Snapcase – Regain Control over Your Predictions with Low-Latency Machine Unlearning

Sebastian Schelter
BIFOLD & TU Berlin
schelter@tu-berlin.de

Stefan Grafberger
BIFOLD & TU Berlin
grafberger@tu-berlin.de

Maarten de Rijke
University of Amsterdam
m.derijke@uva.nl

ABSTRACT

The "right-to-be-forgotten" requires the removal of personal data from trained machine learning (ML) models with machine unlearning. Conducting such unlearning with low latency is crucial for responsible data management. Low-latency unlearning is challenging, but possible for certain classes of ML models when treating them as "materialised views" over training data, with carefully chosen operations and data structures for computing updates.

We present Snapcase, a recommender system that can unlearn user interactions with sub-second latency on a large grocery shopping dataset with 33 million purchases and 200 thousand users. Its implementation is based on incremental view maintenance with Differential Dataflow and a custom algorithm and data structure for maintaining a top-k aggregation over the result of a sparse matrix-matrix multiplication. We demonstrate how interactive low-latency unlearning empowers users in critical scenarios to get rid of sensitive items in their recommendations and to drastically reduce their data's negative influence on other users' predictions.

PVLDB Reference Format:
Sebastian Schelter, Stefan Grafberger, and Maarten de Rijke. Snapcase – Regain Control over Your Predictions with Low-Latency Machine Unlearning. PVLDB, 17(12): 4273 - 4276, 2024.
doi:10.14778/3685800.3685853

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/deem-data/snapcase.

1 INTRODUCTION

Recent law such as the "right to be forgotten" in Europe requires organisations to delete personal user data upon request: "The data subject shall have the right to [...] the erasure of personal data [...] where the data subject withdraws consent." It is insufficient to only delete personal data from primary data stores though. Personal data must also be removed from machine learning models (which may have learned representations of the personal data at training time), giving rise to the problem of *machine unlearning* [9].

The need for low-latency machine unlearning. The data management community emphasized that it is important to provide *low-latency unlearning* [11–13, 16] functionality to users. This does not only allow users to control their personal data usage, but also empowers them to adjust their personal predictions from ML applications in an interactive manner. Existing systems lack this functionality, which can lead to devastating consequences [15].

As an example, imagine a person struggling with alcohol addiction, who is making the decision to get sober and stop consuming alcoholic products. Unfortunately, this person will still be continually exposed to ads and recommendations for alcohol products when using online services, since the underlying ML models will have learned their preference for alcohol from their past consumption patterns. Empowering this person to immediately adjust their recommendations and ads via low-latency unlearning might help reduce their probability of relapsing to their addiction.

This example inspires our proposed demonstration, where we show that low-latency unlearning is possible even in challenging ML use cases with large datasets, and that it allows users to quickly mitigate the impact of unwanted past consumption behavior. The advantage of low-latency unlearning is that it does not require changes to the underlying ML model and, at the same time, puts users in control of their predictions and the (potentially negative) impact of their data on themselves and other users. In our experience, the exact details of what users want to have unlearned and removed from their predictions depend on personal ethics and the context, and are thereby difficult to capture by hardcoded filters in existing platforms.

Machine learning models as materialised views over training data. On a technical level, our vision for low-latency unlearning is to treat ML models as materialised views over their training data [10]. Unlearning personal input data then corresponds to incrementally maintaining the materialised ML model in response to deletions. Unfortunately, existing incremental view maintenance (IVM) techniques are computationally prohibitive for many ML models, where small changes in the input can result in large, expensive model updates due to global aggregations and non-linear operations [9]. However, efficient machine unlearning is an active area of research with recent successes for certain classes of models, e.g., tree-based models [11, 16] and nearest neighbor models [12], when carefully designed update operations and data structures are used.

Snapcase. In this paper, we present the Snapcase system, a reference implementation of our vision of low-latency unlearning for ML models via IVM (Section 3). Snapcase is a recommender system for online grocery shopping, based on a state-of-the-art algorithm for next-basket recommendation [4], which is in production use for several online grocery shopping platforms from our industry partner [14]. Snapcase holds the customers' purchase data in a relational database, maintains the recommendation model as a materialised view over this data, and can update its recommendation model with sub-second latency in response to deletions

Proceedings of the VLDB Endowment, Vol. 17, No. 12 ISSN 2150-8097.
doi:10.14778/3685800.3685853

in the database. Model maintenance is implemented as an incrementalised dataflow program in Differential Dataflow [6], and our system additionally integrates a custom data structure [12] for maintaining an expensive top-k aggregation over the result of a sparse matrix-matrix multiplication with millions of entries (which is the computational bottleneck in unlearning for this model). We provide the source code for Snapcase at https://github.com/deem-data/snapcase.

