Federated LLM Recommendations for Retailers:
a privacy-preserving system for enhanced recommendations for small retailers

John Mildinhall Ph.D
formerly Head of Data Science at Marks & Spencer

November 26, 2026

Abstract

Large Language Models (LLMs) are designed to learn sequences. Recommender systems are also inherently sequential models, as the recommender learns to predict the next purchase of a user. This paper demonstrates that LLMs can be used to create recommender systems using LoRA that perform better than a state-of-the-art matrix factorisation algorithm. Deep Learning methods as used by LLMs are also suited to federated learning, where the data from multiple parties is used to create a single model without sharing the underlying data. This paper shows that an LLM recommender can effectively be trained using federated learning from multiple simulated retailers, and that secure aggregation can be used to preserve the privacy of individual retailers' data. The benefit of federated learning is particularly strong for smaller retailers, and there is some evidence that it also improves larger retailers' recommendations.

1 Introduction

In the retail industry transaction line data is a fundamental asset to the business. In large retailers, powerful and complex algorithmic systems are built with this fundamental resource at their heart, including forecasting, pricing and next best action systems. The power of transaction data also makes this resource highly sensitive, and a mature data governance system normally prevents the sharing of raw transaction data outside the retailer.

Recommender systems are also built using transaction line data. They are valuable to both bricks-and-mortar and e-commerce retailers. Bricks-and-mortar retailers use recommender systems to drive business via loyalty schemes and outbound marketing (often email-based). E-commerce retailer also do this, and in addition add recommendation features such as "customers also bought" and "you might also like" carousels typically at the bottom of product listing pages (PLPs) and on homepages. These recommender systems can capture a large amount of value if they are well-designed and implemented. A sizeable retailer may be able to attribute single percentage point increases in revenue to their recommender systems and this can equate to tens of millions of dollars of incremental revenue.

Traditional recommender systems are difficult to implement. The challenges with these systems include:

1. They require extensive efforts in feature engineering to achieve high quality results. This means building features related to the customer such as what the retailer knows about their demographics, their sizing, any preferences they have expressed and their behaviour on apps and websites. It also means building extensive features about products such as attributes (e.g. number of pockets, fabric constituents, fastenings, cuts).

2. Features must be actively maintained. As the product set evolves, the features that are used may become less useful over time.

3. Features are often curated by human choice, and so non-intuitive features may not be sufficiently explored.

4. Smaller retailers often have cold start challenges where they do not have sufficient transactional data to create effective recommendations. Larger retailers also have cold starts when they introduce a new product category.

5. Recommender systems only know their own retailer's product set - they cannot provide insight into what might happen if a new product is sold, and they cannot point to "missing" products in a range.

6. No semantic knowledge of the products. A collaborative-filtering model sees only co-purchase patterns. It can't use titles, descriptions or general knowledge, such as that a swimsuit and sandals go together, or that a new item resembles an old one.

7. multiple pipelines per surface. Retailers often need multiple instances of their recommenders to supply carousels, outbound marketing, and loyalty schemes among others. An LLM-based system could potentially supply all of these surfaces with the same pipeline.

LLM recommenders Large Language Models predict the next token in a sequence of natural language, and because they are trained on text they can also be adapted to other sequences. A customer's purchase history is such a sequence, so it can be written out as text and the next purchase predicted by the same machinery. Early work recast recommendation as language modelling: P5 converted several recommendation tasks into text-to-text prompts for a single pretrained model [8], and TALLRec showed that a language model tuned on fewer than a hundred examples could outperform conventional baselines on movie and book recommendation [3]. Language models can also rank candidate items without any training, although they are sensitive to the order of the history and of the candidates [11], and a survey organises this literature into discriminative and generative uses [21]. In parallel, industrial systems have moved from hand-built features towards sequence models trained directly on user actions, of which Meta's generative recommenders are the largest example [23]. The closest precedent for our setting is GenRec, which adapts an open-source LLM to Netflix data, verbalises user history, item information and context as text, scores the catalogue with a single forward pass per user and reports significant online gains over Netflix's production ranker in a test on part of its traffic [15].

The retail literature is thinner and more recent. Early work fine-tuned a pretrained language model on product text to improve an e-commerce recommender [22]. For fashion, a sequential framework fine-tunes a pretrained LLM with recommendation-specific prompts and parameter-efficient adaptation [16]. Grocery has the strongest production evidence so far: a two-stage LoRA-tuned model that recommends grocery categories reports a 7.5% relative improvement in cart-adds per impression over the production model [25], and a cascaded generative approach to storefront personalisation reports an estimated 2.7% lift in cart adds per page view in online experiments [10]. Each of these studies a single retailer or marketplace training on its own data. We are not aware of work that trains an LLM ranker jointly across several retailers whose transaction tables cannot be pooled, which is the setting of this paper.

The problem These advances assume that one organisation holds the training data. Retailers do not. Each holds its own transaction and customer tables, data governance normally prevents it from sharing them, and a customer who shops at several retailers is fragmented across their separate records with no common identity. GenRec and the retail systems above are therefore single-tenant: each is trained on one organisation's data. A small retailer, or one launching a

new range, has the least data to train on and so stands to gain most from a model trained on more, yet it is the least able to pool its data with others. This paper asks whether a group of retailers can jointly train one LLM-based ranker without sharing their tables, whether doing so improves on each retailer training alone, and what it leaks if they do.

Federated learning Federated learning trains one model on data that stays where it is. In federated averaging, each participant trains the current model on its own data for a number of steps and sends back the updated weights, and a coordinator averages them into the next global model [17]. Only model updates leave the participant, and the method works on data that is not identically distributed across participants, although the heterogeneity can slow or bias training [17, 14]. Settings are commonly divided into cross-device, with very many unreliable clients such as phones, and cross-silo, with a small number of reliable organisations that each hold a large dataset [14]. Our setting, a handful of retailers, is cross-silo. Federated learning has been applied to recommendation since federated collaborative filtering, which keeps each user's interactions on their own device [2], and has recently been extended to LLM-based recommenders. FELLRec fine-tunes an LLM recommender across clients, exchanging low-rank adapters rather than full weights, keeping privacy-sensitive layers on the client and balancing performance across heterogeneous clients [24]. That work, and the cross-device recommenders before it, treat each client as one user; we treat each client as one organisation holding a whole customer base, and we measure the benefit against a local baseline matched for compute. We run our experiments in the Flower framework [5].

Privacy-preserving techniques in federated learning Keeping the data local does not by itself keep it private, because the updates a participant sends can reveal what it trained on. Individual gradients can be inverted to reconstruct training examples [26], and trained models can reveal whether a given record was in the training set [20] or reproduce rare sequences they were trained on [7]. Two families of defence address this. Differential privacy bounds how much any one person's data can change what is released; in deep learning this is done by clipping each example's gradient and adding noise [1], and applied at the level of a user rather than an example in federated language models [18]. It gives a formal guarantee, but the noise costs accuracy. Secure aggregation is a cryptographic protocol that lets the coordinator learn only the sum of the participants' updates and not any single one, and it tolerates participants dropping out [6]; later protocols reduce its cost by communicating over a sparse graph of participants rather than all pairs [4]. It provides no formal bound on what the sum itself reveals. We study both for our setting: customer-level differential privacy is the subject of Section 4, and secure aggregation of the adapter updates, with an empirical assessment of what remains visible, is the subject of Sections 5 and 6.

Contributions.

1. *The benefit of federation is smaller than naive comparisons suggest, and concentrated in small retailers.* Once local and federated training are matched for compute, federation at equal retailer sizes is not established over local-only training (relative MRR change +2.4%, 95% interval $[-5.1\%, +9.9\%]$ over three seeds). With retailers of very different sizes, only the two smallest gain significantly: +52.6% for the 2k-customer retailer and +10.9% for the 10k retailer. Part of the apparent gain in our first, unmatched comparison reflected batch size and not shared knowledge.

2. *Customer-level differential privacy is not viable in this setting.* The best configuration we found ($\varepsilon = 8$) reaches an MRR of 0.046, against 0.083 for the same federated run without noise and 0.081 for local-only training.

3. *Secure aggregation is numerically exact up to quantisation.* SecAgg+ reproduces plaintext aggregation to within a quantisation error of order 10^{-5}, with no saturated coordinates and about seven times headroom on the clipping range.

4. *A pre-registered empirical leakage suite with offline exposure reconstruction.* We reconstruct from the sampling seed how often each customer was in a training cohort, which lets attacks be stratified by exposure without logging. Attacks on the trained model (membership inference, canary extraction, competitor inference) find no leakage within the pre-registered thresholds. A coordinator-side attack on individual updates reaches an AUC of 0.599 on plaintext updates, which SecAgg+ reduces to 0.526 pooled; per-retailer intervals are too wide to meet the threshold.

2 Setting and Method

2.1 Dataset and data preparation

The H&M recommendations dataset is a large real-world example of transaction line data that was used for a Kaggle contest in 2022 [9]. The data contains roughly 31 million real purchases (customer, article, date, price) made between September 2018 and September 2020, together with a catalogue of about 106,000 articles, each described by a product name, type, department, garment group, colour group and a free-text description. The dataset is large enough to stand in for multiple retailers by splitting across the product hierarchy. Different strategies for splitting the data can simulate different retailers - bigger and smaller retailers, retailers with highly differentiated selections to retailers with identical selections. Different splits result in different customer sets across the retailers. Importantly, this strategy reflects reality in that the same customers are represented across multiple (synthetic) retailers.

All experiments use the same split. Within each of the five H&M index groups (Ladieswear, Menswear, Baby/Children, Sport, Divided) we shuffle the articles with a fixed seed and deal them round-robin across five retailers. (Our first design made each retailer one index group; we replaced it before any federated run, because it left the retailers unlike one another and one of them with most of the transactions.) Every retailer thus held a representative one-fifth slice of every category, a plausible generalist retailer rather than a single-department boutique. Each transaction is assigned to the retailer that owns its article. A retailer sees only the transactions that touch its own products, and each retailer re-indexes its customers with a locally scoped random identifier, so no identifier is shared across retailers and no cross-retailer customer matching is ever possible within the system. The simulation harness retains the true shared identity only to measure overlap and to build the centralised reference used in our initial feasibility experiment (Appendix C); no retailer's client code has access to it. In the case of five same-size synthetic retailers, 86.4% of customers appear at two or more retailers and 13.6% at exactly one.

The five retailers are balanced in size by construction, with between 6.12 and 6.45 million transactions each and catalogues of 21,106 to 21,111 articles (Table 1), and they are used as they are in Experiments 1, 1a, 1b and 3 and in the leakage assessment. Because real retailers differ greatly in scale, Experiment 2 also evaluates heterogeneous training-pool sizes, by subsampling each retailer's customers to a chosen pool size, rather than by changing the split.

Verbalisation. In order to create a dataset that was suitable for a Large Language Model, purchase histories of customers were verbalised into narratives. This took the form of a text prompt for each customer that described their behaviour over time, along with a prompt describing the task of the recommender. The target for each prompt is the next article the customer bought;

Table 1: The five synthetic retailers. Each holds about one fifth of the H&M articles in every product category and all transactions for them, from 2018-09-20 to 2020-09-22. Customers are counted per retailer, with locally assigned identifiers.

