.* This method involves filtering out products belonging to predefined sensitive categories (e.g., Health & Personal Care, Health & Household, and Beauty & Personal Care) from the user's purchase history. By utilizing a fixed mapping of categories to sensitivity, the approach assumes that all products within these categories are inherently sensitive. While this method is computationally efficient and straightforward—requiring no model training—it presents limitations. False positives may occur when nonsensitive products (e.g., general-use beauty items) are misclassified as sensitive, and false negatives may arise when sensitive products outside the predefined categories (e.g., books on health conditions) are overlooked.

*4.1.3 BERT-based Obfuscator.* To address the limitations of the categorical obfuscator, we leverage BERT (Bidirectional Encoder Representations from Transformers) [17], a state-of-the-art transformer-based model, fine-tuned specifically on sensitive product categories, to classify products based on their full textual descriptions rather than relying solely on predefined categories. Additionally, BERT's pretrained architecture allows for efficient fine-tuning on domain-specific datasets, making it well-suited for this task [50].

By employing context-aware classification, nonsensitive products that fall under a broad category will not be misclassified as sensitive. This reduces the difference between a user's actual purchase history and their obfuscated purchase history. As a result, the obfuscated profile remains closer to the original, making the obfuscation process more precise and effective. Further details on the training process and dataset construction for the BERT-based obfuscator are provided in Section 5.3.

*Inference and Classification:* During inference, product descriptions are processed through the fine-tuned BERT model to produce sensitivity scores which are utilized as follows:

- Scores > 0.5: Classified as $P_s$(sensitive).
- Scores ≤ 0.5: Classified as $P_{ns}$(nonsensitive).

### 4.2 Server-Based Recommendation System

The Server-Based Recommendation System generates recommendations for the nonsensitive subset ($P_{ns}$) by leveraging a cloud-based large language model (LLM), ensuring privacy while maximizing utility. By exclusively processing nonsensitive data, the system avoids any potential leakage of sensitive information.

*4.2.1 Model Choice.* We employed ChatGPT-4-o [43] via its API for its state-of-the-art performance in generating contextually rich and relevant product recommendations [19]. Its ability to balance semantic understanding and categorical diversity makes it particularly well-suited for this task. Preliminary evaluations showed that ChatGPT-4-o performs well in terms of recommendation relevance and diversity, making it an appropriate choice for this application.

*4.2.2 Prompt Design and Validation.* To ensure that the recommendations align with the user's purchase history, the system employs a carefully crafted prompt. This prompt ensures a proportional representation of categories and comprehensive coverage across all categories in $P_{ns}$.

The number of nonsensitive recommendations ($n_{ns}$) is calculated as:

$$n_{ns} = n \times \frac{|P_{ns}|}{|P_{ns}| + |P_s|}, \tag{6}$$

where $n$ is the total number of desired recommendations (set to 10 in the experiments), $|P_{ns}|$ is the number of nonsensitive products in the purchase history, and $|P_s|$ is the number of sensitive products in the purchase history. The full prompt used in the server-based recommendation system is provided in Appendix A.1.

## 4.3 Deobfuscator

The Deobfuscator complements the server-based recommendation system by generating recommendations for the sensitive products in $P_s$. This step is performed locally on the user's device to maintain strict privacy standards, ensuring that sensitive data is never transmitted externally.

For sensitive products ($P_s$), their concatenated textual representations are processed locally using Llama 3.2 1B [21], a lightweight language model capable of efficient inference on consumer-grade hardware. By handling sensitive recommendations locally, the system ensures that privacy-critical information remains on the user's device.

*4.3.1 Prompt Design.* The same prompt structure used for nonsensitive recommendations is adapted for sensitive products, see Appendix A.2 for the full prompt.

*4.3.2 Output:* The model generates the sensitive recommendation list ($R_s$) locally. These recommendations are stored securely on the user's device, ensuring strict privacy standards. The number of sensitive recommendations ($n_s$) is calculated as:

$$n_s = n \times \frac{|P_s|}{|P_{ns}| + |P_s|}, \tag{7}$$

This ensures that sensitive recommendations are proportional to the sensitive product share in the purchase history. By processing sensitive data locally, the system safeguards privacy while still providing personalized recommendations.

## 4.4 Combined Recommendation List

The final recommendation list integrates both nonsensitive and sensitive recommendations, providing the final set of personalized suggestions:

$$R'' = R_{ns} + R_s, \tag{8}$$

| Category | #Users | #Items |
|---|---|---|
| Amazon_Fashion | 2.0M | 825.9K |
| Beauty_and_Personal_Care | 11.3M | 1.0M |
| Electronics | 18.3M | 1.6M |
| Health_and_Household | 12.5M | 797.4K |
| Magazine_Subscriptions | 60.1K | 3.4K |
| Books | 10.3M | 4.4M |
| Baby_Products | 3.4M | 217.7K |
| Grocery_and_Gourmet_Food | 7.0M | 603.2K |
| Health_and_Personal_Care | 461.7K | 60.3K |
| Musical_Instruments | 1.8M | 213.6K |

**Table 1: Summary of the selected categories from the Amazon Reviews 2023 dataset, including the number of users and items across sensitive and nonsensitive domains.**

where $R''$ represents the complete recommendation list, including both non-sensitive and sensitive product recommendations. By combining the outputs from the server and the local model, the system delivers personalized recommendations without compromising the user's privacy.

## 5 Experimental Setup

To thoroughly evaluate our privacy-preserving recommendation system we consider three setups: A baseline system with no obfuscation, a system that uses either the Categorical Obfuscator or the BERT-based Obfuscator, and a system that uses either of the obfuscators and the Llama-based Deobfuscator. This experiments are designed to quantify the trade-offs between privacy preservation and utility recovery, providing insights into how effectively the system balances these two objectives.

### 5.1 Dataset

The system leverages the 2023 Amazon Reviews Dataset McAuley Lab [32], a large-scale dataset that includes product reviews and metadata across a variety of categories. This dataset provides a rich resource for evaluating the system's ability to handle both recommendation and privacy-preserving tasks. The selected categories reflect a mix of sensitive and nonsensitive domains, ensuring a diverse range of purchasing behaviors is represented.

