# Controllable and Content-Based Recommendations

**Fırat Öncel**[1,5] **Jihoon Jeong**[3,5] **Emiliano Penaloza**[2,5]
**Mirco Ravanelli**[1,5] **Laurent Charlin**[4,5] **Cem Subakan**[1,3,5]

[1]Concordia University [2]Université de Montréal [3]Laval University
[4]HEC Montréal [5]Mila – Quebec AI Institute

firat.oncel@mila.quebec

## Abstract

Traditional recommendation systems rely on latent (dense) representations, making them difficult to interpret and control. We propose the Controllable and Content-Based Recommendations (CCBR) framework, which builds its recommendations from textual user profile representations. CCBR plugs into collaborative filtering models and introduces controllability via text bottlenecks. We show that CCBR enables text-based and multimodal interventions, allowing users to steer the model towards the directions they prefer. Different from existing controllable recommendation systems, CCBR infers the text summaries directly from item contents (images, audio or video). Across image, audio, and video datasets, we demonstrate that the proposed framework obtains competitive model performance with standard (latent-representation) models while providing controllable model summaries via text. The model also outperforms TEARS, a recent baseline for controllable recommendation systems. Through systematic interventions, we demonstrate the efficacy of the steering mechanism.

**Project Page.** For additional demo materials, please visit: https://firatoncel.github.io/ccbr/.
**Code.** To access project codes, please visit https://github.com/firatoncel/CCBRNeurIPS2026

## 1 Introduction

Recommendation systems play a central role in helping users navigate increasingly large catalogs of music, products, movies, and other digital content [1]. Despite their practical success, modern recommendation systems often remain opaque to users [2, 3]. Many state-of-the-art recommendation systems are based on dense architectures and operate via collaborative filtering [4, 5], which are effective for prediction but difficult to interpret [6]. As a result, users have limited insight into why particular items are recommended and little direct control over the system, especially when their current intent differs from the preferences reflected in their past behavior.

This lack of controllability is particularly limiting in scenarios where preferences are context-dependent [7]. For example, a user shopping for clothing may want to temporarily emphasize color, texture, or aesthetic, rather than receive recommendations that simply reflect their long-term purchase history. Standard collaborative filtering methods are not designed to expose such semantic factors to users [8, 4, 9]. Although they are scalable and well-grounded in the item catalog, their internal representations do not inherently provide a user-editable interface for controlling preferences.

Recent work has explored more transparent and controllable recommendation architectures, including systems that condition their final decisions on interpretable bottleneck layers [10, 11, 12]. These approaches suggest that exposing intermediate preference representations can make recommendation

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

systems more interpretable and steerable. In many real-world domains, user preferences are expressed through rich multimodal content: songs, product images, videos, or other media. A controllable recommender should therefore be able to derive semantic user profiles directly from such content while still allowing users to inspect and edit those profiles in natural language.

In order to realize the full potential of text-based representations, this work proposes Controllable and Content-Based Recommendations (CCBR), a content-based recommendation system that creates scrutable and controllable user summaries through item contents. These summaries aggregate user history in a transparent way, so that user preferences can be presented in natural language. The user can understand how the system represents them, and steer it towards the directions they would like the recommendations to take. Unlike earlier controllable recommendation systems [10, 11, 12], CCBR is content-based, deriving user profiles directly from images, audio, or video. Users can interact with the system through multimodal inputs and by editing its textual representations. For example, in a music recommendation scenario, users can upload some of their favorite songs (that the system has not previously seen) to help the model capture more subtle aspects of their taste in the resulting text summary. This is different than existing controllable recommendation systems, as the system ingests the multimodal content directly, and produces the summary. Alternatively, they can directly edit their text-based user summary (e.g., by adding relevant words or sentences) to steer the model toward their current musical preferences.

To achieve this, CCBR generates text-based user summaries by first encoding items in the user's history with multimodal LLMs and then aggregating these item-level representations using a text-based LLM. The resulting summary is passed to a multi-label classification head that predicts the high-level concepts associated with the user. Finally, we use a latent representation guided by multi-label supervision to regularize collaborative filtering embeddings, making recommendation models more interpretable and controllable. We evaluate CCBR across music, shopping, and movie recommendation tasks, demonstrating its ability to generate relevant user summaries. We also show that our method can be successfully applied on top of several state-of-the-art recommendation systems, including EASE [5], EDLAE [13], DAE [13], Multi-DAE [14], Multi-VAE [14], and MacridVAE [15], making them controllable.

Our contribution is as follows:

- We propose CCBR, a framework that introduces controllability to a given collaborative filtering model in a content-driven, multimodal way through a text-based, editable user summary (Section 3).
- We demonstrate that, despite adding controllability, CCBR does not significantly deteriorate the recommendation performance of several popular models, and outperforms a controllable recommendation system baseline (Section 4.2).
- We show that CCBR can steer recommendations by incorporating previously unseen multimodal inputs across H&M, MovieLens-20M (ML-20M), and Million Song Dataset (MSD) datasets (Sections 4.3 and 4.4).

## 2 Related Works

**Collaborative Filtering-based Recommendation Systems.** Collaborative filtering (CF) is a foundational paradigm in recommender systems, which infers user preferences from historical user-item interactions [16]. CF methods have progressed from similarity-based approaches [17, 18, 19] that rely on user-user or item-item neighborhoods, to linear reconstruction models such as EASE [5] and EDLAE [13], and further to neural or variational autoencoder-based models such as DAE [13], Multi-DAE [14], Multi-VAE [14], and MacridVAE [15]. These methods remain effective in large-scale settings since they directly capture collaborative signals and enable efficient retrieval or ranking over a fixed item catalog.

However, conventional CF models primarily encode user preferences as latent, ID-based representations that are difficult to interpret, inspect, or modify [20, 9]. Existing content-aware CF methods can incorporate side information to improve recommendation accuracy, but such information is typically used internally by the model rather than exposed as an explicit user-editable control interface [21]. This limits their applicability to controllable recommendations, in which users may want to directly adjust the semantic factors that influence their results. To address this gap, CCBR exposes user pref-

erences through editable semantic concepts while preserving the scalability and catalog-grounding benefits of CF. By making preferences accessible through human-understandable concepts, CCBR aligns with the motivation of concept bottleneck models [22, 23], which improve interpretability and enable intervention via explicit intermediate concepts.

**LLM-based Recommendation Systems.** Large language models (LLMs) have recently been explored as a way to incorporate rich textual semantics into recommender systems [24, 25], providing a flexible text-based interface and producing human-readable reasoning. However, directly using LLMs as recommenders remains challenging in large-scale catalog settings, where recommendations must be consistent, grounded in available items, and efficiently retrieved from a fixed item set. Moreover, zero-shot and few-shot methods [26, 27] can suffer from hallucination or incomplete catalog coverage, while fine-tuned LLM recommenders [28, 29] often require task-specific adaptation and still need mechanisms for reliable catalog grounding.

Another line of work uses LLM-generated text to enhance existing recommendation models [30, 31], generating user- or item-level textual attributes and integrating them into conventional collaborative filtering. This can improve the quality of representation by exposing high-level semantic information that is not easily captured by sparse user-item interactions alone [11, 32]. For example, TEARS [11] introduces a controllable system that summarizes user preferences in natural language, enabling users to inspect and edit the summaries. However, TEARS and most related approaches rely primarily on textual metadata, such as item titles, descriptions, or user-written content [32, 33]. As a result, they fail to capture preference-relevant signals that are expressed through non-textual content, such as visual style in products, acoustic characteristics in music, or temporal and aesthetic cues in videos.

Recent work has also explored adapting multimodal large language models (MLLMs) to recommendation [34, 35]. Rec-GPT4V [34] uses MLLMs to summarize item images and reason over user preferences, while MLLM-MSR [35] fine-tunes an MLLM-based recommender for sequential multimodal recommendation. Although these approaches demonstrate the potential of multimodal foundation models for recommendation, they mainly focus on improving recommendation accuracy or reasoning capability. The user preference representation is often implicit in the hidden states, making it difficult for users to directly control the recommendation process. Therefore, CCBR bridges multimodal content understanding and controllable collaborative filtering through a cascade that converts multimodal items into textual summaries and merges them into an editable user-level profile.

**Controllable and Preference-Driven Recommendation.** Controllability has recently become an explicit design goal in both research and recommendation systems in production, with natural language serving as a common interface for it. A useful way to organize these approaches is by where language enters the recommendation pipeline. In narrative-driven recommendation [36], an LLM is used offline to synthesize training queries for a retrieval model, so language serves as a query rather than as a learned user representation. A second group keeps the recommender's representation latent and introduces language alongside it: EXP3RT [37] adds an LLM reasoning layer that summarizes reviews to predict and explain ratings, CTRL-Rec [38] models how natural-language requests affect user-item affinities and uses this as a request-conditioned signal over an engagement score, and Mender [39] conditions a generative retrieval model on a natural-language preference string at inference time. Beyond research systems, controllability has also begun to appear in production, where Instagram lets users adjust and reset the interests that drive their Reels recommendations [40]. CCBR is complementary to these efforts: rather than attaching language to a latent representation, it uses an interpretable text profile as the user representation itself, and CCBR derives the presented profile from raw item content across image, audio, and video rather than from reviews, metadata, or interaction history.

## 3 Methodology

We introduce CCBR, a recommendation framework that enables controllability of a collaborative filtering model. In this framework, user preferences are represented as natural-language profiles derived from the content of items that the user interacted with. Unlike prior LLM-based recommendation systems [25, 41], that incorporate items via item identifiers, CCBR enables the creation of user summaries through multimodal content ingestion. Specifically, CCBR generates user profiles from item *content* (images, audio, or video) and uses the LLM as a summarization engine over the item summaries. The resulting text profile serves as a human-readable, editable representation of user

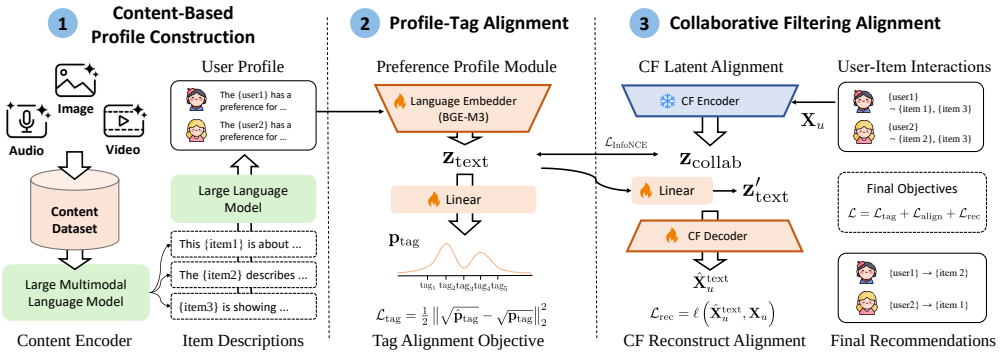


Figure 1: **Overview of the CCBR pipeline.** 1) Item-level content (images, audio, or video) is first verbalized into natural-language descriptions by a multimodal language model, and a language model then aggregates each user's interacted items into a textual preference profile. 2) The profile is mapped to a dense representation through a trainable text encoder and a linear projection head. 3) This text-side representation is aligned with the user representation produced by a frozen collaborative filtering backbone, whose decoder is reused to score candidate items.

taste, while final scoring is performed by a collaborative filtering (CF) backbone that is aligned with the text representation during training. The overall pipeline is given in Figure 1.