Retailer	Catalogue (articles)	Transactions (M)	Customers
1	21,111	6.12	984,431
2	21,110	6.44	999,245
3	21,108	6.44	995,769
4	21,107	6.33	992,845
5	21,106	6.45	999,394

```
Customer purchase history, most recent last:
[2019-02-13] Bought: vermont fancy slacks (1) - Trousers (Garment Lower body). department:
Trouser, garment group: Trousers, colour: Light Grey. Ankle-length trousers in woven fabric
with creases, a regular waist, hook-and-eye fastener and zip fly. Diagonal side pockets and
fake welt back pockets. Price band: high. (paid: 0.0508)
[2019-02-13] Bought: Kellis Lurex - Sweater (Garment Upper body). department: Knitwear,
garment group: Knitwear, colour: Beige. Round-neck jumper in an airy, fine-knit viscose blend
containing glittery threads. Long sleeves, slits in the sides and ribbing around the neckline,
cuffs and hem. Price band: high. (paid: 0.0424)
[2019-06-03] Bought: Mister Muscle LS - Shirt (Garment Upper body). department: Shirt,
garment group: Shirts, colour: White. Shirt woven in a stretch cotton blend with a narrow
turn-down collar, French front and a yoke and darts at the back. Long sleeves with adjustable
buttoning at the cuffs, and a rounded hem. Muscle Fit... Price band: high. (paid: 0.0424)
[2020-04-27] Bought: Essex blouse - Blouse (Garment Upper body). department: Blouse, garment
group: Blouses, colour: White. Blouse in an airy crêpe weave with a flounce-trimmed V-neck,
flutter sleeves, buttons down the front, a yoke at the back and gently rounded hem. Price band:
high. (paid: 0.0305)
[2020-05-15] Bought: Kyoto blouse - Blouse (Garment Upper body). department: Blouse, garment
group: Blouses, colour: Light Beige. Blouse in a softly draping weave made from recycled
polyester. Small, gathered stand-up collar, a small opening and button at the back of the neck,
long sleeves with covered elastication at the... Price band: mid. (paid: 0.0424)
Predict the next item this customer will purchase.
```

Figure 1: A verbalised training example from retailer 1 (real data, unedited). The model reads the prompt and is trained to rank the customer's next purchase, here a dark green lined blazer bought on 2020-09-16 ("Burmilla blazer", price band high), above the other items in the retailer's catalogue.

the model learns to place the prompt's vector close to that article's vector. An example of the narrative and target article is given in below.

Each retailer converts its own tables into text locally, using a schema that is common across retailers. A customer's history becomes a prompt with one line per past purchase, giving the purchase date, the text of the purchased item and the price paid, followed by a fixed instruction:

An item's text is built from its stable catalogue attributes only: product name, product type and group, department, garment group, colour group, a description truncated to 200 characters, and a price band (low, mid or high, the tercile of the item's mean transaction price within that retailer's own data). Prices paid appear only on the history side, since they vary over time independently of the item. We keep the 50 most recent purchases and at most 1,700 tokens per history; longer histories are truncated from the left, so that the oldest purchases are dropped and the most recent ones and the instruction are kept. Item texts are limited to 150 tokens. Using text as the common format means that retailers with different schemas do not need an agreed structured feature set, and it gives items that were never seen in training an embedding from

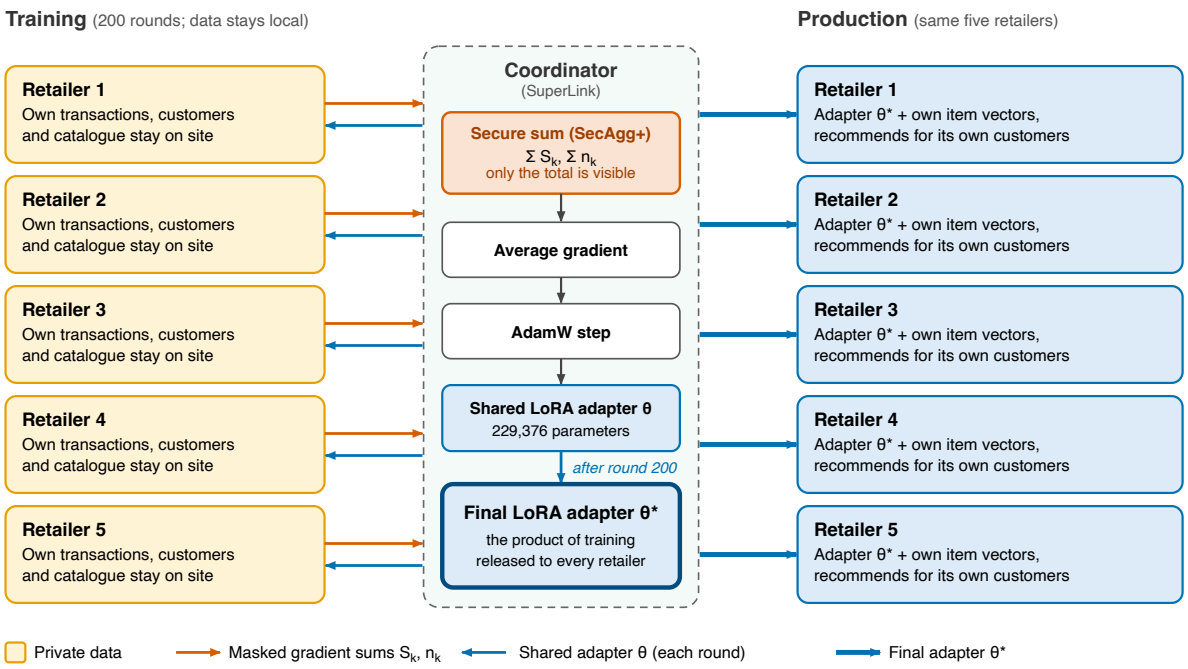


Figure 2: System overview. Five retailers train a shared LoRA adapter without sharing their transactions, customers or catalogues. Each round, a retailer sends its clipped gradient sum and cohort count through secure aggregation (SecAgg+), so the coordinator sees only the totals; the coordinator takes one AdamW step and returns the updated adapter. After the last round the final adapter θ^* is released to every retailer, which uses it locally together with its own item vectors to recommend to its own customers.

their description alone.

2.2 Model

All experiments use Qwen3-1.7B, a performant compact Large Language Model, as the backbone. Two configurations of the recommender are used. The *original configuration* is used only in Experiments 1 and 1a: a rank-16 LoRA adapter (about 17.4 million trainable parameters) on all seven projection matrices in every layer, an item tower trained jointly with the customer tower, in-batch negatives, and a batch of 8 customers per retailer. Because its adapter is large and its objective differs, its results are not comparable with later experiments. The *light configuration* is used in Experiments 1b, 2 and 3 and in the leakage and secure-aggregation experiments, and is the one described in the rest of this section. A customer tower reads a verbalised purchase history and a frozen item tower embeds item descriptions; an item is scored by the scaled dot product of the two vectors. Both towers are built from the same open-weight language model, Qwen3-1.7B [19], which is shared by all retailers and is public. We follow the general approach of GenRec [15], in which an LLM encodes a customer's history and candidate items are ranked in the resulting space, and we adapt it to a setting where the item catalogues of different retailers must stay separate.

Customer tower. The customer tower is the Qwen3-1.7B backbone with low-rank adapters (LoRA) [12]. A history prompt is encoded in a single forward pass and represented by the hidden state of its last non-padding token, giving a vector $u \in \mathbb{R}^d$, where $d = 2048$ is the hidden size of the backbone. The backbone runs in bfloat16 and scores are computed in float32. Only the adapters are trained. We use a deliberately small adapter: rank $r = 4$ (scaling

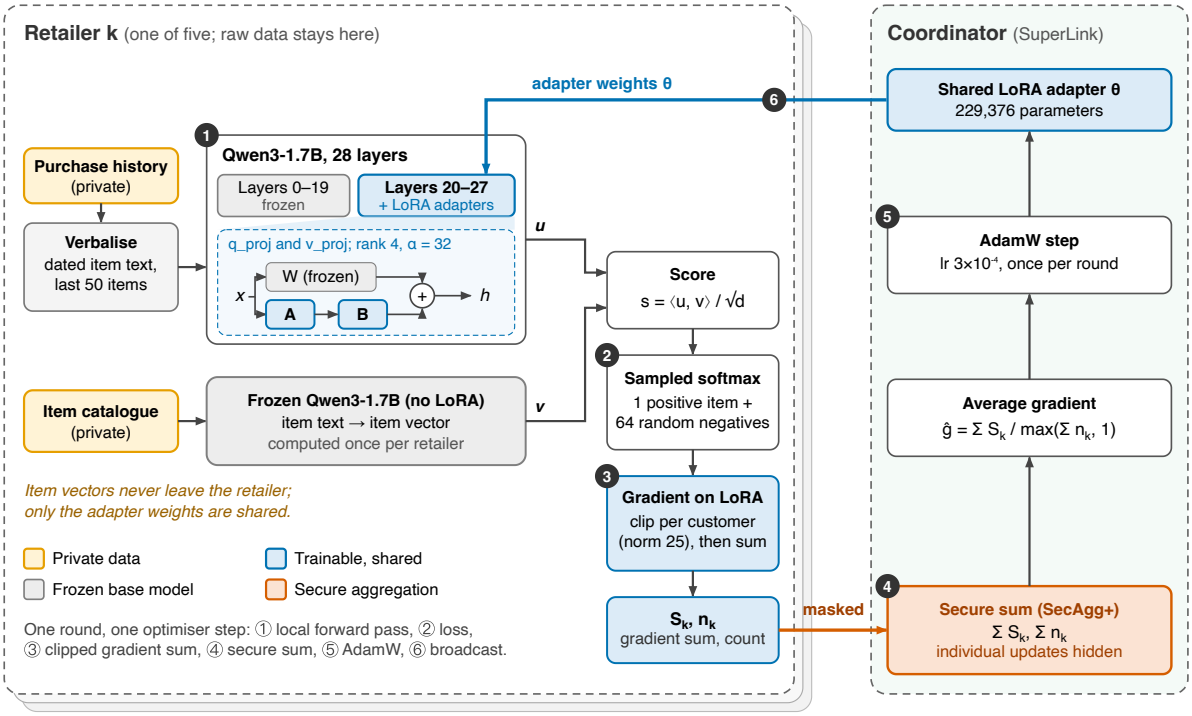


Figure 3: Model and training round at one retailer (left) and at the coordinator (right). A customer's purchase history is verbalised into text and passed through Qwen3-1.7B, whose last eight layers (20–27) carry a rank-4 LoRA adapter on the query and value projections; the final-token representation is the user vector $\mathbf{u}$. Item vectors $\mathbf{v}$ come from the frozen base model and are computed once and kept locally. Items are scored by $s = \langle \mathbf{u}, \mathbf{v} \rangle / \sqrt{d}$ and trained with a sampled softmax over one positive and 64 negatives. Gradients with respect to the adapter are clipped per customer and summed, and the sum S_k and count n_k are sent through SecAgg+. The coordinator forms $\hat{g} = \sum_k S_k / \max(\sum_k n_k, 1)$, takes one AdamW step on the shared adapter, and broadcasts the new weights. Circled numbers give the order of one round.

$\alpha = 32$, dropout 0.05) on the query and value projections of the last eight transformer layers only. The backbone has 28 layers, so the adapters sit in layers 21 to 28. Each layer's query projection maps $2048 \to 2048$ and its value projection maps $2048 \to 1024$, so one layer carries $4\,(2048 + 2048) + 4\,(2048 + 1024) = 28{,}672$ adapter parameters, and eight layers give 229,376 trainable parameters, about 0.01% of the backbone. In the light configuration the adapter is the only quantity that is shared between retailers (Section 2.3). The small size is a consequence of our earlier attempt at differentially private training (Section 4): the cost of privacy mechanisms grows with the number of trained parameters, and we kept the same architecture for the non-private and secure-aggregation experiments so that all arms are comparable.