To evaluate the system's performance comprehensively, ten categories were selected from the dataset, comprising both sensitive and nonsensitive domains. Sensitive categories—"Health & Personal Care," "Health & Household," and "Beauty & Personal Care"—were deliberately chosen for their relevance to privacy concerns. Nonsensitive categories were included to capture a broader diversity of textual characteristics and purchasing contexts, essential for robust system evaluation. Table 1 summarizes the selected categories.

*Classification of Sensitive and Nonsensitive Categories:* Sensitive categories were selected based on their strong potential to reveal private user information. Specifically:

- "Health & Personal Care" and "Health & Household" often include products that relate to medical conditions or personal wellness.

- "Beauty & Personal Care" contains items designed for individuals with specific health concerns, such as sensitive or dry skin or hair, etc.

While it is possible that certain nonsensitive categories could occasionally contain products that may reveal private information depending on how privacy is defined, as already discussed in Section 2.1, we use health, personal care, and body-related items as sensitive since this aligns with common privacy concerns in e-commerce contexts, where such categories are more likely to reveal intimate or personal details about users. Consequently, other categories are treated as nonsensitive.

To construct user purchase histories, we first filter the dataset to "engaged" users (users having at least 30 products in their purchase history), then we sorted each user's products in chronological order and selected the last 20 purchases. This approach maintains a consistent-length input across all users while prioritizing the most recent purchases, which better capture current user preferences and behavior. Choosing 20 purchases provides a balance between computational efficiency and retaining sufficient context to generate high-quality recommendations.

## 5.2 Experimental Setups

For both obfuscation methods we define three experimental setups (as demonstrated in Figure 1) to systematically evaluate the system:

**Baseline ($R$):** The complete purchase history, including sensitive products, is sent to the server-based recommendation system. This serves as the upper bound for recommendation quality (utility) since no privacy-preserving mechanisms are applied. The Baseline provides a reference point to measure the impact of obfuscation and deobfuscation on utility.

**Only Obfuscator ($R'$):** Under this setup, sensitive products are removed from the purchase history using either the Categorical Obfuscator or the BERT-based Obfuscator. The remaining nonsensitive products are then sent to the server-based recommendation system. This condition highlights the privacy-preserving aspect of the system but sacrifices utility, as sensitive products are completely excluded from the recommendations.

**Obfuscator + Deobfuscator ($R''$):** This setup combines the obfuscator with the deobfuscator to address the utility loss introduced by removing sensitive products from the shared purchase history. Nonsensitive recommendations are generated by the server-based system using the obfuscated purchase history. Simultaneously, the deobfuscator generates recommendations for sensitive products locally using Llama 3.2 1B. The final recommendation list merges nonsensitive and sensitive recommendations. This setup aims to preserve privacy while recovering utility, making it the most balanced option.

## 5.3 Obfuscator Training and Testing Details

As already discussed, we use a finetuned BERT-based model for the obfuscation of the sensitive products. The model training details are given below.

*Dataset Construction.* We randomly sampled 10, 000 product entries from predefined sensitive categories to create our dataset. Each product entry comprised a title, its main category, and a list of features. We concatenated these attributes into a single textual prompt to serve as the input to our model. An example prompt format (without showing actual code) included the product's title on the first line, followed by main category information, and then a list of features.

*Labeling.* Products were labeled as either "sensitive" or "nonsensitive" following the definition of sensitive and nonsensitive products outlined in Section 2.1. ChatGPT-4o was used to assign these labels based on the classification criteria. The exact few-shot prompt used for this classification is provided in appendix A.3. The model follows specific instructions for the classification and makes uses of few examples so as to better infer the correct labeling pattern.

*Model and Tokenization.* We fine-tuned a BERT-based model (*bert-base-uncased*) for binary classification of product sensitivity. Each product description was tokenized using the BERT tokenizer with a maximum length of 256, applying right padding and truncation to keep input sizes consistent.

*Training and Hyperparameters.* The dataset was split into training (70%), validation(20%), and test (10%) subsets using stratified sampling to maintain class balance in training, evaluation, and test set. Three actions were taken to improve detection of sensitive products, as misclassifying a sensitive product as nonsensitive is more critical than the reverse.

First, we employ Focal Loss [38] to modify the standard cross-entropy loss. This loss down-weights well-represented examples and concentrates learning on less-represented examples which are harder to classify. The loss function is defined as:

$$L = -\sum_{i=1}^{N} wc_i (1 - pc_i)^{\gamma} \log(pc_i), \tag{9}$$

where $N$ is the total number of samples, $wc_i$ is the weight of the true class of data sample $i$, and $pc_i$ is the predicted probability of the true class of data sample $i$. The focusing parameter $\gamma$ is set to 2 in this implementation. The class weight $wc_i$ is given by:

$$wc_i = \frac{N}{2 \cdot N_c}, \tag{10}$$

where $N_c$ is the number of samples in class $c$ where data sample $i$ belongs to. This approach increases the loss contribution of the ("sensitive") labeled products as they tend to be less represented.

Second, we lower the decision threshold for the "sensitive" class to 0.3. This change increases recall by classifying more products as sensitive, thereby reducing the risk of mistakenly labeling truly sensitive items as nonsensitive.

| Hyperparameter | Value |
|---|---|
| Learning Rate | $2 \times 10^{-5}$ |
| Weight Decay | 0.01 |
| Batch Size | 16 |
| Epochs | 5 |

**Table 2: Hyperparameters used for training the BERT-based product sensitivity classification model.**

| Component | Specification |
|-----------|---------------|
| **GPU** | 2× NVIDIA RTX A6000 (49GB VRAM) |
| **CPU** | AMD EPYC 7662 (64 Cores, 128 Threads) |
| **RAM** | 128GB DDR4 |
| **Storage** | 1TB NVMe SSD |
| **Networking** | 1 Gbps Ethernet (for API calls to global LLM) |

**Table 3: Hardware Configurations of the System**

Finally, model selection is based on the best F1-score on the validation set. This metric optimizes the trade-off between precision and recall, aligning with our goal of accurately identifying sensitive products while minimizing misclassification.

Table 2 summarizes the hyperparameters used during the training process. During each training epoch, the model was optimized to minimize the loss mentioned in Eq. 9. A checkpoint was saved at the end of every epoch, and the best-performing checkpoint on the validation set (based on F1 score) was automatically loaded at the end of training.