Demonstration. We demonstrate Snapcase using a large grocery shopping dataset with more than 33 million purchases and more than 200 thousand users from the Instacart platform (a grocery delivery service in the US). This dataset is representative for real-world recommendation workloads in online grocery shopping, since its catalog size matches or exceeds the catalog sizes of large grocery shopping chains in the US and Europe [1, 3, 8].

Scenarios. We offer *three responsible data management scenarios* to attendees, centered around consumption behavior with respect to alcohol addiction, obesity, and a high carbon footprint. In each scenario, we assume that the user wants to make an ethically motivated positive life decision and get rid of *sensitive items* (e.g., wine and liquor in the alcohol addiction scenario) from their purchase data and recommendations, and reduce their negative influence on other users' recommendations as well.

Interactive unlearning. Attendees choose a scenario and can then inspect the sensitive items and categories in the user's purchases, model state, and personalised recommendations. Next, attendees can interactively trigger low-latency machine unlearning for sensitive item purchases of the scenario user. They will be able to inspect the detailed changes to the model state and recommendations resulting from the IVM updates, and they will observe how the user's model state and recommendations gradually lose the connection to sensitive items and categories. If the correct purchases are unlearned, sensitive items will completely disappear from the recommendations, and the user will also stop negatively influencing the recommendations of other users.

2 BACKGROUND

We briefly introduce background knowledge required for the remainder of the paper.

Next-basket recommendation. The ML use case in our demonstration is next-basket recommendation (NBR). The input data for NBR is historical purchase information $\{\mathbf{b}_{u_1}, \mathbf{b}_{u_2}, \ldots, \mathbf{b}_{u_n}\}$ of a user u, where the binary vector $\mathbf{b}_{u_j} \in \{0,1\}^{|I|}$ in the item space I denotes the set of items bought together in the j-th shopping basket. The goal of NBR models is to predict a user's next set of items $\mathbf{b}_{u_{n+1}}$ (i.e., the next shopping basket). Nearest-neighbor methods that take the repeat behavior of grocery shopping into account dominate this field [5] and outperform neural approaches [2].

We use the TIFU [4] algorithm, which models the temporal dynamics of the frequency information in the users' past baskets and conducts a hierarchical aggregation to create a sparse embedding $\phi_{\mathbf{u}}$ per user u. Next, it computes the top-k neighborhood graph over the embeddings of all users and uses this graph to create recommendations for a user u as a linear combination $\alpha\,\phi_u + (1-\alpha)\sum_{j \in N_u}\phi_j$

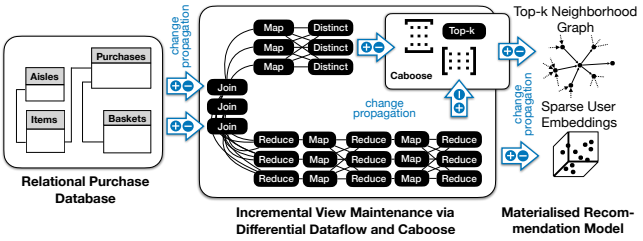


Figure 1: Overview of the Snapcase system – a recommendation model is maintained as a materialised view over a relational database via Differental Dataflow and Caboose.

of the users' embedding and the embeddings of the k nearest neighbors N_u of u in embedding space.

Efficient unlearning for nearest neighbor-based recommendation. The computational challenge in unlearning for the TIFU model is the maintenance of the top-k neighborhood graph under deletions. In earlier work, we implemented this computation with locality-sensitive hashing [9], but ran into severe scalability issues for larger datasets. Therefore, we recently developed a custom algorithm and data structure called Caboose [12], which efficiently maintains an expensive top-k aggregation over the result of a sparse matrix-matrix multiplication. Caboose can update this aggregation result in less than a second even for input matrices with a million rows [12] .

Differential Dataflow. Differential Dataflow [6, 7] executes data-parallel dataflow computations over partitioned data, with support for user-defined functions applied in standard operators such as map, join, filter, and group-by. It provides automatic incrementalisation of programs built from these operators, by internally treating data collections as a monotonically growing set of update records and using corresponding delta-based operator implementations, which only perform work as input collections change and efficiently compute the resulting changes to their output collections.