Item tower. "In the light configuration, item vectors $v_i \in \mathbb{R}^d$, of the same dimension d, are produced by the same backbone with the adapter inactive, that is, by the pretrained model alone. Each retailer encodes its own catalogue once and stores the vectors locally. They are never updated, receive no gradient, and are never communicated. Because they depend only on the public pretrained weights and on the retailer's own item descriptions, no information about catalogues or sales passes between retailers through this tower. This asymmetry also makes the training loss decompose cleanly over customers: each customer's contribution to the loss depends only on their own history and on fixed item vectors, which is what allows per-customer gradient clipping to be well defined (Section 2.3).

Scoring and objective. The score of item i for a customer with vector u is $s(u,i) = \langle u, v_i \rangle / \sqrt{d}$, where the factor $1/\sqrt{d}$ keeps the scale of the scores independent of the vector dimension. At evaluation time we rank the retailer's whole catalogue by this score. At training time scoring the whole catalogue is unnecessary, and we use a sampled softmax: for each customer, the positive is the item actually bought next, and 64 negatives are drawn uniformly from the retailer's catalogue by a random number generator keyed on the run seed, step and retailer, independently of the customers in the cohort. The loss for one customer is the cross-entropy of the positive against the sampled negatives,

$$\ell(u, i^+) \;=\; -\log \frac{\exp s(u,i^+)}{\exp s(u,i^+) + \sum_{j \in \mathcal{N}} \exp s(u,j)}, \qquad |\mathcal{N}| = 64,$$

and the training objective is the sum of these losses over a step's cohort. We do not filter the positive out of the sampled negatives; with catalogues of about 21,000 items this affects roughly 0.3% of draws. The same negatives are shared by the (up to four) customers processed together on one device.

2.3 Federated round

Gradients rather than weights. Training differs from standard federated averaging in one respect that matters for the privacy analysis: retailers do not train locally for several steps and send back weights. Instead, each round is exactly one optimiser step, and what a retailer sends is a single summed, clipped gradient. The coordinator holds the optimiser state and applies the update. Let $N = 5$ be the number of retailers, θ the adapter parameters (Section 2.2), and θ_t their value at round t. Each round proceeds as follows.

1. *Broadcast.* The coordinator sends θ_t to every retailer. All N retailers take part in every round.

2. *Cohort.* Retailer k holds a fixed pool $\mathcal{P}_k$ of customers, one training example each (Section 2.5). It forms this round's cohort $\mathcal{C}_k^t$ by Poisson sampling: each customer in the pool is included independently with probability $q_k = b/|\mathcal{P}_k|$, where $b = 64$ is the target cohort size. The cohort size is therefore random, with expectation b.

3. *Clipped gradient sum.* For each customer $c \in \mathcal{C}_k^t$, retailer k computes the gradient g_c of that customer's loss ℓ (Section 2.2) with respect to θ_t and clips its ℓ_2 norm to $C = 25$: $\bar{g}_c = g_c \cdot \min(1, C/\|g_c\|_2)$. It returns the local sum $S_k^t = \sum_{c \in \mathcal{C}_k^t} \bar{g}_c$ and the cohort size $n_k^t = |\mathcal{C}_k^t|$. Gradients are computed in micro-batches of four customers to bound memory, and an empty cohort returns zeros.

4. *Central step.* The coordinator forms

$$\hat{g}_t \;=\; \frac{\sum_{k=1}^{N} S_k^t}{\max(\sum_{k=1}^{N} n_k^t,\, 1)}, \qquad \theta_{t+1} = \mathrm{AdamW}(\theta_t, \hat{g}_t),$$

   with learning rate 3×10^{-4} and the default AdamW moment and weight-decay settings.

We run $T = 200$ rounds. Because $\hat{g}_t$ divides by the total number of customers rather than by the number of retailers, each customer in the round has equal weight, and a round with $N = 5$ retailers behaves like one optimiser step on a batch of about $5b = 320$ customers drawn from five separate pools. Raw transactions, customer tables and item catalogues never leave a retailer; the only quantities that cross the retailer perimeter are θ_t in one direction and (S_k^t, n_k^t) in the other.

Deterministic sampling. All randomness in a round that depends on a retailer, namely which pool members are in the cohort and which negatives are drawn, comes from a generator keyed on the triple (run seed, step, retailer) through SHA-256. Two runs with the same seed therefore see exactly the same cohorts and negatives, whether they are plaintext or secure-aggregation runs, which lets us compare arms along matched training trajectories. It also means that the number of times any given customer was sampled during training can be recomputed offline from the seed alone, without logging anything during training. Section 6 uses this to stratify attacks by exposure. One consequence for the heterogeneous experiments is that every retailer contributes about b customers per round regardless of the size of its pool, so a customer in a small pool is sampled far more often than one in a large pool: over $T = 200$ rounds the expected number of times a customer is sampled is $Tb/|\mathcal{P}_k|$, which is 6.4 for a pool of 2,000 and about 0.02 for a pool of 700,000.

Secure aggregation. In the secure-aggregation arm, the sums S_k^t are not sent to the coordinator in the clear. We use the SecAgg+ protocol [4], a variant of secure aggregation [6] with sparse pairwise masking, as implemented in Flower [5]. Each retailer adds pairwise masks that cancel when all contributions are summed, so the coordinator can recover the sum of the N vectors but not any individual one. The protocol natively computes a weighted average. We give every retailer the same weight, so that it computes a plain average, and the coordinator multiplies the result by the public retailer count N to recover $\sum_k S_k^t$. The cohort size n_k^t is appended to the vector as one extra entry, so it is also protected, and the coordinator learns only $\sum_k n_k^t$. Values are quantised to integers before masking, with a per-coordinate bound of ± 2000 and a quantisation range of 2^{22}. Any coordinate outside this bound would be clipped silently, so each retailer checks its vector against the bound locally before sending it and logs the result; we report these checks, and the numerical error introduced by quantisation, in Section 5. The protocol uses 5 shares with a reconstruction threshold of 3 (a strict majority). We assume an honest-but-curious coordinator and retailers and do not evaluate dropped or malicious participants in this study.

Role of clipping. Clipping each customer's gradient to norm C bounds that customer's influence on the update and also bounds the magnitude of every coordinate that enters quantisation. We add no noise to the aggregate, so the procedure carries no differential-privacy guarantee. Our earlier attempt to add calibrated noise is reported in Section 4.

2.4 Local-only baseline

Whether federation pays is a comparison between a retailer that joins the federation and the same retailer training alone. The comparison is only informative if the two arms are given the same training effort, so the local-only baseline is constructed to match the federated arm in everything except the data it sees.

Construction. For each retailer k we train a separate adapter $\theta^{(k)}$ on that retailer's pool alone, with no communication. The training procedure is the one in Section 2.3 with a single participant: Poisson-sampled cohorts, per-customer clipping to the same norm $C = 25$, the same sampled softmax with 64 negatives, the same optimiser and learning rate, no noise, and $T = 200$ steps. The one change is the cohort size. In a federated round retailer k contributes a target of $b = 64$ customers and the five retailers together contribute about $5b = 320$. A local-only run therefore draws a target of 320 customers per step from the retailer's own pool, so the sampling probability is $320/|\mathcal{P}_k|$ instead of $64/|\mathcal{P}_k|$. Both arms then process about 64,000 customer gradients over the run and take the same number of optimiser steps, each step using a batch of about 320 customers. We call this matching *compute-matched*: it equalises customer gradient computations and optimiser steps. It does not equalise wall-clock time or communication, both of which are

higher for the federated arm. Sampling uses a generator keyed by the run seed, step and retailer, as in the federated arm, and all arms use the same seeds.

What the comparison isolates. Given these choices, the only systematic difference between a retailer's local-only adapter and the federated adapter is that the federated update at each step is computed from customers drawn from five pools (five retailers' transaction fragments) instead of one. A federated advantage therefore reflects the value of the other retailers' data, and not the effect of a larger batch, more optimiser steps or more compute. In the equal-size experiments every retailer's pool contains 100,000 customers. In the heterogeneous experiments the local-only arm uses retailer k's own pool size $|\mathcal{P}_k|$, between 2,000 and 700,000. A small pool is then sampled more intensively by the local-only arm than by the federated arm: with $|\mathcal{P}_k| = 2{,}000$ a customer is sampled about 32 times in 200 steps locally and about 6.4 times in the federated run.

Why matching matters. The choice of baseline changes the answer. Our first comparison trained each local-only model with a batch of 8 customers per step and compared it with a federation averaging five such retailers (Appendix C). Batch size is a confound: in a later control on retailer 1 with a pool of 100,000 customers, raising the local-only batch from 64 to 320, with no federation involved, increased MRR from 0.0630 to 0.0709 (a 12.5% relative gain). Part of any federated gain measured against a smaller-batch baseline is therefore a batch-size effect. We use the matched baseline throughout the main experiments and report the earlier comparison only in the appendix.

Limitations of the baseline. Both arms use one hyperparameter setting, chosen in the earlier experiments; we did not tune the learning rate, clipping norm or number of steps separately for each arm, so the comparison is conditional on this configuration. The local-only baseline also trains only on the retailer's own data, and it does not cover other ways a retailer could use external information, such as fine-tuning a federated model locally. We discuss these in Section 7.

2.5 Evaluation

We split time globally, with the same boundaries for all retailers: the final seven days of the data form the test window, the preceding seven days form a validation window, and everything earlier is training data. A *(history, target)* example is a customer's purchases up to a point in time together with the next purchase. For training, each retailer draws at most one example per customer, chosen uniformly among that customer's qualifying training-period positions, up to a pool size cap (100,000 customers per retailer in the equal-size experiments, and 2,000 to 700,000 in the heterogeneous ones). Experiments 1 and 1a use 10,000 customers per retailer and train for two epochs. This spreads a fixed budget across distinct customers rather than over-weighting heavy buyers, and it makes the unit of training a customer, which is also the unit that our leakage analysis in Section 6 reasons about. For evaluation, every qualifying position in the retailer's test window is eligible, and we draw a random subsample of 5,000 examples per retailer to bound evaluation cost. Each retailer is evaluated only against its own held-out examples and ranks only its own catalogue; no retailer is ever scored against another's items.

3 Utility Experiments

3.1 Experiment 1. Proof of concept: can an LLM rank retail purchases?

We first ask whether the pipeline learns a useful ranking at all, using a single retailer's own data and no federation. Each of the five synthetic retailers trains its own adapter on its own transactions only. The model is Qwen3-1.7B with a standard LoRA adapter (rank 16, $\alpha = 32$, applied to all seven projection matrices in every layer), a jointly trained item tower, and in-batch

Table 2: Experiment 1. Single-retailer performance (local-only training, single seed). MRR and Recall@k on 5,000 held-out examples per retailer, ranking the full catalogue of about 21,100 items.

Retailer	MRR	Recall@5	Recall@10	Recall@20
1	0.124	0.140	0.151	0.162
2	0.120	0.143	0.157	0.169
3	0.128	0.147	0.157	0.168
4	0.138	0.158	0.174	0.185
5	0.133	0.156	0.169	0.180
Mean	0.129	0.149	0.162	0.173

negatives. Each retailer trains on 10,000 of its own customers for two epochs with a batch of 8. Evaluation uses the 5,000 held-out examples per retailer described in Section 2.5, ranking the retailer's full catalogue of about 21,100 items.

The models learn a strong ranking signal (Table 2). Across the five retailers MRR ranges from 0.120 to 0.138 (mean 0.129), and Recall@20 from 0.162 to 0.185 (mean 0.173), so for roughly one customer in six the true next purchase is among the top 20 of about 21,100 items. A ranker that ordered the catalogue at random would score an MRR of about 5×10^{-4} and a Recall@20 of about 10^{-3}, so the trained models place the true next purchase more than two orders of magnitude higher than chance. The five retailers have similar catalogue sizes and transaction volumes by construction, and the results are correspondingly consistent across them.