*Evaluation Metrics.* Model performance was evaluated using four standard classification metrics: accuracy, precision, recall, and F1-score. The final model achieved an evaluation loss of 0.181, an accuracy of 0.911, an F1-score of 0.868, a precision of 0.815, and a recall of 0.928. These results indicate that the model effectively differentiates between sensitive and nonsensitive products while maintaining a trade-off between capturing sensitive items and avoiding unnecessary misclassification.

*Testbed Configuration.* Table 3 presents the hardware configuration of the testbed used for our experiments. We evaluate system hardware usage and end-to-end latency in Section 7.

## 6 Evaluation

### 6.1 Utility and Privacy Results

Table 4 presents the performance of the system under different configurations, focusing on the trade-offs between utility and privacy. Utility is evaluated using two key metrics: HR@10 (Categorical and Semantic) and category distribution vector distances (L1 and L2) compared to the baseline. Privacy is assessed through Binary Privacy Leakage ($PL_b$) and Semantic Privacy Leakage ($PL_s$). Below we compare in detail the performance of the different baselines as well as of our system; see Section 5.2 and Figure 1.

The Baseline ($R$), which uses the full purchase history without obfuscation, achieves the highest utility, with Categorical HR@10 of 0.6263 and Semantic HR@10 of 0.3045, serving as the upper bound for recommendation quality. However, it results in 100% Privacy Leakage, sharing all sensitive products with the server.

Categorical Obfuscation ($R'$) impacts both recommendation relevance and diversity due to the complete removal of sensitive categories. This results in a Categorical HR@10 of 0.4742 and a Semantic HR@10 of 0.2444. These metrics reflect a decrease in the system's ability to recommend items that align with the user's preferences. This configuration also impacts recommendation diversity, as indicated by the L1 distance of 0.8462 and L2 distance of 0.4860, which measure deviations in category distributions compared to

the baseline. While this approach ensures perfect privacy (0% Privacy Leakage), it limits the system's ability to generate diverse and relevant recommendations.

BERT Obfuscation ($R'$) improves utility by selectively removing sensitive products, achieving a Categorical HR@10 of 0.5960 and a Semantic HR@10 of 0.2866. Category distribution deviations are smaller compared to categorical obfuscation, with an L1 distance of 0.4610 and L2 distance of 0.2672. The use of BERT leverages contextual understanding to classify sensitive products more effectively, allowing for greater retention of nonsensitive data. However, this method introduces 22.2833% Privacy Leakage, as not all sensitive products are successfully identified and removed.

An analysis of the missed sensitive products reveals that their average sensitivity is 0.2022, which is well below the overall mean sensitivity of all assessed products (0.5422). This difference (over 0.3 points) exceeds the standard deviation of the sensitivity scores of the products (0.2317), supporting the claim that the classifier primarily fails to detect items of lower sensitivity.

Adding deobfuscation ($R''$) addresses the limitations introduced by obfuscation, where sensitive data removal reduces recommendation relevance and diversity. By locally generating sensitive recommendations, deobfuscation helps restore utility while ensuring sensitive data remains private and is not transmitted to the server.

Categorical Obf + Deobf ($R''$) enhances utility by increasing Categorical HR@10 to 0.5258 and Semantic HR@10 to 0.2994, compared to the obfuscation-only configuration. It also improves category distribution alignment, with L1 and L2 distances reduced to 0.4773 and 0.2847, respectively. Privacy leakage remains at 0%, as sensitive products are entirely excluded from the data sent to the server. However, due to the complete removal of sensitive categories during obfuscation, the diversity within those categorAnalysis of Average Category Distribution Distances ies cannot be fully recovered, limiting the effectiveness of deobfuscation in this configuration.

BERT Obf + Deobf ($R''$) achieves the highest utility among privacy-preserving configurations, with Categorical HR@10 of 0.6061 and Semantic HR@10 of 0.3117, closely approaching the baseline. This configuration also shows improved category distribution alignment, with L1 and L2 distances reduced to 0.3747 and 0.2267, respectively. However, Privacy Leakage remains at 22.2833%, reflecting the limitations of the BERT classifier in perfectly identifying sensitive products.

Deobfuscation leverages a lightweight local model (Llama3.2 1B) to generate sensitive recommendations directly on the user's device, ensuring that sensitive data is not transmitted to the server. While the Only Local configuration also avoids any sensitive data exposure and achieves 0% Privacy Leakage, it falls short in utility, compared to the higher scores achieved by BERT Obf + Deobf ($R''$), as observed in Table 4. This utility gap highlights the limitations of relying solely on local models, which lack access to the broader nonsensitive data available on the server, leading to reduced recommendation diversity and relevance.

By combining server-based recommendations for nonsensitive products with locally generated sensitive recommendations, BERT Obf + Deobf ($R''$) achieves the best balance between utility and privacy. It recovers lost sensitive recommendations while preserving privacy and closely aligning with the baseline's category distributions, with L1 and L2 distances minimized to 0.3747 and 0.2267,

| Scheme | HR@10 | | Distance | | Privacy Leakage (%) | |
|---|---|---|---|---|---|---|
| | Categorical | Semantic | L2 Distance | L1 Distance | $PL_b$ (%) | $PL_s$ (%) |
| Baseline ($R$) | 0.6263 | 0.3045 | 0 | 0 | 100% | 100% |
| Only Local | 0.4667 | 0.2398 | 0.4538 | 0.7711 | **0%** | **0%** |
| Categorical Obf Only ($R$) | 0.4742 | 0.2444 | 0.4860 | 0.8462 | **0%** | **0%** |
| Categorical Obf + Deobf ($R''$) | 0.5258 | 0.2994 | 0.2847 | 0.4773 | **0%** | **0%** |
| BERT Obf Only ($R'$) | 0.5960 | 0.2866 | 0.2672 | 0.4610 | 22.2833% | 10.9899% |
| BERT Obf + Deobf ($R''$) | **0.6061** | **0.3117** | **0.2267** | **0.3747** | 22.2833% | 10.9899% |

Table 4: Performance metrics for various system configurations, comparing utility (Categorical HR@10, Semantic HR@10, and category distribution distances L1 and L2 with the Baseline ($R$)) and privacy leakage($PL_b$% and $PL_s$%). The results highlight how different approaches balance recommendation relevance and privacy preservation, with BERT Obf + Deobf ($R''$) achieving the best trade-off compared with others.