3 SYSTEM OVERVIEW

Overview. Snapcase is a Rust-based reference implementation for maintaining an ML model as a materialised view over a database, focusing on the challenging use case of next-basket recommendation. Its architecture is shown in Figure 1: the purchase data of users is held in various tables of a traditional relational database. The recommendation model (Section 2) is implemented as a combination of an incrementalised dataflow program in Differential Dataflow and our Caboose data structure. Model maintenance requires maintaining the sparse embeddings as well as the top-k neighborhood graph between user embeddings.

Machine unlearning as incremental view maintenance. Snapcase assumes that the relational database is under the control of an external e-commerce application. Deletions in the database are propagated to our system via change data capture and trigger the immediate update of the recommendation model via low-latency unlearning with IVM. The sparse user embeddings require a hierarchical aggregation over the shopping baskets with time-decayed weights [4], which we model with a sequence of map and reduce operations in our dataflow program. Maintenance of these embeddings

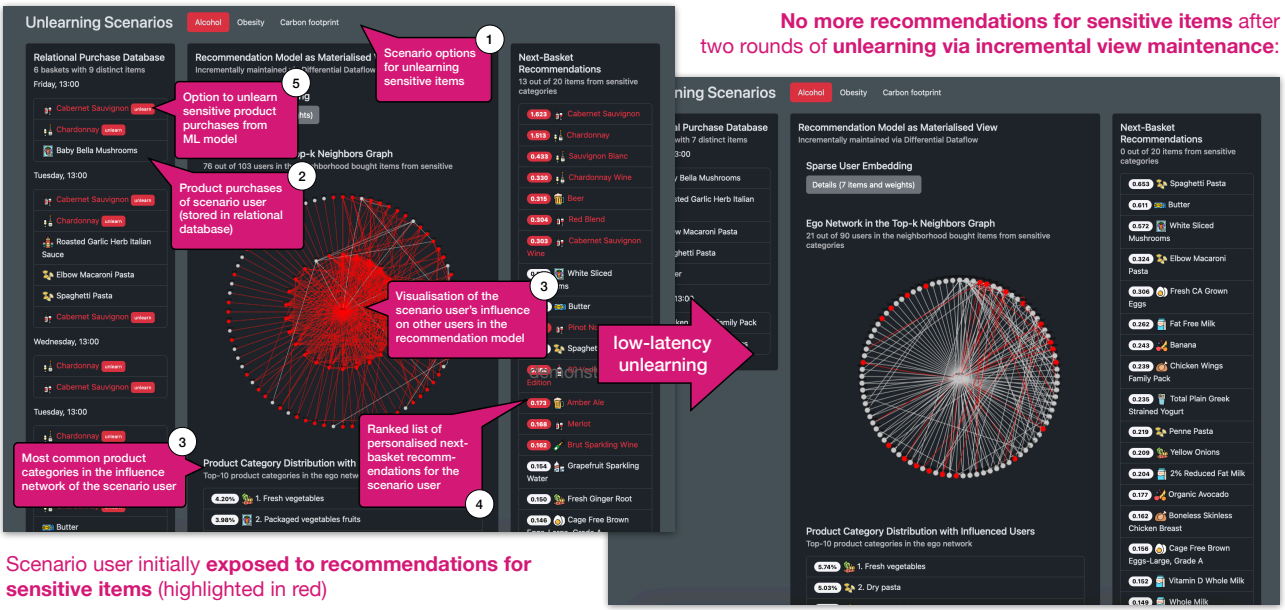


Figure 2: Demonstration interface for the addiction scenario, where the scenario user bought alcoholic items, is subsequently exposed to further alcoholic items in their recommendations, and is closely connected to other users in the dataset, who also consume alcoholic items. Attendees trigger low-latency unlearning of alcohol purchases for the scenario user and will observe how this gradually changes their recommendations and corresponding model state to get rid of the sensitive alcohol items.

is fast, as updates can be isolated per user. The final embeddings are held in an in-memory hashmap, which mirrors the changes in the indexed state from Differential Dataflow [7]. The top-k neighborhood graph is more difficult to maintain, since the deletion of a single interaction with a single item from a user history potentially impacts the top-k neighbors of all users who share one item with the user to unlearn for. The set of potentially impacted users can often be as large as half of the existing user base due to popular items in the heavy-tailed interaction distribution common for recommendation data. Therefore, we use Differential Dataflow to compute the changes to the embeddings and delegate the maintenance of the top-k neighbors graph to Caboose. The actual recommendations are not materialised, but computed online. This is reasonable from a latency perspective since the recommendations only require a linear combination of already materialised embedding vectors, followed by a sorting operation for the most highly weighted items.

4 DEMONSTRATION DETAILS

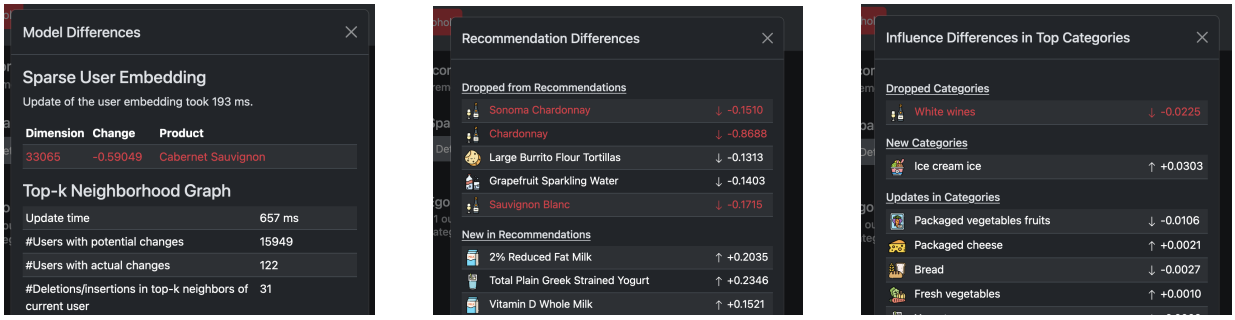
For our demonstration, we run Snapcase in a Rust-based web socket server, which asynchronously communicates with a Javascript-based web interface for the attendees. We use a large dataset of grocery shopping purchase data[1] from the Instacart delivery service in the US. The dataset contains 33,819,106 purchases from 206,209 users in 3,421,083 shopping baskets, for a product catalog of 49,685 distinct items. The top-k neighborhood graph maintained in Snapcase for this dataset contains around 10 million edges. This dataset is representative of real-world workloads, since the catalog size matches or exceeds publicly reported catalog sizes from large grocery shopping chains in the US and Europe [1, 3, 8].

Scenarios. We center our demonstration around *three responsible data management scenarios*. In each scenario, we assume that a user decides to make an ethically motivated positive change in their life, with regard to their consumption behavior. In particular, we focus on persons struggling with *addiction*, who want to stop consuming alcohol, persons with *obesity* problems, who want to stop consuming unhealthy food, and persons who want to stop consuming meat products for ecological reasons to reduce their *carbon footprint*. We refer to the items that a person wants to stop consuming as *sensitive items* and assume that these items are part of their purchase history on online platforms due to past consumption. As a result, the person will still be exposed to such sensitive items by the recommender system on online grocery shopping platforms (which learned their past behavior). Furthermore, their purchase history also contributes to other users being given recommendations for the undesirable sensitive items.

Our demonstration is meant to showcase that low-latency unlearning functionality empowers users of online platforms to instantaneously curate their recommendations to remove sensitive items and, at the same time, drastically reduce their negative influence on other users' recommendations. We use an off-the-shelf NBR model from existing research [4], which has no notion of sensitive items.

Demonstration. We detail our proposed demonstration and the provided interface for attendees (see Figure 2 for an example of a user deciding to stop consuming alcoholic products). At the beginning of the demonstration, attendees pick the scenario (addiction, obesity, carbon footprint) of their choice (1). For each chosen scenario, we visualise the data of a sample user from the Instacart

[1] https://www.kaggle.com/c/instacart-market-basket-analysis



(a) Model update times and changes. **(b) Recommendation changes.** **(c) Changes in influence of other users.**

Figure 3: Attendees are presented with details on the changes and update times incurred by unlearning for the materialised recommendation model, the item recommendations and the influence of the scenario user on other users.

dataset, who bought items from the corresponding sensitive product categories and receives recommendations for sensitive items from the recommendation model.

Inspecting the purchase history, model state and recommendations for the scenario. After choosing the scenario, attendees can inspect the database and model state for the scenario user in detail. They can inspect the shopping basket history of the user (2), as stored in the relational database, with items from sensitive categories highlighted in red. Furthermore, they can inspect the current state of the materialised recommendation model for the scenario user (3), including the user's sparse embedding representation, the ego network (all one-hop connections) of the user in the top-k neighborhood graph (where we also highlight connected users in red, who bought items from sensitive categories) and a list of the top-10 product categories present in the purchase histories of the connected users. (4) On the right side, attendees are presented with a list of the top-20 next basket recommendations for the scenario user, where we again highlight sensitive items in red. Figure 2 shows this for the alcohol scenario: repeated purchases of different wines lead to an ego network for the scenario user which is dominated by other users who consumed alcohol. As a consequence, the recommendations for this scenario user are also dominated by alcoholic items.

Unlearning purchases to rid the user of recommendations for sensitive items. In the main part of the demonstration, attendees can interactively trigger the low-latency unlearning of item interactions for the scenario user (5). This results in the deletion of the sensitive items from all the user's shopping baskets in the relational database, which subsequently propagates the changes to Differential Dataflow to incrementally maintain the recommendation model. Attendees are presented with a detailed dialogue, which shows the individual changes and IVM update times for the database, the recommendation model (Figure 3a) and the recommendations (Figure 3b). They can observe the impact of the deletion on both the scenario user and users influenced by them (Figure 3c). The statistics shown also highlight the computational difficulty of unlearning, as the top-k connections of several thousands of users have to be inspected for unlearning a single item interaction. Most importantly, attendees can observe how unlearning sensitive items gradually changes the model state for the scenario user by reducing the amount of sensitive items in their recommendations, and by removing sensitive categories from the user's influence network. Attendees will experience that repeated rounds of unlearning often completely remove the exposure to sensitive items, by moving the scenario user to a different point in the model space. We see this in Figure 2 (right side), where the ego network in the neighborhood graph contains almost no more (red) users with sensitive items after unlearning.

We would additionally like to note that our ego network visualisation also allows attendees to navigate to other users than the preselected scenario users. Furthermore, we provide our demonstration and system code under an open license, which allows other researchers to extend Snapcase to new algorithms and datasets.

REFERENCES

[1] Albert Heijn. 2023. Van Land tot Klant: Onze Ketens. Retrieved July 5, 2024 from https://www.ah.nl/over-ah/duurzaamheid/onze-ketens
[2] Maurizio Ferrari et al. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. *RecSys* (2019).
[3] Fortune. 2010. Inside the Secret World of Trader Joe's. Retrieved July 5, 2024 from https://fortune.com/2010/08/23/inside-the-secret-world-of-trader-joes/
[4] Haoji Hu et al. 2020. Modeling Personalized Item Frequency Information for Next-Basket Recommendation. *SIGIR* (2020).
[5] Ming Li et al. 2023. A Next Basket Recommendation Reality Check. *ACM TOIS* 41, 4 (2023), 1–29.
[6] Frank McSherry et al. 2013. Differential Dataflow. *CIDR* (2013).
[7] Frank McSherry et al. 2020. Shared Arrangements: Practical Inter-query Sharing for Streaming Dataflows. *PVLDB* 13, 10 (2020).
[8] REWE. 2024. Über das Unternehmen - Struktur und Vertriebslinien. Retrieved July 5, 2024 from https://www.rewe-group.com/de/unternehmen/struktur-und-vertriebslinien/rewe/
[9] Sebastian Schelter. 2020. Amnesia - A Selection of Machine Learning Models That Can Forget User Data Very Fast. *CIDR* (2020).
[10] Sebastian Schelter. 2021. Towards Efficient Machine Unlearning via Incremental View Maintenance. *Workshop on Challenges in Deploying and Monitoring ML Systems@ICML* (2021).
[11] Sebastian Schelter et al. 2021. Hedgecut: Maintaining Randomised Trees for Low-latency Machine Unlearning. *SIGMOD* (2021).
[12] Sebastian Schelter et al. 2023. Forget Me Now: Fast and Exact Unlearning in Neighborhood-based Recommendation. *SIGIR* (2023).
[13] Julia Stoyanovich et al. 2022. Responsible Data Management. *Commun. ACM* 65, 6 (2022), 64–74.
[14] Maria Vechtomova et al. 2023. Databricks AI Summit: Streamlining API Deploy ML Models Across Multiple Brands: Ahold Delhaize's Experience on Serverless. Retrieved July 5, 2024 from https://www.youtube.com/watch?v=GSJFyoBiCXk
[15] Vincent Warmerdam. 2021. Beyond Broken. Retrieved July 5, 2024 from https://koaning.io/posts/beyond-broken/
[16] Zhaomin Wu et al. 2023. Deltaboost: Gradient Boosting Decision Trees with Efficient Machine Unlearning. *SIGMOD* (2023).