To anchor these scores against a conventional collaborative-filtering method, we also trained iALS (implicit-feedback alternating least squares [13]) separately for each retailer, so that nothing is shared across retailers. We used the `implicit` library with 512 factors, regularisation 0.1, confidence weight 100 and 15 iterations. These were tuned on validation examples from retailer 1 only, and the optimum was flat. At test time each customer's vector is solved in closed form from the same 50 most recent purchases that the language model reads. iALS is scored on the same 5,000 held-out examples per retailer, ranking the full catalogue with the same rank convention (Table 3). Trained on every customer's training-period purchases, iALS reaches a mean MRR of 0.053, against 0.129 for the local-only language model (2.4×). Restricted to the 10,000 customers the language model trains on, it reaches 0.044 (2.9×). The gap narrows at deeper cutoffs: at Recall@20 the language model's advantage is 1.3× over the all-customer model and 1.7× over the matched-pool model, so the advantage is concentrated at the top of the ranking. The comparison is not data-matched: the language model sees one target per customer, whereas iALS sees all of a customer's training purchases, which is why both iALS variants are reported.

This establishes that the pipeline works, namely that verbalised purchase histories, a shared backbone with a LoRA adapter, and text-derived item vectors suffice to rank a retailer's catalogue. It does not by itself show that the model exploits a customer's history rather than item popularity, because retail purchases are heavily concentrated on a small number of popular items. The baselines in Table 3 address this: ranking by popularity alone scores an MRR of 0.011, and a matrix-factorisation model that does use each customer's history scores 0.053, both well below the language model. The results are from a single seed and a configuration (a large adapter, a trained item tower, in-batch negatives) that differs from the lighter architecture used in all later experiments, so we use them as a sanity check and as the reference point against which federation is later compared, not as a measurement to be quoted against the later results.

Table 3: Experiment 1. Baseline comparison (mean over the five retailers, single seed). All rows use the same 5,000 held-out examples per retailer and rank the full catalogue. iALS: matched pool uses the same 10,000 training customers as the language model; all customers uses every customer's training-period purchases. Both score each customer from the same 50 most recent purchases that the language model reads.

Model	MRR	Recall@5	Recall@10	Recall@20
Most-popular ranking	0.011	0.013	0.023	0.038
iALS, matched pool (10,000 customers)	0.044	0.061	0.081	0.102
iALS, all customers	0.053	0.072	0.100	0.129
Language model, local-only	0.129	0.149	0.162	0.173

3.2 Experiment 1a. Proof of concept: Federated recommendations

We next ask whether pooling the retailers' training signal helps, using the same configuration as above. Two further arms are trained alongside the local-only arm. In the *federated* arm the five retailers train one shared adapter with FedAvg for 10 rounds of 250 local steps each (batch 8 per retailer), so that each retailer performs the same number of local steps as in the local-only arm and synchronisation is the only difference. In the *centralised* arm one adapter is trained on the undivided transaction table. This is an upper bound that no real deployment could reach, since it requires pooling the retailers' private data. All three arms are evaluated on the same 5,000 held-out examples per retailer.

Federated training scores higher than local-only training on MRR for four of five retailers and is level on retailer 3 (0.1279 against 0.1281), and it is higher on Recall@20 for all five (Table 4). Averaged over retailers, MRR rises from 0.129 to 0.135 and Recall@20 from 0.173 to 0.179. On MRR the federated arm matches or exceeds the centralised arm on every retailer, with retailer 3 a tie, although the centralised arm remains ahead on Recall@20 on every retailer (mean 0.188 against 0.179). Federation closes between 20% (retailer 3) and 74% (retailer 5) of the Recall@20 gap between local-only and centralised training.

We treat this as motivation rather than as evidence of a federation benefit, for three reasons. First, each arm was run once, so no interval can be attached to the differences, and several are of the order of 0.001 to 0.003. Second, the arms are not matched in effective batch size: each local-only model updates on batches of 8 examples from one retailer, whereas federated averaging combines updates computed on five retailers' data, so part of any gap may reflect the larger effective batch rather than knowledge shared across retailers. Third, the configuration (a large adapter, a trained item tower and in-batch negatives) differs from the lighter architecture used in every later experiment. The matched-compute, multi-seed comparison in Section 3.3 revisits the question with these confounds removed and finds a much smaller effect.

3.3 Experiment 1b. Matched-compute, multiseed comparison of local versus federated

Experiment 1a left open whether federation helps, because its arms differed in batch size and it used a single seed. We therefore repeated the local-versus-federated comparison under a matched design. All five retailers have training pools of 100,000 customers. Local-only training uses one adapter per retailer with an effective batch of 320 customers per step, while federated training uses a cohort of 64 customers per retailer per step across the five retailers, so the two arms consume the same total number of examples per step. Both arms draw their cohorts and negative samples from the same keyed random streams, so differences between them are not caused by

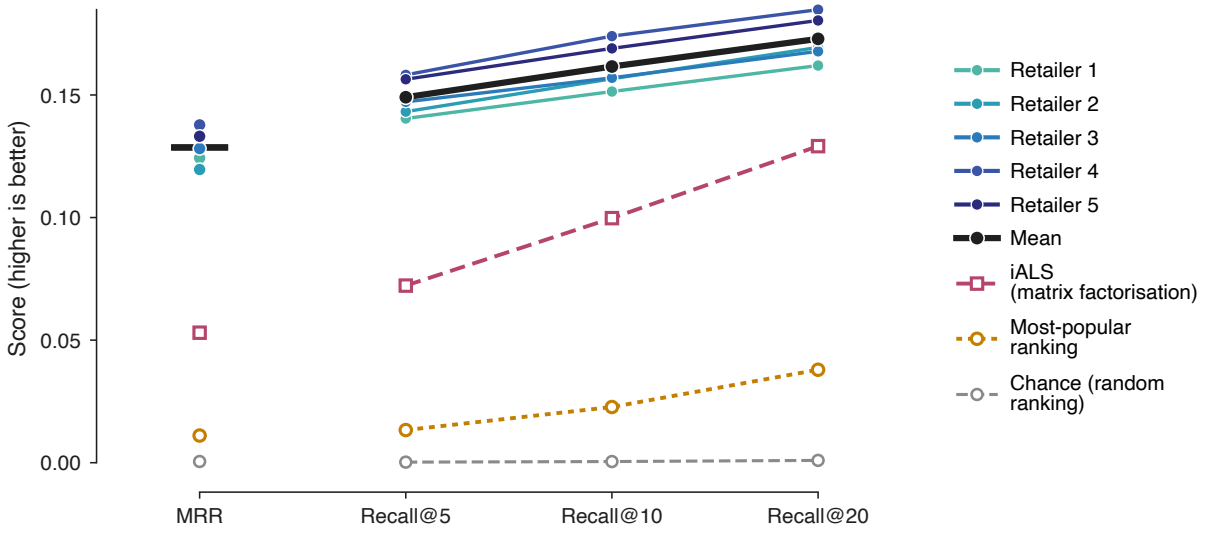


Figure 4: Experiment 1. Ranking quality of a single retailer's model, trained on its own data only. Mean reciprocal rank (MRR) and Recall@k for $k = 5$, 10 and 20, for each of the five retailers; the black marks show the mean over retailers, and lines join the Recall@k values for each retailer. The dotted amber line is a baseline that ranks items by how often they were bought in the training period (all-time popularity, mean over retailers), and the dashed grey line is the expected score of a random ranking of the catalogue (MRR $\approx 5 \times 10^{-4}$; Recall@$k = k/N$ for a catalogue of $N \approx 21{,}100$ items). The dashed magenta line with square markers is an iALS matrix-factorisation baseline trained on all customers (mean over retailers). Each point uses the same 5,000 held-out examples per retailer, ranked against the retailer's full catalogue (single seed).

different data draws. We ran seeds 0, 1 and 2. No torch seed is set, so adapter initialisation also varies between runs. All other settings follow the common setup of Section 2: a rank-4 LoRA on the query and value projections of the last eight layers, 200 steps, and 5,000 held-out test examples per retailer.

Table 5 gives the results. Federation scores slightly higher than local-only training on every metric, but the margin is small relative to the variation between seeds. Mean MRR over retailers and seeds is 0.0810 for local-only and 0.0828 for federation. This is a relative lift of +2.4% on average over seeds, with a 95% interval of $[-5.1\%, +9.9\%]$, computed as a t-interval over the three seeds. The interval spans zero, so the data do not establish that federation helps at equal retailer sizes. The seed-by-seed results show why. Seed 0 gave a lift of +5.8% with federation ahead at all five retailers, but seeds 1 and 2 gave +0.9% and +0.4%, with federation ahead at only two and three retailers respectively. The per-retailer signs also change between seeds; retailer 3, for example, moves between +2.6%, −4.0% and −5.6%. We therefore read the single-seed gain in Experiment 1a, and the first seed here, as a favourable draw rather than a stable effect.

This experiment has limits. Three seeds give wide intervals, so we can exclude large effects in either direction but not effects of a few percent. The retailers are fixed rather than sampled, so the intervals describe variation over seeds only, not over retailers. Training lasts 200 steps, and we did not tune hyperparameters separately for each arm.

3.4 Experiment 2: Heterogeneous retailer sizes

The equal-size experiments leave open whether federation helps a small retailer more than a large one, since real retailers differ in size by orders of magnitude. We repeat the matched design with the five retailers holding training pools of 2,000, 10,000, 50,000, 200,000 and 700,000

Table 4: Experiment 1a. Local-only, federated and centralised training (single seed, not batch-matched). MRR and Recall@20 on 5,000 held-out examples per retailer, ranking the full catalogue.

	MRR			Recall@20		
Retailer	Local	Fed.	Central.	Local	Fed.	Central.
1	0.124	0.127	0.126	0.162	0.166	0.174
2	0.120	0.137	0.133	0.169	0.180	0.192
3	0.128	0.128	0.128	0.168	0.171	0.184
4	0.138	0.139	0.137	0.185	0.192	0.202
5	0.133	0.142	0.139	0.180	0.187	0.190
Mean	0.129	0.135	0.133	0.173	0.179	0.188

Table 5: Experiment 1b. Matched-compute comparison of local-only and federated training at equal retailer sizes (five retailers, three seeds). Metrics are means over retailers and seeds; ± is the standard deviation across the three seed means. Lift is the relative MRR change of federated over local-only in each seed, averaged over retailers.

Arm	MRR	Recall@5	Recall@10	Recall@20
Local-only	0.0810 ± 0.0002	0.1068 ± 0.0004	0.1245 ± 0.0010	0.1396 ± 0.0011
Federated	0.0828 ± 0.0025	0.1103 ± 0.0030	0.1262 ± 0.0016	0.1409 ± 0.0020
Relative MRR lift of federated over local-only (%)				
Seed 0 / 1 / 2	+5.84 / +0.92 / +0.38			
Mean, 95% CI	$+2.38, [-5.1, +9.9]$			

customers (retailers 1 to 5), all with the same catalogue size and held-out examples as before. The federated arm uses plaintext aggregation with a target cohort of 64 customers per retailer per step, so every retailer contributes the same number of examples per step whatever its pool size. Each retailer's local-only baseline trains alone on its own pool for 200 steps with batches of 320, the same compute as in Section 3.3. Both arms use the same keyed random streams, and we run five seeds. We report the relative change in MRR from federated over local-only training, $\mathrm{MRR}_{\mathrm{fed}}/\mathrm{MRR}_{\mathrm{local}} - 1$, computed per seed, with a 95% t-interval over the five seeds.