| Scheme | Nonsensitive | | Sensitive | |
|---|---|---|---|---|
| | Avg L2 Distance | Avg L1 Distance | Avg L2 Distance | Avg L1 Distance |
| Baseline ($R$) | 0 | 0 | 0 | 0 |
| Only Local | 0.0227 | 0.0343 | 0.0404 | 0.041 |
| Categorical Obf Only ($R'$) | 0.0508 | 0.0769 | 0.0984 | 0.1025 |
| Categorical Obf + Deobf ($R''$) | 0.0269 | 0.0393 | 0.0623 | 0.0673 |
| BERT Obf Only ($R'$) | 0.0158 | 0.0223 | 0.0547 | 0.0553 |
| BERT Obf + Deobf ($R''$) | **0.0144** | **0.0203** | **0.0401** | **0.0430** |

Table 5: Average L1 and L2 distances for nonsensitive and sensitive categories across configurations. Nonsensitive distances improve with deobfuscation as category proportions approach the baseline. Sensitive distances, initially high due to exclusion in obfuscation-only configurations, are noticeably reduced with deobfuscation, highlighting its role in restoring alignment for sensitive recommendations.

| Scheme | Recovery (%) | |
|---|---|---|
| | L2 | L1 |
| **Categorical Obf + Deobf ($R''$)** | 41.42 | 43.59 |
| **BERT Obf + Deobf ($R''$)** | 15.16 | 18.72 |

Table 6: Alignment recovery through obfuscation.

respectively. This hybrid approach demonstrates its effectiveness in maintaining both recommendation quality and privacy in real-world applications.

## 6.2 Privacy-Utility Trade-offs

The baseline configuration, which transmits the entire purchase history, achieves the highest HR@10 scores, reflecting the optimal recommendation quality. However, it leads to 100% privacy leakage, as all sensitive products are exposed to the server.

Categorical obfuscation ($R'$) achieves complete privacy protection by removing all products from predefined sensitive categories. While this method fully preserves privacy, it reduces HR@10 and also noticeably shifts category distribution. As shown in Table 4, categorical obfuscation results in a notable shift, with higher L1 and L2 distances from the baseline.

BERT-based Obfuscation mitigates this utility drop by selectively removing products classified as sensitive using context-aware classification, resulting in flagging less products as sensitive, preserving

more non-sensitive products. This results in a noticeable improvement in the recommendation quality (higher HR@10 and noticeably lower categorical distribution difference), while due to imperfect classification there is 22% privacy leakage, leaving some sensitive products exposed.

## 6.3 Category Distribution Alignment Analysis

Table 5 provides a detailed evaluation of the category distribution deviations for both nonsensitive and sensitive categories across different configurations. The metrics, average L1 and L2 distances, capture the alignment of category distributions in the recommendations with the baseline. These distances are normalized by the number of categories, making them directly comparable across nonsensitive and sensitive groups.

The results reveal notable differences between nonsensitive and sensitive category alignment:

**Nonsensitive Categories:** With obfuscation-only configurations ($R'$), the sum of nonsensitive category proportions in the category distribution vector is always one, as sensitive categories are completely removed. This mismatch leads to larger nonsensitive distances compared to configurations with deobfuscation. For example, Categorical Obf Only ($R'$) results in Avg L2 Distance of 0.0508 and Avg L1 Distance of 0.0769, reflecting a mismatch in nonsensitive proportions due to the absence of sensitive recommendations. When deobfuscation is added ($R''$), nonsensitive category proportions become more balanced. For BERT Obf + Deobf, the

distances improve to Avg L2 Distance of 0.0144 and Avg L1 Distance of 0.0203, as sensitive recommendations are restored locally, reducing the nonsensitive-only bias.

**Sensitive Categories:** Sensitive categories show higher deviations in obfuscation-only configurations, as they are completely excluded from the recommendations. For Categorical Obf Only ($R'$), sensitive recommendations have Avg L2 Distance of 0.0984 and Avg L1 Distance of 0.1025, indicating complete misalignment. Deobfuscation noticeably reduces these deviations by reintroducing sensitive recommendations. For BERT Obf + Deobf, the sensitive distances improve to Avg L2 Distance of 0.0401 and Avg L1 Distance of 0.0430, showing that sensitive recommendations align more closely with the baseline.

## 6.4 Recovering Utility: via Deobfuscation

When applied to Categorical Obfuscation ($R'$), deobfuscation reintroduces previously removed sensitive categories, improving HR@10 scores and reducing L1 and L2 distances to the baseline, recovering over 41% (L2) [1] and over 43% (L1) of the original alignment, see Table 6. However, because sensitive categories are entirely erased during obfuscation, full recovery remains infeasible, as the local model lacks access to the complete distribution of user interests.

For BERT-based obfuscation ($R'$), deobfuscation achieves better utility than categorical approaches. Since BERT obfuscation retains more nonsensitive data, the combined BERT obfuscation and deobfuscation configuration ($R''$) leads to closer alignment with the baseline, with higher HR@10 and lower category distribution deviations. As shown in Table 6, deobfuscation improves alignment by reducing L1 and L2 distances by approximately 15%. The results demonstrate that BERT-based obfuscation combined with local deobfuscation outperforms categorical obfuscation in preserving both privacy and recommendation quality, offering a more effective balance between privacy protection and utility retention.

Last, as discussed in Section above, configurations with deobfuscation restore sensitive categories, ensuring the proportions of nonsensitive and sensitive recommendations align more closely with the baseline.

## 7 Computational Overhead
## 7.1 Hardware Requirements

Our system is designed to function efficiently on widely available consumer hardware, eliminating the need for cloud-based infrastructure. The two local obfuscator and deobfuscator modules used in our pipeline—BERT-base (110M parameters) [17] and Llama-3.2-1B (1B parameters) [2] —are well-known for fitting into edge and mobile devices[3, 17].

To analyze the memory and computational overhead of these local modules, we provide a detailed breakdown of system resource consumption during the evaluation phase in Table 7.

The following values were measured during the experiments and averaged across all users in the evaluated dataset:

- *System Memory (MB):* The total RAM consumed by the system while running the model, including OS overhead and background processes.
- *Process Memory (MB):* The actual RAM allocated exclusively to the model process, excluding system overhead.
- *Peak GPU Memory (MB)*: The highest amount of allocated GPU memory actively used by tensors, and reserved GPU memory, which includes additional caching.
- *CPU Usage (cores):* The number of logical CPU cores utilized by the model.