**(1) User profile generation.** The raw content of each item is passed through a modality-specific multimodal foundation model to produce a concise natural-language description. This step is performed once per item, offline, and the resulting descriptions are cached and reused across all subsequent training and inference. For each user, the descriptions of items in their interaction history are then passed to an instruction-tuned text-only LLM, which produces a natural-language profile summarizing the user's preferences. Crucially, item identifiers such as titles, product IDs, or artist names are never exposed to either the multimodal or the text-only model, so the generated profile reflects only a content-derived signal, and no pretraining knowledge about specific items can leak into the user representation. The specific models used for each modality are detailed in Section 4.

**(2) Text encoding with tag-supervised pretraining.** We encode the natural-language user profile with BGE-M3 [42], producing a text representation $\mathbf{z}_{\text{text}} \in \mathbb{R}^{1024}$. While general-purpose text encoders capture broad semantic similarity, they are not optimized for the recommendation domain: two summaries that read alike at the surface level may correspond to very different taste profiles, and conversely, summaries phrased differently may describe the same underlying preferences. Dataset-specific tags, movie genres, product categories and colors, music tags, provide a compact and human-interpretable view of user taste that is directly tied to the catalog the CF backbone scores over. Supervising the text encoder with tag distributions therefore serves two purposes: (i) it grounds the textual representation in the same content vocabulary used to evaluate controllability in Section 4.3, ensuring that lexical edits to the profile translate into meaningful shifts along interpretable axes, and (ii) it injects a recommendation-aware inductive bias into the encoder prior to CF alignment, allowing the subsequent alignment stage to start from representations whose geometry already reflects item-content structure rather than generic textual similarity. We train the text encoder in two stages.

In the *first stage*, we leverage dataset-specific tag annotations available for each item. For each user, we construct a target tag distribution $\mathbf{p}_{\text{tag}} \in \Delta^{n_{\text{tags}}}$ by aggregating the tags of items in their interaction history and normalizing to a valid probability distribution. A linear head projects the text embedding to a predicted tag distribution,

$$\hat{\mathbf{p}}_{\text{tag}} = \text{softmax}(\mathbf{W}\,\mathbf{z}_{\text{text}} + \mathbf{b}), \qquad \mathbf{W} \in \mathbb{R}^{n_{\text{tags}} \times d}, \tag{1}$$

and the encoder is trained to match the target distribution under the squared Hellinger distance [43],

$$\mathcal{L}_{\text{tag}} = \frac{1}{2}\left\|\sqrt{\hat{\mathbf{p}}_{\text{tag}}} - \sqrt{\mathbf{p}_{\text{tag}}}\right\|_2^2. \tag{2}$$

To further bridge the gap between the textual representation and the collaborative signal, we introduce a stochastic embedding-substitution mechanism at this stage. At each training step, with equal probability, we either feed the `[CLS]` token embedding of the textual profile into the linear tag

head, or replace it with the user's item embedding average obtained from a pretrained and frozen matrix factorization (MF) model similar to [44]. This stochastic mixing trains a single tag head to be compatible with both representations, which yields two complementary inference modes. When recommendations are made for a user with a known interaction history, the MF-derived embedding can be used at inference to exploit collaborative signals and achieve higher recommendation accuracy. When no interaction history is available, a cold-start user, an edited summary, or a purely text-driven query, the same head decodes directly from the `[CLS]` embedding of the textual profile, allowing the system to operate in a fully content-grounded regime without relying on any prior interaction. We would like to mention that we have also experimented with using a text LLM to obtain a text-encoding of the user profile. We present the recommendation results from this system in Appendix D.

**(3) Collaborative Filtering alignment and scoring.** CCBR is agnostic to the choice of collaborative filtering backbone; we instantiate it with EASE, EDLAE-Full Rank (FR), EDLAE-Low Rank (LR), Linear-DAE, Multi-DAE, Multi-VAE, and MacridVAE, spanning linear item-item, denoising, and variational families. Item-item models are factorized by Singular Value Decomposition (SVD) to create an encoder-decoder architecture. The Collaborative filtering model that we use as a backbone is trained on user-item interactions in the standard way, and we align the text representation from Stage 2) with the collaborative filtering user latent via an InfoNCE [45] objective: for each user, $\mathbf{z}'_{\text{text}}$ and $\mathbf{z}_{\text{collab}}$ form the positive pair, and all other users' collaborative latents in the batch serve as negatives. The CF decoder is copied and finetuned for the text branch in this stage. Stage-wise optimization settings are summarized in Appendix A.1, and detailed definitions of $\mathcal{L}_{\text{align}}$, $\mathcal{L}_{\text{rec}}$ are provided in Appendix A.2.

# 4 Experiments

## 4.1 Experimental setup

We conduct experiments across three data modalities using a strong generalization approach, in which the model is evaluated on users not seen during training. These include recommendations for image-based clothing products, movie recommendations, and music.

**Datasets and data modalities.** The datasets that we use are the H&M fashion dataset [46], which contains images for various clothing items, the MovieLens 20M (ML-20M) dataset [47] (where we use trailers [48] for content), and the Million Song Dataset (MSD) [49] for music recommendations. We follow the strong generalization protocol [14], partitioning users disjointly into training, validation, and test sets so that no user in the held-out splits is observed during training. This protocol is a stricter test of generalization than the more common weak generalization setting, in which held-out interactions come from users seen during training. It requires the model to form a useful representation of an entirely new user from their interaction history alone, which aligns directly with CCBR's content-driven design, since a new user's profile is generated from scratch from the items they have interacted with, with no user-specific parameters learned during training. After preprocessing, H&M contains $1{,}943$ items and $70{,}201$ users (with $1{,}000$ users each held out for validation and test), ML-20M contains $11{,}355$ items and $135{,}269$ users ($5{,}000$ each for validation and test), and MSD contains $31{,}046$ items and $166{,}188$ users ($5{,}000$ each for validation and test).

**User profile creation pipeline.** To create a user profile for a given user, we first generate a concise natural-language description for each item in their interaction history using a modality-specific MLLM. For the H&M dataset, item images are described using Qwen3-VL-32B-Instruct [50]; for ML-20M, movie trailers are described using VideoLLaMA 3 [51]; and for MSD, audio clips are described using MusicFlamingo [52]. This per-item summarization is performed once, offline, and the resulting descriptions are cached and reused across all subsequent training and inference. Once item-level summaries are available, the descriptions of the items in a user's interaction history are aggregated into a single natural-language user profile by an instruction-tuned text-only LLM, for which we use Qwen3-30B-A3B-Instruct-2507 [53]. Throughout the pipeline, item identifiers such as titles, product IDs, or artist names are withheld from both the multimodal and text-only models, ensuring that the resulting profiles reflect only content-derived signals and that no pretraining knowledge about specific items leaks into the user representation.

**Controllability evaluation pipeline.** To quantify the controllability that CCBR enables, we define a systematic editing pipeline over a fixed set of human-interpretable concepts. The concepts are

drawn from the tag vocabulary natively associated with each dataset: movie genres for ML-20M, product categories and colors for H&M, and tag-based descriptors for MSD. This choice grounds the evaluation in attributes that are both meaningful to end users and directly verifiable against item-level metadata, so that any shift in the recommendations induced by an edit can be measured against a concrete content axis rather than a free-form notion of similarity.

Given a user summary, we first identify which of these concepts are explicitly or implicitly mentioned, using an instruction-tuned LLM, Qwen3.5-27B [54], prompted with the closed concept list and asked to return only concepts that appear in it. We then perform two complementary interventions on the summary, again via the same LLM. In the removal intervention, we take a concept currently present in the profile and rewrite the summary so that all explicit and implicit references to it are excised, while the remainder is preserved; this is repeated for each concept, producing a separate edited summary for each present concept. In the addition intervention, we take a concept that is absent from the profile and rewrite the summary so that a single mention of it is woven in without disturbing the existing references; this too is repeated one concept at a time, producing a separate edited summary for each absent concept. The edited summaries are passed through CCBR, and the resulting recommendations are scored against the catalog's concept matrix: a controllable system should produce monotonic shifts in the recommended items, suppressing items that carry a removed concept and promoting items that carry an introduced one. We defer the precise definitions of the intervention effects, normalization, and aggregation to Section 4.3. Dataset-specific backbone settings are detailed in Appendix A.3.

### 4.2 Recommendation performance

In Table 1, we showcase the performance of CCBR across three domains (fashion, music, movies) and seven collaborative filtering backbones. We emphasize that the relevant claim is not uniform performance gains (which would be unrealistic for a deliberately low-bandwidth, human-readable representation) but rather that CCBR remains competitive with strong CF baselines while offering an interpretable and editable user profile, whose controllability we quantify in Section 4.3. CCBR rows in Table 1 further confirm an expected trend: when used in isolation, the text-bottlenecked representation yields lower accuracy than the collaborative backbones, since granular item-level predictions are inherently harder to recover from a compact textual profile than from a collaborative signal trained directly on user-item interactions. Note that, in this and subsequent sections, we only compare with TEARS [11] with VAE-CF models as the original paper provides results with VAE backbones.