Federation helps the smallest retailers and has no reliable effect on the larger ones (**??** and Table 6). Retailer 1, with 2,000 customers, gains +52.6% (95% interval $[39.3, 65.9]$) and its lift is positive in all five seeds. Retailer 2, with 10,000 customers, gains +10.9% ($[1.1, 20.7]$). For retailers 3 to 5 the estimates are −1.5%, +3.0% and +3.8%, and all three intervals include zero (Table 6). The effect is therefore concentrated in the retailers with the least local data, and it vanishes by a pool of about 50,000 customers.

The mechanism is visible in the absolute scores. Local-only MRR rises steeply with pool size, from 0.051 at 2,000 customers to 0.084 at 700,000, whereas federated MRR is nearly flat, from 0.078 to 0.087. The shared model gives the smallest retailer roughly the accuracy of a much larger one. Two features of the design make the small-retailer gain larger than it would be in general, and we do not want to overstate it. The first is that the local-only baseline is weak for a pool of 2,000: with 200 steps of 320 it sees every customer about 32 times, so part of the lift is a limit of the baseline and not a pure gain from pooled data. The second is that every retailer contributes 64 customers per step regardless of pool size, so a customer at the smallest retailer is sampled about 6.4 times under federation while one at the largest is sampled about 0.02 times

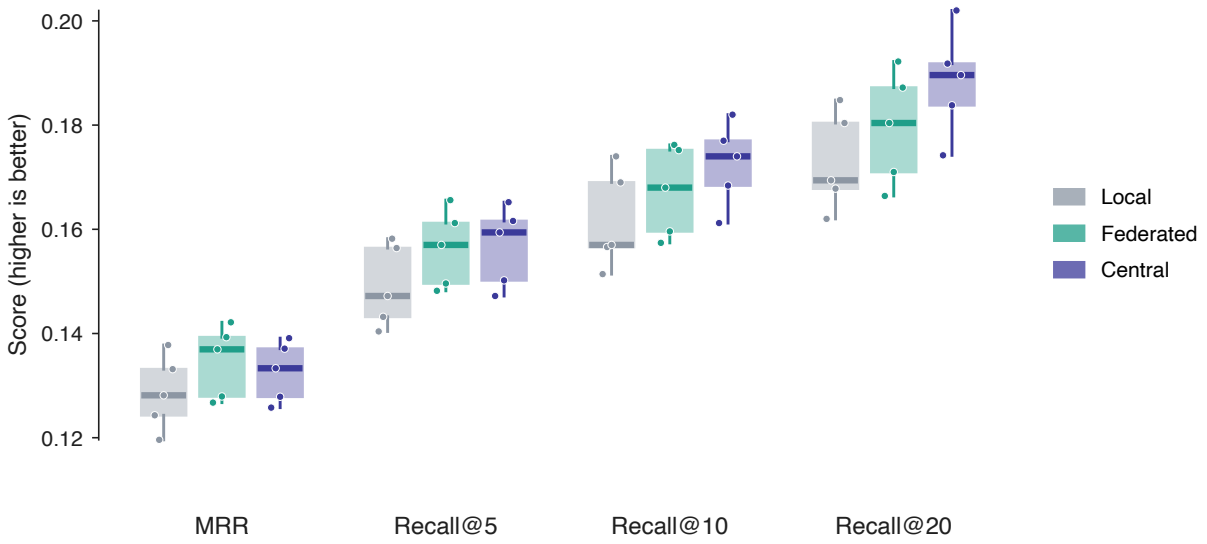


Figure 5: Experiment 1a. Local-only, federated and centralised training compared on MRR and Recall@k for $k = 5$, 10 and 20. Each box summarises the five retailers (line: median; box: interquartile range; whiskers: full range; dots: individual retailers). Federated Learning consistently shows better performance than local data four all four metrics. In the case of MRR and Recall@5, federated learning is seen to outperform similarly to centralised learning.

Table 6: Heterogeneous retailer sizes (five seeds). Mean MRR for local-only and plaintext federated training, and the mean relative lift of federated over local-only with a 95% t-interval over seeds (4 degrees of freedom).

Retailer	Pool size	Local	Federated	Lift (%)	95% interval
1	2,000	0.051	0.078	+52.6	[39.3, 65.9]
2	10,000	0.074	0.081	+10.9	[1.1, 20.7]
3	50,000	0.082	0.081	−1.5	[−6.9, 4.0]
4	200,000	0.082	0.084	+3.0	[−5.3, 11.4]
5	700,000	0.084	0.087	+3.8	[−1.3, 8.9]

on average. We have not separated these effects from the value of the other retailers' data, and we treat exposure as a candidate explanation, not a tested one.

Several limitations apply. The five retailers are fixed and not sampled from a population, so the intervals reflect seed variation only, and we apply no multiple-comparison correction across the five retailers. The interval for retailer 2 has a lower end of 1.1%, so we regard it as suggestive, and it would not survive a Bonferroni correction. The experiment has no SecAgg+ arm. By the equal-size results SecAgg+ is utility-neutral, but we have not measured it at these sizes. An earlier single-seed run with an unmatched local baseline suggested that federation hurt retailers 3 and 4, and that did not replicate once the baseline was matched.

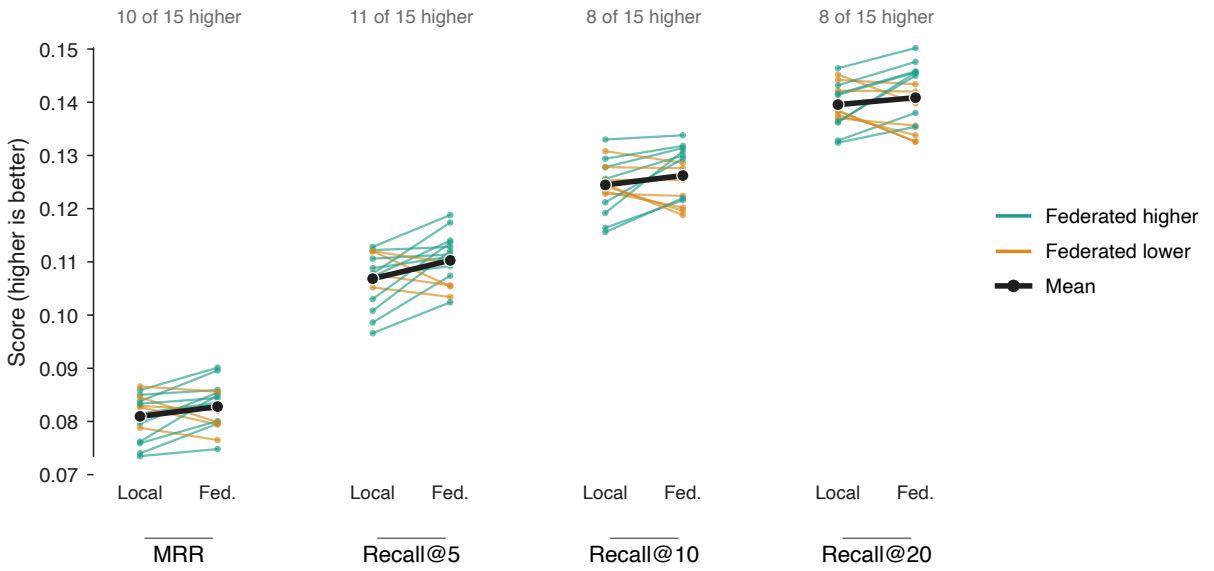


Figure 6: Experiment 1b. Paired comparison of local-only and federated training under matched compute. Each line joins the local-only and federated (Fed.) scores of one retailer in one seed (five retailers × three seeds = 15 lines per metric); teal lines show federation scoring higher, amber lines lower, and the black line is the mean over the 15 pairs. The number above each metric is the count of pairs in which federation scored higher. The pairs share retailers across seeds and are not independent. The vertical axis does not start at zero.

4 Differential Privacy

4.1 Experiment 3. Differential privacy is not practical for retail recommendations

Customer-level differential privacy has the property of giving federation a formal guarantee, so we tested it in the most favourable configuration we found. The five retailers' clipped gradient sums are combined and a single Gaussian noise draw is added to the total, as if by a trusted aggregator. This simulates the best case for DP, because one noise draw is shared across all five retailers rather than each retailer adding its own. Each retailer uses a pool of 100,000 customers, a cohort of 64 per step, a clip norm of 25, 200 steps and the tiny LoRA of Section 2.2, and the noise is calibrated with a PRV accountant to $(\varepsilon, \delta) = (8, 10^{-6})$ (achieved $\varepsilon = 7.99$, noise multiplier 0.36). We compare the noisy run with an identical run without noise, and with compute-matched local-only training from Section 3.3. Earlier configurations, including a larger adapter, per-retailer noise, a frozen-projection architecture and smaller pools, performed worse and are not reported here.

Adding the noise cuts MRR by roughly 45%, from 0.083 to 0.046 on average over the five retailers, and the noisy model falls below compute-matched local-only training on every retailer (Table 7). Even with a single noise draw, the signal-to-noise ratio of the aggregated update is about 0.2 for the whole run. The best private federated model therefore performs worse than a retailer training alone with no privacy mechanism, so for this model and dataset the formal guarantee costs more than federation can return. We did not pursue larger privacy budgets, because the question is whether a guarantee is affordable at a meaningful ε, and we treat this as a negative result for this setting rather than for DP in general. It also motivates the weaker, empirical protection that we study instead (Section 5 and **??**).

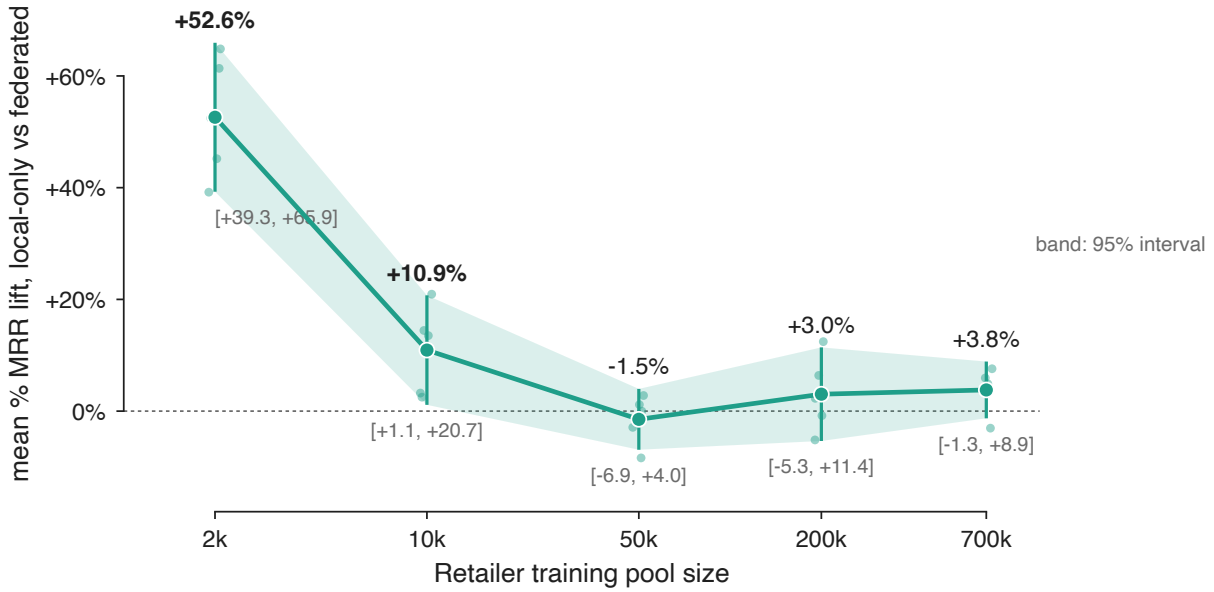


Figure 7: Experiment 2. Mean relative MRR lift of federated over compute-matched local-only training as a function of the retailer's training-pool size. The line is the mean over five seeds, the band and whiskers show the 95% confidence interval (Student t, 4 degrees of freedom), and dots are individual seeds.

Table 7: Distributed DP federation at $\varepsilon = 8$ (single run). MRR on 5,000 held-out examples per retailer. DP and no-noise federation use identical settings apart from the noise. Local-only is the compute-matched baseline of Section 3.3 (mean of three seeds).

Retailer	Fed., DP	Fed., no noise	Local-only
1	0.046	0.077	0.075
2	0.044	0.085	0.080
3	0.046	0.082	0.083
4	0.045	0.085	0.082
5	0.048	0.088	0.085
Mean	0.046	0.083	0.081

5 Secure Aggregation

5.1 Protocol and configuration

Secure aggregation lets the coordinator learn the sum of the retailers' updates without seeing any individual update. We use Flower's SecAgg+ protocol [4, 6], in which each retailer masks its update with pairwise random masks that cancel exactly in the aggregate, and shares the key material needed to recover a retailer that drops out among the others using Shamir secret sharing. What each retailer submits is the pair (S_k, n_k) of Section 2.3, its clipped gradient sum and cohort count, and the coordinator receives only the aggregate over the five retailers. We configure the protocol with five shares and a reconstruction threshold of three. The threshold plays two roles: any three retailers can reconstruct a dropped retailer's masking key, so it is both the number of simultaneous dropouts the protocol tolerates (two of five) and the number of colluding parties that would break confidentiality. We chose a strict majority, $\lfloor N/2 \rfloor + 1 = 3$, as a default. Our threat model is an honest-but-curious coordinator, which follows the protocol but may inspect everything it receives; an actively malicious coordinator and confidential-computing

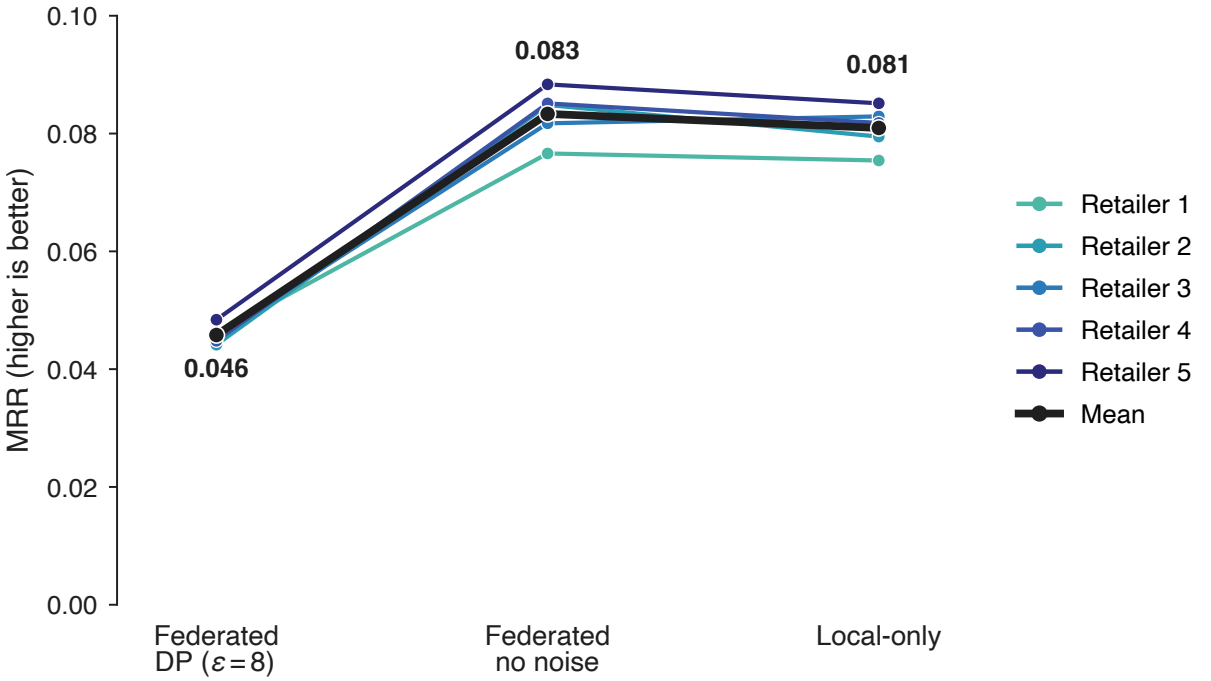


Figure 8: Experiment 3. Retrieval quality (MRR) of federated training under customer-level differential privacy ($\varepsilon = 8$, $\delta = 10^{-6}$), the same federated run without noise, and compute-matched local-only training, for each of the five retailers (coloured lines) and their mean (black). The federated runs use a pool of 100,000 customers per retailer. Local-only values are the mean over three seeds of the matched-compute local arm (Section 3.3); the other two conditions are single runs. Adding noise costs every retailer roughly 40–48% of its MRR.

infrastructure are out of scope.

SecAgg+ in Flower computes a weighted average, not a sum. Each client scales its vector by its weight, quantises it to bounded integers, and the server divides by the total weight. We give every retailer the same weight, so that the result is the uniform average $\frac{1}{N}\sum_k S_k$, and the server multiplies it by $N = 5$, which is public and adds no error. The cohort count n_k travels as one extra array element, so the coordinator recovers $\sum_k n_k$ in the same way and forms $\hat{g}_t = \sum_k S_k / \max(\sum_k n_k, 1)$ exactly as in the plaintext arm. Quantisation uses a range of 2^{22} and a per-coordinate clipping range of ± 2000; values outside that range would be clipped silently. The coordinator then takes one AdamW step on the shared adapter, as before.

5.2 Numerical fidelity

The masking itself is exact, because the pairwise masks cancel in modular arithmetic, so the only source of difference between SecAgg+ and plaintext aggregation is the conversion of floating-point updates to bounded integers and back. We measure it in two ways. In an isolated test, five synthetic vectors of 229,376 dimensions, matching the adapter size and the observed gradient norms, pass through Flower's own quantisation and dequantisation functions. The relative error of the aggregate is 7.8×10^{-6} at our quantisation range of 2^{22} (5×10^{-4} at 2^{16} and 1.2×10^{-7} at 2^{28}), so the error falls with the quantisation range as expected (Figure 9, left). In a second test, identical per-retailer inputs with unequal cohort sizes pass through both aggregation paths for 200 AdamW steps. With the deployed clipping range of ± 2000 the recovered aggregate matches the plaintext one to a relative error of about 10^{-4}, and the cohort-count normalisation to better than 10^{-4}. The weight trajectories nevertheless drift apart by about 1% of the distance moved, which comes from AdamW amplifying small gradient differences and does not depend on the

Figure 9: SecAgg+ numerics. Left: relative error of the aggregate against the quantisation range (log–log); the deployed range is 2^{22}. Right: per-round maximum absolute update coordinate for each retailer in a full SecAgg+ run, against the clipping range of 2000.

quantisation range we tried. The two paths are therefore faithful in what they compute but not bit-identical, and a utility comparison of the trained models is still needed.

We also check that the clipping range is never reached. In every round of a full SecAgg+ run each retailer records, locally and before quantisation, the largest absolute coordinate of its update. Across the 1,000 retailer-rounds of that run no coordinate lies outside ± 2000, and the largest value is 289, which leaves a headroom of 6.9× (Figure 9, right). The largest values in three other runs are 212, 269.5 and 295. Because the bound is much larger than needed, a narrower range would give finer quantisation, and we have not tuned it.

5.3 Utility

Under the matched design of Section 3.3, SecAgg+ federation is indistinguishable from plaintext federation. Averaged over retailers and seeds the MRR is 0.0829 for SecAgg+ and 0.0828 for plaintext federation. The mean per-seed relative difference is +0.09%, with a 95% t-interval over three seeds of $[-1.6\%, +1.8\%]$. SecAgg+ therefore costs no measurable accuracy, to within about $\pm 2\%$, which is the resolution of three seeds and not a proof of equivalence. A first SecAgg+ run scored 3–8% below plaintext federation; the cause was a client random stream synchronised across retailers, not the protocol, and the matched design removes it by giving both arms the same keyed random streams.

5.4 Scope

These results describe a single-machine Flower simulation in which the real cryptographic protocol runs for 200 rounds with all five retailers present in every round. We have not tested retailer dropout, malformed or replayed updates, or a deployment with separate hosts, TLS and authenticated nodes. SecAgg+ protects individual updates in transit. It does not protect the released model, which every retailer receives, and the leakage from that model and from the coordinator's view of the updates is the subject of **??**.

6 Empirical Leakage Assessment

6.1 Threat model and pre-registered thresholds

We assess leakage empirically, not by proof, because the federation carries no formal privacy guarantee (Section 4). There are three observers. The *coordinator* sees each round's updates, in plaintext or as the SecAgg+ sum, and is assumed honest-but-curious. A *participating retailer* receives the released shared adapter and can query it offline. Both are also assumed to know the training setup, including the sampling scheme. Secure aggregation changes only what the coordinator sees. It does not change the released adapter, so any leakage through the final model should be the same with and without SecAgg+, and only the attack on the coordinator's view can show what SecAgg+ buys.

We ran four attacks, each posed as a plain question and each with a pass rule fixed on 2 November 2026, before any attack result existed. The thresholds are the author's proposal, were adopted provisionally to allow the runs to start, have not been agreed with privacy or legal stakeholders, and were not changed after we saw the results. Rules are stated on the upper end of a 95% confidence interval, so that a wide interval cannot pass by default. A pass means that no leakage above the threshold was detected, with the stated power; it is not evidence that there

is none. The AUC threshold of 0.55 is applied at its two significant figures. Attack A's rule as written named customers sampled once; we apply the same threshold to customers sampled three or more times, a choice made after seeing the results, and report it as such.

Our sampling is deterministic. Each retailer's cohort at each step is drawn from a keyed random stream, so the number of training cohorts that included a given customer (its *exposure*) can be reconstructed offline from the seed, with no logging by the trainer. We checked the reconstruction against the logged cohort sizes of all 3,000 retailer-rounds of the SecAgg+ runs with zero mismatches. Exposure matters because with 200 steps and a pool of 100,000 customers per retailer a customer is sampled 0.13 times on average, so most pool members are never trained on, and treating all of them as "members" would dilute any attack. Attacks A and B are therefore reported as a function of realised exposure.

6.2 Attack A: membership inference

Given the finished model, can someone tell whether a particular customer's history was used to train it, and does that depend on how often the customer was sampled?

For each retailer we score customers by the model's loss and rank on their next purchase, and compute the AUC for separating customers of a given exposure from customers who were never sampled. Controls separate never-sampled customers from customers outside the pool, and split the outside-pool customers into random halves. The pass rule is that for customers sampled exactly once the upper confidence limit of the AUC is at most 0.55 and the true-positive rate at a 1% false-positive rate is at most 2%, and that both controls sit at 0.5, since otherwise the test is invalid. We evaluate the local-only, plaintext and SecAgg+ models for three seeds and five retailers.