The BERT Obfuscator module exhibits low computational overhead, utilizing 4.45 CPU cores and requiring 369.46 MB of system memory with a 402.95 MB process memory. Its GPU demand remains modest, with peak memory usage reaching 471.89 MB allocated and 601.55 MB reserved.

Similarly, Llama-3.2-1B demonstrates resource efficiency within edge device capabilities. It operates with an average of 5.61 CPU cores, consuming 1331.41 MB of system memory and 872.28 MB of process memory. For GPU inference, its peak memory usage reaches 3044.50 MB allocated and 3225.77 MB reserved, aligning with the capabilities of consumer-grade devices.

These results confirm that both models maintain a manageable memory footprint and low computational demands, making them well-suited for deployment on widely available consumer devices without reliance on cloud infrastructure.

## 7.2 Real-Time Computational Analysis

If we denote the time execution of the obfuscator, deobfuscator, and server-based recommendation system by $T_{obf}$, $T_{deobf}$, and $T_{rec}$ respectively, the additional end-to-end delay of the system compared to the baseline (only having $T_{rec}$) equals:

$$T_{total} = T_{obf} + \max(T_{rec}, T_{deobf}) - T_{rec}, \qquad (11)$$

since the deobfuscator and the server-based recommendation system can run simultaneously as they are independent modules.

To evaluate the inference performance in different computing environments, we conducted experiments on a consumer-grade laptop, equipped with an AMD Ryzen 9 5900HX CPU, an NVIDIA RTX 3050 Ti Laptop GPU (4GB VRAM), and 30GB RAM. Due to the GPU's memory constraints, we employed an 8-bit quantized version of the Llama model to enable efficient inference. The average inference time for obfuscation is 0.1808 and for deobfuscation is 6.3816. Given that the average $T_{rec}$ is 2.7428 seconds, the total additional inference time for the consumer Laptop equal 3.8196 seconds, indicating the real-time feasibility of the privacy-preserving framework on edge devices.

## 8 Limitations

*Sensitivity Classification Errors.* The effectiveness of obfuscation relies on the BERT-based classifier, yet misclassifications can impact both privacy and utility. False negatives lead to privacy leakage, exposing sensitive products, while false positives unnecessarily remove nonsensitive products, reducing recommendation quality. This limitation arises because the system prioritizes on-device efficiency, requiring a model that balances accuracy and computational

---

[1]Looking at Table 4, obfuscation increases L2 distance from 0 to 0.4860, and deobfuscation reduces it to 0.2847 or by (0.4860-0.2847)/0.4860=41.52%.

| Model | Peak GPU Memory (MB) | | Memory Usage (MB) | | CPU Usage (cores) |
|---|---|---|---|---|---|
| | Allocated | Reserved | System Memory | Process Memory | |
| **Obfuscator** | 471.89 | 601.55 | 369.46 | 402.95 | 3.69 |
| **Deobfuscator** | 3225.77 | 3044.50 | 1331.41 | 872.28 | 5.61 |

**Table 7: Average Hardware overhead of the local obfuscator-deobfuscator modules during the experimental phase.**

feasibility. A more complex classifier could improve sensitivity detection but may not be practical for local execution.

*Fixed Privacy Definition and Lack of User Adaptability.* The current approach defines sensitivity based on health-related categories, which may not fully reflect individual privacy preferences. However, sensitivity is subjective, and factors such as marital status, income level, and gender may also be considered private depending on the user. We believe it is straightforward to expand the system to incorporate a broader range of sensitive attributes. User-adaptive privacy controls can improve its flexibility and applicability across diverse contexts.

*Cumulative Purchase History Privacy Leakage.* The obfuscation system evaluates products individually but does not account for patterns in purchase history. Items like general pain relievers or nutritional supplements may seem nonsensitive on their own, but frequent or large-scale purchases could suggest conditions like chronic pain or nutritional deficiencies. This creates a privacy risk, as long-term analysis may expose personal health details that were not intended to be disclosed. Our approach does not address this form of privacy inference, as it focuses on product-level sensitivity rather than cumulative patterns.

*Input and Output Length Constraints.* The input and output capacities of local LLM-based deobfuscator models and server-based recommendation systems are inherently limited [2]. When purchase histories exceed the model's input capacity or the number of requested recommendations surpasses its output limit, text chunking and multiple iterations become necessary. This can impact recommendation quality, as the model may lose context by processing only parts of the purchase history at a time.

## 9 Related Work

Recent studies have explored the role of LLMs in recommendation systems, emphasizing their ability to generate recommendations without additional training [13, 23, 30, 55]. [13] analyzed ChatGPT's recommendation performance, demonstrating its effectiveness across different ranking paradigms and its superiority over other LLMs. Beyond general recommendation tasks, LLMs have been investigated for conversational recommendation systems, leveraging their natural language understanding capabilities [24–26]. For instance, [26] introduced ChatRec, a system that transforms user profiles and past interactions into structured prompts, improving both top-k recommendation and zero-shot rating predictions. However, these studies largely overlook privacy concerns,

which remain a critical issue in LLM-based recommendation systems. Recent research has highlighted the privacy risks of black-box models, particularly their potential for extensive user profiling [36].

Privacy-preserving techniques in LLMs have gained attention [9, 11, 34, 40, 52], though their applicability to recommendation systems is limited. Hide and Seek (HaS) [11] anonymizes private entities in user prompts such as names before LLM processing and later restores them, but their restoration method is inapplicable to recommendation systems. InferDPT [52] applies differential privacy (DP) perturbations to input prompts which may sizably degrade personalization especially in the context of recommendations. Similarly, [34] apply multiple rounds of text sanitization to remove sensitive information, potentially further limiting recommendation relevance. [40] propose a split-processing approach that adds noise to token embeddings for local differential privacy which potentially severely affects recommendation relevance. In general, all approaches degrade personalization, as modifying product names, brands, key details, embeddings, etc. will disrupt the LLM's ability to generate tailored recommendations [42].

[40] propose a split-processing approach that adds noise to token embeddings for local differential privacy before denoising in the cloud. However, this technique is incompatible with text-based LLMs, as it relies on embedding perturbations rather than direct text obfuscation, making it ineffective for protecting sensitive textual inputs such as purchase history.