Table 1: Recommendation accuracy on H&M, MSD, and ML-20M. We evaluate seven collaborative filtering backbones, our proposed CCBR model with a text-bottlenecked user representation aligned via alignment loss. For TEARS models, we consider merged model with $\alpha = 0.5$ for fair comparison.

| | **H&M** | | | **MSD** | | | **ML-20M** | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) |
| EASE | 0.2621 | 0.2440 | 0.1638 | 0.4519 | 0.3792 | 0.4290 | 0.4909 | 0.5162 | 0.4226 |
| CCBR (Ours) | 0.2212 | 0.2116 | 0.1359 | 0.3271 | 0.2636 | 0.2979 | 0.4629 | 0.4869 | 0.4008 |
| EDLAE (FR) | 0.2693 | 0.2460 | 0.1662 | 0.4616 | 0.3849 | 0.4351 | 0.5028 | 0.5267 | 0.4323 |
| CCBR (Ours) | 0.2215 | 0.2148 | 0.1360 | 0.3129 | 0.2521 | 0.2824 | 0.4792 | 0.5057 | 0.4108 |
| EDLAE (LR) | 0.2695 | 0.2447 | 0.1661 | 0.4619 | 0.3858 | 0.4373 | 0.5032 | 0.5275 | 0.4338 |
| CCBR (Ours) | 0.2215 | 0.2148 | 0.1360 | 0.3233 | 0.2609 | 0.2946 | 0.4802 | 0.5070 | 0.4119 |
| Linear-DAE | 0.2372 | 0.2234 | 0.1432 | 0.3830 | 0.3129 | 0.3499 | 0.4982 | 0.5316 | 0.4261 |
| CCBR (Ours) | 0.2069 | 0.2013 | 0.1211 | 0.3146 | 0.2544 | 0.2819 | 0.4667 | 0.5011 | 0.3904 |
| Multi-DAE | 0.2160 | 0.2080 | 0.1286 | 0.3632 | 0.3000 | 0.3246 | 0.5042 | 0.5423 | 0.4310 |
| CCBR (Ours) | 0.2098 | 0.2055 | 0.1264 | 0.3300 | 0.2687 | 0.2988 | 0.4787 | 0.5282 | 0.4055 |
| Multi-VAE | 0.2244 | 0.2135 | 0.1360 | 0.3843 | 0.3144 | 0.3523 | 0.5148 | 0.5468 | 0.4383 |
| TEARS | 0.2006 | 0.1964 | 0.1161 | 0.2946 | 0.2357 | 0.2602 | 0.4836 | 0.5116 | 0.4105 |
| CCBR (Ours) | 0.2106 | 0.2064 | 0.1252 | 0.3376 | 0.2746 | 0.3065 | 0.4964 | 0.5370 | 0.4263 |
| MacridVAE | 0.2285 | 0.2206 | 0.1413 | 0.3873 | 0.3151 | 0.3542 | 0.4957 | 0.5218 | 0.4111 |
| TEARS | 0.1906 | 0.1855 | 0.1081 | 0.2731 | 0.2158 | 0.2358 | 0.4582 | 0.4814 | 0.3872 |
| CCBR (Ours) | 0.2016 | 0.1992 | 0.1188 | 0.3247 | 0.2641 | 0.2942 | 0.4829 | 0.5204 | 0.4111 |

### 4.3 Controllability Evaluation via Systematic User Summary Editing

Recommendation accuracy alone does not establish that CCBR's text bottleneck is *causally engaged*: a text summary could, in principle, carry information that the CF backbone ignores at inference time. To test whether modifying the natural-language user profile produces a faithful change in the recommended items, we design a pair of counterfactual interventions, Leave-One-Out (LOO) removal and Add-One-In (AOI) addition. A controllable recommendation system should respond monotonically to these edits: removing an attribute from the summary should suppress items carrying it, and adding one should promote them.

**Setup and notation.** We instantiate *attributes* from the tag vocabularies natively associated with each dataset. For movies (ML-20M), attributes correspond to movie genres ($C = 18$); for H&M clothing products, we use dominant colors ($C = 16$) and clothing categories ($C = 51$); for music (MSD), we use tag-based genre descriptors ($C = 80$). Let $\mathcal{A} = \{a_1, \ldots, a_C\}$ denote this attribute set, and let $\mathcal{A}_i \subseteq \{1, \ldots, C\}$ denote the indices of attributes carried by item $i$. For every user $u$, we have a natural-language summary $s_u$ produced by the profile generator pipeline of Section 4.1, and an attribute index set $\mathcal{A}_u \subseteq \{1, \ldots, C\}$ obtained by prompting an instruction-tuned LLM to identify which attributes are explicitly or implicitly mentioned in $s_u$. Given a summary $s$, the trained CCBR model produces a score vector $\mathbf{y}(s) \in \mathbb{R}^N$ over the item catalog. We measure the effect of an edit by comparing the number of top-$K$ recommended items that carry the targeted attribute before and after the edit. For each user $u$, we construct two kinds of edited summaries using the same instruction-tuned LLM.

**Add-One-In (AOI) additions and Leave-One-Out (LOO) removals.** An addition edit takes an attribute $c$ that is absent from $\mathcal{A}_u$ and weaves a single mention of $a_c$ into $s_u$ without disturbing the existing references, yielding $\tilde{s}_u^{+c}$. A removal edit takes an attribute $c$ that is present in $\mathcal{A}_u$ and rewrites $s_u$ so that all explicit and implicit references to $a_c$ are excised while the remainder is preserved, yielding $\tilde{s}_u^{\setminus c}$. Each edit targets one attribute at a time, so a user with $n_u = |\mathcal{A}_u|$ mentioned attributes contributes $n_u$ removal edits and $C - n_u$ addition edits. Users with $n_u < 2$ are excluded from the removal analysis, since no attribute remains after the edit. Note that these edits are achieved through prompting an LLM. We assess the locality of H&M color and category edits using two independent LLM-as-a-judge models, finding that more than $98\%$ of edits preserve non-target profile information (Appendix B.1). The exact prompt are shown in Appendix B.2.

**Measuring the effect of an edit.** For a summary $s$, we measure how strongly attribute $c$ is represented in the top-$K$ recommendations by the per-attribute recall

$$r_c^{(K)}(s) \;=\; \frac{1}{K} \sum_{i \in \mathrm{Top}_K(s)} \mathbf{1}\big[a_c \in \mathcal{A}_i\big], \tag{3}$$

i.e., the fraction of top-$K$ items that carry attribute $c$. The per-user effect of an edit is the difference in this recall between the edited and original summaries,

$$\Delta r_c^{(K)}(u) \;=\; r_c^{(K)}(\tilde{s}_u^{\bullet c}) - r_c^{(K)}(s_u), \tag{4}$$

where $\tilde{s}_u^{\bullet c}$ denotes the edited user profile (either $\tilde{s}_u^{+c}$ for additions or $\tilde{s}_u^{\setminus c}$ for removals). A controllable system should yield negative $\Delta r_c^{(K)}$ for removals and positive $\Delta r_c^{(K)}$ for additions. We aggregate these per-user differences by first averaging over all users for whom the attribute was a valid target (present users for removal, absent users for addition), and then averaging across attributes to obtain a global controllability score at cutoff $K$.

**Controllability analysis.** In Figure 2, we showcase the controllability results obtained with CCBR applied to seven different collaborative filtering methodologies presented earlier. We selectively target a concept shown on the x-axis of each panel. We conduct experiments to remove and add concepts. We report the aggregate concepts that are derived from the recommended items after the edit on the y-axis of Figure 2.

We observe that CCBR is able to very significantly remove the targeted concepts as showcased in the right panel of Figure 2. We also see that, for additions, CCBR manages to add certain movie genres to a significant degree and others to a lesser degree. This difference is potentially due to differences in frequencies across movie genres and the presence of multiple-labels for a given movie. For instance the drama genre ($5{,}723$ occurrences across $11{,}355$ items) is a genre relatively harder to

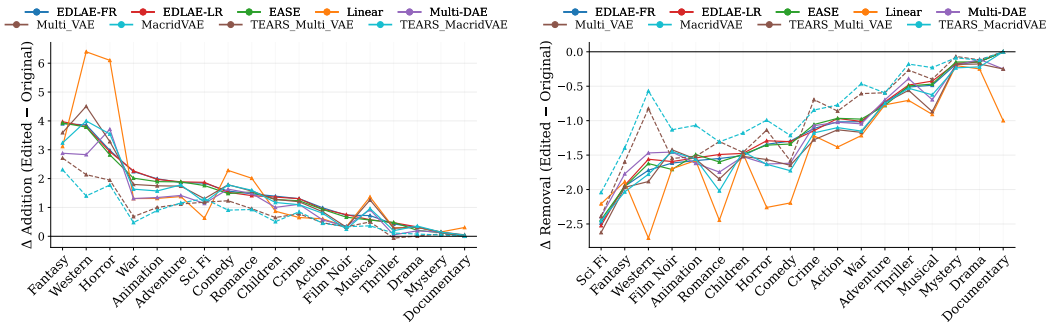


Figure 2: Controllability (@top-20) on ML-20M across seven collaborative filtering backbones (solid) and two TEARS-trained variants (dashed, same colors as their backbones). Curves show the difference (edited − original) in the per-genre share of recommended items, for addition **(left)** and removal **(right)**. Our models consistently outperform their TEARS equivalents: addition deltas are larger (more positive) and removal deltas are larger in magnitude (more negative), meaning summary edits steer the recommender more strongly in the intended direction.

remove or insert because of its high prevalence, and large number of interactions with other genres (e.g., about 1,200 movies with romance tag also have the drama tag). On the other extreme, only few items have 'documentary' tag (775), and only four users in the intervention set have watched movies with the documentary tag. We would like to also emphasize that for VAE-based collaborative filtering backbones, CCBR is able to more significantly affect the output compared to TEARS. We also show the editing results for clothing items and music in Appendix C.1 and C.2.

### 4.4 Editing the user summary with multimodal inputs

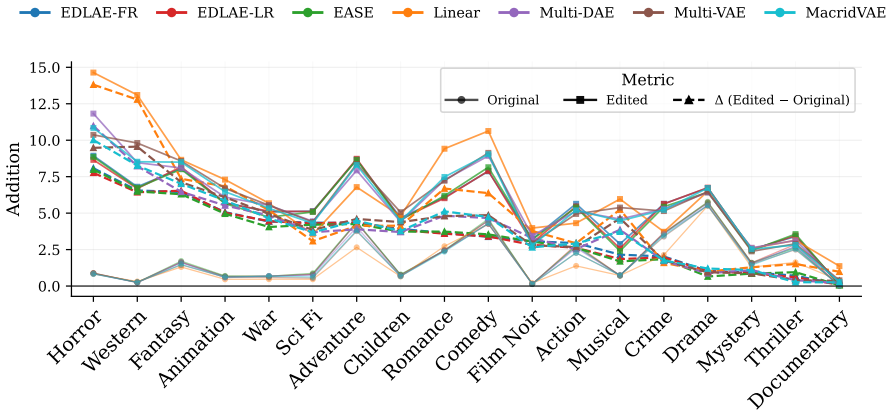


Figure 3: Showcasing controllability on adding unseen items on the ML-20M dataset. We show the controllability for addition, for seven different collaborative filtering backbones as shown in the legend.

The addition intervention of Section 4.3 edits the user summary at the text level: an instruction-tuned LLM is prompted to insert a single mention of a target attribute into the user summary $s_u$, or to excise references to it for removals. While this is sufficient to test whether the text bottleneck responds to lexical changes, it leaves open the more demanding question of whether CCBR can ingest *new content* through multimodal inputs (images, videos, or audio) that were not seen at training time and route it faithfully through the user profile. We address this with a complementary intervention in which the addition is performed at the item level, using items held out from the catalog during training.

For an additional edit, we maintain a pool of items absent from both the CF training set and the user histories used for profile generation, and obtain their content-derived descriptions using the

same modality-specific pipeline as in Section 3. For a user $u$ and a target attribute that is not already present in the user's summary ($c \notin \mathcal{A}_u$), we sample unseen items carrying the new concept $a_c$, and re-summarize to produce an edited profile $\tilde{s}_u^{+c}$ that reflects the newly injected items. The key difference from the text-level intervention is that the new concept enters the profile via the modality-specific encoder rather than through direct lexical insertion, allowing us to measure whether CCBR can pick up genuinely unseen content from raw multimodal signals.

This experiment provides two guarantees that the text-level addition cannot. First, it isolates a genuine cold-start regime: the inserted item has no CF representation, so any influence on the recommendation list must be mediated by the text channel. Second, it tests the full pipeline end-to-end (modality encoder, profile editor, text encoder, and CF-aligned decoder) under the same conditions in which CCBR would be deployed when new items enter the catalog after training. We showcase the obtained editing results in Figure 3. We observe that with multimodal inputs, CCBR is able to obtain a level of controllability similar to the level obtained with text-only summary edits. Results for the unseen setting on H&M and MSD are shown in Appendices C.1 and C.2.

### 4.5 Steering the Model with Negative Preferences

In Section 4.3 we examined controllability through the removal of concepts from the text summary. That is, we tested whether CCBR can be steered by *negating* a concept, i.e., explicitly stating that the user dislikes the targeted attribute rather than merely deleting references to it. The H&M and MSD datasets expose only positive interaction signals (purchases, plays), and the user summaries produced in Section 3 accordingly describe taste in purely affirmative terms. ML-20M, by contrast, provides explicit five-star ratings, giving us access not only to what a user liked but also to what they actively disliked. This asymmetry lets us probe whether CCBR's text bottleneck can carry *signed* preference information and whether the CF backbone responds to it as expected.

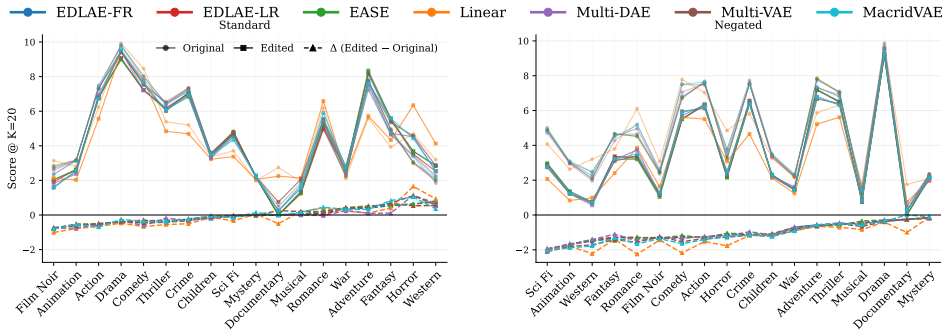


Figure 4: Negation intervention on ML-20M, where the targeted genre in the user summary is rewritten as a dispreference (e.g., "the user enjoys horror" $\rightarrow$ "the user dislikes horror"). Solid lines show per-genre recall at $k = 20$ for the original and edited summaries; dashed lines show the difference between them. Standard model (**left**) is trained on positive summaries while negated model (**right**) is trained on signed summaries.

**Summary generation with negative preferences.** We treat ratings $\geq 4$ as positive interactions and additionally retain ratings $\leq 2$ as *negative* interactions. For each user we partition the history into a positive set $\mathcal{H}_u^+$ and a negative set $\mathcal{H}_u^-$, and prompt the profile-generation LLM with both sets, instructing it to summarize $\mathcal{H}_u^+$ in affirmative language (*"the user likes..."*) and $\mathcal{H}_u^-$ in dispreference language (*"the user dislikes..."*). As in the positive-only setting, item identifiers are withheld, and the LLM only sees content-derived descriptions, so the resulting profiles remain free of pretraining leakage. CF training is left unchanged: only positively-rated items contribute to $\mathcal{H}_u^+$ used by the collaborative backbone, ensuring that any improvement attributable to the signed summary comes through the text channel.

Figure 4 compares the negation intervention on two CCBR variants: a *Standard* model trained only on positive interactions, and a *Negated* model trained on signed summaries. Only the Negated model produces a consistent downward shift in the targeted genre's top-$K$ recall, confirming that the text bottleneck transmits signed preference information end-to-end and that the CF backbone responds to negative steering in the expected direction.

**Computational cost.** We separate two regimes for computational cost comparison: a one-time offline cost per item, and the online cost of modeling a new user. The multimodal description of each item is computed once and cached, then reused across all training and inference, so it never recurs at serving time; for the video modality, VideoLLaMA 3 takes $4672.9 \pm 2615.4$ ms per item on a single H100-80GB ($\sim$14.7 GPU-hours for the 11,355-item ML-20M catalog), with the image and audio encoders at comparable per-item cost. At serving time, we simulate 100 users per modality (batch size 1) with $\mathrm{Uniform}(10, 30)$ interacted items on a single A100-80GB, across three stages: (A) generating the user summary, (B) encoding it, and (C) recommending top-20 items (Table 2). The recurring recommendation-time cost is negligible, embedding takes $\sim$20 ms and scoring under 1 ms, independent of catalog size; the only substantial costs are the cached per-item description and the per-user summary generation ($\sim$10–13 s), incurred once when a profile is created or edited and not on every recommendation.

Table 2: Online (serving-time) latency per new user across the three CCBR stages, on a single A100-80GB over 100 simulated users per modality (batch size 1), each with $\mathrm{Uniform}(10, 30)$ interacted items.

| Modality (#items) | A: summary (ms) | B: embed (ms) | C: recommend (ms) |
|---|---|---|---|
| Image (1,943 items) | $10611.8 \pm 1000.6$ | $20.16 \pm 12.55$ | $0.104 \pm 0.010$ |
| Audio (31,046 items) | $12748.9 \pm 974.6$ | $21.36 \pm 13.87$ | $0.317 \pm 0.960$ |
| Video (11,355 items) | $10673.2 \pm 880.4$ | $20.07 \pm 14.82$ | $0.251 \pm 0.859$ |

## 5 Conclusions

We present CCBR, a content-based recommendation framework that introduces controllability to standard collaborative filtering models through an editable, natural-language user profile. By verbalizing items with modality-specific foundation models (images, audio, or video), aggregating them into a textual profile via an instruction-tuned LLM, and aligning the resulting representation with a CF backbone via InfoNCE, CCBR creates a human-readable bottleneck that users can inspect and modify, while preserving the catalog grounding and ranking quality of established collaborative filtering methods. Across three modalities (clothing images, movie trailers, music audio) and seven CF backbones, we observe that CCBR does not significantly deteriorate the recommendation performance while adding the ability to steer the system towards user preferences.

Our systematic editing experiments show that CCBR is causally engaged: leave-one-out and add-one-in interventions on the user summary produce monotone, attribute-faithful shifts in the recommended items. The framework further supports multimodal interventions by ingesting previously unseen items through the modality-specific encoder and reflecting them in the recommendations. The model also supports dis-preference statements that can suppress the targeted concept. Together, these results indicate that CCBR offers a practical path toward recommendation systems that are accurate, interpretable, and directly steerable by their users.

Taken together, our results point to controllability as a property that can be added to existing recommenders at modest cost rather than as a competing objective. Across three modalities and seven collaborative filtering backbones, the editable text bottleneck preserves most of the ranking quality of the underlying models while exposing a representation that users can read and intervene on, and the leave-one-out, add-one-in, multimodal, and negative-preference interventions all move the recommendations in the intended direction. Because the collaborative model enters CCBR only as a scorer aligned to the text latent, the same mechanism applies uniformly across the linear, denoising, and variational families we consider, suggesting that the interpretability and steerability CCBR provides are largely decoupled from the specific choice of backbone.

A natural avenue for future work is to extend CCBR beyond per-user offline profile generation to interactive settings, where the user iteratively refines the textual summary in dialogue with the system, and to study how the framework composes with personalization signals beyond interaction histories, such as explicit goals or contextual constraints.

## Acknowledgments

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Digital Research Alliance of Canada (alliancecan.ca). Cem Subakan, and Mirco Ravanelli are respectively supported by the NSERC projects RGPIN-2023-05759, and RGPIN-2022-05088. Mirco Ravanelli also is supported by Translated through the Imminent research grant. Emiliano Penaloza acknowledges the support of the NSERC PGS-D grant and the Bourse en intelligence artificielle provided by Université de Montréal. Laurent Charlin recognizes the support of the Canada CIFAR AI Chair Program, the Canada First Research Excellence Fund and IVADO. We have not received any support for this work except the regular research funding indicated above, and we declare no competing interests.

## References

[1] Francesco Ricci, Lior Rokach, and Bracha Shapira. Recommender systems: Techniques, applications, and challenges. *Recommender systems handbook*, pages 1–35, 2021.

[2] Behnoush Abdollahi and Olfa Nasraoui. Explainable restricted boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1606.07129*, 2016.

[3] Marta Caro-Martínez, Guillermo Jiménez-Díaz, and Juan A Recio-García. Conceptual modeling of explainable recommender systems: an ontological formalization to guide their design and development. *Journal of Artificial Intelligence Research*, 71:557–589, 2021.

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

[5] Harald Steck. Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference*, WWW '19, page 3251–3257. ACM, May 2019.

[6] Faisal M Almutairi, Nicholas D Sidiropoulos, and Bo Yang. Xpl-cf: Explainable embeddings for feature-based collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2847–2851, 2021.

[7] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender systems handbook*, pages 217–253. Springer, 2010.

[8] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[9] Sheshera Mysore, Mahmood Jasim, Andrew McCallum, and Hamed Zamani. Editable user profiles for controllable text recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 993–1003, 2023.

[10] Filip Radlinski, Krisztian Balog, Fernando Diaz, Lucas Dixon, and Ben Wedin. On natural language user profiles for transparent and scrutable recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

[11] Emiliano Penaloza, Olivier Gouvert, Haolun Wu, and Laurent Charlin. Tears: Text representations for scrutable recommendations. In *Proceedings of the ACM on Web Conference*, 2025.

[12] Fırat Öncel, Emiliano Penaloza, Haolun Wu, Shubham Gupta, Mirco Ravanelli, Laurent Charlin, and Cem Subakan. Audio prototypical network for controllable music recommendation. In *IEEE MLSP*, 2025.

[13] Harald Steck. Autoencoders that don't overfit towards the identity. In *NeurIPS*, 2020.

[14] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *The World Wide Web Conference*, 2018.

[15] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. Learning disentangled representations for recommendation. In *NeurIPS*, 2019.

[16] Hyeyoung Ko, Suyeon Lee, Yoonseo Park, and Anna Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[18] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

[19] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[20] Krisztian Balog, Filip Radlinski, and Shushan Arakelyan. Transparent, scrutable and explainable user models for personalized recommendation. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*, pages 265–274, 2019.

[21] Wen-Hao Chen, Chin-Chi Hsu, Yi-An Lai, Vincent Liu, Mi-Yen Yeh, and Shou-De Lin. Attribute-aware recommender system based on collaborative filtering: Survey and classification. *Frontiers in big Data*, 2:49, 2020.

[22] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In *International conference on machine learning*, pages 5338–5348. PMLR, 2020.

[23] Sungbin Shin, Yohan Jo, Sungsoo Ahn, and Namhoon Lee. A closer look at the intervention procedure of concept bottleneck models. In *International conference on machine learning*, pages 31504–31520. PMLR, 2023.

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

[25] 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 *Proceedings of the 17th ACM conference on recommender systems*, pages 1007–1014, 2023.

[26] Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*, 2023.

[27] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM conference on recommender systems*, pages 1126–1132, 2023.

[28] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. Do llms understand user preferences? evaluating llms on user rating prediction. *arXiv preprint arXiv:2305.06474*, 2023.

[29] Zhi Zheng, Wenshuo Chao, Zhaopeng Qiu, Hengshu Zhu, and Hui Xiong. Harnessing large language models for text-rich sequential recommendation. In *Proceedings of the ACM Web Conference 2024*, pages 3207–3216, 2024.

[30] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM international conference on web search and data mining*, pages 806–815, 2024.

[31] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. Representation learning with large language models for recommendation. In *Proceedings of the ACM web conference 2024*, pages 3464–3475, 2024.

[32] Hanjia Lyu, Song Jiang, Hanqing Zeng, Yinglong Xia, Qifan Wang, Si Zhang, Ren Chen, Chris Leung, Jiajie Tang, and Jiebo Luo. Llm-rec: Personalized recommendation via prompting large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 583–612, 2024.

[33] Bin Yin, Junjie Xie, Yu Qin, Zixiang Ding, Zhichao Feng, Xiang Li, and Wei Lin. Heterogeneous knowledge fusion: A novel approach for personalized recommendation via llm. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 599–601, 2023.

[34] Yuqing Liu, Yu Wang, Lichao Sun, and Philip S Yu. Rec-gpt4v: Multimodal recommendation with large vision-language models. *arXiv preprint arXiv:2402.08670*, 2024.

[35] Yuyang Ye, Zhi Zheng, Yishan Shen, Tianshu Wang, Hengruo Zhang, Peijun Zhu, Runlong Yu, Kai Zhang, and Hui Xiong. Harnessing multimodal large language models for multimodal sequential recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

[36] Sheshera Mysore, Andrew Mccallum, and Hamed Zamani. Large language model augmented narrative driven recommendations. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 777–783, New York, NY, USA, 2023. Association for Computing Machinery.

[37] Jieyong Kim, Hyunseo Kim, Hyunjin Cho, SeongKu Kang, Buru Chang, Jinyoung Yeo, and Dongha Lee. Review-driven personalized preference reasoning with large language models for recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 1697–1706, New York, NY, USA, 2025. Association for Computing Machinery.

[38] Micah Carroll, Adeline Foote, Kevin Feng, Marcus Williams, Anca Dragan, W. Bradley Knox, and Smitha Milli. Ctrl-rec: Controlling recommender systems with natural language, 2025.

[39] Fabian Paischer, Liu Yang, Linfeng Liu, Shuai Shao, Kaveh Hassani, Jiacheng Li, Ricky T. Q. Chen, Zhang Gabriel Li, Xiaoli Gao, Wei Shao, Xue Feng, Nima Noorshams, Sem Park, Bo Long, and Hamid Eghbalzadeh. Preference discerning with LLM-enhanced generative retrieval. *Transactions on Machine Learning Research*, 2025.

[40] Instagram. Control your Instagram Reels algorithm. `https://about.instagram.com/blog/announcements/reels-algorithm-control`, 2025. Accessed: 2026-11-04.

[41] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

[42] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

[43] Ernst Hellinger. Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, 1909(136):210–271, 1909.

[44] Guy Tennenholtz, Yinlam Chow, ChihWei Hsu, Jihwan Jeong, Lior Shani, Azamat Tulepbergenov, Deepak Ramachandran, Martin Mladenov, and Craig Boutilier. Demystifying embedding spaces using large language models. In *ICLR*, 2024.

[45] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748, 2018.

[46] Carlos García Ling, ElizabethHMGroup, FridaRim, inversion, Jaime Ferrando, Maggie, neuraloverflow, and xlsrln. H&m personalized fashion recommendations. `https://kaggle.com/competitions/h-and-m-personalized-fashion-recommendations`, 2022. Kaggle.

[47] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 2015.

[48] Sami Abu-El-Haija, Joonseok Lee, Max Harper, and Joseph Konstan. Movielens 20m youtube trailers dataset. In *MovieLens*, 2018.

[49] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[50] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[51] Boqiang Zhang, Kehan Li, Zesen Cheng, Zhiqiang Hu, Yuqian Yuan, Guanzheng Chen, Sicong Leng, Yuming Jiang, Hang Zhang, Xin Li, Peng Jin, Wenqi Zhang, Fan Wang, Lidong Bing, and Deli Zhao. Videollama 3: Frontier multimodal foundation models for image and video understanding. *arXiv preprint arXiv:2501.13106*, 2025.

[52] Sreyan Ghosh, Arushi Goel, Lasha Koroshinadze, Sang-gil Lee, Zhifeng Kong, Joao Santos, Ramani Duraiswami, Dinesh Manocha, Wei Ping, Mohammad Shoeybi, and Bryan Catanzaro. Music flamingo: Scaling music understanding in audio language models. In *International Conference on Learning Representations*, 2026.

[53] Qwen Team. Qwen3 technical report. 2025.

[54] Qwen Team. Qwen3.5: Towards native multimodal agents. February 2026.

[55] Z.ai. GLM-4-32B-0414. `https://huggingface.co/zai-org/GLM-4-32B-0414`, 2025.

[56] Gemma Team. Gemma 4 Technical Report. 2026.

[57] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 639–648, New York, NY, USA, 2020. Association for Computing Machinery.

# A Training Details

## A.1 General Training Setup

For each CF backbone, we perform hyperparameter tuning by sweeping between $30$ and $50$ configurations, with the exact number depending on the model and dataset.

We begin by training a matrix factorization (MF) model on the training interactions and store the resulting $1024$-dimensional item embedding table. A weight decay term is applied during MF training so that the token norm distribution of the learned embeddings matches that of the text encoder. After this step, only the item embedding table is retained.

In the first stage, the text encoder backbone is finetuned with a learning rate of $5\mathrm{e}{-6}$ and the linear heads with a learning rate of $3\mathrm{e}{-4}$ for $100$ epochs, using only the Hellinger objective. We use a batch size of $64$ and, with probability $0.5$, replace the `[CLS]` embedding with the mean of the user's item embeddings. We also apply dropout with rate $0.2$ to user interactions.

In the second stage, the text encoder backbone learning rate is lowered to $3\mathrm{e}{-6}$. The CF decoder is copied and finetuned for the text branch with a learning rate of $3\mathrm{e}{-5}$, while the loss coefficient for tag prediction is reduced by a factor of $10$. Dropout on user interactions is increased to $0.6$ in this stage. The linear head mapping the text encoder output to the text latent is trained with a learning rate of $3\mathrm{e}{-4}$. We use a batch size of 96 in this stage.

For reconstruction, we use normalized MSE loss for EASE, EDLAE-LR, and EDLAE-FR, and a multinomial objective for the remaining models. For EASE, EDLAE-LR, EDLAE-FR, Linear-DAE, and Multi-DAE, we apply $\ell_2$ normalization to the latents and use InfoNCE alone as the alignment loss. For VAE-based backbones, we omit $\ell_2$ normalization and add a Smooth-L1 loss to match the norms of the text and VAE latents.

## A.2 Training Objectives

We now formalize the losses used in each training stage. Let $\mathbf{x} \in \{0,1\}^{|\mathcal{I}|}$ denote a user's binary interaction vector over the item set $\mathcal{I}$, and let $\hat{\mathbf{x}}$ denote the reconstruction logits produced by the CF decoder of the text branch. Let $\mathbf{z}'_{\text{text}}$ denote the projected text latent and $\mathbf{z}_{\text{collab}}$ the user latent of the frozen CF backbone, both in $\mathbb{R}^d$, where $d$ is the latent dimensionality of the backbone (Appendix A.3).

**Tag loss.** $\mathcal{L}_{\text{tag}}$ is the squared Hellinger distance between the predicted and target tag distributions $\hat{\mathbf{p}}_{\text{tag}}$ and $\mathbf{p}_{\text{tag}}$, as defined in Section 3.

**Alignment loss.** For a batch of $B$ users, the InfoNCE objective with temperature $\tau$ is

$$\mathcal{L}_{\text{InfoNCE}} \;=\; -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp\Big(\langle\tilde{\mathbf{z}}'^{(i)}_{\text{text}},\,\tilde{\mathbf{z}}^{(i)}_{\text{collab}}\rangle/\tau\Big)}{\sum_{j=1}^{B}\exp\Big(\langle\tilde{\mathbf{z}}'^{(i)}_{\text{text}},\,\tilde{\mathbf{z}}^{(j)}_{\text{collab}}\rangle/\tau\Big)}. \tag{5}$$

For non-VAE backbones (EASE, EDLAE-LR, EDLAE-FR, Linear-DAE, Multi-DAE), both latents are $\ell_2$-normalized, $\tilde{\mathbf{z}} = \mathbf{z}/\|\mathbf{z}\|_2$, and $\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{InfoNCE}}$. For VAE-based backbones (Multi-VAE, MacridVAE), we omit the normalization, $\tilde{\mathbf{z}} = \mathbf{z}$, and additionally match the latent norms with a Smooth-L1 term:

$$\mathcal{L}_{\text{align}} \;=\; \mathcal{L}_{\text{InfoNCE}} \;+\; \mathrm{SmoothL1}\big(\|\mathbf{z}'_{\text{text}}\|_2,\;\|\mathbf{z}_{\text{collab}}\|_2\big)\,. \tag{6}$$

**Reconstruction loss.** For EASE, EDLAE-LR, and EDLAE-FR, we use a normalized MSE loss between the decoder logits and the interaction vector:

$$\mathcal{L}_{\text{nMSE}} \;=\; \frac{1}{|\mathcal{I}|}\left\|\frac{\hat{\mathbf{x}}}{\|\hat{\mathbf{x}}\|_2} - \frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right\|_2^2. \tag{7}$$

For the remaining backbones (Linear-DAE, Multi-DAE, Multi-VAE, and MacridVAE), we use the multinomial log-likelihood:

$$\mathcal{L}_{\text{mult}} \;=\; -\sum_{k=1}^{|\mathcal{I}|} x_k\,\log\mathrm{softmax}(\hat{\mathbf{x}})_k. \tag{8}$$

**Stage 1.** The text encoder and linear heads are trained with the tag loss alone, $\mathcal{L}_{\text{stage 1}} = \mathcal{L}_{\text{tag}}$.

**Stage 2.** The tag loss is down-weighted by a factor of 10, and the alignment and reconstruction losses are added with weights $\alpha_2$ and $\alpha_3$:

$$\mathcal{L}_{\text{stage 2}} \;=\; 0.1 \cdot \mathcal{L}_{\text{tag}} \;+\; \alpha_2 \cdot \mathcal{L}_{\text{align}} \;+\; \alpha_3 \cdot \mathcal{L}_{\text{rec}}, \tag{9}$$

where $\mathcal{L}_{\text{rec}}$ is $\mathcal{L}_{\text{nMSE}}$ for EASE, EDLAE-LR, and EDLAE-FR and $\mathcal{L}_{\text{mult}}$ otherwise.

### A.3 Dataset-specific Settings

**ML-20M.** We follow the general recipe described above. For the variant that incorporates negated summaries, we additionally train a second MF model on the negative interactions extracted from the user histories. The EASE and EDLAE-FR models are factorized to a $1024$-dimensional latent space via SVD.

**MSD.** It contains approximately $31{,}046$ items, and a $1024$-dimensional latent space is insufficient for the linear models in this regime. We therefore use a $4096$-dimensional latent space for EASE, EDLAE-FR, EDLAE-LR, and Linear-DAE on this dataset. For Multi-DAE, Multi-VAE, and MacridVAE, increasing the latent dimensionality did not yield further gains, so we retain the $1024$-dimensional latent space for these backbones.

**H&M.** Roughly one third of the items in H&M are black, which introduces a substantial class imbalance during the first stage of training. To mitigate this, we augment the training set by intervening on the user summaries to modify or remove explicit references to the color black, and use this augmented set for the first stage of training. The second stage is then trained on the original, unaugmented summaries.

## B Controllability Details

### B.1 Independent Validation of Edit Locality

We assess whether H&M color and category edits preserve non-target profile information using two independent LLM-as-a-judge models, GLM-4 32B-0414 [55] and Gemma-4 12B [56]. For each (before, after, target attribute) triple, each judge returns a binary `preserved` label. An edit is judged preserved only when all non-target preferences retain their meaning and no new preferences are introduced. As shown in Table 3, both judges consistently find that the edits preserve non-target profile information across addition and removal interventions, with preservation rates above $98\%$ for every intervention. This suggests that the edits are generally localized to the target attribute rather than substantially rewriting the user profile.

Table 3: Independent LLM-as-a-judge evaluation of whether H&M profile edits preserve non-target information. Values report the percentage of edits judged to preserve the remainder of the profile.

| **Intervention** | **GLM-4 32B-0414** | **Gemma-4 12B** |
|---|---|---|
| Remove color | 100.00% | 99.83% |
| Remove category | 99.98% | 98.01% |
| Add color | 100.00% | 99.64% |
| Add category | 99.99% | 99.73% |

### B.2 Prompt Details

**Attribute extraction prompt.**

```
"You are a precise movie genre extractor.  Given a user watching summary,
identify which movie genres are mentioned or clearly implied.
IMPORTANT RULES:
- Only extract genres that are explicitly mentioned or very strongly implied by
the summary.
- Map related terms to the closest valid genre using these rules:
  - "science fiction" or "sci fi" → Sci-Fi
  - "noir" or "film noir" → Film-Noir
  - "kids" or "family" → Children
  - "animated" or "cartoon" → Animation
  - "suspense" or "psychological thriller" → Thriller
  - "romantic" or "love story" or "rom-com" → Romance
  - "dark comedy" or "satire" or "slapstick" → Comedy
  - "war film" or "military" or "combat" → War
  - "detective" or "whodunit" or "murder mystery" → Mystery
  - "gangster" or "heist" or "mob" or "mafia" → Crime
  - "superhero" or "martial arts" or "kung fu" → Action
  - "spy" or "espionage" → Thriller
  - "zombie" or "slasher" or "supernatural" or "gothic" → Horror
  - "fairy tale" or "mythological" or "magical" → Fantasy
  - "space opera" or "dystopian" or "cyberpunk" or "time travel" → Sci-Fi
  - "cowboy" or "frontier" or "wild west" → Western
  - "biographical" or "biopic" or "courtroom" or "coming of age" → Drama
  - "song and dance" or "concert" → Musical
  - "nature documentary" or "docudrama" → Documentary
  - "quest" or "expedition" or "survival" → Adventure
- Do NOT hallucinate genres not present or implied in the summary.
VALID GENRES:
{genres}
Respond ONLY with valid JSON, no other text:
{"genres":  ["Genre1", "Genre2", ...]}"
```

**Removal edit prompt.**

```
"You are a precise text editor.  Your task is to edit a movie watching summary to
REMOVE all mentions and implications of ONE specific movie genre, while keeping
the rest of the summary natural and coherent.
GENRE TO REMOVE: {genre_to_remove}
ORIGINAL SUMMARY:
{summary}
RULES:
- Remove all explicit and implicit references to the specified genre.
- Do NOT add any new movie genres or attributes.
- Keep the overall structure, tone, and remaining content intact.
- The edited summary should read naturally, as if that genre was never part of
the user's taste.
- If removing the genre leaves an awkward gap or broken sentence, smooth it out
minimally.
- Do NOT remove any other genres or attributes beyond the one specified.
Respond with ONLY the edited summary text, nothing else."
```

**Addition edit prompt.**

```
"You are a precise text editor.  Your task is to edit a movie watching summary to
naturally INCORPORATE mentions of ONE specific new movie genre, while keeping the
rest of the summary coherent.
GENRE TO ADD: {genre_to_add}
ORIGINAL SUMMARY:
{summary}
RULES:
- Weave the specified genre naturally into the existing summary.
- It should feel like a genuine part of the user's watching preferences.
- Do NOT remove any existing genres or attributes.
- Keep the overall structure and tone consistent.
- Add minimal text, just enough to incorporate the new genre naturally.
- The genre should be clearly identifiable in the edited text (explicitly
mentioned or very strongly implied).
Respond with ONLY the edited summary text, nothing else."
```

# C Plots

## C.1 H&M

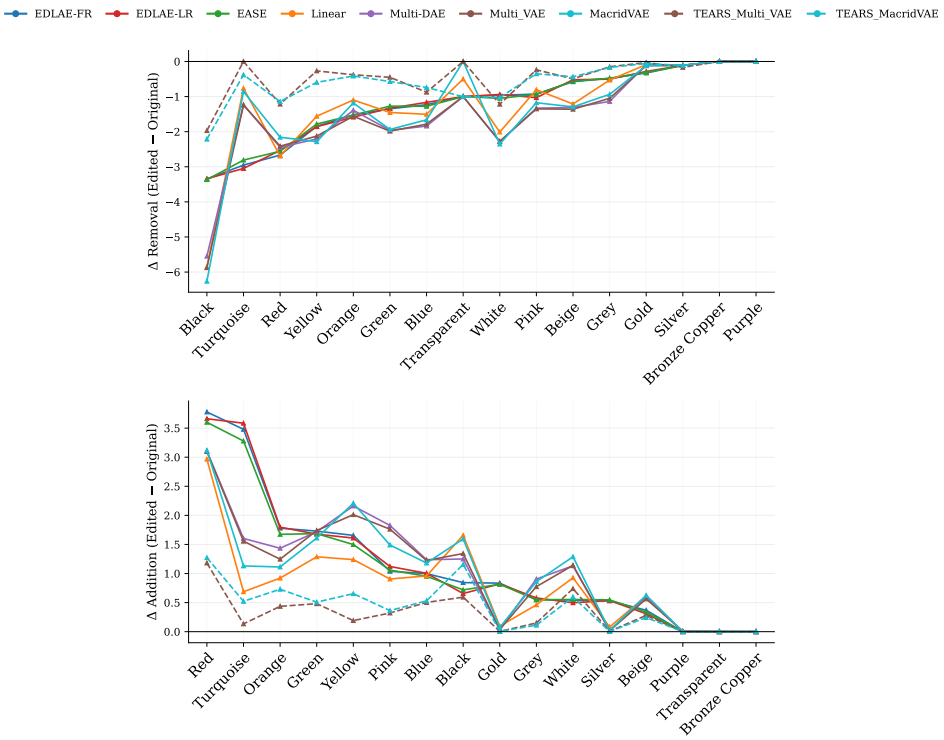


Figure 5: Per-color edit effects on the H&M dataset at $k$=20. Top: $\Delta$ Removal (Edited $-$ Original); bottom: $\Delta$ Addition (Edited $-$ Original). Colors are sorted by frequency. TEARS variants (dashed) respond more weakly than the standard models, especially on rare colors, indicating that gradient-based steering through the text encoder produces smaller but more selective shifts than direct interaction-level edits.




Figure 6: Per-product-type edit effects on the H&M dataset at $k$=20. Top: $\Delta$ Removal; bottom: $\Delta$ Addition. Product types are sorted by frequency. All models respond strongly on frequent items (Socks, Pyjama Set, Boots) and converge to near-zero deltas on rare ones (Slippers, Underwear Body, Swimwear Set), reflecting the sparsity floor below which edits cannot meaningfully influence rankings.

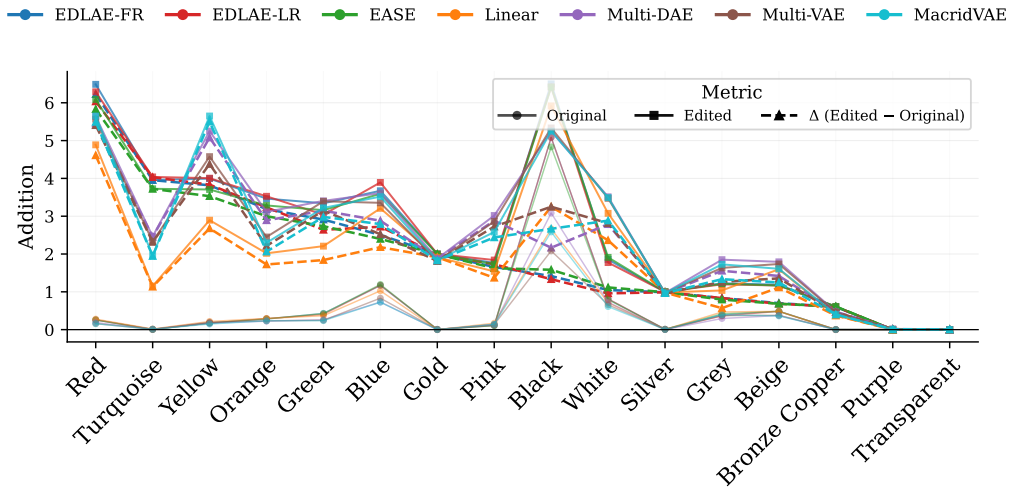


Figure 7: Per-color addition effects on the H&M dataset at $k$=20 for the unseen (multimodal-input) setting. Solid lines show Original and Edited recommendations; dashed lines show $\Delta$ (Edited − Original). Edits propagate cleanly even for colors absent from the user's history, showing that text-conditioned models can introduce previously unseen attributes rather than merely amplify existing ones.

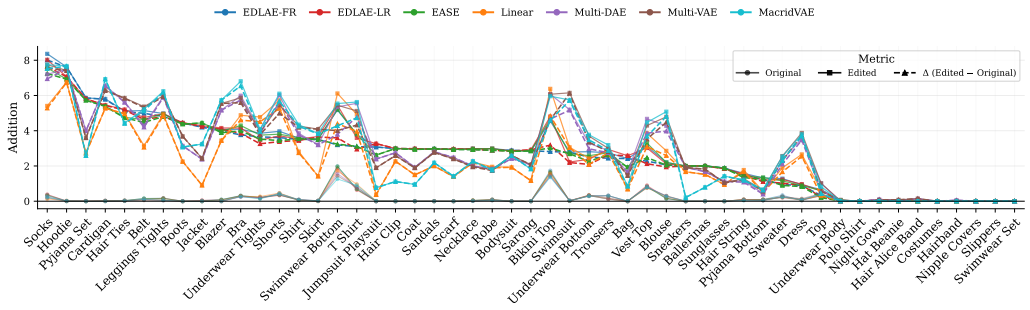


Figure 8: Per-product-type addition effects on the H&M dataset at $k$=20 for the unseen setting. Solid lines: Original and Edited recommendations; dashed: $\Delta$. The largest deltas concentrate on common product types and decay sharply for rare ones, confirming that the unseen-attribute setting inherits the same long-tail behavior observed in the seen case.

### C.2 MSD

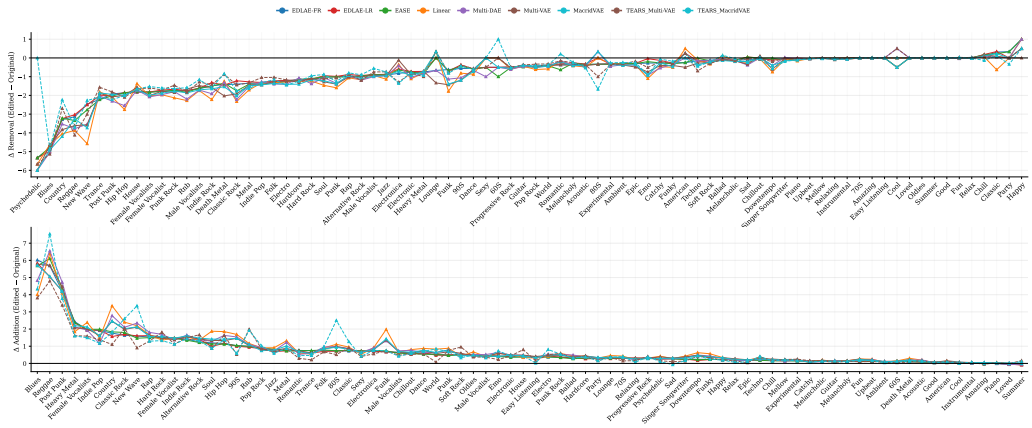


Figure 9: Per-tag edit effects on the MSD (Million Song Dataset) at $k$=20, covering 80 music tags spanning genres, eras and mood descriptors. Top: $\Delta$ Removal; bottom: $\Delta$ Addition (Edited $-$ Original); TEARS variants are dashed. Removal effects are largest on Psychedelic, Blues and Country, while addition effects peak on Blues, Reggae and Post Punk. The asymmetry between top removal and top addition tags shows that the two operations engage different parts of the tag space rather than being symmetric inverses.

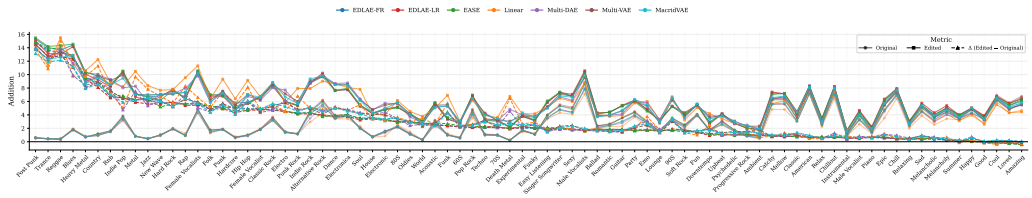


Figure 10: Per-tag addition effects on the MSD at $k$=20 for the unseen (multimodal-input) setting, across the seven collaborative filtering backbones. Line styles distinguish the Original and Edited recommendations and their difference $\Delta$ (Edited $-$ Original). Injecting unseen items that carry a target tag increases the share of that tag in the recommendations for most tags, with effect sizes varying across tags.

## D CCBR with an LLM Text Encoder

In the main experiments, CCBR encodes the natural-language user profile with a BERT-style text encoder (BGE-M3). Here we study an alternative instantiation in which the text encoder is replaced by a similarly-sized LLM encoder, Qwen3.5-0.8B [54], while the rest of the pipeline is left unchanged and the user profile remains fully editable. Unlike the two-stage recipe of Section 3, the LLM encoder is trained in a single stage, where tag supervision, alignment, and reconstruction are optimized jointly rather than split across a tag-pretraining stage followed by a CF-alignment stage.

Table 4 reports the recommendation accuracy obtained with this LLM encoder across all seven collaborative filtering backbones. Relative to the BGE-M3 encoder, the LLM encoder improves NDCG@100 by $+7.7\%$ on average on H&M and $+14.1\%$ on MSD, while remaining on par on ML-20M ($+1.2\%$). The improvements are most pronounced on MSD, the dataset with the largest catalog ($\sim$31K items), where a stronger text encoder helps the compact textual profile recover more of the granular item-level signal. These results indicate that CCBR is agnostic to the choice of text encoder and benefits directly from a more capable one, without any change to the editable user profile or the single-stage training setup described above.

Table 4: Recommendation accuracy on H&M, MSD, and ML-20M with the single-stage LLM text encoder (Qwen3.5-0.8B). For each collaborative filtering backbone we report the backbone and the corresponding CCBR model.

| | H&M | | | MSD | | | ML-20M | | |
|---|---|---|---|---|---|---|---|---|---|
| **Model** | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) |
| EASE | 0.2621 | 0.2440 | 0.1638 | 0.4519 | 0.3792 | 0.4290 | 0.4909 | 0.5162 | 0.4226 |
| CCBR (Ours) | 0.2434 | 0.2252 | 0.1500 | 0.3778 | 0.3091 | 0.3490 | 0.4658 | 0.4854 | 0.4034 |
| EDLAE (FR) | 0.2693 | 0.2460 | 0.1662 | 0.4616 | 0.3849 | 0.4351 | 0.5028 | 0.5267 | 0.4323 |
| CCBR (Ours) | 0.2500 | 0.2303 | 0.1544 | 0.3793 | 0.3103 | 0.3496 | 0.4852 | 0.5092 | 0.4203 |
| EDLAE (LR) | 0.2695 | 0.2447 | 0.1661 | 0.4619 | 0.3858 | 0.4373 | 0.5032 | 0.5275 | 0.4338 |
| CCBR (Ours) | 0.2466 | 0.2243 | 0.1536 | 0.3830 | 0.3141 | 0.3546 | 0.4717 | 0.4951 | 0.4060 |
| Linear-DAE | 0.2372 | 0.2234 | 0.1432 | 0.3830 | 0.3129 | 0.3499 | 0.4982 | 0.5316 | 0.4261 |
| CCBR (Ours) | 0.2155 | 0.2020 | 0.1309 | 0.3525 | 0.2889 | 0.3179 | 0.4883 | 0.5226 | 0.4129 |
| Multi-DAE | 0.2160 | 0.2080 | 0.1286 | 0.3632 | 0.3000 | 0.3246 | 0.5042 | 0.5423 | 0.4310 |
| CCBR (Ours) | 0.2183 | 0.2059 | 0.1316 | 0.3579 | 0.2939 | 0.3154 | 0.5010 | 0.5387 | 0.4273 |
| Multi-VAE | 0.2244 | 0.2135 | 0.1360 | 0.3843 | 0.3144 | 0.3523 | 0.5148 | 0.5468 | 0.4383 |
| CCBR (Ours) | 0.2110 | 0.2003 | 0.1265 | 0.3696 | 0.3051 | 0.3314 | 0.4847 | 0.5191 | 0.4101 |
| MacridVAE | 0.2285 | 0.2206 | 0.1413 | 0.3873 | 0.3151 | 0.3542 | 0.4957 | 0.5218 | 0.4111 |
| CCBR (Ours) | 0.2237 | 0.2181 | 0.1370 | 0.3675 | 0.3014 | 0.3291 | 0.4899 | 0.5243 | 0.4143 |

# E Beyond Linear and Autoencoder Backbones

CCBR is agnostic to the choice of collaborative backbone: the CF model acts as a frozen scorer aligned to the text latent via InfoNCE, so the controllability layer only requires the backbone to expose a user-level representation that the text latent can be aligned to, after which the decoder is reused to score items. To demonstrate that this design extends beyond the linear item-item and autoencoder families considered in the main text, we integrate two additional backbones that learn fundamentally different user-item interactions: NeuMF [4], which learns a nonlinear interaction rather than a simple factorization, and LightGCN [57], a graph-convolutional recommender. Both slot into CCBR with no change to the framework; a new backbone only needs to be trained under strong generalization and re-run through the alignment stage.