The rule is met (**??** and Table 8). For customers sampled once, the pooled AUC is 0.500 (95% interval $[0.496, 0.504]$) for plaintext federation and 0.500 ($[0.496, 0.505]$) for SecAgg+, with a true-positive rate of 1.0%, and the controls give 0.501 and 0.498. The plaintext and SecAgg+ models are indistinguishable, as expected. Customers sampled three or more times are few (463 pooled), so their intervals are wide. Their AUC is 0.519 ($[0.490, 0.550]$) for plaintext and 0.521 ($[0.501, 0.547]$) for SecAgg+. Under the same threshold both pass, although the plaintext upper limit is 0.550 to three decimal places and so sits at the threshold. The local-only models, which sample each customer far more often (0.64 against 0.13 mean exposures), show a small signal (AUC 0.504 to 0.508), consistent with leakage growing with exposure.

6.3 Attack B: canary extraction

If we plant fake customers with unmistakable purchase patterns in one retailer's data, can someone with the model recover them?

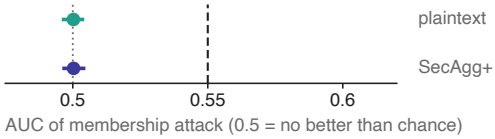
In each run we insert 40 real canaries and 40 sham canaries into retailer 1's training pool, replicated 1, 4, 16 or 64 times, train through the federated pipeline, and measure for each canary how highly the model ranks its secret purchase. The extraction ratio is the mean reciprocal rank of the real canaries over that of the sham ones, so a ratio of 1 means no extraction. The rule is that the confidence interval includes 1 for canaries of realised exposure at most one. Canaries with more exposure, and the oversampled replicas, are reported as a stress test and not as a pass or fail. We run three seeds for plaintext federation and one for SecAgg+.

For plaintext federation at realised exposure of at most one, the pooled extraction ratio is 1.72 ($[0.86, 3.29]$, $n = 69$), and for SecAgg+ it is 2.54 ($[0.49, 6.95]$, $n = 22$), so the rule is met (**??**). At exposure of four or more the ratio is 0.93 ($[0.52, 1.61]$, $n = 33$). The power is low, because there

Attack A customers sampled once; 3 seeds × 5 retailers, n = 30,000

Question: Can someone tell from the finished model whether a customer's history was used to train it?

Rule: upper 95% CI of the AUC ≤ 0.55, and true-positive rate at 1% false-positive rate ≤ 2%



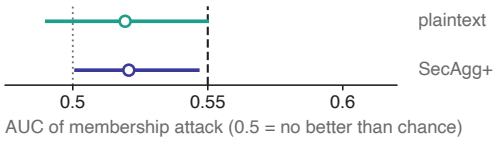
Pass

Both arms score 0.50 with intervals of about 0.496–0.505, well under 0.55, and the true-positive rate at 1% false positives is 1.0%.

Attack A customers sampled three or more times; n = 463

Question: Can someone tell from the finished model whether a customer's history was used to train it?

Rule: the same rule, at the two significant figures of the 0.55 threshold (the rule as pre-registered named one-time sampling)



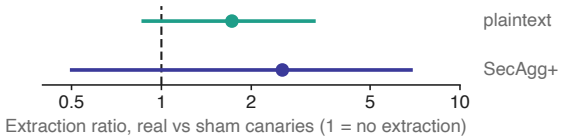
Pass

Upper CIs are 0.550 (plaintext) and 0.547 (SecAgg+), both 0.55 at the threshold's precision. The intervals are wide because only 463 customers fall in this stratum.

Attack B canaries at exposure ≤ 1; plaintext 3 seeds (n = 69), SecAgg+ seed 0 (n = 22)

Question: If fake customers with unmistakable purchase patterns are planted in one retailer's data, can they be extracted from the model?

Rule: 95% CI of the extraction ratio includes 1 (real canaries ranked no better than sham ones)



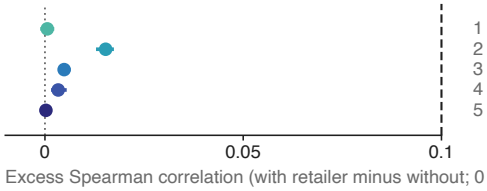
Pass

Both intervals include 1, but they are wide (up to about 7), so an extraction ratio below roughly 3 could not be ruled out.

Attack C seed 0; one leave-one-retailer-out model per retailer (1–5)

Question: Does the shared model reveal which products a specific retailer sells best?

Rule: upper 95% CI of the excess Spearman correlation ≤ 0.10, and top-50 best-seller recovery no better than without the retailer



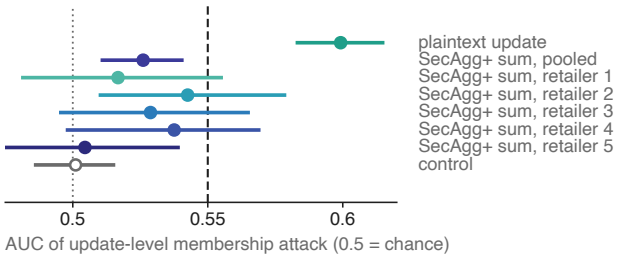
Pass

The largest upper CI is 0.017, far under 0.10. Top-50 recovery differs by 2–3 items of 50 for retailers 2 and 4, and the rule has no tolerance, so that part is unresolved.

Attack D seed 0, power re-run; 1,859 members pooled, about 360–400 per retailer

Question: Watching the updates arrive, can the coordinator tell whether a customer was in a retailer's batch?

Rule: upper 95% CI of the AUC against the SecAgg+ five-retailer sum ≤ 0.55, pooled and for each retailer



Fail

Four of five per-retailer upper CIs exceed 0.55 (retailer 5 is 0.539), through interval width alone. The pooled SecAgg+ sum passes (upper CI 0.541), and plaintext updates are attackable (AUC 0.599).

dotted line: no-leakage value dashed line: pre-registered threshold bars: 95% interval open marker: control or smaller stratum

Figure 10: Summary of the pre-registered leakage tests: for each, the question, the rule, the estimate with its 95% interval on its own scale, and the outcome. Dotted line: no-leakage value; dashed line: pre-registered threshold. The 0.55 AUC threshold is applied at two significant figures and, for Attack A, to both exposure strata (the rule as pre-registered named one-time sampling). Attacks B–D use limited seeds, and the Attack D rows are the power re-run, added after seeing the initial result (see text).

are few canaries: a ratio below about 3 could not be ruled out. The bin at exposure two to three has a ratio of 12 ($[0.5, 43]$) that is produced by a single canary landing in the top 100, is not monotone in exposure, and we regard it as noise.

6.4 Attack C: competitor inference

Does the shared model reveal which products a specific retailer sells best, beyond what the model learns without that retailer?

For each retailer we train a model on the other four, then compare the Spearman correlation between each model's item affinities and the retailer's sales, and also how many of the retailer's 50 best sellers each model recovers. The rule is that the excess correlation of the full model over the leave-one-out model has an upper confidence limit of at most 0.10, and that top-50 recovery is no better than without the retailer. This uses one seed.

The correlation rule is met (**??**). The excess is 0.001, 0.015, 0.005, 0.003 and 0.000 for retailers 1 to 5, and the largest upper limit is 0.017, so nearly all of the model's ability to predict a retailer's sales ($\rho \approx 0.53$, against 0.51 to 0.53 without that retailer) comes from public item text, category and price. Top-50 recovery is equal for retailers 1, 3 and 5, but nominally higher with the retailer for retailer 2 (0.16 against 0.12) and retailer 4 (0.14 against 0.08), a difference of two or three items of fifty. The rule as written has no tolerance, and a single seed cannot distinguish a difference of this size from retraining variance, since each leave-one-out model is a separate run with a different batch composition. We therefore count Attack C as passed, on the correlation criterion and on that reading of the top-50 comparison. The interval on the correlation reflects item sampling and not retraining variance.

6.5 Attack D: update-level membership

Watching the updates arrive each round, can the coordinator tell whether a given customer was in a retailer's batch, and does SecAgg+ stop that?

At six training steps we compute the cosine similarity between a candidate customer's clipped gradient and three signals: a retailer's own plaintext update, the SecAgg+ sum of all five retailers' updates, and a control sum built from the other retailers' updates only, which has no information about the candidate. We then compute the AUC for separating customers who were in the retailer's cohort from those who were not. The rule is that the AUC against the SecAgg+ sum has an upper confidence limit of at most 0.55, both pooled across retailers and for each retailer, and the plaintext AUC is reported as the contrast. This uses one seed.

The first run, with 48 members per step, gave a per-retailer AUC against the SecAgg+ sum of 0.548, 0.531, 0.562, 0.534 and 0.481 and failed the rule for individual retailers through interval width. After seeing it we added a power re-run that uses every cohort member (1,859 pooled across retailers). This re-run was a post hoc addition, we report both runs, and the threshold did not change. In the re-run, the coordinator who sees one retailer's plaintext update reaches an AUC of 0.599 ($[0.583, 0.615]$), which is a real signal. Against the SecAgg+ sum the AUC is 0.526 ($[0.510, 0.541]$), against 0.501 ($[0.486, 0.516]$) for the control (**??**). Pooled across retailers the SecAgg+ result meets the rule, but retailer by retailer it does not: the upper limits are 0.556, 0.579, 0.566, 0.570 and 0.540, four of them above 0.55, because each retailer has only about 360 to 400 members. Because the rule applies to each retailer as well as to the pool, we record Attack D as a fail. SecAgg+ therefore removes about three quarters of the coordinator's advantage over chance (0.099 to 0.026) but not all of it, and the remaining signal is statistically above 0.5.

Table 8: Attack suite against the pre-registered rules (thresholds proposed by the authors, not agreed with privacy or legal stakeholders, unchanged after the results). Intervals are 95%.

Attack	Rule	Result	Verdict
A. Membership	AUC upper limit $\leq$ 0.55 at 1× exposure (also applied to $3+\times$), TPR@1%FPR $\leq$ 2%, controls at 0.5	1×: plaintext 0.500 $[0.496, 0.504]$, SecAgg+ 0.500 $[0.496, 0.505]$, TPR 1.0%. $3+\times$ ($n = 463$): plaintext 0.519 $[0.490, 0.550]$, SecAgg+ 0.521 $[0.501, 0.547]$	Pass
B. Canaries	Ratio interval includes 1 at exposure ≤ 1	Plaintext 1.72 $[0.86, 3.29]$; SecAgg+ 2.54 $[0.49, 6.95]$	Pass
C. Competitor	Excess ρ upper limit $\leq$ 0.10; top-50 no better	Excess $\leq$ 0.015, upper limit 0.017; top-50 higher by 2–3 items for two retailers	Pass
D. Update-level	SecAgg+ AUC upper limit ≤ 0.55, pooled and per retailer	Pooled 0.526 $[0.510, 0.541]$; per retailer upper limits 0.540–0.579; plaintext 0.599	Fail

6.6 Summary against thresholds

Table 8 gives the outcome of each pre-registered rule. Attacks A, B and C pass, and Attack D fails. The released model leaks nothing we could detect about customers at any exposure we could test, or about canaries planted in a retailer's data, and its predictions of a retailer's sales ranking are almost entirely explained by public item information. The failure is at the coordinator, where SecAgg+ substantially reduces but does not remove an attacker's edge: the pooled result meets the threshold and the per-retailer results do not. These conclusions are limited by the power of each test, by the single seed for Attacks C and D, and by the setting: a public-data simulation, a tiny adapter and 200 training steps. They are not formal guarantees and should not be read as "no leakage".