Several privacy techniques have been proposed in traditional recommendation systems, some of which could theoretically be adapted to LLM-based recommenders [8, 18, 33, 59, 61]. Obfuscation-based approaches, such as AdNauseam [33], disrupt user profiling by introducing noise (e.g., clicking random ads)Recent advancements in privacy-preserving obfuscation for structured-data-driven recommendation systems include PBooster [8], which injects interest-aligned noise into user interactions, and Harpo [61], which applies reinforcement learning to selectively obfuscate preferences while preserving core interests. De-Harpo [59] refines this process with denoising mechanisms to filter out irrelevant obfuscations. While these methods maintain recommendation quality in traditional systems, they do not directly transfer to LLM-based recommenders due to the unstructured nature of LLM inputs and their reliance on free-text representations. Recent studies confirm that standard obfuscation methods struggle to maintain both privacy and personalization in LLM-powered systems [60].

Beyond obfuscation, other privacy-preserving strategies such as federated learning (FL) [41], and encryption [10, 31] have been explored. FL has been widely studied for training LLM-based recommender models while preserving user privacy [57, 58, 62], but its application to inference in LLM-based systems remains under-explored. Encryption-based approaches such as CipherGPT [31]

---

[2]Llama 3.2 1B supports a maximum input of 128,000 tokens and generates up to 2,048 tokens per response.

employ homomorphic encryption for private inference, but such solutions are impractical for LLM-based recommendation due to high computational costs, communication overhead, and incompatibility with server-hosted LLM APIs. Similarly, other encrypted processing techniques [10] require model modifications on the server side, which is infeasible for most cloud-based LLM services.

## 10 Conclusion

We proposed a hybrid privacy-preserving framework for LLM-based recommendation systems that balances user privacy with recommendation utility. Our approach first identifies and removes products that may reveal sensitive information through a fine-tuned BERT-based obfuscator module. By filtering out sensitive products from the modified purchase history sent to the server-side recommender, our method offers meaningful privacy. However, this degrades recommendation quality. To address this, we introduce local deobfuscation, a complementary step that recovers sensitive-category relevance without transmitting privacy-critical data to the server. Our experiments highlight the system's effectiveness in aligning recommendations with the original distribution for both sensitive and non-sensitive items. Additionally, hardware evaluations confirm the feasibility of our approach on standard consumer devices. These findings underscore the practical viability of our method in real-world scenarios, where users demand both privacy protection and high-quality recommendations.

## References

[1] Armin Abdollahi, Mehdi Kamal, and Massoud Pedram. 2025. RocketPPA: Ultra-Fast LLM-Based PPA Estimator at Code-Level Abstraction. *arXiv preprint arXiv:2503.21971* (2025).

[2] Meta AI. 2024. Llama 3.2 Model Card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md. Accessed: 2025-02-01.

[3] Meta AI. 2024. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/ ccessed: 2025-02-08..

[4] Amazon. 2025. Amazon.com Privacy Notice. https://www.amazon.com/gp/help/customer/display.html?nodeId=GX7NJQ4ZB8MHFRNJ. Accessed: 2025-01-25.

[5] Amazon. 2025. Amazon.com. Spend less. Smile more. https://www.amazon.com/ Accessed: 2025-01-25.

[6] Anthropic. 2025. Claude: Anthropic's AI Assistant. https://www.anthropic.com/claude Accessed: 2025-02-08.

[7] Seyedarmin Azizi, Mohammad Erfan Sadeghi, Mehdi Kamal, and Massoud Pedram. 2024. Efficient Noise Mitigation for Enhancing Inference Accuracy in DNNs on Mixed-Signal Accelerators. *arXiv preprint arXiv:2409.18553* (2024).

[8] Ghazaleh Beigi, Ruocheng Guo, Alexander Nou, Yanchao Zhang, and Huan Liu. 2019. Protecting user privacy: An approach for untraceable web browsing history and unambiguous user profiles. In *Proceedings of the twelfth ACM international conference on web search and data mining*. 213–221.

[9] Aldo Gael Carranza, Rezsa Farahani, Natalia Ponomareva, Alex Kurakin, Matthew Jagielski, and Milad Nasr. 2023. Synthetic Query Generation for Privacy-Preserving Deep Retrieval Systems using Differentially Private Language Models. *arXiv preprint arXiv:2305.05973* (2023).

[10] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216* (2022).

[11] Yu Chen, Tingxin Li, Huiming Liu, and Yang Yu. 2023. Hide and seek (has): A lightweight framework for prompt privacy protection. *arXiv preprint arXiv:2309.03057* (2023).

[12] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems* (Boston, Massachusetts, USA) *(RecSys '16)*. Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[13] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in

[14] Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2024. Security and privacy challenges of large language models: A survey. *Comput. Surveys* (2024).

[15] Google DeepMind. 2025. Gemini: Google's Multimodal AI Model. https://deepmind.google/technologies/gemini Accessed: 2025-02-08.

[16] Chroma Developers. 2025. Chroma: The AI-native open-source embedding database. https://github.com/chroma-core/chroma. Accessed: 2025-02-23.

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Jill Burstein, Christy Doran, and Thamar Solorio (Eds.). Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[18] Angela Di Fazio. 2024. Enhancing Privacy in Recommender Systems through Differential Privacy Techniques. In *Proceedings of the 18th ACM Conference on Recommender Systems* (Bari, Italy) *(RecSys '24)*. Association for Computing Machinery, New York, NY, USA, 1348–1352. https://doi.org/10.1145/3640457.3688019

[19] Dario Di Palma, Giovanni Maria Biancofiore, Vito Walter Anelli, Fedelucio Narducci, Tommaso Di Noia, and Eugenio Di Sciascio. 2023. Evaluating chatgpt as a recommender system: A rigorous approach. *arXiv preprint arXiv:2309.03613* (2023).

[20] Cynthia Dwork. 2006. Differential Privacy. In *International Colloquium on Automata, Languages and Programming*. https://api.semanticscholar.org/CorpusID:2565493

[21] Hugging Face. 2025. meta-llama/Llama-3.2-1B. https://huggingface.co/meta-llama/Llama-3.2-1B Accessed: 2025-01-03.

[22] Hugging Face. 2025. sentence-transformers/all-MiniLM-L6-v2. https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2 Accessed: 2025-01-03.

[23] Arya Fayyazi, Mehdi Kamal, and Massoud Pedram. 2025. FACTER: Fairness-Aware Conformal Thresholding and Prompt Engineering for Enabling Fair LLM-Based Recommender Systems. *arXiv preprint arXiv:2502.02966* (2025).