Table 5 reports results for these backbones and their CCBR counterparts on the H&M (image) and ML-20M (movie) domains. Beyond preserving recommendation accuracy, we observe that attaching CCBR can in fact improve the final recommendation performance in several cases.

Table 5: CCBR applied to non-linear and graph-based backbones (NeuMF and LightGCN) on the H&M (image) and ML-20M (movie) domains. Both backbones integrate into CCBR without any change to the framework.

| | H&M | | | ML-20M | | |
|---|---|---|---|---|---|---|
| **Model** | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) | NDCG@100 (↑) | R@50 (↑) | R@20 (↑) |
| NeuMF | 0.1842 | 0.1759 | 0.1148 | 0.4485 | 0.4841 | 0.3981 |
| CCBR (Ours) | 0.1958 | 0.1924 | 0.1175 | 0.4634 | 0.4864 | 0.3981 |
| LightGCN | 0.2041 | 0.1954 | 0.1231 | 0.4187 | 0.4468 | 0.3483 |
| CCBR (Ours) | 0.2007 | 0.1974 | 0.1221 | 0.4621 | 0.4989 | 0.3958 |

More broadly, the core mechanism of CCBR is not specific to the CF/MF setting. The controllability layer only requires the backbone to expose a user-level representation that the text latent can be aligned to; once this target exists, the decoder is reused to score items. Two-tower models are an especially natural fit, as the user tower already emits a user embedding that can serve directly as the alignment target, with the item tower playing the role of the decoder. Sequential and transformer-based recommenders can be accommodated similarly, by aligning the text latent to the final user hidden state. The main requirement is simply that a user-side representation exists to align against; architectures that output only an opaque score, with no explicit user vector, would first need such a target to be defined.

## F Broader Impacts

**Positive impacts.** CCBR represents each user through a natural-language profile that the user can read and edit, rather than through an opaque latent vector. This makes it possible for users to see how a recommendation system represents them, to correct inaccurate or outdated inferences, and to steer recommendations toward their current goals, for example by temporarily emphasizing a style, genre, or mood that differs from their long-term history. Such transparency and control can increase user agency and trust, and can make it easier for users and auditors to identify when a system has formed an undesirable picture of a user. Because CCBR can be attached to existing collaborative filtering models, these benefits do not require replacing deployed recommenders.

**Negative impacts and mitigations.** CCBR is an interpretable-by-design recommender. While we see no immediate negative societal impacts, the explicit user profiles it exposes could be misused: they may reveal sensitive user preferences. In addition, LLM-generated profiles can inherit biases from the underlying model. Treating profiles as personal data, auditing them for bias, and letting users view and edit their own profiles can reduce these risks.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 make three claims: (i) CCBR introduces controllability into a given CF backbone through an editable, content-derived text bottleneck; (ii) the resulting models remain competitive with their underlying CF backbones and outperform the controllable baseline TEARS; and (iii) recommendations can be steered through text edits, previously unseen multimodal inputs, and negative preference statements. Claim (i) is described in Section 3. Claim (ii) is supported by Table 1, which also reports the accuracy gap to the backbones, most notably on MSD. Claim (iii) is supported by Sections 4.3, 4.4, and 4.5, with additional results in Appendices C.1 and C.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 4.2 acknowledges that the text-bottlenecked representation yields lower accuracy than the underlying CF backbones, with the largest gap on MSD, the dataset with the largest catalog. The computational cost analysis (Table 2) reports that generating or editing a user summary takes about 10 to 13 seconds per user; this is a one-time cost per profile, but it remains the dominant serving-time expense. Section 4.3 notes that controllability is weaker for highly prevalent or rare attributes and that the evaluated edits are produced by prompting an LLM. The Conclusions note that CCBR is currently evaluated in an offline profile-generation setting and leave interactive refinement to future work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.

- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not include theoretical results; the contribution is methodological and empirical.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 3 describes the full CCBR pipeline (profile generation, tag-supervised text encoding, and InfoNCE alignment with the CF backbone). Section 4.1 specifies the datasets, the strong generalization protocol with the resulting split sizes, and the exact models used at each stage (Qwen3-VL-32B-Instruct, VideoLLaMA 3, MusicFlamingo, Qwen3-30B-A3B-Instruct-2507, BGE-M3, and Qwen3.5-27B). Appendix A reports the stage-wise training setup, learning rates, batch sizes, loss definitions, and dataset-specific settings, and Appendix B.2 provides the prompts used for attribute extraction and summary editing. The released code contains the full preprocessing pipeline and the training configurations.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All datasets used (H&M, MovieLens 20M with the accompanying YouTube trailers mapping, and the Million Song Dataset) are publicly available, and all models used in the pipeline (Qwen3-VL-32B-Instruct, VideoLLaMA 3, MusicFlamingo, Qwen3-30B-A3B-Instruct-2507, BGE-M3, Qwen3.5-27B, Qwen3.5-0.8B, GLM-4-32B-0414, and Gemma-4 12B) have publicly released weights. The CCBR codebase is publicly available at `https://github.com/firatoncel/CCBRNeurIPS2026`, including preprocessing scripts, training configurations for all CF backbones, and the controllability evaluation pipeline; the accompanying README provides instructions for reproducing the reported results.

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

Justification: Section 4.1 reports the datasets, the split sizes under strong generalization, and the models used for profile generation and editing. The training objectives are defined in Section 3 and Appendix A.2. Appendix A.1 reports the hyperparameter search (30 to 50 configurations per backbone and dataset), learning rates, batch sizes, dropout rates, and loss weighting for both training stages, and Appendix A.3 reports the dataset-specific latent dimensionalities and the H&M color augmentation. Remaining details, including the optimizer, the InfoNCE temperature, the loss weights, and the selected configuration for each backbone and dataset, are provided in the configuration files of the released code.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We do not report error bars for the recommendation accuracy results in Tables 1, 4, and 5, due to the computational cost of repeating the full pipeline (profile generation, two-stage text-encoder training, and alignment) with multiple seeds for every backbone and dataset. The controllability results in Sections 4.3, 4.4, and 4.5 are averages over many per-user, per-attribute interventions, and the same qualitative trends hold across all CF backbones. Latency measurements in Table 2 are reported as mean $\pm$ standard deviation over 100 simulated users.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The computational cost analysis in Section 4.5 and Table 2 report the hardware and execution time of each stage of the pipeline. Offline item verbalization with VideoLLaMA 3 takes $4672.9 \pm 2615.4$ ms per item on a single H100-80GB GPU (about 14.7

GPU-hours for the ML-20M catalog), and serving-time summary generation, summary encoding, and top-20 scoring for a new user are measured on a single A100-80GB GPU.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly available datasets in line with their stated terms of use, does not involve new data collection from human subjects, and does not collect or release personally identifying information. We have reviewed the NeurIPS Code of Ethics and the research conforms to its requirements.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix F discusses both.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release new large pretrained models, image generators, or scraped datasets. It uses publicly available datasets and publicly released foundation models in their intended modes of use.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets, foundation models, and CF baselines (EASE, EDLAE, Linear-DAE, Multi-DAE, Multi-VAE, MacridVAE, NeuMF, LightGCN) are cited at their first use. Datasets: H&M Personalized Fashion Recommendations (Kaggle competition data terms), MovieLens 20M (GroupLens usage license, non-commercial research use), the MovieLens 20M YouTube Trailers mapping (CC BY 4.0), and the Million Song Dataset (research terms of use). Models: Qwen3-VL-32B-Instruct, Qwen3-30B-A3B-Instruct-2507, Qwen3.5-27B, Qwen3.5-0.8B, VideoLLaMA 3, and Gemma-4 12B (Apache 2.0); BGE-M3 and GLM-4-32B-0414 (MIT); MusicFlamingo (NVIDIA OneWay Noncommercial License). All assets are used for non-commercial academic research in accordance with these terms.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: The only new asset is the CCBR codebase, released at `https://github.com/firatoncel/CCBRNeurIPS2026` together with configuration files for the reported experiments and a README describing data preparation, training, and the controllability evaluation pipeline. No new datasets are released.

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects, so no IRB approval is required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs are a core component of the CCBR pipeline and are described in Sections 3 and 4.1. Modality-specific multimodal foundation models (Qwen3-VL-32B-Instruct for images, VideoLLaMA 3 for videos, and MusicFlamingo for audio) verbalize items, and an instruction-tuned text-only LLM (Qwen3-30B-A3B-Instruct-2507) aggregates item descriptions into user profiles. Qwen3.5-27B is used for attribute extraction and counterfactual summary editing in the controllability experiments, GLM-4-32B-0414 and Gemma-4 12B serve as independent judges of edit locality (Appendix B.1), and Qwen3.5-0.8B is evaluated as an alternative text encoder (Appendix D). Beyond this core methodological role, LLMs were also used for editing (e.g., grammar, spelling, word choice) and for facilitating and running experiments; all LLM-generated outputs and edits were reviewed and verified by the authors.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.