7 Discussion and Limitations

7.1 What the results support

The evidence supports a narrower claim than "federation helps". At equal retailer sizes, and with compute matched, we could not establish that joining the federation beats training alone: the three-seed interval on the relative change in MRR spans -5.1% to $+9.9\%$, and the larger apparent benefit in earlier comparisons came in part from unmatched batch size. The benefit appears where a retailer has little data of its own. With heterogeneous pools, the two smallest retailers (2,000 and 10,000 customers) gain 52.6% and 10.9%, and the three largest show no reliable effect. For a deployment this suggests that the case for a retailer joining depends on how much it has to lose by training alone, and that a consortium of similar, well-supplied retailers should not expect large gains from pooling.

Customer-level differential privacy was not affordable in this setting: the best configuration reached about half the accuracy of the same federation without noise, and fell below a retailer training alone. Secure aggregation, in contrast, cost no measurable accuracy (a relative difference of $+0.09\%$, interval $[-1.6\%, +1.8\%]$) and removed about three quarters of the coordinator's advantage in the one attack that targets the updates. Together these suggest a pragmatic posture of secure aggregation for infrastructure confidentiality, with empirical leakage tests in place of a formal guarantee, but this is a recommendation for further evaluation and not a demonstration that the posture is safe.

7.2 What the leakage results do and do not show

The thresholds in Section 6.1 are our proposal. They were fixed before the results and not changed, but they have not been agreed with privacy or legal stakeholders, and a different organisation might reasonably set them lower. A pass means that no leakage above the threshold was detected with the stated power, and the power is limited: the canary test could not exclude an extraction ratio below about 3, the stratum of customers sampled three or more times is small (463 customers) and its plaintext upper limit sits at the threshold„ and the per-retailer results for Attack D fail the rule through interval width although the pooled result passes. Attacks C and D use one seed, and Attack B uses one seed for SecAgg+. The power re-run of Attack D was added after we saw the first run, and we report both. Our findings are also limited to the attacks we ran. We tested membership, canary, competitor-inference and update-level attacks, but not, for example, attribute inference, model inversion, or an adversary who combines the released adapter with outside data.

Secure aggregation cannot remove information that the released model already contains. The shared adapter goes to every participating retailer, so everything the model reveals about customers or retailers is available to all of them whether or not SecAgg+ is used. That is why the attacks on the final model gave the same results with and without SecAgg+, and why leakage through the model grows with how often a customer is sampled: the local-only models, which sample each customer far more often, show a small signal (AUC 0.504 to 0.508) where the federated models do not. Our setting has low per-customer exposure (0.13 samples on average), which favours the defender. Longer training, smaller pools or repeated sampling would raise exposure and would need to be re-tested.

7.3 Limitations

Data and setting. We use one public dataset, H&M, split into five synthetic retailers by dealing each product category round-robin. The retailers therefore have the same product mix, the same customers' tastes, and a high customer overlap (86.4% of customers appear at two or more retailers). Real retailers differ in assortment, customer base and overlap, and the benefit of federation could be larger or smaller. Our overlap is probably favourable to federation, because a customer's history is fragmented across retailers, but we have not tested lower overlap. The heterogeneous experiment varies only pool size, with fixed retailers, so we cannot separate the effect of size from the identity of a particular retailer. The time split is a single one-week test window.

Model and training. The adapter is tiny (229,376 parameters, on the last 8 layers of a 1.7B model), the item tower is frozen, and training lasts 200 steps, chosen for tractability. We did not tune hyperparameters separately for each arm. Federated training, local-only training and SecAgg+ may respond differently to tuning, and the lift for the smallest retailer is partly a result of a weak local baseline, since the local-only model sees each of 2,000 customers about 32 times. The designs in our feasibility plan for a private per-retailer item head and residuals were not implemented for these results, so the experiments describe the shared part of the model only.

Statistics. Interval estimates rest on three seeds in the equal-size experiments and five in the heterogeneous experiment, with retailers treated as fixed and no multiple-comparison correction across the five retailers. The interval for retailer 2 has a lower end of 1.1% and should be read as suggestive. No random seed controls the adapter initialisation, so initialisation varies between runs in all arms. Phase 0 compared the arms with a single seed and without matching batch size, and we use it only as motivation.

Differential privacy. The result is negative for one simulated configuration in which a single noise draw covers all five retailers, as if applied by a trusted aggregator, and we did not explore larger privacy budgets. We also have not verified that the accounting treats the same person appearing at several retailers correctly, so the stated ε should not be read as a formal

customer-level guarantee across the federation.

Secure aggregation. The protocol ran in a single-machine Flower simulation, with all five retailers participating in every round. We did not test dropout, malformed or replayed updates, separate hosts, TLS or authenticated nodes. The threat model assumes an honest-but-curious coordinator, and a malicious coordinator, or a coalition of three or more retailers, is outside what we claim. We use confidential-computing infrastructure in none of the experiments.

7.4 Future work

The most useful next steps are a pilot with real retailers, whose heterogeneity, scale and overlap we cannot reproduce; extending the heterogeneous experiment to SecAgg+ and to more seeds for the attacks; agreeing leakage thresholds with privacy and legal stakeholders before any deployment; implementing the private per-retailer head and measuring whether it changes both utility and leakage; and fault and multi-host tests of the secure-aggregation pipeline. If a deployment found leakage above its agreed thresholds, bounded mitigations that do not require noise include earlier stopping, deduplicating customer sequences, stronger regularisation, minimum cohort sizes, restricted access to the released adapter and rate-limited queries; if those were not enough, the options are to return to formal DP at its measured utility cost, to redesign the learning problem, or to stop.

8 Conclusion

We first investigated whether an LLM-based recommender is feasible and practical. Our results show that LLMs can be an effective recommender, and considerably outperform a cutting edge out-of-the-box matrix factorisation algorithm. We do not show the relative improvability of either method through, in the case of matrix factorisation, feature engineering and hyperparameter tuning, or in the case of LLM-based recommenders, prompt engineering. This remains an open question.

We adapted the LLM-based recommender to a federated, multi-retailer setting, in which retailers train a shared adapter on a frozen backbone without sharing their transaction tables, customer tables or catalogues, and we tested three questions: whether federation pays, whether it can be made formally private, and whether the infrastructure leaks. In a simulation with five synthetic retailers built from H&M purchase data, a single retailer's model learns a strong ranking signal but cannot be made formally private with current techniques.

Federation pays most for the retailers that have the least data. When compute is matched, a benefit at equal retailer sizes is not established (a three-seed interval on the relative change in MRR of -5.1% to $+9.9\%$), whereas with pools from 2,000 to 700,000 customers the two smallest retailers gain 52.6% and 10.9% and the three largest show no reliable effect. Customer-level differential privacy was not viable in this setting: the best configuration retained about half the accuracy of the same federation without noise and fell below a retailer training alone. Secure aggregation, by contrast, cost no measurable accuracy (a relative difference of $+0.09\%$, interval $[-1.6\%, +1.8\%]$) and quantised the updates with a relative error of 7.8×10^{-6}, with no coordinate near the clipping range.

A pre-registered suite of four attacks found no detectable leakage from the released model about customers or about planted canaries„ and found that the coordinator can recover a small membership signal from the update it receives. Secure aggregation removes about three quarters of that signal but not all of it. These are empirical results with stated power, and they are not a formal privacy guarantee. They come from one dataset, a tiny adapter, short training and a single-machine simulation, and several attacks use one seed. We view this as a modest risk that can be mitigated through contractual obligations and good infosec practice.

The main practical message is that LLM-based recommenders are competitive and practical.

The benefit of federated learning for LLM-based recommenders depends on the relative sizes of the retailers. The next steps are a pilot with real retailers, agreement on leakage thresholds with privacy and legal stakeholders, and tests of the secure-aggregation pipeline under dropout and across separate hosts.

References

[1] Martin Abadi et al. "Deep Learning with Differential Privacy". In: *ACM CCS*. 2016.

[2] Muhammad Ammad-ud-din et al. *Federated Collaborative Filtering for Privacy-Preserving Personalized Recommendation System*. 2019. arXiv: 1901.09888.

[3] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. "TALLRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation". In: *ACM Conference on Recommender Systems (RecSys)*. 2023.

[4] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. "Secure Single-Server Aggregation with (Poly)Logarithmic Overhead". In: *ACM CCS*. 2020.

[5] Daniel J. Beutel et al. *Flower: A Friendly Federated Learning Research Framework*. 2020. arXiv: 2007.14390.

[6] Keith Bonawitz et al. "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *ACM CCS*. 2017.

[7] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. "The Secret Sharer: Evaluating and Testing Unintended Memorization in Neural Networks". In: *USENIX Security*. 2019.

[8] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. "Recommendation as Language Processing (RLP): A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5)". In: *ACM Conference on Recommender Systems (RecSys)*. 2022.

[9] H&M Group. *H&M Personalized Fashion Recommendations*. Kaggle competition dataset. 2022.

[10] Moein Hasani et al. *A Cascaded Generative Approach for e-Commerce Recommendations*. 2026. arXiv: 2605.11118.

[11] Yupeng Hou et al. "Large Language Models are Zero-Shot Rankers for Recommender Systems". In: *European Conference on Information Retrieval (ECIR)*. 2024.

[12] Edward J. Hu et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *ICLR*. 2022.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. "Collaborative Filtering for Implicit Feedback Datasets". In: *Proceedings of the Eighth IEEE International Conference on Data Mining (ICDM)*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22.

[14] Peter Kairouz, H. Brendan McMahan, et al. *Advances and Open Problems in Federated Learning*. 2019. arXiv: 1912.04977.

[15] Ying Li, Shradha Sehgal, Arjun Rao, Rein Houthooft, Yaochen Zhu, and Ashish Rastogi. *GenRec: An LLM-Backed Recommendation Ranker at Netflix*. 2026. arXiv: 2608.10257.

[16] Han Liu et al. *Sequential LLM Framework for Fashion Recommendation*. 2024. arXiv: 2410.11327.

[17] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. "Communication-Efficient Learning of Deep Networks from Decentralized Data". In: *International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2017.

[18] H. Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. "Learning Differentially Private Recurrent Language Models". In: *International Conference on Learning Representations (ICLR)*. 2018.

[19] Qwen Team. *Qwen3 Technical Report*. arXiv:2505.09388. 2025.

[20] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership Inference Attacks Against Machine Learning Models". In: *IEEE S&P*. 2017.

[21] Likang Wu et al. *A Survey on Large Language Models for Recommendation*. 2023. arXiv: 2305.19860.

[22] Nuofan Xu and Chenhui Hu. *Enhancing E-Commerce Recommendation using Pre-Trained Language Model and Fine-Tuning*. 2023. arXiv: 2302.04443.

[23] Jiaqi Zhai et al. "Actions Speak Louder than Words: Trillion-Parameter Sequential Transducers for Generative Recommendations". In: *International Conference on Machine Learning (ICML)*. 2024.

[24] Jujia Zhao, Wenjie Wang, Chen Xu, See-Kiong Ng, and Tat-Seng Chua. "A Federated Framework for LLM-based Recommendation". In: *Findings of the Association for Computational Linguistics: NAACL*. 2025.

[25] Yuan Zhong, Chuanwei Ruan, Moein Hasani, Tejaswi Tenneti, Haixun Wang, and Fenglong Ma. *GrocLM: Grocery Category Recommendation in E-Commerce with Large Language Models*. 2026. arXiv: 2607.24764.

[26] Ligeng Zhu, Zhijian Liu, and Song Han. *Deep Leakage from Gradients*. 2019. arXiv: 1906.08935.

A Hyperparameters

[TODO:]

B Full DP sweep

[TODO:]

C Phase 0 original three-arm comparison

[TODO: Present with the caveat that arms were not batch-matched.]

D Pre-registered thresholds

[TODO:]

E Reproducibility

[TODO: Deterministic keyed RNG, drivers, data build script.]