[24] Yue Feng, Shuchang Liu, Zhenghai Xue, Qingpeng Cai, Lantao Hu, Peng Jiang, Kun Gai, and Fei Sun. 2023. A large language model enhanced conversational recommender system. *arXiv preprint arXiv:2308.06212* (2023).

[25] Luke Friedman, Sameer Ahuja, David Allen, Zhenning Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. 2023. Leveraging large language models in conversational recommender systems. *arXiv preprint arXiv:2305.07961* (2023).

[26] Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524* (2023).

[27] Google Shopping - Shop Online, Compare Prices &amp; Where to Buy. 2025. google-shopping. https://shopping.google.com/ Accessed: 2025-01-25.

[28] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.

[29] Yunlong He, Lingtao Wei, Fei Chen, Hanlin Zhang, Jia Yu, and Haiyang Wang. 2025. Fedai: Federated recommendation system with anonymized interactions. *Expert Syst. Appl.* 271 (2025), 126564. https://api.semanticscholar.org/CorpusID:275768413

[30] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*. 720–730.

[31] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen jie Lu, Cheng Hong, and Kui Ren. 2023. CipherGPT: Secure Two-Party GPT Inference. Cryptology ePrint Archive, Paper 2023/1147. https://eprint.iacr.org/2023/1147

[32] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952* (2024).

[33] Daniel C. Howe and Helen Nissenbaum. 2017. Engineering Privacy and Protest: A Case Study of AdNauseam. In *IWPE@SP*. https://api.semanticscholar.org/CorpusID:26168377

[34] 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 preprint arXiv:2306.08223* (2023).

[35] Omnia Kelany, Sherin Aly, and Mohamed A. Ismail. 2020. Deep Learning Model for Financial Time Series Prediction. In *2020 14th International Conference on Innovations in Information Technology (IIT)*. 120–125. https://doi.org/10.1109/IIT50501.2020.9299063

[36] Tina Khezresmaeilzadeh, Elaine Zhu, Kiersten Grieco, Daniel J Dubois, Konstantinos Psounis, and David Choffnes. 2024. Echoes of Privacy: Uncovering the Profiling Practices of Voice Assistants. *arXiv preprint arXiv:2409.07444* (2024).

[37] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang,

and Weinan Zhang. 2025. How Can Recommender Systems Benefit from Large Language Models: A Survey. *ACM Trans. Inf. Syst.* 43, 2, Article 28 (Jan. 2025), 47 pages. https://doi.org/10.1145/3678004

[38] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[39] Sichun Luo, Wei Shao, Yuxuan Yao, Jian Xu, Mingyang Liu, Qintong Li, Bowei He, Maolin Wang, Guanzhi Deng, Hanxu Hou, et al. 2024. Privacy in LLM-based Recommendation: Recent Advances and Future Directions. *arXiv preprint arXiv:2406.01363* (2024).

[40] Peihua Mai, Ran Yan, Zhe Huang, Youjia Yang, and Yan Pang. 2023. Split-and-denoise: Protect large language model inference with local differential privacy. *arXiv preprint arXiv:2310.09130* (2023).

[41] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[42] Peter Müllner, Elisabeth Lex, Markus Schedl, and Dominik Kowald. 2024. The impact of differential privacy on recommendation accuracy and popularity bias. In *European Conference on Information Retrieval*. Springer, 466–482.

[43] OpenAI. 2025. ChatGPT-4o. Large Language Model developed by OpenAI. Available at https://openai.com/.

[44] OpenAI. 2025. ChatGPT: OpenAI's Conversational AI Model. https://openai.com/chatgpt Accessed: 2025-02-08.

[45] Parsa Razmara, Tina Khezresmaeilzadeh, and B Keith Jenkins. 2024. Fever detection with infrared thermography: Enhancing accuracy through machine learning techniques. *arXiv preprint arXiv:2407.15302* (2024).

[46] Amazon Web Services. 2024. Selecting the Right Metadata to Build High-Performing Recommendation Models with Amazon Personalize. https://aws.amazon.com/blogs/machine-learning/selecting-the-right-metadata-to-build-high-performing-recommendation-models-with-amazon-personalize/ Accessed: 2025-02-10.

[47] Erez Shmueli and Tamir Tassa. 2020. Mediated Secure Multi-Party Protocols for Collaborative Filtering. *ACM Trans. Intell. Syst. Technol.* 11, 2, Article 15 (Feb. 2020), 25 pages. https://doi.org/10.1145/3375402

[48] Robin Staab, Mark Vero, Mislav Balunović, and Martin Vechev. 2023. Beyond memorization: Violating privacy via inference with large language models. *arXiv preprint arXiv:2310.07298* (2023).

[49] Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. 2024. Is Cosine-Similarity of Embeddings Really About Similarity?. In *Companion Proceedings of the ACM Web Conference 2024* (Singapore, Singapore) *(WWW '24)*. Association for Computing Machinery, New York, NY, USA, 887–890. https://doi.org/10.1145/3589335.3651526

[50] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification?. In *Chinese computational linguistics: 18th China national conference, CCL 2019, Kunming, China, October 18–20, 2019, proceedings 18*. Springer, 194–206.

[51] Latanya Sweeney. 2002. k-anonymity: A model for protecting privacy. *International journal of uncertainty, fuzziness and knowledge-based systems* 10, 05 (2002), 557–570.

[52] Meng Tong, Kejiang Chen, Jie Zhang, Yuang Qi, Weiming Zhang, Nenghai Yu, Tianwei Zhang, and Zhikun Zhang. 2023. InferDPT: Privacy-preserving inference for black-box large language model. *arXiv preprint arXiv:2310.12214* (2023).

[53] U.S. Department of Health & Human Services. 1996. Health Insurance Portability and Accountability Act (HIPAA). https://www.hhs.gov/hipaa Accessed: February 9, 2025.

[54] Shang Wang, Tianqing Zhu, Bo Liu, Ming Ding, Xu Guo, Dayong Ye, Wanlei Zhou, and Philip S Yu. 2024. Unique security and privacy threats of large language model: A comprehensive survey. *arXiv preprint arXiv:2406.07973* (2024).

[55] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, et al. 2024. A survey on large language models for recommendation. *World Wide Web* 27, 5 (2024), 60.

[56] Lanling Xu, Junjie Zhang, Bingqian Li, Jinpeng Wang, Sheng Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2025. Tapping the Potential of Large Language Models as Recommender Systems: A Comprehensive Framework and Empirical Analysis. arXiv:2401.04997 [cs.IR] https://arxiv.org/abs/2401.04997

[57] Huimin Zeng, Zhenrui Yue, Qian Jiang, and Dong Wang. 2024. Federated recommendation via hybrid retrieval augmented generation. In *2024 IEEE International Conference on Big Data (BigData)*. IEEE, 8078–8087.

[58] Chunxu Zhang, Guodong Long, Hongkuan Guo, Xiao Fang, Yang Song, Zhaojie Liu, Guorui Zhou, Zijian Zhang, Yang Liu, and Bo Yang. 2024. Federated adaptation for foundation model-based recommendations. *arXiv preprint arXiv:2405.04840* (2024).

[59] Jiang Zhang, Hadi Askari, Konstantinos Psounis, and Zubair Shafiq. 2022. A Utility-Preserving Obfuscation Approach for YouTube Recommendations. *arXiv preprint arXiv:2210.08136* (2022).

[60] Jinghao Zhang, Yuting Liu, Qiang Liu, Shu Wu, Guibing Guo, and Liang Wang. 2024. Stealthy attack on large language model based recommendation. *arXiv*

[61] preprint arXiv:2402.14836 (2024).

[61] Jiang Zhang, Konstantinos Psounis, Muhammad Haroon, and Zubair Shafiq. 2021. Harpo: Learning to subvert online behavioral advertising. *arXiv preprint arXiv:2111.05792* (2021).

[62] Jujia Zhao, Wenjie Wang, Chen Xu, Zhaochun Ren, See-Kiong Ng, and Tat-Seng Chua. 2024. Llm-based federated recommendation. *arXiv preprint arXiv:2402.09959* (2024).

[63] Yandong Zheng, Hui Zhu, Rongxing Lu, Songnian Zhang, Fengwei Wang, Jun Shao, and Hui Li. 2025. Efficient and Privacy-Preserving Weighted Range Set Sampling in Cloud. *IEEE Transactions on Dependable and Secure Computing* 22, 1 (2025), 534–548. https://doi.org/10.1109/TDSC.2024.3408816

[64] Erion Çano and Maurizio Morisio. 2017. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis* 21, 6 (Nov. 2017), 1487–1524. https://doi.org/10.3233/ida-163209

# A  Prompts

## A.1  Prompt for Server-Based Recommendation System

**Task:** Based on the user's purchase history provided below, recommend $n_{ns}$ product descriptions. Ensure that:

(1) The proportion of product categories in the recommendations matches the proportion in the purchase history. For example, if 50% of the purchases are from the 'fashion' category, 50% of the recommendations should also be from 'fashion.'

(2) All categories present in the purchase history are represented in the recommendations.

(3) The output consists of a numbered list of the recommended product descriptions.

Provide only the final recommendations as a numbered list.

**User's purchase history:** ...

## A.2  Prompt for Local Deobfuscator

**Task:** You are an assistant providing product recommendations. Based on the items listed below, suggest only $n_s$ other products that are relevant and specific to similar needs.

**User's purchase history:** ...

## A.3  Prompt for Product Sensitivity Classification

To classify product sensitivity, we used few-shot prompting with ChatGPT-4o. The model assigns a label of either "nonsensitive" or "sensitive" to each given product description. The following prompt was used:

**Task:** Assign a label ('"nonsensitive"' or '"sensitive"') to a product based on its implications for medical conditions.
**Classification Criteria:**

- **nonsensitive** → Products that a normal person may buy for general use.
  Includes common wellness, personal care, and household items that do not imply a medical condition.
  *Examples:* "multivitamins", "herbal supplements", "general skincare", "fluoride toothpaste", "protein powder".
- **sensitive** → Products that imply a medical condition or health concern.
  Includes anything specifically marketed for symptom relief, condition management, or treatment.
  *Examples:* "prescription medications", "medical devices", "condition-specific supplements".

**Examples:**

- *Product:* "Daily multivitamin with Vitamin D & Zinc"
  *Label:* "nonsensitive"
- *Product:* "Fragrance-free laundry detergent for sensitive skin"
  *Label:* "nonsensitive"
- *Product:* "Joint pain relief cream with menthol"
  *Label:* "sensitive"
- *Product:* "Blood glucose monitoring kit for diabetes"
  *Label:* "sensitive"
- *Product:* "Prescription-strength corticosteroid cream for eczema"
  *Label:* "sensitive"

**Instruction:** Write only the label. Do not write anything else.
**Here is the product to classify:**

**Task:** Assign a sensitivity score between 0 and 1 to a product based on its implications for medical conditions.
**Scoring Criteria:**

- **0.0** → General-use products with no medical implications (e.g., vitamins, cosmetics, general wellness items).
  *Example:* "Multivitamins", "herbal teas", "fluoride toothpaste".
- **0.1 - 0.4** → Health-related products not tied to a specific medical condition (e.g., sleep aids, general pain relief).
  *Example:* "Fragrance-free detergent for sensitive skin" (0.3), "melatonin gummies" (0.4).
- **0.5 - 0.7** → Products suggesting a potential health concern but commonly used for wellness (e.g., symptom relief, targeted supplements).
  *Example:* "Liver detox supplements" (0.7), "pain-relief patches" (0.6).
- **0.8 - 1.0** → Products designed for treating, monitoring, or managing a specific medical condition (e.g., prescription drugs, medical aids).
  *Example:* "Blood glucose monitor" (1.0), "prescription eczema cream" (1.0).

**Examples:**

- "Moisturizing hand lotion with aloe vera" → 0.0 (General-use cosmetic, no sensitivity.)
- "Multivitamin with Vitamin D & Zinc for daily health" → 0.0 (A normal purchase for general wellness.)
- "Fragrance-free laundry detergent for sensitive skin" → 0.3 (Health-related but not condition-specific.)
- "Melatonin sleep aid gummies" → 0.4 (Used for sleep but not strictly medical.)
- "Prescription-strength corticosteroid cream for eczema relief" → 1.0 (Directly treats a medical condition.)

**Instruction:** Write only the score. Do not write anything else.
**Here is the product to score:**

## A.4 Prompt for Sensitivity Score Assignment

To assign sensitivity scores to products, we use few-shot prompting with ChatGPT-4o. The following prompt was used: