Modeling Multiple Tasks in Recommendation Systems

DO Dinh Hieu

SINGAPORE MANAGEMENT UNIVERSITY

2025

Modeling Multiple Tasks in Recommendation Systems

by
DO Dinh Hieu

Submitted to School of Computing and Information Systems in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science

Dissertation Committee:

Hady W. LAUW (Supervisor/Chair)
Associate Professor of Computer Science
Singapore Management University

NGO Chong Wah
Professor of Computer Science
Singapore Management University

FANG Yuan
Associate Professor of Computer Science
Singapore Management University

Jisun AN
Assistant Professor
Indiana University Bloomington

Singapore Management University
2025

Abstract

Modeling Multiple Tasks in Recommendation Systems

by Do Dinh Hieu

Traditional research in recommendation systems has largely centered on the static offline supervised learning setting. In this paradigm, all available user-item interaction data is collected and partitioned into fixed training, validation, and test sets. Models are developed and evaluated in this controlled environment, where the underlying data distribution is assumed to remain unchanged. This approach offers clear advantages: it simplifies experimentation, enables reproducible benchmarking, and allows for straightforward comparisons between algorithms.

However, this static offline setting does not reflect the realities faced by modern recommendation systems. In real-world applications, data is dynamic and ever-evolving, where new users and items are constantly emerged, user preferences shift over time, and interactions arrive as a continuous stream. Moreover, data is often fragmented across multiple platforms or domains, each with its own characteristics and challenges. These factors introduce complexities such as the need for continual adaptation and transferring knowledge across domains.

Recognizing these limitations, this dissertation aims to bridge the gap by formulating recommendation problems that more accurately reflect real-world scenarios. The primary goal is to design dynamic frameworks that efficiently learn from new data streams, capably handle evolving user behaviors and item catalogs, and effectively address data fragmentation across platforms by enabling knowledge transfer between various tasks and domains.

To this end, this dissertation makes three principal contributions in the context of recommendation systems. First, it introduces a novel dual-target cross-domain

recommendation framework. This framework significantly enhances recommendation quality by effectively leveraging knowledge from similar, yet distinct, datasets across multiple domains. Second, the dissertation proposes a continual learning framework for recommendation systems, enabling models to seamlessly adapt to the sequential arrival of new users, items, and interactions, crucially retaining valuable knowledge from previous tasks. This aims to provide effective solutions for the ever-expanding nature of real-world recommender systems. Third, it presents a session-aware recommendation framework that integrates both short-term and long-term user preferences. By strategically combining the strengths of experts in short-term and long-term preferences, this approach yields more accurate and personalized recommendations that dynamically align with users' evolving interests.

Through these contributions, this dissertation seeks to enhance the effectiveness of recommendation systems, ensuring their ability to meet the dynamic and evolving demands of real-world applications.

Publications During Enrollment

This dissertation includes extended or modified versions of conference and journal papers that have been published or are currently under review.

Chapter 3

- Jaime Hieu Do and Hady W. Lauw. 2025. Dual-Target Disjointed Cross-Domain Recommendation Mediated via Latent User Preferences. In *ACM SIGKDD Explorations Newsletter, 27(1), 52–61.*

Chapter 4

- Jaime Hieu Do and Hady W. Lauw. 2023. Continual Collaborative Filtering Through Gradient Alignment. In *Proceedings of the 17th ACM Conference on Recommender Systems (RecSys '23), 1133–1138.*

Chapter 5

- Jaime Hieu Do, Trung-Hoang Le, and Hady W. Lauw. 2025. Compositions of Variant Experts for Integrating Short-Term and Long-Term Preferences. Under Revision at *ACM Transactions on Recommender Systems.*

Contents

List of Figures

List of Tables

Acknowledgments

Obtaining a PhD with no prior research experience was as challenging as it was rewarding. This is an accomplishment that I owe to the immense guidance, support, and encouragement of many incredible people.

I extend my foremost and deepest gratitude to my advisor, Prof. Hady W. Lauw, for entrusting me with the opportunity to pursue this PhD, for teaching me the principles of research, and for providing a balance between granting me the freedom to explore and offering invaluable guidance when I needed it. I am also very grateful to Dr. Duc-Trong Le, whose early encouragement was instrumental in my decision to begin my doctoral studies. My sincere thanks also go to my dissertation committee members: Prof. Ngo Chong Wah, Prof. Fang Yuan, and Prof. Jisun An. Your insightful feedback and discussions significantly improved the quality of this dissertation.

I want to express special acknowledgements to Dr. Le Trung-Hoang. I was privileged to have him as both a collaborator and a secondary mentor. His technical support and critical second opinions were crucial, helping me improve significantly as a researcher through the years.

I also want to thank all members of the PreferredAI research group for the valuable time we spent together in academic discussions, mutual encouragement, and outdoor activities. My PhD journey would not have been complete without sharing these memorable moments with you all. Thank you to Dr. Le Duy Dung, Dr. Aghiles Salah, Dr. Maksim Tkachenko, Dr. Truong Quoc Tuan, Dr. Le Trung-Hoang, Dr. Zhang Ce, Dr. Chia Chong Cher, Dr. Lee Ween Jiann, Dr. Tran Nhu-Thuat, Dr. Konstantinos Theocharidis, Binh Tran, Lim Jia Peng, Darryl Ong Rong Sheng, Hongtuo Nie, Ezekiel Ong Young, Le Thi Phuong, Nguyen Minh Quang, Dilan Dinushka, Ngo Huu Manh Khanh, Dong Viet Hoang.

Beyond academia, I was fortunate to have friends who supported my personal growth. With Trung-Hoang and Quang, I learned to find balance through physical activity. Together we conquered the 36-km coast-to-coast trail twice, and their encouragement pushed me to complete two half-marathons and cycle around the island of Singapore twice, activities which provided valuable new perspectives. This spirit of adventure continued, and my sincere thanks go to Dr. Le Trung-Hoang, Dr. Quang Pham, Dr. Le Duy Dung, Dr. Tai Nguyen, Dong Viet Hoang, Pham Hoang Quoc Viet, Nguyen Nhat Minh, and Dao Quang Huy for all the hiking routes, conversations, and laughs we shared.

Finally, and most importantly, I offer my heartfelt thanks to my family. To my parents, thank you for your unwavering belief and support in every choice I made. To my wife, your unconditional love, patience, and encouragement were my constant source of support throughout this journey. This accomplishment is as much yours as it is mine.

Chapter 1

Introduction

In the past decade, artificial intelligence (AI) has made remarkable strides, achieving or even surpassing human-level performance in various tasks, especially those involving vision and language. The advent of deep learning techniques has propelled AI systems to unprecedented levels of accuracy and efficiency. However, despite these advancements, there's one crucial area that remains a significant challenge: recommendation systems.

Recommendation systems play a pivotal role in modern information ecosystems, facilitating personalized content delivery and enhancing user experience across diverse platforms. From e-commerce giants to social media platforms, recommendation algorithms serve as the backbone of content curation, product suggestions, and user engagement strategies. Despite their widespread adoption and undeniable impact, recommendation systems pose unique challenges that necessitate innovative approaches. The prevailing paradigm in recommendation system research revolves around offline learning settings, wherein all available data is partitioned into distinct subsets for training, validation, and testing. The models try to learn the underlying distribution of the training set, the best parameters are selected based on performance on the validation set and are expected to generalize to new unseen data (i.e., test set). While this approach simplifies model development and evaluation, it often fails to capture the dynamic nature of real-world recommendation scenarios. In practice, recommendation systems operate in online or streaming

environments, where users, items, and their interactions arrive sequentially, posing significant challenges in model adaptation, scalability, and real-time decision-making. The core challenge in ever-growing recommendation systems is the constantly evolving nature of the data. User preferences are not static; they shift over time, influenced by the trends, seasons, and natural changing behaviors. A user who was previously interested in action movies might suddenly develop a passion for documentaries. Similarly, the catalog of items is in perpetual flux, with new products being added and old ones being removed. This dynamic nature of data necessitates a shift from traditional offline learning paradigms to more adaptive approaches that can effectively handle the continuous influx of new information.

Furthermore, cold-start, sparsity, and long-tail problems represent fundamental challenges that afflict recommendation systems, limiting their effectiveness in capturing user preferences and providing accurate recommendations. Cold-start scenarios arise when new users or items lack sufficient historical data, making personalized recommendations difficult. Similarly, sparsity issues arise when the available data is scarce or incomplete, hindering the system's ability to model user-item interactions effectively. Compounded by the power law distribution, wherein the majority of users and items have minimal interactions (i.e., long-tail users/items), the model becomes skewed toward popular entities, complicating the accurate learning of all user preferences. These challenges necessitate the development of more sophisticated recommendation framework that can effectively handle dynamic data streams, adapt to evolving user preferences, and leverage knowledge transfer across domains and tasks.

The foci of this dissertation are to (i) formulate more dynamic settings for recommendation systems based on the fragmentation and growing nature, being able to handle new data streams coming into the systems and enable information transfer between domains; and (ii) leverage multiple *tasks* and *models* in different ways to mutually improve performances, avoiding interferences, and enhancing the effectiveness of recommendation systems.

Tasks in Machine Learning

We provide an introduction to the concept of tasks in machine learning and discusses different learning paradigms for handling multiple tasks.

In broad terms, a machine learning *problem* denotes the goal that one machine learning model aims to address. Common problems include image classification, text generation, and top-K recommendation. When provided with specific input and output (i.e., data), a machine learning problem becomes a learning *task*. Similar tasks from distinct underlying data distributions are referred to as *domains*. We can consider *domain* as a more specific term than *task* (i.e., two different domains can be seen as two tasks, but not vice versa).

Supervised Learning. The typical learning *task* we refer to is supervised learning, which is the most prevalent learning paradigm. Given the dataset $\mathcal{D} = \{(X, y)\}$ with an unknown underlying relation $y = g(X)$, the objective is to find an approximated function f parameterized by θ that maps $f_\theta(X) \to y$. The quality of this mapping function is measured using a designated loss function ℓ between the model prediction and ground truth output $\ell(f_\theta(X), y)$. By optimizing the loss function, we aim to derive a "good" model that can generalize well to unseen data. While supervised learning proves effective in many scenarios, it simplifies the *problem* into a stationary setting, which may fail to capture dynamics present in numerous real-world situations.

Online Learning. Unlike supervised learning, online learning deals with data streams where information arrives sequentially [52, 121]. The model must learn from each data point efficiently, as opposed to iterating over the entire dataset multiple times like in supervised learning.

Multi-Domain Learning. This approach involves training a model on the *same task* across *multiple related domains* simultaneously [27, 60]. The goal is to leverage shared parameters between domains to improve generalization and avoid overfitting. Multi-domain learning assumes all domain data is available for training beforehand (offline setting).

Multi-Task Learning. This approach tackles multiple related tasks simultaneously, leveraging shared parameters between them [143, 155]. The goal is to improve generalization and avoid

overfitting by exploiting knowledge gained from solving multiple tasks. Similar to multi-domain, multi-task learning assumes all task data is available beforehand.

Continual Learning, also known as lifelong or incremental learning, addresses the problem of learning from an *infinite stream of tasks* [24, 36, 107, 108]. The goal is to leverage accumulated knowledge from all past tasks to enhance future learning, while continually refining previously learned tasks with each new task coming.

Transfer Learning. In transfer learning, knowledge acquired from a source task (often rich in data and containing generalizable features) is transferred to a target task [155, 165]. Fine-tuning is a popular technique where a pre-trained model on a large source task (denoted by θ_S^*) is used to initialize learning for the target task.

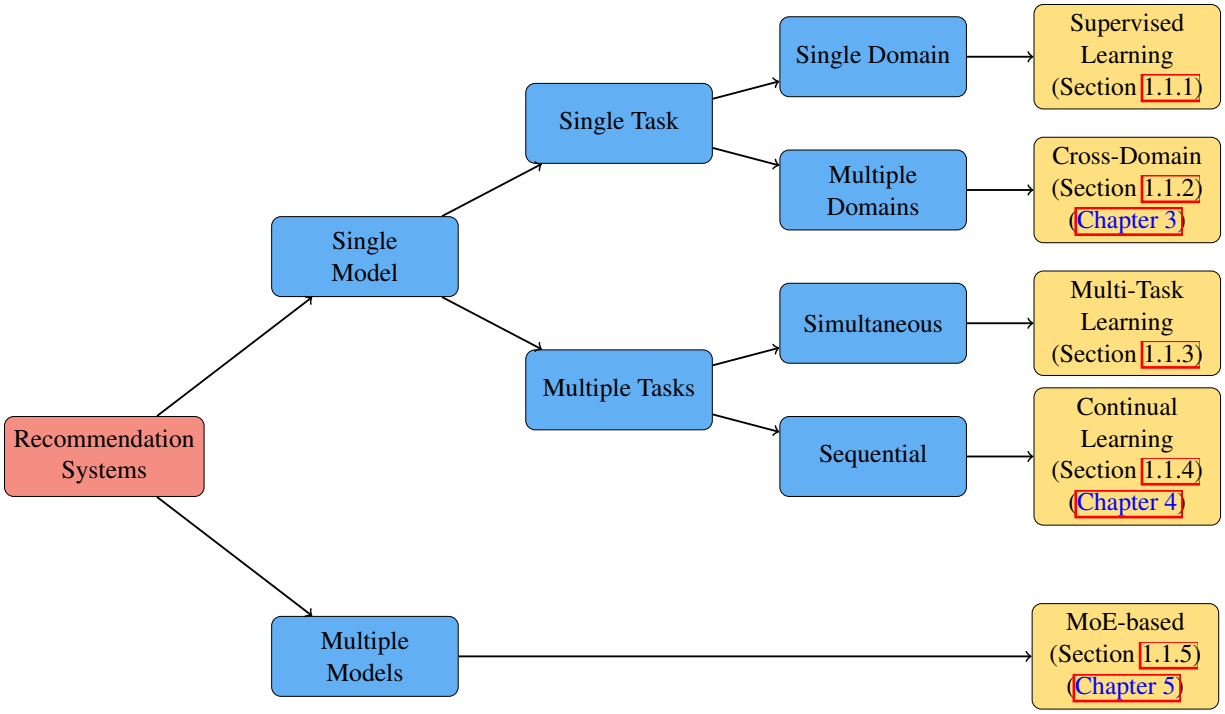
Domain Adaptation [5, 151]. This is a specialized form of transfer learning where source and target tasks are similar but drawn from different domains. The target domain data might be unlabeled. The objective is to adapt the source-trained model for good performance on this unlabeled target data. This relaxes the assumption of classical machine learning, where training and testing data share the same underlying distribution.

Meta Learning [31, 32, 102]. The concept of meta-learning has evolved over time. Traditionally, it referred to a model's ability to improve its learning efficiency across an increasing number of tasks. Currently, the focus is on "few-shot learning", where a model is trained on a set of tasks and then, during testing, presented with a few examples of new tasks, the model needs to quickly learn and perform well on these unseen tasks. While this concept overlaps with continual learning, the key difference is that meta-learning trains on the tasks with offline setting.

1.1 Tasks in Recommender Systems

Two-sidedness is a key characteristic that distinguishes recommendation systems from other machine learning problems. In recommender systems, there are two types of entities: users and items. The objective is to match users with items they are likely to prefer. The most common

Figure 1.1: Diagram of different recommendation system learning paradigms.



form of data in recommendations is the interaction matrix between users and items. These interactions can be explicit (e.g., ratings and reviews) or implicit (e.g., views and purchases).

In this section, we categorize recommendation systems into different learning paradigms based on the models and tasks involved, as illustrated in Figure 1.1. We will discuss each paradigm in detail, including supervised recommendation systems, cross-domain recommendation, multi-task recommendation, continual learning for recommendation, and mixture of experts-based recommendation.

1.1.1 One Model, One Task: Supervised Recommendation Systems

Given a set of users $\mathbf{U}$, a set of items $\mathbf{I}$, and their interactions represented by a typically sparse matrix $\mathbf{R} \in \mathbb{R}^{|\mathbf{U}| \times |\mathbf{I}|}$, the general objective is to reconstruct the interaction matrix: $f_\theta(\mathbf{U}, \mathbf{I}) \rightarrow \mathbf{R}$. Here, we aim to learn a model f parameterized by θ that predicts the score for each user i and item j as $r_{ij} := f_\theta(U_i, I_j)$.

Model training is performed by minimizing the loss between the predicted output $f_\theta(\mathbf{U}, \mathbf{I})$ and the ground truth $\mathbf{R}$, formulated as:

$$\arg\min_{\theta} \ell\Big(f_\theta(\mathbf{U}, \mathbf{I}), \mathbf{R}\Big)$$

where ℓ represents the loss function that measures the discrepancy between the predicted and actual interaction scores.

In supervised recommendation task, there is a single model f, a single loss function ℓ, and a single dataset $\mathbf{R}$. However, real-world scenarios involve multiple tasks addressed simultaneously or sequentially. This leads to the emergence of various learning paradigms in recommendation systems, including multi-task learning, cross-domain recommendation, and continual learning.

1.1.2 One Model, Multiple Domains: Cross-Domain Recommendation

Cross-domain recommendation (CDR) entails leveraging shared information across two or more *domains* to enhance recommendation performance. The conventional approach is *single-target* CDR, where knowledge from domains rich in information (such as explicit/implicit feedback and user demographics) is utilized to enhance recommendations in a sparser target domain. Specifically, let $\mathbf{U}^s, \mathbf{I}^s$ and $\mathbf{U}^t, \mathbf{I}^t$ denote the user and item sets in the source and target domains, respectively, with their interactions represented by rating matrices $\mathbf{R}^s$ and $\mathbf{R}^t$. Here, the focus lies solely on improving recommendation performance in the target domain $\mathbf{R}^t$, while the source domains $\mathbf{R}^s$ serve as supplementary information aiding the learning process. Single-target CDR aims to learn both domains with a single model f_θ that can leverage information from both domains jointly. The objective is formulated as:

$$\arg\min_{\theta} \ell\Big(f_\theta(\mathbf{U}^t, \mathbf{I}^t \mid \mathbf{U}^s, \mathbf{I}^s, \mathbf{R}^s), \mathbf{R}^t\Big)$$

Previous research has also explored *dual-target* and *multi-target* CDR scenarios. In these

setups, different domains interact reciprocally with the aim of enhancing recommendation accuracy across all involved domains. The objective of dual-target CDR can be represented in two forms, (i) losses combination:

$$\arg\min_{\theta} \Big[\alpha\ell_1\Big(f_\theta(\mathbf{U}^s, \mathbf{I}^s), \mathbf{R}^s\Big) + (1-\alpha)\ell_2\Big(f_\theta(\mathbf{U}^t, \mathbf{I}^t), \mathbf{R}^t\Big)\Big]$$

and (ii) joint modeling:

$$\arg\min_{\theta} \ell\Big(f_\theta(\mathbf{U}^s, \mathbf{I}^s, \mathbf{U}^t, \mathbf{I}^t), \mathbf{R}^s, \mathbf{R}^t\Big)$$

Cross-domain recommendation bears resemblance to multi-task learning, where multiple task objectives are jointly optimized. However, dual/multi-target CDR represents a more specialized context, typically involving domains with consistent data formats. In contrast, multi-task learning encompasses various *tasks*, such as rating prediction, ranking prediction, and explainability, which can vary independently. Consequently, techniques from multi-task learning are also applied to address cross-domain recommendation problems.

In Chapter 3, we introduce a novel dual-target cross-domain recommendation setting, named NO3, which stands for *no* overlap of users, *no* overlapping items, and *no* side information. In this setting, we aim to learn two similar tasks from two distinct domains without any user or item overlap, and without relying on side information.

1.1.3 One Model, Multiple Tasks: Multi-Task Recommendation

Multi-Task Recommender Systems (MTRS) in recommendation systems aims to learn multiple related tasks in a unified model to mutually improve performances based on their shared information. Compared to cross-domain recommendation, MTRS formulation is more general and not limited to multiple domains of the same tasks. MTRS may involve multiple recommendation tasks, including but not limited to: rating prediction, ranking prediction, explainability, sequential recommendation, user profile prediction, and so on. The most common approach is to learn

multiple tasks in parallel, where the model learns all tasks simultaneously. The objective can be formulated as weighted losses combination:

$$\arg\min_{\theta} \sum_{i=1}^{n} \alpha_i \ell_i \Big(f_\theta(\mathbf{U}^i, \mathbf{I}^i), \mathbf{Y}^i \Big)$$

where ℓ_i is the loss function for task i, α_i is the weight for task i, f_θ is the unified model for all tasks, and $\mathbf{U}^i, \mathbf{I}^i, \mathbf{Y}^i$ represent the set of users, items, and general target output for task i. Here, we use $\mathbf{Y}^i$ to denote the target output, which can be ratings, ranking scores, or any other task-specific outputs. The model learns to optimize all tasks simultaneously, sharing parameters across tasks to improve generalization and avoid overfitting.

1.1.4 One Model, Sequential Tasks: Continual Learning for Recommendation

While various learning approaches have been extensively investigated for recommender systems, continual learning remains relatively overlooked. Continual learning addresses the problem of learning from an *infinite stream of tasks*. The goal is to leverage accumulated knowledge from all past tasks to enhance future learning, while continually refining previously learned tasks with each new task coming. In continual learning, we aim to use a single model f_θ to learn new tasks as they arrive, while simultaneously retaining knowledge from previously learned tasks within the same model.

When a new task t is introduced, we leverage the accumulated knowledge from all preceding tasks, distilled in θ^*_{t-1}, to boost the learning of the new task using dataset $\mathbf{R}^t$, deriving of new optimal parameters θt^*. This process can be formulated as minimizing the expected loss function:

$$\theta^*_t = \arg\min_{\theta_t} \mathbb{E}_{\mathbf{R}^t} \left[\ell \Big(f_{\theta_t}(\mathbf{U}^t, \mathbf{I}^t \mid \theta^*_{t-1}), \mathbf{R}^t \Big) \right] \tag{1.1}$$

Throughout this learning process, the model is required to retain useful knowledge from θ^*_{t-1}.

This paradigm seems particularly apt for recommendation systems, where the users, items, and their interactions continuously expand. Continual learning offers a structured framework that aligns with the evolving dynamics of a recommendation platform. However, the adaptation of continual learning to recommendation systems has been limited by the absence of a robust formulation for managing the ongoing influx of tasks.

In Chapter 4, we introduce a standardized framework for Continual Collaborative Filtering, which provides a structured approach to training, testing, and evaluating incremental collaborative filtering models. This framework aims to facilitate the development of continual learning models that can effectively adapt to the dynamic nature of recommendation systems.

1.1.5 Multiple Models, One/Multiple Task(s): Mixture of Experts based Recommendation

In some scenarios, it is beneficial to use multiple models to handle a single task. In the past, this approach was usually referred to as *ensemble* methods, where multiple models are trained independently and their outputs are combined to produce the final recommendation. In recent years, *Mixture of Experts* (MoE) has been proposed as a more sophisticated approach to ensemble learning. MoE involves training multiple specialized sub-networks, known as *experts* and a *gating network* that dynamically selects the most relevant experts for each input. This architecture allows individual experts to specialize, for instance, in a multi-task setting, each expert can handle a distinct task, while in a single-task setting, they can focus on different facets of the user preferences. This specialization allows the model to significantly increase its capacity and capture a wider range of patterns, while maintaining efficiency, as only a targeted subset of experts is activated for any given input.

The general formulation of MoE can be represented as:

$$\mathbf{y} = \sum_{i=1}^{n} g(\mathbf{x})_i \cdot f_i(\mathbf{x})$$

where $\mathbf{x}$ is the input, $g(\mathbf{x})$ is the gating function that determines the relevance of all experts for input $\mathbf{x}$, $g(\mathbf{x})_i$ is the $i-$th component which indicates the importance of expert i, and $f_i(\mathbf{x})$ is the output of expert i. The gating function can be learned to assign higher weights to experts that are more relevant to the input. Subsequently, the output $\mathbf{y}$ can be used to passed to the next layer or to produce the final recommendation.

In recommendation systems, MoE-based approaches have been employed mostly to handle multi-task learning scenarios, where different experts are trained to be specified for their target tasks. Chapter 5 presents a novel framework that models both short-term and long-term preferences to enhance recommendation performance.

1.2 Main Contributions

The objective of this dissertation is to advance the modeling of multiple domains and tasks within recommendation systems, extending beyond the confines of traditional offline supervised learning. It aims to propose model-agnostic solutions to enhance learning within these paradigms.

First, we investigate dual-target cross-domain recommendation, where two related domain datasets originate from distinct platforms. We introduce the NO3 setting, characterized by *no* user overlap, *no* item overlap, and *no* side information. We begin by presenting a multi-task learning framework in which the model learns both domains simultaneously. Subsequently, we propose a methodology to "tie" similar users, fostering closer representations, and leverage mediated latent user preferences to improve recommendation quality in both domains.

Next, we address the more intricate challenge of continual learning in recommendation systems. Here, the recommender must adapt to the arrival of new users, items, and interactions over time. We introduce a standardized framework for the training, testing, and evaluation of incremental collaborative filtering. Furthermore, we propose a model-agnostic approach that can be seamlessly integrated with any traditional recommendation model. This approach aims to maximize transferability and mitigate interference among multiple tasks.

We also explore the effects of short-term and long-term user preferences in sequential recommendation and propose a novel framework that leverages the strengths of both preferences to enhance recommendation performance. By integrating these two aspects, we aim to provide more accurate and personalized recommendations that align with users' evolving interests.

The remainder of this dissertation is organized as follows: Chapter 2 provides a broad overview of related work across various settings. Our work on dual-target cross-domain recommendation is detailed in Chapter 3. In Chapter 4, we present the continual collaborative filtering framework. Chapter 5 discusses our work on session-aware recommendation, focusing on the integration of short-term and long-term user preferences. Finally, Chapter 6 outlines future directions for modeling multiple tasks in recommendation systems.

Chapter 2

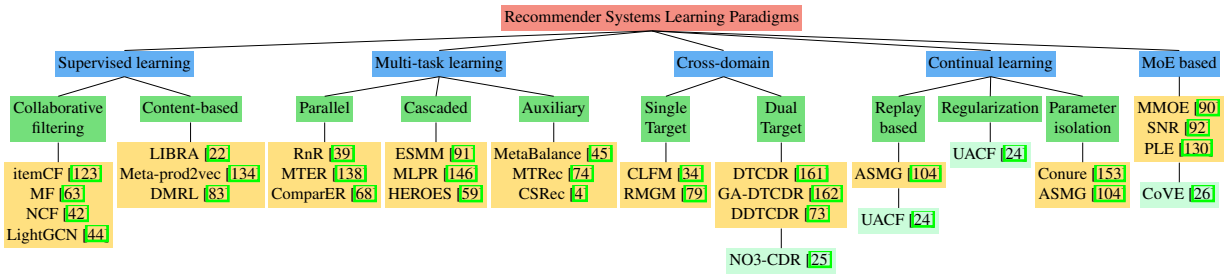
Literature Survey

As discussed in Chapter 1, one recommendation *problem* involves one or multiple *models* handling a single or multiple *tasks*. Prior work has delved into both single and multiple tasks. Traditional recommender systems typically address individual tasks, while rising attention shifts towards formulating multiple tasks in recommendations. Of particular focus in this thesis are multi-task learning, cross-domain, continual learning, and Mixture of Experts (MoE) based recommendation.

2.1 Traditional Recommender Systems

In recent years, the landscape of recommender systems has undergone a significant transformation. This transformation has been fueled by the integration of cutting-edge methodologies from

Figure 2.1: Prior work with different learning paradigms in recommendation systems.



diverse fields such as deep neural networks, Transformer, and Large Language Models (LLM). While LLM-based approaches have garnered attention for their zero-shot capability, classical methods remain essential in practical applications and continue to be a focal point of ongoing research endeavors. In this section, we discuss three main categories of recommendation systems: collaborative filtering, content-based recommendation, and sequential recommendation.

2.1.1 Collaborative Filtering

Collaborative filtering is a foundational technique in recommendation systems, traditionally oriented towards the batch learning paradigm. This approach leverages the social connections between users and items, utilizing the known preferences of a group of users to generate recommendations for others. Collaborative filtering methods are generally categorized into two types: memory-based methods and model-based methods.

Memory-based methods involve identifying similar users or items to make recommendations. In a typical memory-based algorithm, given a candidate user, we try to find k most similar items based on all items that the user has rated [123]. Advances in memory-based methods focus on finding better similarity metrics, such as Pearson correlation [97, 117] or vector cosine similarity [122, 123]. This approach is *non-parametric*, straightforward, easy to implement, and flexible for adding new data; however, it relies heavily on a dense rating matrix and performs poorly with sparse data or cold-start users/items.

Model-based methods focus on designing models that can capture user preferences and item characteristics in dense vector representations. Examples of model-based recommendation systems range from basic linear Matrix Factorization (MF [55, 63, 100, 114, 115]) to more complex neural network-based models such as Neural Collaborative Filtering (NCF [42]), variational autoencoders [80, 133], and graph-based models [13, 41, 44, 57]. These models learn high-quality user/item representations used to calculate rating scores or item rankings.

Collaborative filtering, while a powerful technique in recommendation systems, does come

with its own set of weaknesses. One major limitation is the "cold start" problem, where new items or users lack sufficient data for accurate recommendations. This can lead to poor recommendations for new users or items until they have generated enough interaction data.

2.1.2 Content-Based Recommendation

The general concept of content-based recommenders is to suggest items based on their similarities with user profiles or items that users previously interacted with. This method analyzes the key attributes of a user's favorite items by analyzing their descriptions. These preferences are then used to build the user's profile. New candidate items that closely match this profile will be recommended.

For example, [85] recommends webpages based on tag similarities, while LIBRA [22] integrates tags with item descriptions for improved recommendations. Meta-prod2vec [134] incorporates metadata as additional information to better learn item embeddings through user-item interactions.

Content-based recommendation systems are effective in cold-start scenarios for both users and items, where users provide their initial interests or demographic information, and items' metadata and descriptions are used to analyze item properties. However, because the feature representations of items are partially designed manually, this method requires significant domain knowledge. Additionally, content-based recommendation systems can only suggest items based on existing user interests, limiting their ability to broaden those interests.

2.1.3 Sequential Recommendation

Sequential recommendation aims to recommend items to the user by modeling their past behavior sequences and characterizing their dynamic interests. We assume that modeling sequence data more accurately captures short-term preferences. Session-based recommendation approaches often ignore the personalized context (e.g., user identity is omitted). Previous works used Markov

Chains (MC) to capture local sequential patterns [46, 116, 137]. To address sequential data, [50] uses recurrent neural networks (RNN) such as Gated Recurrent Unit (GRU), [150] applies the attention mechanism, [61, 105, 131] use the Transformer architecture [135]. [71] models contemporaneous basket sequences for next-item prediction using twin networks. [142] applies multiple contrast signals for sequential recommendation. [139, 140] model sequences using hypergraphs, [149] further model multibehaviours. [7, 112] focus on the common repeat consumption behaviors. Others use convolutional neural networks (CNN) for faster parallel computation [129, 152] Another line of work is session-aware recommendation, which further leverages cross-session information to model longer item associations from one session to previous sessions. [70] models user historical baskets. [109] carries the hidden state of interactions from previous sessions into the current session. Another line of work is the next-basket recommendation, recommending a set of items given historical baskets [76].

2.1.4 Other Recommender System Formulations

In complex scenarios, hybrid approaches [8, 9, 21, 37, 145] integrate collaborative filtering and content-based models to harness the strengths of each method and mitigate their weaknesses. Collaborative filtering struggles with cold-start users and items due to their limited interactions, while content-based models overlook user preferences in recommendations. Hybrid approaches offer a balanced solution, addressing the limitations of both individual methods.

Prior studies have also explored the use of side information to enhance recommendation performance. For instance, social networks [89], topic modeling [144], reviews [132, 159], questions [69], and images [11] have been utilized to improve recommendations. These auxiliary information sources can help alleviate the cold-start problem by providing additional context about users and items.

The utilization of a large language model (LLM) in recommendation systems has garnered significant attention in recent research. Language model architectures have been extensively

adopted in recommendation systems due to their inherent similarities in sequential characteristics, particularly in sequential recommendation tasks, as evidenced by models such as SASRec [61] and BERT4Rec [128]. The incorporation of large language models (LLMs) into recommendation systems is a growing trend. This integration is primarily driven by the LLMs' capacity for textual understanding, which enables text-based recommendations and the incorporation of textual modalities into recommendation systems [35, 113]. However, their role in ID-based recommendation remains limited, often functioning as a supplementary component rather than a core model [14]. There is no clear evidence showing that LLM can easily outperform traditional models in ID-based recommendation [53, 77, 154]. On the other hand, the high computational cost and input token limitations for datasets with large number of items outweigh their performance benefits, making their integration a challenging trade-off.

Recent years have witnessed several modern recommender systems formulations, such as explainable or reinforcement-based recommendations, though not in the scope of this thesis. We refer to [2, 15, 30, 120, 158] for comprehensive surveys on these formulations.

2.2 Multi-Task Recommendation

Previous research in this domain can be classified into three main types: (i) parallel [12, 39, 138], (ii) cascaded [91, 147], and (iii) auxiliary [45, 74].

Parallel multi-task recommendation concurrently optimizes two or more recommendation tasks using a weighted sum of their objective functions. For instance, RnR [39] combines ranking and rating prediction tasks for personalized video recommendations, while MTER [138] and ComparER [68] integrate explanation generation alongside recommended items.

Cascaded multi-task recommendation refers to a sequential chain of tasks that must be performed in a strict order, modeling user behavior stages such as moving from impression to click, then to add-to-cart, and finally to purchase. This approach is distinct from sequential recommendation. An example of early work in this domain is ESMM [91], which addresses

sparsity and sample selection bias through an "impression → click → conversion" sequence.

Auxiliary multi-task recommendation. In the auxiliary task relation, one task is designated as the main task, with other tasks serving as auxiliary tasks to enhance the main task's performance. This approach is similar to single-target cross-domain recommendation. MetaBalance [45] aims to reduce the gradient magnitude of auxiliary tasks to prioritize the target task objective, while MTRec [74] incorporates link prediction to support the primary recommendation task.

In recent years, the concept of multi-scenario recommendation has emerged as a sub-category of multi-task learning. This approach involves addressing various scenarios to make recommendations in different contexts. These scenarios include situations such as providing recommendations on a homepage, suggesting similar products on individual product pages, and offering suggestions as users review their carts during the checkout process.

2.3 Cross-Domain Recommendation

CDR has garnered significant attention from the research community, encompassing various problem settings. Prior studies can be categorized using different criteria.

2.3.1 Single-Target, Dual-Target, and Multi-Target

Foundational work [6, 16, 28, 33, 43] formulates single-target setting, which aims to mitigate data sparsity by utilizing redundant data or information from other domains to enhance the original domain. For instance, CBT [78] generates a codebook matrix to extract cluster-level ratings from an auxiliary domain to support the target domain. TALMUD [101] expands on this by incorporating multiple source domains with varying relevance rates.

The research then extends into the multi-target CDR [29, 34, 82, 110, 111]. CLFM [34] adopts a multi-target approach, dividing the cluster-level codebook into common and domain-

specific sections. RMGM [79] integrates multiple sparse domains sharing common latent cluster-level patterns into a generative model.

Recently, dual-target CDR [73, 161, 162] have gained more attention, aiming to improve recommendation quality across both domains. DTCDR [161] first formulates dual-target setting by sharing user knowledge across domains. GA-DTCDR [162] enhances this framework using graph and attention mechanisms to learn better representations of overlapping users.

2.3.2 User Overlapping

Several studies [6, 54, 84, 93] address scenarios where there is full user overlap across domains, treating each domain as a vertical partition of the rating matrix. Techniques such as tri-factorization [54] and graph convolutional networks [38] are employed to align user preferences across domains.

Conversely, [16, 34, 148, 156] focus on the problem of non-overlapping users, leveraging user tags [16] and item features [126] as auxiliary information.

Further research explores the concept of partial user overlap [95, 110, 141, 163, 164] using methods like collective matrix factorization [110] and representation combination [161].

2.3.3 Using Side Information

Auxiliary knowledge, such as user tags [16, 148] and textual descriptions [66, 126], is also utilized to enhance recommendations. FUSE [16] integrates social network information by utilizing tensor factorization using users' generated tags. SCD [66] discovers similar items across domains by extracting semantical information from items' textual information.

2.4 Continual Learning for Recommendation

Continual learning seems to perfectly fit the growing nature of recommender systems; however, it has not been widely studied and remains under-explored.

While a few studies explore incremental learning in recommendation systems, limitations in their settings and evaluation protocols hinder a true continual learning framework. Ader [98] uses knowledge distillation for online session-based recommendations, splitting sessions by time and fine-tuning the model in online learning manner. Conure [153] formulates multi-platform recommendations as a continual learning problem. After each task, a small portion of the most "active" parameters (i.e., parameters with the largest absolute values) are retained and fine-tuned, while the others are set as inactive. The refined parameters are then frozen, and the redundant parameters are used to learn new tasks. However, Conure cannot accommodate new users; it can only learn from users who existed in previous tasks for every new task. ASMG [104] proposes a framework that refines a generative model over previously learned tasks to create a more up-to-date serving model at test time.

While continual learning methods aim to mitigate forgetting and maximize positive forward effects, the experimental evaluations in [98, 104, 153] only assess post-learning model performance. They overlook the effects across tasks, such as the retention of previously learned tasks and how prior tasks affect the ability to learn new ones.

2.5 Mixture of Experts Based Recommendation

Mixture of Experts (MoE) has emerged as a powerful framework for recommendation systems, particularly in multi-task learning scenarios [90, 92, 130] as the alternative to shared bottom architecture. MoE allows for the specialization of different models (experts) to handle specific aspects of the recommendation task, enabling higher capacity, more efficient learning.

For instance, MMOE [90] employs multiple gating networks to determine relevant experts

for each task in multi-task recommendation setting. SNR [92] extends MMOE with sub-network routing mechanism. PLE [130] proposes progressive layer extraction to address the seesaw phenomenon, where one task's performance improves at the expense of another's. M^3oE [157] introduces a framework for multi-domain multi-task mixture of experts recommendation.

Chapter 3

Dual-Target Disjointed Cross-Domain Recommendation

In the modern era, users increasingly interact with content across multiple online platforms, leading to a fragmented ecosystem where data is dispersed among distinct but related domains. While these platforms often host overlapping or similar items, their user interaction data remains isolated, resulting in significant sparsity within each individual domain. This data sparsity, especially acute in item-rich environments, undermines recommendation quality—particularly for users or items with limited interaction histories.

Cross-domain recommendation (CDR) aims to mitigate such issues by enabling knowledge transfer across different domains. Prior approaches often rely on overlapping users or items, or exploit side information such as user demographics or item metadata. However, these assumptions rarely hold in practice due to privacy constraints, system heterogeneity, or simply a lack of shared identifiers.

In this chapter, we explore a more challenging and realistic scenario for cross-domain recommendation, namely **NO3-CDR**, where there are *no overlapping users*, *no overlapping items*, and *no side information* between the domains. This setting is motivated by practical environments such as independent platforms offering similar services or products, where user identity linkage

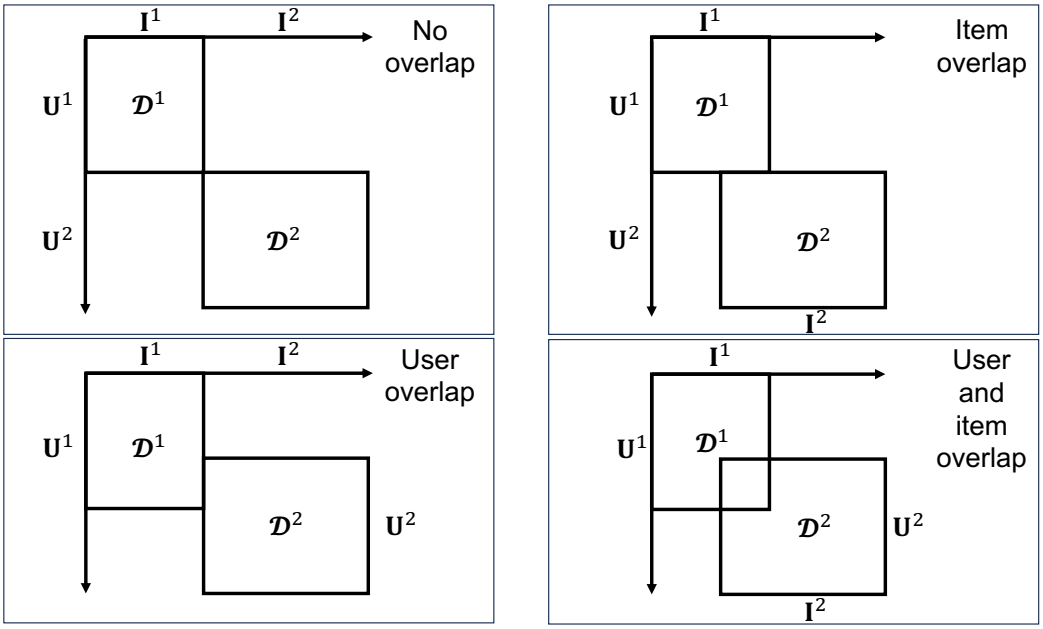


Figure 3.1: Four scenarios of overlapping user and item.

is infeasible.

To address this challenge, we introduce a novel framework that establishes a bridge between domains through implicit structural alignment of user preferences. We propose two strategies, *hard user matching* and *soft user matching*, to learn from analogous user behaviors across domains without relying on direct correspondence. These methods allow the model to align latent user representations and transfer learned patterns, even in the complete absence of shared entities or auxiliary data.

3.1 Problem Formulation

Cross-Domain Recommendation (CDR) in General. The original CDR is useful when data from one domain (known as the source domain), such as user-item interactions, is utilized to improve the recommendation process in a different but related domain (referred to as the target domain). The primary goal of CDR is to address challenges like data sparsity and the cold-start problem in the target domain by exploiting knowledge from the source domain.

Single-Target vs. Dual-Target Approaches. Prior research in CDR systems has explored methodologies aiming to transfer knowledge between distinct recommendation domains. Early work focuses on single-target approaches which typically entail exploiting redundant information from a source domain to a less abundant target domain. In these scenarios, the rich user or item information acquired from the source domain assists the learning process for the sparser target task. Techniques such as domain adaptation and transfer learning have been employed to improve recommendation performance specifically towards target domains.

Recently, there has been a growing interest in dual-target approaches, focusing on enhancing user and item recommendations across both domains. These methods seek to elevate recommendations by pinpointing and leveraging the common ground between user preferences and item attributes, thereby catering to the diverse interests of users across various domains.

Overlapping vs. Non-overlapping Data. Based on the overlap of users and items, cross-domain recommendations can be categorized into four scenarios as illustrated in Figure 3.1:

- *No overlap*: $\mathcal{U}^1 \cap \mathcal{U}^2 = \varnothing$ and $\mathcal{I}^1 \cap \mathcal{I}^2 = \varnothing$. There is no overlap between users and items.
- *User overlap*: $\mathcal{U}^1 \cap \mathcal{U}^2 \neq \varnothing$. There are shared users in both domains.
- *Item overlap*: $\mathcal{I}^1 \cap \mathcal{I}^2 \neq \varnothing$. There are shared items in both domains.
- *User and item overlap*: $\mathcal{U}^1 \cap \mathcal{U}^2 \neq \varnothing$ and $\mathcal{I}^1 \cap \mathcal{I}^2 \neq \varnothing$. There are overlaps between both users and items.

CDR with overlapped users/items seeks to capitalize on cross-domain information to enrich recommendations within the focal domain. Traditionally, such approaches presume users engaging across both domains, aiming to suggest source items to target users or mitigate cold-start issues for users new to the target domain. Yet, the constraint of overlapped users lacks practicality in the real world, considering that real user identities are not widely available.

Due to the limitations of assuming overlapped entities across domains, previous studies address the more general scenario of non-overlapping CDR, where they can leverage auxiliary information such as demographics and textual data across domains.

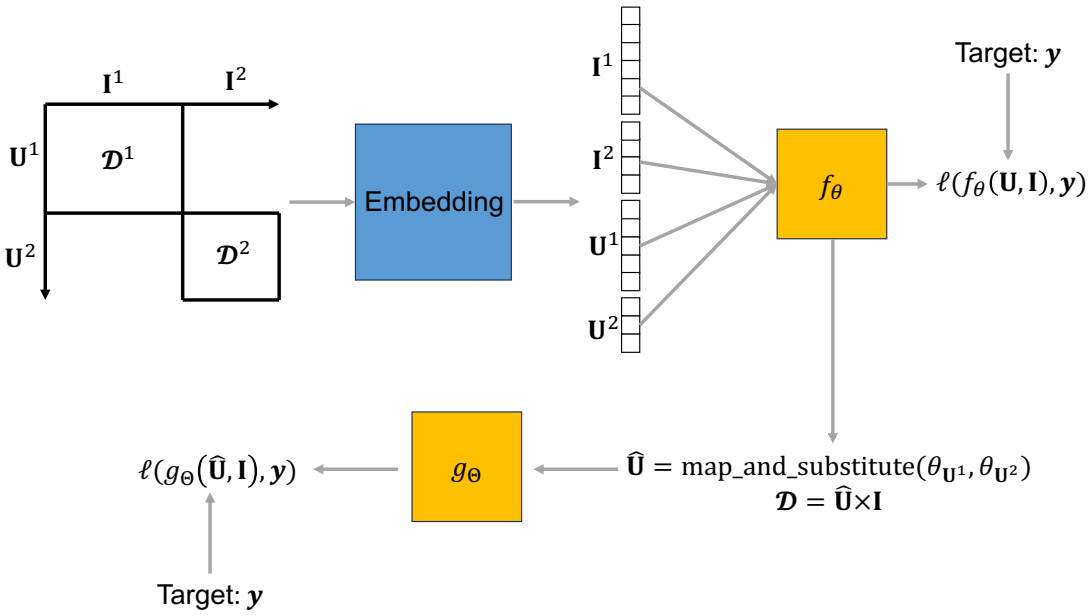


Figure 3.2: HNO3-CDR step-by-step workflow. Users, items, and ratings go through the embedding layer and recommendation model f_θ. Here, a generic recommender loss ℓ is computed by model prediction and target $\boldsymbol{y}$. Subsequently, based on the learned user representation, users from the two domains are mapped and substituted into new data $\mathcal{D}$. This new dataset is passed through a new recommendation model as an independent learning task.

However, in scenarios where additional side information is unavailable or disregarded, the recommendation task relies solely on the historical user-item interactions. This situation poses challenges in bridging the gap between the two domains.

In this chapter, we address the novel setting of *dual-target, non-overlapping, cross-domain recommendation, where auxiliary information is unavailable*. Our objective is to bridge the gap in user preferences between the two domains by aligning the underlying shared preferences of users across domains, distinguishing our novel problem setting from previous studies.

3.2 Methodology

In the context of two distinct yet related tasks, $\mathcal{D}^1 \in \mathbb{R}^{|\mathcal{U}^1|\times|\mathcal{I}^1|}$ and $\mathcal{D}^2 \in \mathbb{R}^{|\mathcal{U}^2|\times|\mathcal{I}^2|}$, our objective is to develop a recommender model f parameterized by θ, denoted as f_θ, capable of capturing user preferences while enhancing recommendation performance for both tasks. Notably, we

operate under the assumption that there is no predefined relationship between the sets of users $(\mathcal{U}^1, \mathcal{U}^2)$ and items $(\mathcal{I}^1, \mathcal{I}^2)$. Our focus is on the generalized scenario where user identities remain anonymous and cannot be directly mapped, and no additional item-related information, such as descriptions or reviews, is available.

Dual-target CDR. The dual-target framework is designed to optimize recommendation accuracy across domains. We aim to learn a unified model f_θ, that performs effectively in both domains:

$$\theta^* = \arg\min_\theta \Big(\ell(\mathcal{D}^1 \mid \theta) + \ell(\mathcal{D}^2 \mid \theta) \Big) \tag{3.1}$$

Here, ℓ represents a general model-agnostic loss function, such as Root Mean Squared Error (RMSE) for Matrix Factorization or Binary Cross-Entropy (BCE) for Neural Collaborative Filtering (NCF).

Optimizing vanilla dual-target CDR is equivalent to a simultaneous multi-task learning objective through a shared objective:

$$\theta^* = \arg\min_\theta \ell(\mathcal{D}^1, \mathcal{D}^2 \mid \theta) \tag{3.2}$$

In this scenario, the set of users, denoted as $\mathcal{U}$, is the union of two distinct individual user sets, i.e., $\mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2$, with $|\mathcal{U}| = |\mathcal{U}^1| + |\mathcal{U}^2|$. Similarly, the set of items, denoted as $\mathcal{I}$, is the union of individual item sets, i.e., $\mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$, with $|\mathcal{I}| = |\mathcal{I}^1| + |\mathcal{I}^2|$.

3.2.1 HNO3-CDR: User Hard-Matching for Cross-Domain Recommendation

In the first attempt to bridge the connection of users in two domains, we find the hard-matching of every user from one domain to one corresponding user in the other domain, maximizing the similarities of matched users. Hungarian Algorithm [65] is a widely employed method to solve assignment problems. This classic algorithm minimizes the total cost of assignments in

Algorithm 1: HNO3-CDR Learning Algorithm

Input : $\mathcal{D}^1, \mathcal{D}^2, \mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2, \mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$

$\theta^* = \arg\min_\theta \ell(\mathcal{D}_1 \cup \mathcal{D}_2 \mid \theta)$ ▷ Derive representations

$row\_ind, col\_ind = \text{Hungarian}(\theta^*_{\mathcal{U}^1}, \theta^*_{\mathcal{U}^2})$ ▷ Users matching

$\hat{\mathcal{U}} = \text{map}(\mathcal{U}, row\_ind, col\_ind)$ ▷ Mapping user indices

$\mathcal{D} \in \mathbb{R}^{|\hat{\mathcal{U}}| \times |\mathcal{I}|}$ ▷ New dataset from substituted users

$\Theta^* = \arg\min_\theta \ell(\mathcal{D} \mid \Theta)$ ▷ Learn until convergence

Output: Θ^*

bipartite graphs, offering an efficient solution for various contexts. One user from the first domain can be assigned to at most one user in the other domain and vice versa. This results in a *hard* one-to-one user-matching across the two domains. Algorithm 1 and Figure 3.2 illustrate the step-by-step hard-matching learning algorithm for CDR. First, we obtain the optimal user representations from both domains in a multi-task learning setting, where the domain-specific datasets are combined as $\mathcal{D}^1 \cup \mathcal{D}^2$. The optimal parameters are learned by optimizing $\theta^* = \arg\min_\theta \ell(\mathcal{D}^1 \cup \mathcal{D}^2 \mid \theta)$. Next, we produce the mapping of the two user sets using the Hungarian algorithm. The resulting matching is then used to substitute users from one domain with their counterparts in the other. For example, if user $u_i^1 \in \mathcal{U}^1$ is matched with user $u_j^2 \in \mathcal{U}^2$, we replace u_j^2 with u_i^1. This creates a full overlapping user scenario, where the matched users are merged into a single unified set, denoted as $\hat{\mathcal{U}}$. Finally, using the substituted user set $\hat{\mathcal{U}}$, we construct a new dataset $\mathcal{D} \in \mathbb{R}^{|\hat{\mathcal{U}}| \times |\mathcal{I}|}$ and optimize a new model g_Θ accordingly.

3.2.2 SNO3-CDR: Soft-Matching End-To-End Cross-Domain Recommendation

HNO3-CDR faces several challenges. Firstly, it adopts a step-by-step learning process, where each step is executed discretely without a seamless flow, posing difficulties in optimization. Secondly, the mapping process occurs after the initial learning phase, creating uncertainty regarding the meaningfulness of the connection between the two user sets. Once this mapping is done, adjustments to enhance its suitability are not possible. To address these issues, we propose a

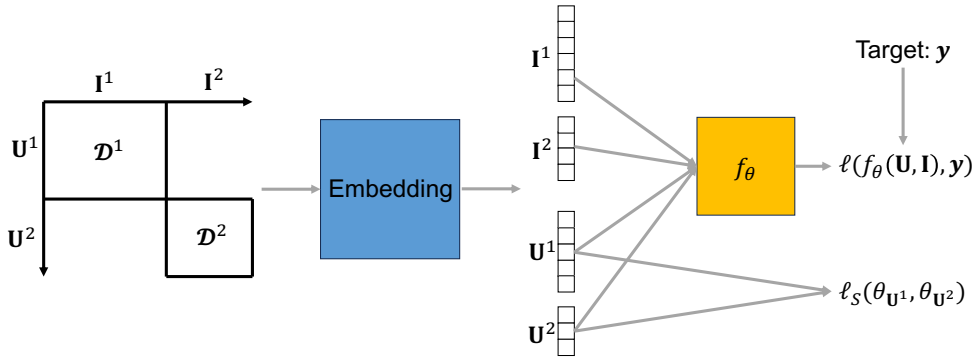


Figure 3.3: SNO3-CDR workflow. Users, items, and ratings go through the normal embedding layer and recommendation model f_θ to derive generic recommender loss ℓ between model prediction and target $\boldsymbol{y}$. Sinkhorn distance ℓ_S between two user sets acts as a bridge of users between the two domains and is combined with generic loss.

solution that involves user soft-matching and functions as an end-to-end learning model. This model streamlines the learning process into a continuous flow and prioritizes the optimization of general recommendations alongside the meaningful mapping of users. The end-to-end architecture ensures a continuous and adaptable mapping process, allowing for continuous enhancement of user representation with a focus on fostering meaningful connections throughout the model optimization process.

Sinkhorn distance

Optimal transport algorithms try to minimize transportation cost from *source/producer* to *target/consumer* given the producer' capacities and consumers' needs:

$$d = \min \sum_{i,j} P_{i,j} C_{i,j}$$

$$\text{Subject to:} \quad P_{i,j} \geq 0 \quad \text{for all } i, j$$

$$\sum_j P_{i,j} = r_i \quad \text{for all } i$$

$$\sum_i P_{i,j} = c_j \quad \text{for all } j$$

Algorithm 2: SNO3-CDR Learning Algorithm

Input : $\mathcal{D} = \mathcal{D}^1 \cup \mathcal{D}^2, \mathcal{U} = \mathcal{U}^1 \cup \mathcal{U}^2, \mathcal{I} = \mathcal{I}^1 \cup \mathcal{I}^2$, learning rate α

for *iteration = 1, . . . , w* **do**

  $\theta = \theta - \alpha \times \nabla \ell(\mathcal{D} \mid \theta)$ ▷ $w-$step warmup iterations

$\theta^* = \arg\min_\theta \ell(\mathcal{D} \mid \theta) + \gamma \ell_S(\theta_{\mathcal{U}^1}, \theta_{\mathcal{U}^2})$ ▷ Optimize until convergence

Output: θ^*

where $P_{i,j}$ is the amount to transport from P_i to C_j, $C_{i,j}$ is cost to transport from P_i to C_j, r_i is capacity of P_i, and c_j is C_j's need.

Sinkhorn algorithm [3, 19] can be applied to transform the optimal transportation problem into the mapping of two "point clouds", where we transport "mass" from one set of points to another. [19] rewrites the original optimization formulation into Lagrange form:

$$d_S(P, C) = \sum_{i,j} P_{i,j} C_{i,j} - \frac{1}{\lambda} h(P) + \sum_i m_i \left(\sum_j P_{i,j} - r_i \right) + \sum_j n_j \left(\sum_i P_{i,j} - c_j \right)$$

with m_i and n_j are Lagrange multipliers.

The derivative w.r.t. P can be easily derived by:

$$\frac{\partial d_S}{\partial P_{i,j}} = C_{i,j} + \frac{1}{\lambda} + \frac{1}{\lambda} \log P_{i,j} + m_i + n_j$$

This differentiable Sinkhorn distance can be seamlessly incorporated into any general objective of recommender models.

Mediate Latent Preferences by Sinkhorn Distance.

We constrain users from two domains to be close to each other without binding them tightly one-to-one. We define the Sinkhorn distance between two sets (i.e., point clouds) of user representations, $\mathcal{U}^1$ and $\mathcal{U}^2$, as:

$$\ell_S(\theta_{\mathcal{U}^1}, \theta_{\mathcal{U}^2}) = d_S(\mathcal{U}^1, \mathcal{U}^2) + d_S(\mathcal{U}^2, \mathcal{U}^1) \tag{3.3}$$

Here, $d_S(\mathcal{U}^1, \mathcal{U}^2)$ denotes the standard uni-directional Sinkhorn distance from point cloud $\mathcal{U}^1$ to $\mathcal{U}^2$, calculated using an arbitrary ground distance function (e.g., Euclidean, cosine) as the transportation cost between points in $\mathcal{U}^1$ and $\mathcal{U}^2$. This results in a symmetric, bi-directional distance measure. ℓ_S is differentiable with respect to both sets of representations, $\theta_{\mathcal{U}^1}$ and $\theta_{\mathcal{U}^2}$, making it suitable for gradient-based optimization within a recommender system framework. Alternatively, we could employ a standard uni-directional Sinkhorn distance, using either $d_S(\mathcal{U}^1, \mathcal{U}^2)$ or $d_S(\mathcal{U}^2, \mathcal{U}^1)$. Section 4.3 will show the impact of bi-directional and uni-directional formulations.

We incorporate ℓ_S into the training objective to mediate the latent preferences of users across domains. This encourages the user representations to be similar while retaining the capacity to capture domain-specific preferences. Conceptually, this can be formulated as a constrained optimization problem:

$$\theta^* = \arg\min_{\theta} \ell(\mathcal{D}^1, \mathcal{D}^2 \mid \theta)$$
$$\text{Subject to:} \quad \ell_S(\theta_{\mathcal{U}^1}, \theta_{\mathcal{U}^2}) \leq \alpha^2$$

where $\ell(\mathcal{D}^1, \mathcal{D}^2 \mid \theta)$ is the primary recommendation loss function for data from domains $\mathcal{D}^1$ and $\mathcal{D}^2$, and α^2 is a positive tolerance threshold.

By rewriting the constraint as $\ell_S(\theta_{\mathcal{U}^1}, \theta_{\mathcal{U}^2}) - \alpha^2 \leq 0$, the final objective function for our end-to-end learning framework using the Lagrange multiplier is derived as:

$$\begin{aligned} \theta^* &= \arg\min_{\theta} \ell(\mathcal{D}^1, \mathcal{D}^2 \mid \theta) + \gamma \left(\ell_S(\theta_{\mathcal{U}^1}, \theta_{\mathcal{U}^2}) - \alpha^2\right) \\ &\propto \arg\min_{\theta} \ell(\mathcal{D}^1, \mathcal{D}^2 \mid \theta) + \gamma \ell_S(\theta_{\mathcal{U}^1}, \theta_{\mathcal{U}^2}) \end{aligned} \tag{3.4}$$

This augmented objective effectively balances the optimization of the primary recommendation task ℓ with the continuous and flexible mapping process ℓ_S, therefore promoting the transfer and adaptation of user preferences across domains by aligning their representations.

Alternatively, this augmented objective can be interpreted within a multi-task learning frame-

Table 3.1: Datasets stats for four scenarios

Dataset	Stats	Generic		Superset		Subset		Common	
		$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^1$	$\mathcal{D}^2$	$\mathcal{D}^1$	$\mathcal{D}^2$
Books	#ratings	8,898,041	982,619	8,898,041	967,196	1,319,803	982,619	1,319,803	967,196
—	#users	367,982	61,934	367,982	61,236	61,236	61,934	61,236	61,236
Kindle	#items	603,668	68,223	603,668	68,079	256,019	68,223	256,019	68,079
Electronics	#ratings	6,387,916	1,109,521	6,387,916	648,026	1,230,678	1,109,521	1,230,678	648,026
—	#users	694,953	154,813	694,953	81,381	81,381	154,813	81,381	81,381
Cell Phones	#items	157,693	47,607	157,693	46,996	134,621	47,607	134,621	46,996
CDs	#ratings	1,377,008	123,518	1,377,008	42,872	181,705	123,518	181,705	42,872
—	#users	107,546	12,381	107,546	3,720	3,720	12,381	3,720	3,720
Music	#items	71,943	9,906	71,943	9,113	49,898	9,906	49,898	9,113
AMZ Books	#ratings	223,302	197,140	-	-	-	-	-	-
—	#users	3,353	2,578	-	-	-	-	-	-
Book Crossing	#items	5,752	4,313	-	-	-	-	-	-

work, where minimizing the transportation cost ℓ_s serves as an auxiliary task that supports and improves the performance of the primary recommendation task.

The overall learning process, illustrated in Algorithm 2 and Figure 3.3, involves an initial warm-up phase to learn meaningful user representations, followed by concurrently optimizing the augmented objective, which incorporates both the recommendation loss and the transportation cost.

3.3 Experiments

Datasets. For experiments, the first three pairs of datasets are from Amazon[1]: *Books - Kindle Store*; *Electronics - Cell Phones and Accessories*; and *CDs and Vinyl - Digital Music*, chosen based on the assumption that users' preferences are likely shared between the two domains. For example, users who enjoy reading books may also be interested in similar Kindle e-books. To further diversify our analysis, we construct a fourth dataset from two sources: *Amazon Books - Book Crossing*[2], where the two share the same category of items but from different user sets and sources.

[1]https://nijianmo.github.io/amazon/index.html
[2]https://grouplens.org/datasets/book-crossing/

Four Scenarios. For comprehensive analysis, we explore four distinct scenarios, based on the overlap of two user sets $\mathcal{U}^1$ and $\mathcal{U}^2$, from the generic case with no constraint of users, to the extreme scenario where only users overlap between two domains are allowed, and the two middle ground scenarios.

- Scenario 1 (Generic): Any $\mathcal{U}^1$ and $\mathcal{U}^2$
- Scenario 2 (Superset): $\mathcal{U}^1 \supset \mathcal{U}^2$
- Scenario 3 (Subset): $\mathcal{U}^1 \subset \mathcal{U}^2$
- Scenario 4 (Common): $\mathcal{U}^1 = \mathcal{U}^2$

In all four cases, regardless of overlapping, *user identities are masked* so that the model treats the same user in two domains as two different users. Table 3.1 summarizes the respective statistics of the datasets under each of the four experimental scenarios, where $\mathcal{D}^1$ and $\mathcal{D}^2$ denote the two domains (e.g., in *Books - Kindle Store*, $\mathcal{D}^1$ is *Books* and $\mathcal{D}^2$ is *Kindle Store*).

Rating and Ranking Tasks. For evaluation, we employ two recommendation tasks: rating prediction and ranking prediction. We apply our model-agnostic proposed methods to two representative backbone models: Matrix Factorization (MF [63]) and Neural Collaborative Filtering (NCF [42]) and evaluate their performance. We use Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for rating prediction, and Normalized Discounted Cumulative Gain (NDCG) and Recall with $k = 50$ for ranking prediction.

Comparative Methods. Due to its novel setting, there is no direct baseline for NO3-CDR. Previous dual-target cross-domain recommendation studies either (i) utilize shared parameters from the same users or items, which assumes user or item overlap–an assumption that does not hold in our setting–or (ii) leverage other data modalities as side information, which are also unavailable in our case. Therefore, we consider the comparative methods below:

- Base models: We use MF [63] and NCF [42] as backbone models for rating and ranking tasks, respectively. We combine data from two domains and train with one single model, with objective function in Equation 3.2.

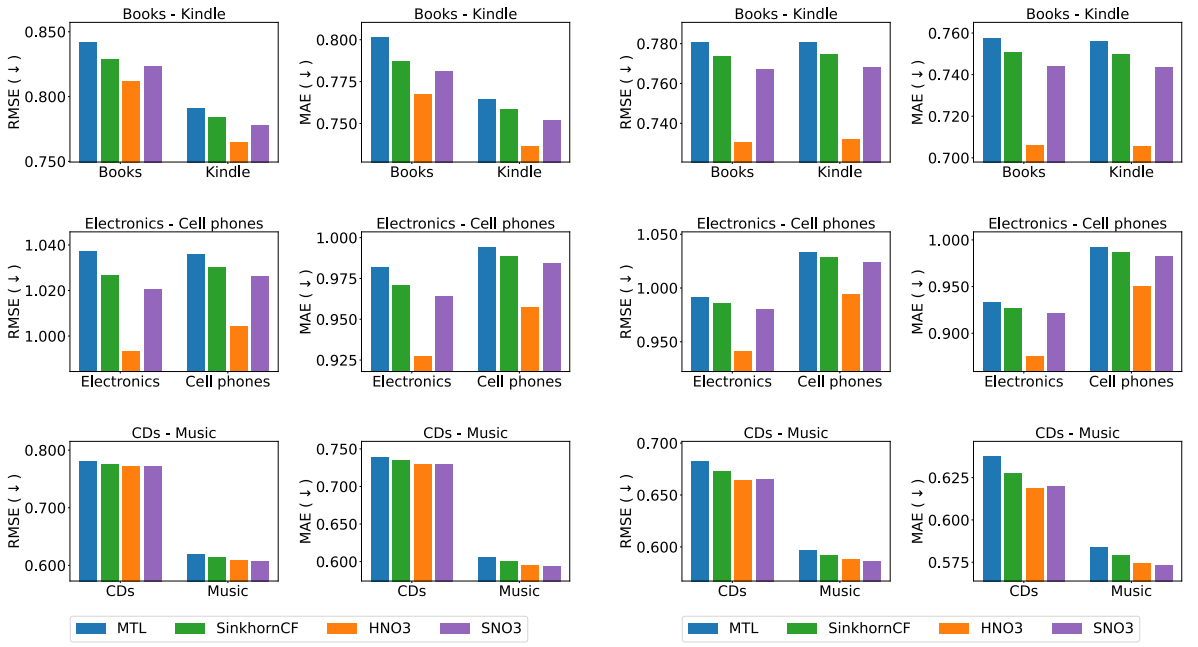
Table 3.2: The effects of aggregating user identities across domains for *Amazon CDs - Digital Music* dataset. Better results are in bold.

Training	CDs		Digital Music	
	RMSE(↓)	MAE(↓)	RMSE(↓)	MAE(↓)
Separately	0.6612	0.6103	0.5959	0.5729
Together	**0.6299**	**0.5775**	**0.5848**	**0.5621**

- SinkhornCF [75]: Infuses Sinkhorn divergence of items into the learning objective. It can be applied to MF (i.e., SinkhornMF) and NCF (i.e., SinkhornNCF).
- NMF [72]: As recent studies [87, 88] suggest Non-negative Matrix Factorization (NMF) to be superior to the original MF, NMF is included as a baseline for rating prediction.
- VAECF [80] and its variants are widely used due to their non-linear probabilistic generative modeling. We include VAECF as a baseline for the ranking prediction task.

We adopt NMF and VAECF, which are considered superior to the backbone models MF and NCF, to evaluate whether the proposed methods can enhance the backbone models sufficiently to outperform these two baselines.

Hyper-parameter Tuning. Each dataset is partitioned into training, validation, and test sets using a chronological proportional split as described in prior works [47, 94], with a ratio of 60/20/20 for training, validation, and test sets. All methods are trained on the training set, tuned for optimal performance and model selection based on the validation set, and the best models are evaluated on the test set. We perform random search for hyper-parameter tuning, with the search space for some key hyper-parameters as follows: learning rate $\in [0.001, 0.1]$, embedding size $\in \{64, 100, 128, 256\}$, and control parameter $\gamma \in [0.1, 1.0]$. The number of warm-up iterations for SNO3 and HNO3-CDR is set to $w = 5$.



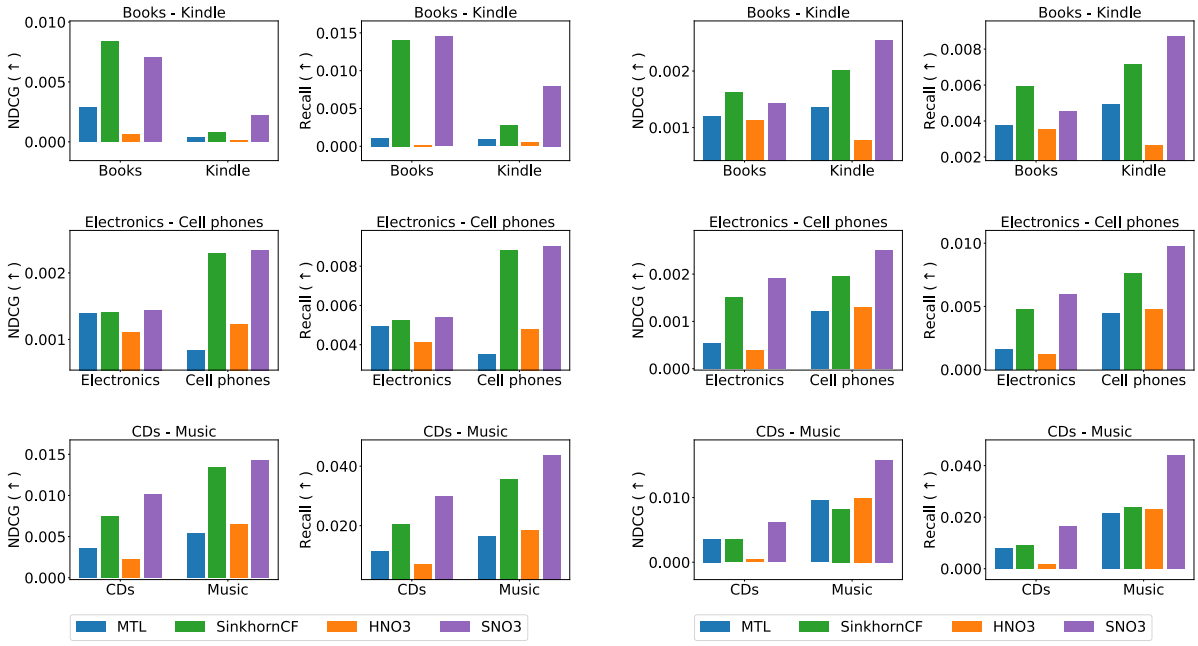
(a) Generic scenario.

(b) Superset scenario.

(c) Subset scenario.

(d) Common scenario.

Figure 3.4: Rating prediction performances in four scenarios. For RMSE and MAE, the lower values (↓) indicate better results.



(a) Generic scenario.

(b) Superset scenario.

(c) Subset scenario.

(d) Common scenario.

Figure 3.5: Ranking prediction performances in four scenarios. For NDCG and Recall, higher values (↑) indicate better results.

Table 3.3: Results of different "target" domain on *CDs - Music*'s four scenarios. Best results are in bold.

(a) Common scenario

Target domain	Rating prediction				Ranking prediction			
	D^1		D^2		D^1		D^2	
	RMSE (↓)	MAE (↓)	RMSE (↓)	MAE (↓)	NDCG (↑)	Recall (↑)	NDCG (↑)	Recall (↑)
None	0.7026	0.6611	0.6854	0.6683	0.0054	0.0134	0.0092	0.0274
$\mathcal{D}^1$	**0.6942**	**0.6523**	**0.6808**	**0.6637**	0.0065	0.0164	0.0082	0.0237
$\mathcal{D}^2$	0.7129	0.6716	0.6892	0.6719	**0.0069**	**0.0169**	**0.0121**	**0.0360**
Auto	**0.6942**	**0.6523**	**0.6808**	**0.6637**	**0.0069**	**0.0169**	**0.0121**	**0.0360**

(b) Superset scenario

Target domain	Rating prediction				Ranking prediction			
	D^1		D^2		D^1		D^2	
	RMSE (↓)	MAE (↓)	RMSE (↓)	MAE (↓)	NDCG (↑)	Recall (↑)	NDCG (↑)	Recall (↑)
None	0.6103	0.5900	0.5922	0.5790	0.0045	0.0134	0.0124	0.0386
$\mathcal{D}^1$	**0.6041**	**0.5836**	**0.5864**	**0.5731**	**0.0046**	**0.0146**	0.0135	0.0395
$\mathcal{D}^2$	0.6161	0.5958	0.5969	0.5837	0.0028	0.0085	0.0135	**0.0414**
Auto	**0.6041**	**0.5836**	**0.5864**	**0.5731**	**0.0046**	**0.0146**	0.0135	0.0395

(c) Subset scenario

Target domain	Rating prediction				Ranking prediction			
	D^1		D^2		D^1		D^2	
	RMSE (↓)	MAE (↓)	RMSE (↓)	MAE (↓)	NDCG (↑)	Recall (↑)	NDCG (↑)	Recall (↑)
None	0.6726	0.6277	0.5922	0.5790	0.0035	0.0082	0.0082	0.0234
$\mathcal{D}^1$	**0.6650**	**0.6200**	**0.5864**	**0.5731**	**0.0036**	**0.0089**	**0.0095**	**0.0237**
$\mathcal{D}^2$	0.6824	0.6373	0.5969	0.5837	0.0023	0.0071	0.0048	0.0217
Auto	**0.6650**	**0.6200**	**0.5864**	**0.5731**	**0.0036**	**0.0089**	**0.0095**	**0.0237**

(d) Generic scenario

Target domain	Rating prediction				Ranking prediction			
	D^1		D^2		D^1		D^2	
	RMSE (↓)	MAE (↓)	RMSE (↓)	MAE (↓)	NDCG (↑)	Recall (↑)	NDCG (↑)	Recall (↑)
None	0.7757	0.7349	0.6138	0.6014	0.0035	0.0107	0.0047	0.0130
$\mathcal{D}^1$	0.7806	0.7394	0.6192	0.6068	0.0025	0.0085	0.0044	0.0110
$\mathcal{D}^2$	**0.7709**	**0.7301**	**0.6074**	**0.5948**	**0.0036**	**0.0112**	**0.0055**	**0.0163**
Auto	**0.7709**	**0.7301**	**0.6074**	**0.5948**	**0.0036**	**0.0112**	**0.0055**	**0.0163**

3.3.1 Research Questions (RQ) and Discussions

RQ1: The effects of using user identities across domains.

We first investigate the potential benefits of having user identities across domains. We carry out an experiment to compare the performance of training the model separately and together on the *CDs - Music* dataset. We first filter only users who have presented in both domains, then train the MF model in two different settings: (i) separately, where we train the model on each domain independently, and (ii) together, where we combine user-item interactions from both domains

Table 3.4: Results for *Amazon Books - Book Crossing* dataset. Note that in Amazon Books, the rating scale is from 1 to 5, while for Book Crossing is from 1 to 10. Best results are in bold, while second-best results are in *italic*.

(a) Rating Prediction

Model	AMZ Books		Book Crossing	
	RMSE	MAE	RMSE	MAE
MF	0.9080	0.8429	3.3754	3.1218
SinkhornMF	0.8878	0.8289	3.3116	3.1498
NMF	**0.8853**	0.8226	3.2920	3.0703
MF-HNO3	*0.8864*	**0.8193**	**3.2564**	**3.0700**
MF-SNO3	0.8865	*0.8195*	*3.2771*	*3.0701*

(b) Ranking Prediction

Model	AMZ Books		Book Crossing	
	NDCG (%)	Recall (%)	NDCG (%)	Recall (%)
NCF	*0.1075*	*0.3494*	0.1324	0.3268
SinkhornNCF	0.0938	0.2784	*0.1667*	*0.3582*
VAECF	0.0786	0.2310	0.1308	0.3453
NCF-HNO3	0.0890	0.2069	0.1168	0.2954
NCF-SNO3	**0.1198**	**0.3709**	**0.1709**	**0.3838**

and train on the whole data.

Table 3.2 contrasts training the model separately versus jointly on the *CDs - Digital* dataset. The results show that joint training reduces both RMSE and MAE for *CDs* and *Music*, demonstrating improved performance over separate training. It aligns with the intuition that shared user identities can improve predictions across domains. Therefore, effective mechanisms for aligning user identities can be leveraged to enhance recommendations.

RQ2: How do the two variants NO3-CDR perform?

Figures 3.4 and 3.5 present results across three *Amazon* datasets for two prediction tasks under four different scenarios. Comparing against benchmark baselines, we observe distinct behaviors in each task. For rating prediction (Figure 3.4), the MF-HNO3 and MF-SNO3 variants outperform SinkhornMF and NMF, both of which surpass traditional MF. Notably, MF-HNO3 con-

Table 3.5: Case study in CDs-Music dataset

User in CDs: A117WAVHO1WAIE		User in Music: A8QZWK9SUH66P	
Items rated	Items categories	Items rated	Items categories
The Commodores	R&B, Funk, Pop	Doo-Wops & Hooligans	Pop, R&B
Earth Wind & Fire	R&B, Funk, Soul	Waking Up	Pop, Rock
Song of Solomon	Rock, Pop	X	Pop, R&B
Carpenters Gold	Pop	Here's To The Good Times	Pop, Rock
Piano Prophet	Jazz, R&B	The Fault In Our Stars	Rock
		The Hunting Party	Rock

User in CDs: A28DBLK5JB17P3		User in Music: A167KI3P7XN1AM	
Items rated	Items categories	Items rated	Items categories
Led Zeppelin: Box	Rock, Metal		
Led Zeppelin I	Rock, Metal	Made In The A.M.	Pop, Rock
Led Zeppelin II	Rock, Metal		
Houses of the Holy	Rock, Metal	Somewhere In Time LP	Rock, Metal
At Your Service	Pop, Rock		

User in CDs: A28DBLK5JB17P3		User in Music: A1VFOUHOYX29YP	
Items rated	Items categories	Items rated	Items categories
Led Zeppelin: Box	Rock, Metal	Light Me Up	Rock, Metal
Led Zeppelin I	Rock, Metal	Hit Me Like A Man	Rock, Metal
Led Zeppelin II	Rock, Metal	Bad Magic - Motörhead	Rock, Metal
Houses of the Holy	Rock, Metal	Dystopia - Megadeth	Rock, Metal
At Your Service	Pop, Rock	XI Metal - Church	Rock, Metal

sistently achieves the best performance, yielding significantly lower RMSE and MAE, followed by MF-SNO3 as the second-best performer. In contrast, for ranking prediction (Figure 3.5), the Hungarian-based NCF-HNO3 fails to surpass the NCF baseline, while SinkhornNCF and VAECF have superior performance over vanilla NCF. Among the NO3 variants, NCF-SNO3 consistently enhances the NCF backbone, achieving the best overall performance. It surpasses the two strongest NCF-based baselines in most cases, particularly in terms of NDCG and Recall. The only exceptions are NDCG on the *Books* domain (Figure 3.5a, top-left) and both NDCG and Recall on the *Electronics* and *Cell Phones* domains (Figure 3.5c, middle row), where Sinkhorn-

NCF marginally outperforms NCF-SNO3.

The choice between HNO3 and SNO3 depends on the specific recommendation task: HNO3 is more effective for rating prediction, while SNO3 excels in ranking tasks. This is likely due to the different ways the two backbone models generate item scores. In MF, ratings are directly predicted from user-item embeddings, which aligns well with the one-to-one matching of the HNO3 variant. In contrast, NCF generates user-item scores indirectly through multiple feed-forward neural network layers, which benefits more from the flexible matching SNO3 for ranking tasks.

RQ3: The scenarios involving two different data sources.

Table 3.4 presents the results of experiments conducted on datasets from two different sources: *Amazon Books* and *Book Crossing*. For the rating task, MF-HNO3 delivers the best performance in terms of RMSE and MAE, except for RMSE on *Amazon Books*, where it ranks second to NMF. SNO3-CDR closely follows behind. In the item ranking task, NCF-SNO3 outperforms the others in terms of NDCG and Recall, while NCF-HNO3 does not improve upon the NCF baseline. These findings align with previous results from the three *Amazon* dataset pairs. This supports the idea that aggregating data from multiple fragmented platforms can enhance performance. While more data does not always guarantee better results, effectively guiding the learning process allows the model to leverage richer information. The results also demonstrate the effectiveness of the proposed methods in improving recommendations across diverse data sources.

RQ4: Uni-directional versus bi-directional SNO3.

SNO3-CDR offers the flexibility to transport bi-directionally between two "point clouds" $\mathcal{U}^1$ and $\mathcal{U}^2$. To see whether uni-directional or bi-directional yields superior recommendation, and whether there is an optimal assignment to each domain as source or target, we analyze three cases: (i) bi-directional transportation (i.e., no designated "target"), (ii) $\mathcal{U}^1$ as "target" point cloud, and (iii) $\mathcal{U}^2$ as "target" nodes.

Table 3.3 compares bi-directional and uni-directional MF-SNO3 and NCF-SNO3 across all four scenarios of the *CDs-Music* dataset. In all cases, the best uni-directional method outper-

forms the bi-directional method, improving results in both domains. No domain consistently outperforms the other. In three out of four scenarios, selecting one target domain enhances both rating and ranking predictions. The exception is the *Common* scenario: for ratings, selecting $\mathcal{D}_1$ as the target improves results, while for ranking, choosing $\mathcal{D}_2$ yields better performance.

In pursuit of optimal results for the one-sided SNO3, we propose an *automatic* method to identify the better "target" domain by selecting the domain with higher user representation variance. After warm-up epochs, we calculate and compare variances, choosing the domain with higher variance as the target. This *Auto* method achieves the best SNO3 results in most cases (see Table 3.3), except in the Superset scenario, where *Auto* performs better in $\mathcal{D}^1$ but not in $\mathcal{D}^2$. This discrepancy arises due to the extreme imbalance in dataset sizes (Table 3.1): $\mathcal{D}^1$ has over 1 million ratings, while $\mathcal{D}^2$ has only 42,872 ratings.

RQ5: Should we prioritize matching the same user across domains to enhance recommendation performance?

Users may portray different preferences across platforms, such as purchasing classical music on *CDs and Vinyl* and modern trending songs on *Digital Music*. Our goal is to enhance recommendations on both platforms rather than focusing solely on matching users across domains, as we assume no overlap in users.

However, *though not used in the learning as presumed non-existent*, the availability of user identity information allows us to investigate whether the algorithms match users across domains correctly. We investigate the user mapping accuracy in *CDs - Music* dataset's Common scenario, using MF-HNO3, since it performs best in rating prediction; and NCF-SNO3 for ranking. Surprisingly, out of 3,720 users across both domains, MF-HNO3 accurately maps only 1 to 3 users on different runs. While NCF-SNO3 does not output user mapping, we derive the mapping based on the closest Sinkhorn distances of final user representations, and the result is 0 to 3 correct user pairs.

HNO3 is a step-by-step learning process and mapping quality solely relies on user representation derived from the initial learning model. For SNO3, the control variable γ in Equation 3.4 can

be adjusted to balance recommendation and transportation objectives. However, as γ increases (favoring user mapping), recommendation performance gradually decreases. The Sinkhorn distance in SNO3 acts as a flexible bridge between domains, where matching users is not prioritized to achieve the best recommendation quality.

3.3.2 Case Study: Example Matched User Pairs

Table 3.5 presents three user pairs from the *CDs* domain along with their corresponding matches from the *Music* domain.

In the first pair, both users show similar preferences for a mix of R&B, Pop, and Rock. User A117WAVHO1WAIE has a diverse taste, enjoying artists like *The Commodores*, *Earth Wind & Fire*, and *The Carpenters*, ranging from classic R&B and funk to pop. Interestingly, her match in the music domain, user A8QZWK9SUH66P, also appreciates Pop and R&B, with selections like *Bruno Mars' "Doo-Wops & Hooligans"* and *Florida Georgia Line's "Here's To The Good Times"*, showcasing a similar inclination to pop and rock.

The second pair, user A28DBLK5JB17P3 in *CDs* and user A167KI3P7XN1AM in *Music*, exhibited more distinct common preferences. They are deeply rooted in rock and metal, especially classic metal rock. In the third pair, user from the second pair, A28DBLK5JB17P3, is also the best match for the user in *Music*, A1VFOUHOYX29YP, who also roots for rock albums, such as The Pretty Reckless' *"Light Me Up"* and *"Hit Me Like A Man"*.

NCF-SNO3 effectively captures the similarities among intricate user preferences. The consolidation of these identified parallels among matched user pairs serves to reinforce the notion of preference bridging, rather than prioritizing the enhancement of correct matching accuracy. While the optimal match for a user across domains may not fully align with their unique preferences, they may exhibit a greater degree of similarity in their preferences compared to their own preferences in different domains.

3.4 Discussion

This chapter addresses the challenge of scarce data in recommendation systems. We introduce the novel scenario of NO3-CDR framework and propose a unique approach to enhance recommendation systems by leveraging connections across distinct yet conceptually similar datasets from multiple platforms based on user underlying preferences. Our methodology focuses on bridging the gap between these platforms, enabling mutual improvements in recommendation quality while respecting user privacy. Empirical experiments demonstrate the effectiveness of our approach in improving recommendation quality, showcasing its potential to address data scarcity challenges in fragmented cross-domain recommendation systems.

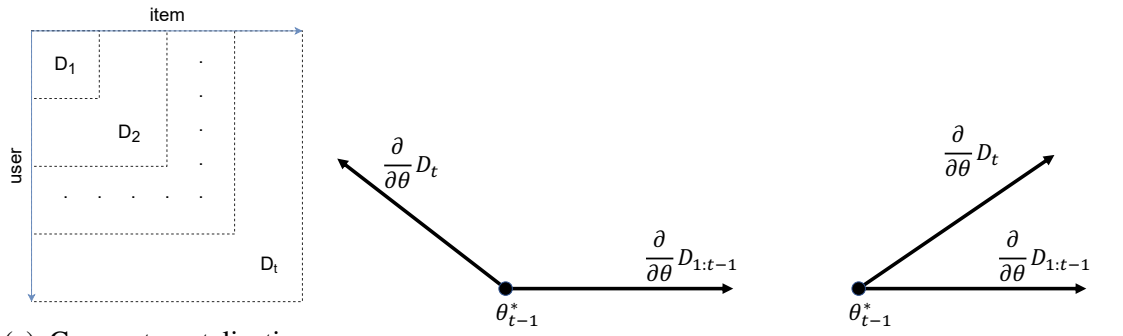
Chapter 4

Continual Collaborative Filtering Through Gradient Alignment

Most real-world recommendation systems operate in dynamic environments, where new users, new items, and new interactions are continuously introduced. However, the majority of collaborative filtering systems are designed as static models, as they are trained once on a fixed dataset and deployed without subsequent adaptation. This static assumption fails to capture the evolving nature of user preferences and item availability, especially in fast-paced domains such as e-commerce, streaming platforms, and social networks.

There need to be mechanisms to handle the dynamic nature of recommendation systems. The common remedy is to periodically retrain the model using updated datasets. While this allows the system to incorporate new data, it suffers from high computational overhead and latency. Additionally, frequent retraining may be infeasible due to privacy constraints, where older data cannot be retained or reused.

An alternative approach involves online learning [81, 127] or model fine-tuning, where the model is incrementally updated with new user-item interactions. Although this reduces the computational burden, it suffers from *catastrophic forgetting*, where newly acquired knowledge may overwrite previously learned preferences. This leads to degradation in recommendation quality



(a) Compartmentalization of rating data into tasks

(b) Interference among tasks

(c) Transfer between tasks

Figure 4.1: If two gradients give same loss for D_t, we prefer the gradient that leverages transfer between tasks over interference (Fig. 4.1c over 4.1b).

for older users or items.

To address these challenges, we propose a novel framework named *Continual Collaborative Filtering (CCF)*. This framework formulates the ongoing influx of new users, items, and interactions as a continual learning problem, where the system is exposed to a sequence of tasks over time, and the system must learn to make recommendations based on both current and past data, without relying on full retraining or storing all historical data. Our framework tries to balance the trade-off between *stability* (to retain past preferences) and *plasticity* (to rapidly learn new examples) without growing capacity proportionally with streaming data. This, in a nutshell, is the crux of this chapter: *designing a continual learning framework for collaborative filtering, investigating and analyzing methods for balancing learning ability and retention in continual collaborative filtering setting*.

4.1 Problem Formulation

To better assess the *stability-plasticity dilemma*, we propose a Continual Learning framework and evaluation protocol for recommender systems with clear definitions of tasks, goals, and metrics.

Tasks and Goals Over time, new users keep appearing and existing users continue to interact and explore new items. We could treat each group of users as a new "task". However, old

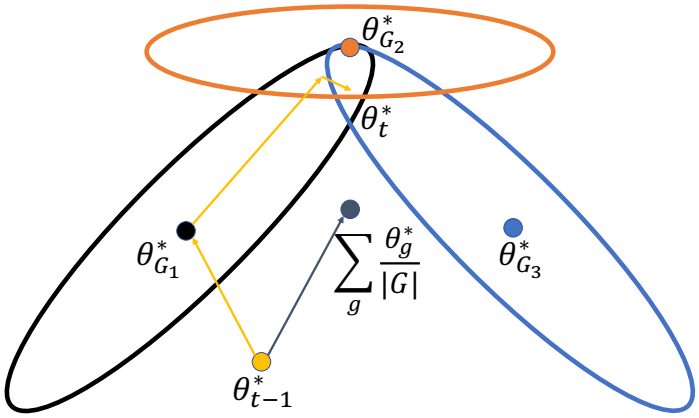


Figure 4.2: UACF desired learning path. Ovals are low-loss regions for individual groups. Normally, model updates towards average gradient over all groups of users, and finally ends up at the center of the triangle. Our algorithm learns group by group to find the optimal path to the intersection.

items might still attract users even as new items are offered to users. Thus, we propose a novel continual learning setting for recommender systems, where interactions keep expanding on both user and item sides. We define a task as compartmentalization of users and items, as illustrated in Figure 4.1a. Suppose we order the dimensions of $\mathbf{R}$, i.e., users and items, according to the time they first emerge. In that case, a task corresponds to a block of users and items that appear over a specified period of time[1]. Within a new task, we see ratings corresponding to earlier users on new items, new users on earlier items, new users on new items, and potentially new ratings by existing users on old items. While observing the data periodically, the objective is to learn a generalized recommendation model that not only could capture recent preferences but also retain previous users' interest.

Notation: We index users by $u \in \{1, \ldots, U\}$ and items by $i \in \{1, \ldots, I\}$, which form the user-item interaction matrix $\mathbf{R} \in \mathbb{N}^{U \times I}$. We denote T as the number of tasks in continual learning setting, which in theory is infinite. We refer to the set of observations within a task t as $\mathcal{D}_t \in \mathbf{R}^{U_t \times I_t}$, where U_t, I_t are the number of users and items in task t, respectively. $U_{t+1} >$

[1]Note that there is flexibility in how a task is defined, depending on specific application scenarios. For instance, it would also be possible to define a task in terms of only new items, with the assumption that all users are present throughout.

$U_t, I_{t+1} > I_t$, which reflects ever-growing systems acquiring more users and items. We use ℓ as model-agnostic loss function for general collaborative filtering algorithm.

"Multi-Task" Collaborative Filtering In a typical offline batch learning manner, the objective is to learn the set of user and item parameters θ that minimize the loss over i.i.d. distribution samples within dataset $\mathcal{D} \in \mathbf{R}^{U \times I}$:

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{r_{ui} \sim \mathcal{D}} \left[\ell \left(r_{ui} \mid \theta \right) \right] \tag{4.1}$$

The above assumes a task-agnostic setting with an arbitrary loss function. Given multiple tasks, the learning objective turns into minimizing total loss for every observed user and item, over all the tasks:

$$\theta^* = \arg\min_{\theta} \sum_{t=1}^{T} \mathbb{E}_{r_{ui} \sim \mathcal{D}_t} \left[\ell \left(r_{ui} \mid \theta \right) \right] \tag{4.2}$$

However, this loss function presumes that all tasks are observed at once.

"Online" Task Learning In the continual learning context of interest in this work, each task appears sequentially. Access to past tasks' data is prohibited. We learn each task consecutively. At each time step t, we leverage the prior parameters θ^*_{t-1} as initialization (i.e., transfer learning), to minimize loss over current dataset $\mathcal{D}_t$ to arrive at new parameters θ_t. This is equivalent to the task-level *online learning* objective.

$$\theta^*_t = \arg\min_{\theta_t} \mathbb{E}_{r_{ui} \sim \mathcal{D}_t} \left[\ell \left(r_{ui} \mid \theta^*_{t-1}, \theta_t \right) \right] \tag{4.3}$$

The sequential nature of tasks implies that as we handle tasks over time, the model parameters tend to favor recent tasks. Over time, they may no longer fit the earlier tasks well, resulting in lower performance for older users and items. This phenomenon is known as *catastrophic forgetting* [36, 96].

Training and Test Protocol By default, at task t, the model is initialized from previous model $f_{\theta_{t-1}}$ and can be trained in offline batch learning setting using data of current task $\mathcal{D}_t$,

while access to past samples $\mathcal{D}_{1,\ldots,t-1}$ is forbidden. After learning, the model performs on test set of every seen task (i.e., $\mathcal{D}_1^{test}, \ldots, \mathcal{D}_t^{test}$) without revealing task identities (i.e., single-head setting). This protocol is called *fine-tuning*.

Metrics Conventionally for Collaborative Filtering, we have two sets of metrics: rating and ranking. In this chapter, we adopt Mean Square Error (MSE) for rating metric. For ranking, Recall (Hit ratio) is used to measure the ranking effectiveness of recommendation model. Besides measuring performance for individual tasks, it is also crucial to monitor how the learning process *affects across tasks*. The goal is to quickly learn current task to satisfy immediate new users while preserving existing preferences. For each base metric above, we construct the matrix $\mathbf{a} \in \mathbb{R}^{T \times T}$, where a_{ij} is performance on task j after observing task i. To aggregate performance across tasks, we derive these two metrics:

- Learning Average (LA) $\frac{1}{T}\sum_{i=1}^{T} a_{i,i}$: to measure learning ability, how good a model is at learning tasks and whether it benefits from previously learned tasks.
- Retained Average (RA) $\frac{1}{T}\sum_{i=1}^{T} a_{T,i}$: to measure learning retention and improvement after learning the final task. The overall average will be the average performance of every learned task up to the time of testing.

With limited capacity, one system cannot preserve perfect recall of previously learned tasks while also absorbing huge data from an arbitrary number of tasks, which refers to the stability-plasticity dilemma. Given that contradiction, we further propose a combined measure, which is the harmonic mean of learning average and retained average. This metric is the aggregated number that conveys the overall goodness of a Continual Collaborative Filtering model.

$$\text{Harmonic mean} = \frac{2 * LA * RA}{LA + RA}$$

Since rating metrics are better when lower, the corresponding continual metrics are better when lower. Conversely, ranking metrics are better when higher.

4.2 Methodology

Continual learning optimization resembles a tug-of-war, where tasks, users, or instances desire to pull the parameters towards their own specific objectives. When numerous "players" participate, pulling parameters in different directions, this process may become unending. We propose an algorithm that leverages user embeddings to identify and group similar users with similar underlying preferences. Specifically, we employ k-means clustering within the user embedding space. The rationale is that users with proximate embeddings are likely to share common preferences. By aligning the learning process across these user groups and optimizing them sequentially, we aim to mitigate gradient interference among these disparate users.

The grouping offers several benefits: it enhances collaboration within each group, as users with similar preferences "agree" on parameter updates, facilitating modeling for less active users. It also reduces the number of "players" in the tug-of-war, leading to more stable task learning. Additionally, the algorithm can be extended to scenarios with immediate rewards, such as real-time systems where user-item interactions are streamed. In such cases, the system can learn and satisfy a group of users promptly through online learning before proceeding.

The transfer of user grouping occurs within individual tasks, so in addition to in-task transfer, we aim to maximize transfer across tasks by utilizing episodic memory $\mathcal{M}$, which is a fixed-size memory that represents all past tasks that we sample throughout the learning process by reservoir sampling [136]. We apply task alignment between current task $\mathcal{D}_t$ and episodic memory $\mathcal{M}$ to further reach greater "agreement" among users. In summary, ***U**ser **A**lignment for **C**ollaborative **F**iltering* (UACF) objective is:

$$\theta_t^* = \arg\min_{\theta_t} \mathbb{E}\left(\left(\sum_{g \in \mathbf{G}^t} \ell(G_g \mid \theta_t) - \beta \sum_{j=1}^{g-1} \frac{\partial \ell(G_g)}{\partial \theta_t} \times \frac{\partial \ell(G_j)}{\partial \theta_t} \right) - \alpha \frac{\partial \ell(\mathcal{D}_t)}{\partial \theta_t} \times \frac{\partial \ell(\mathcal{M}_t)}{\partial \theta_t} \right) \quad (4.4)$$

where $\mathbf{G}^t$ is the set of user groups at task t, obtained through k-means clustering from $\mathcal{D}_t$. The

parameters α and β are the control regularization for episodic memory and group alignment, respectively. The visualization of this concept is shown in Figure 4.2.

4.3 Experiments

The main experimental objective is to show evidence that catastrophic forgetting indeed occurs, and how the continual learning approaches address it towards balancing the learning ability and retention of previously learned tasks.

4.3.1 Experimental Settings

Datasets We experiment on different sources of dataset: three categories of Amazon Product Review Dataset[2] (*Books*, *Kindle store*, *Movies and TV*) and MovieLens Ratings Dataset[3]. These are categories where older items remain relevant for a long time. To simulate the expansion in users and items over time, we split the datasets chronologically into five tasks as illustrated in Figure 4.1a. Five tasks were chosen, as they are commonly used in popular continual learning datasets (e.g., Split-MNIST, Split-SVHN, Split-CIFAR [1, 103]). Each task was divided to introduce a roughly equal number of new users and items. Table 4.1 summarizes the task-wise splits.

Base Models Our proposed methods are model-agnostic that can plug into different collaborative filtering models. In the experiments, we adopt two base models from basic linear model MF [63] to neural networks-based, NCF [42]. These are among the major methodologies for collaborative filtering.

Comparative Methods Under investigation is the framework of dealing with the continual learning setting itself. To that extent, the most apt baselines would be those considered classical approaches to continual learning (fine-tuning, experience replay). In addition, we com-

[2]http://jmcauley.ucsd.edu/data/amazon
[3]https://grouplens.org/datasets/movielens

Table 4.1: Task-wise statistics of four datasets

Dataset	Stats	Total	Task 1	Task 2	Task 3	Task 4	Task 5
books	#ratings	1,960,674	228,399	348,435	365,906	455,571	562,363
	#users	38,121	7,624	15,061	19,721	22,796	27,916
	#items	35,736	7,147	13,153	18,487	23,719	27,593
kindle	#ratings	438,994	35,736	68,602	90,899	110,106	133,651
	#users	8,533	1,706	3,353	4,873	6,102	7,359
	#items	10,068	2,011	4,023	5,931	7,563	8,704
movies	#ratings	684,453	61,447	105,429	111,300	170,052	236,225
	#users	16,002	3,104	6,191	8,763	11,449	13,541
	#items	9,774	1,954	3,900	5,683	7,554	9,330
movielens	#ratings	1,000,209	39,962	137,431	202,555	299,256	321,005
	#users	6,040	1,198	2,408	3,610	4,820	6,007
	#items	3,706	699	1,391	2,039	2,808	3,485

pare the various adopted gradient alignment-based methods from the literature (Reptile [102], MER [118]). However, rather than simply measuring which method is better or worse, we set out to show and understand the trade-off between stability and plasticity that afflicts various models in different ways.

- *Finetuning*: standard transfer learning setting, the objective is to enrich the model with new data (Equation 4.3), favoring current data for immediate good performance. However, without revision of previous users, the data-driven model quickly forgets and provides poor recommendations for distant users.
- *Experience Replay (ER)*: common approach for continual learning [119]. It involves keeping some samples from previous tasks in a memory buffer $\mathcal{M}$ and replaying them with novel inputs, using reservoir sampling to select samples with equal probability. The objective, as shown in Equation 4.5, is to learn parameters that fit the current tasks $\mathcal{D}_t$ as well as samples from earlier tasks in $\mathcal{M}$. While this lessens forgetting, it may not completely prevent it, as the remnants of each task in $\mathcal{M}$ are relatively small. Additionally, since the model capacity is limited, paying attention to revision would affect its learning ability.

$$\theta_t^* = \arg\min_{\theta_t} \mathbb{E}_{r_{ui} \sim \mathcal{D}_t \cup \mathcal{M}} \left[\ell \left(r_{ui} \mid \theta_{t-1}^*, \theta_t \right) \right] \tag{4.5}$$

Table 4.2: Finetuning results with MovieLens dataset

(a) MSE(↓) for MF-based model.

FT	T1	T2	T3	T4	T5
T1	1.1805	-	-	-	-
T2	1.2140	1.0871	-	-	-
T3	1.2674	1.2623	1.1464	-	-
T4	1.2797	1.2749	1.2713	1.1514	-
T5	1.2781	1.2740	1.3142	1.3313	1.1726

(b) Recall(↑) for NCF-based model.

FT	T1	T2	T3	T4	T5
T1	0.7691	-	-	-	-
T2	0.0891	0.7100	-	-	-
T3	0.0137	0.0180	0.7096	-	-
T4	0.0035	0.0046	0.0022	0.7316	-
T5	0.0014	0.0011	0.0005	0.0032	0.7254

- *Reptile*: gradient alignment-based method which maximizes transfer between current task and memory buffer. The objective function is shown in Equation 4.6.

$$\theta_t^* = \arg\min_{\theta_t} \mathbb{E}\left(\ell(\mathcal{D}_t \mid \theta_{t-1}^*, \theta_t) - \alpha \frac{\partial \ell(\mathcal{D}_t)}{\partial \theta_t} \times \frac{\partial \ell(\mathcal{M}_t)}{\partial \theta_t} \right) \tag{4.6}$$

- *MER* [118]: derived gradient alignment at two levels, instance- and user-level.

Metrics Our objective is to closely monitor the base models in order to track performance changes over time. Consequently, we employ the traditional rating metric, Mean Squared Error (MSE) for MF-based models, as MF is renowned for rating prediction and optimized for MSE. Conversely, for NCF, which is optimized for ranking, we utilize the ranking-based Recall@5 metric. We follow training and test protocol described in Section 4.1 to obtain metric matrices. We report the overall matrices of metrics to observe model's behavior for each method and compare *Harmonic mean* that balances learning average (LA) and retained average (RA).

Hyper-parameters tuning. For all methods, at each task, the task's ratings are divided into train/validation/test sets. With rating prediction model, MF, we adopt proportional split in [47, 94] by ratio 60/20/20; while for NCF, we follow *leave-one-out* evaluation as original paper [42], the last item of each user is used for test and the second to last for validation. The number of user groups is fixed at 10 for all datasets. For each method, we carry out experiments with multiple (i.e., 5) random seeds and report the average over all runs.

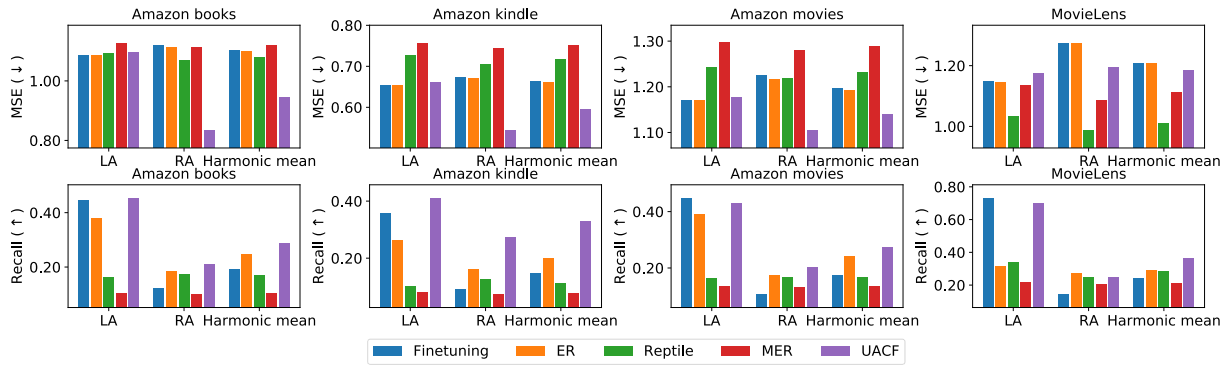


Figure 4.3: Results of comparative methods on four datasets. MSE on MF-based model is on the left while Recall for NCF is on the right. Note that for MSE, the lower is the better performance and for Recall, the higher indicate better result.

4.3.2 Results and Insights

Research Question 1: *Does the forgetting issue exist in Continual Collaborative Filtering setting?* Parameter sharing is the cause of catastrophic forgetting. Learning new tasks without proper revision of previous ones leads to forgetting of learned tasks. In MF, which is a linear model, users share parameters on representation of items; while neural networks-based NCF model, both user and item sides further share weights of layers. Intuitively, MF-based model would be affected less and NCF-based would suffer severe forgetting issue. We follow the setting in Section 4.1 and observe matrices of *MSE* performances with MF-based model and *Recall* results with NCF-based model for MovieLens dataset in Table 4.2, where each column T_i is performance of task i over time, while each row T_j is result of all the task after training task j. As it proceeds to learn new tasks, the MF model slowly performs worse, which is shown in the gradual increase of MSE. On the other hand, NCF dramatically forgets all the learned tasks, as indicated by how Recall drops to near 0. This showcases that catastrophic forgetting indeed occurs in the context of collaborative filtering.

Research Question 2: *Overall, how does UACF perform?* Further results on four datasets, Amazon books, kindle, movies, and MovieLens are shown in Figure 4.3. For MF-based model, UACF achieves similar learning average and better retained average than other methods in three

Amazon datasets out of the four.

With neural networks-based model, NCF, the results are consistent over all four datasets, where UACF acquires competitive learning ability as Finetuning and the best retention rate. Overall, UACF with NCF base performs the best on harmonic mean of LA and RA.

4.4 Discussion

We show that catastrophic forgetting indeed occurs in collaborative filtering and formulate a continual learning framework for recommendation systems. While Experience Replay stems the extent of forgetting, this comes at the cost of not being as quick to react to new observations. For a better balance, we explore gradient alignments novelly at user level. Experiments demonstrate that the proposed method, UACF, alleviates forgetting significantly while being flexible for learning new tasks, and together achieve a better balance between learning ability and retention. However, as the proposed UACF requires clustering users at each task, this requires computational overhead compared to Reptile and MER. This trade-off might be acceptable for small to medium datasets, but it may not be feasible for large-scale datasets. Future work could explore more efficient clustering methods or alternative approaches to enhance transfer and mitigate interference without clustering.

Chapter 5

Compositions of Variant Experts for Integrating Short-Term and Long-Term Preferences

Session-based recommendation systems have gained significant attention in recent years, as they reflects the dynamic nature of user preferences and sequential behaviors. These systems aim to predict the next item a user is likely to interact with, based on their previous interactions *within a session*. While session-based models emphasize the importance of short-term behavior within a single session, user preferences are not solely determined by their recent interactions; they are also influenced by their long-term preferences, which are shaped by their historical behavior over a more extended period. This chapter explores the integration of short-term and long-term preferences in session-aware recommendation systems, proposing a novel framework namely Compositions of Variant Experts (CoVE) to effectively capture and leverage these preferences.

This chapter investigates the integration of short-term and long-term user preferences in session-aware recommendation systems. In particular, it explores how different temporal aspects of user behavior can be modeled to improve the accuracy and personalization of recommendations. An illustrative example is shown in Figure 5.1, where a user with a long-standing interest

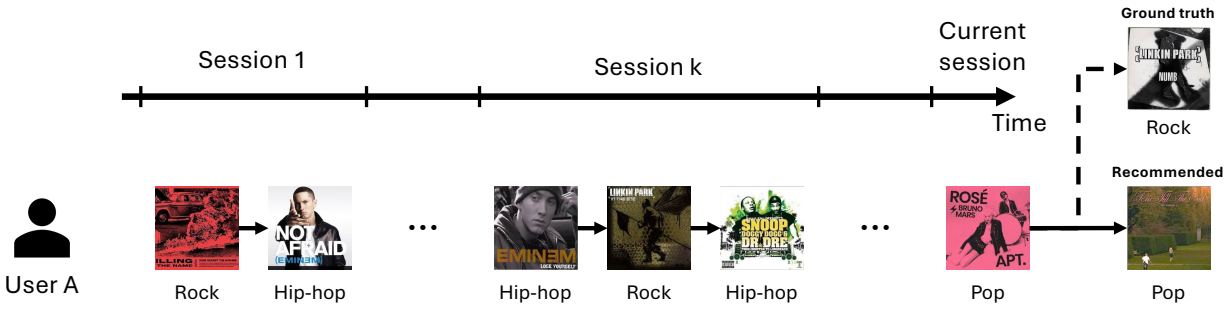


Figure 5.1: User A might want to be recommended a rock song based on their long-term interest in rock and hiphop, rather than a pop song based on their short-term interest in a trending pop song.

in rock and hip-hop music is currently listening to a trending pop track. A system that naively relies on the current session may recommend more pop music, disregarding deeper user inclinations. Conversely, ignoring the session context may result in irrelevant suggestions that fail to respond to recent interests. Understanding and effectively exploiting these preferences can lead to more accurate and personalized recommendations.

Integrating short-term and long-term preferences poses several challenges. Existing session-based recommendation models often focus on capturing short-term preferences, neglecting long-term preferences. On the other hand, traditional recommendation models that capture long-term preferences may not be suitable for session-aware recommendations due to the lack of ability to model sequential user behavior. We propose novel compositions of variant experts (CoVE) framework that can integrate both short- and long-term preferences to enhance recommendation performance by leveraging the strength of multiple types of experts working in concert.

This chapter contributions are threefold. First, we explore the existence and impact of short- versus long-term preferences in session-aware recommendation systems in section 5.1. Second, we propose the Compositions of Variant Experts (CoVE) with two main variants, which dynamically integrate short- and long-term preferences in section 5.2. Third, in section 5.3, we conduct extensive experiments to demonstrate the effectiveness of the proposed models and analyze the impact of the variant expert types, the number of experts, as well as the gating mechanism.

Table 5.1: Notations

Symbol	Description
$\mathcal{P} = \{1, \ldots, p, \ldots, N\}$	the set of N items, $N = \lvert\mathcal{P}\rvert$
$\mathcal{U} = \{1, \ldots, u, \ldots, M\}$	the set of M users, $M = \lvert\mathcal{U}\rvert$
$S_t^u = \langle p_{t,1}, p_{t,2}, \ldots, p_{t,K} \rangle$	user u's session at time period t with a total of K items
$C_{1:T}^u = \langle S_1^u, S_2^u, \ldots, S_T^u \rangle$	set of user u's sessions from time period 1 up to time period T
$\{f_1, f_i, \ldots, f_n\}$	the set of n experts
$f\left(u, C_{1:T}^u\right) \rightarrow \mathbb{R}^N$	predicted scores of N items
h_t^u	user u representation up to time t
$\Psi(p)$	item p representation
$\beta(p)$	item p bias
$g(p_{m,t})$	$n-$dimensional output of the gating function w.r.t. item input $p_{m,t}$
$g(p_{m,t})_i$	$i-$th component of $g(p_{m,t})$

5.1 Problem Formulation

Let $\mathcal{P}$ be the universal set of items, where $N = |\mathcal{P}|$ is the total number of items. Let $\mathcal{U}$ be the set of users, $M = |\mathcal{U}|$ is the total number of users. A user $u \in \mathcal{U}$ interacts with several items within a short span of time, denoted t, forming a *session* $S_t^u = \langle p_{t,1}, p_{t,2}, \ldots, p_{t,K} \rangle$, where $p_{t,k} \in \mathcal{P}, \forall k \in \{1, 2, \ldots, K\}$, K is the total number of items in session S_t^u. All of these user interactions from multiple sessions in chronological order $C_{1:T}^u = \langle S_1^u, S_2^u, \ldots, S_T^u \rangle$ reflect both user *short-term* within a session and *long-term* preferences which are items across all the sessions. We presuppose the most recent session S_T^u of $C_{1:T}^u$ is the current browsing session of user u. The main task is to predict the next probable item $p_{T,K+1}$, where K is the total number of items in the current session S_T^u.

Let f be an "expert", in this case, a model, defined as a function $f\left(u, C_{1:T}^u\right) \rightarrow \mathbb{R}^N$ that maps an input set of sessions from user u, $C_{1:T}^u$, to an $N-$dimensional vector of real values, where each value denotes the predicted ranking score for an item. In this article, we use the terms "expert" and "preference model" interchangeably.

Here, we discuss the categorization of f into two, i.e., short-term expert and long-term expert. The following properties are useful in determining to which category an expert belongs.

Property 1 (SINGLE SESSION PROPERTY) The model processes only one session at a time, without any cross-session interactions. Specifically, when provided with multiple sessions, the model considers only the most recent one, i.e., $f\left(u, C_{1:T}^u\right) = f(u, S_T^u)$.

Property 2 (USER INDEPENDENCE PROPERTY) Given an identical sequence of input items, the model, denoted as f, always produces the same output regardless of user-specific information. Formally, for any two users $u \neq u'$, $f(u, C_{i:j}^u) = f(u', C_{i':j'}^{u'}), \forall C_{i:j}^u = C_{i':j'}^{u'}, i \leq j, i' \leq j'$.

Based on these two properties, short-term and long-term preference model are formally defined as follows:

Definition 1. *(SHORT-TERM PREFERENCE MODEL) A model is considered as a short-term preference model if both Property 1 and Property 2 hold. In particular, a short-term preference model $f\left(u, C_{1:T}^u\right)$ takes as input a single sequence of items S_T^u (following Property 1) and aims to predict the next probable item $p_{T,K+1}$. The model is user independent, meaning for the same input sequence, it always produces the same output regardless of the user, i.e., $f\left(u, S_t^u\right) = f(u', S_{t'}^{u'}), \forall S_t^u = S_{t'}^{u'}, u \neq u'$.*

Definition 2. *(LONG-TERM PREFERENCE MODEL) A model is considered as a long-term preference model if either Property 1 or Property 2 does not hold. A long-term preference model $f\left(u, C_{1:T}^u\right)$ considers historical sessions $C_{1:T}^u$ or user u, or both as inputs to predict the most probable item $p_{T,K+1}$. The long-term model is user-dependent, meaning for two users u and u', $u \neq u'$, the model may produce different outputs given the same historical sessions, i.e., $\exists f(u, C) = f(u', C)$. Additionally, a model that considers more than one session is also a long-term preference model. The long-term preference model may learn user underlying preferences, concentrated into a dense representation h^u or model the inter-session relations instead of one single session.*

A short-term preference model operates solely on the interactions occurring within the current session, without incorporating any external information such as user demographics, histori-

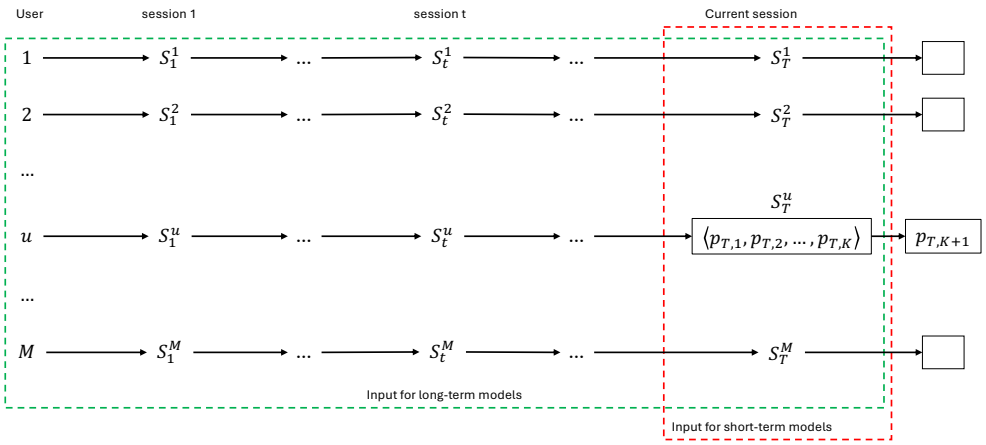


Figure 5.2: An illustrative example distinguishing inputs for long-term and short-term preference models.

cal preferences, or side data. This makes it well-suited for scenarios where long-term user data is unavailable or irrelevant. A notable example of a short-term preference model is GRU4REC [51], which utilizes Gated Recurrent Unit (GRU) [17], widely used to model sequential data such as text. GRU4REC processes a sequence of recently interacted items to generate recommendations, focusing exclusively on short-term engagement patterns rather than long-term user history. For better illustration, we highlight the inputs of short-term preference model in the red box in Figure 5.2.

A long-term preference model leverages stored user information–such as user representation or historical interactions–to enhance future recommendation predictions. It could vary from general collaborative filtering to sequential models which utilize previous sessions together with the current one. For instance, Bayesian Personalized Ranking (BPR) [114] takes a user as input to produce a personalized list of recommended items (green box in Figure 5.2). Another example instance of long-term preference model, HGRU4REC [109], uses an additional GRU layer to model all previous sessions of the same user into a condensed representation, which is combined with the current session for next-item prediction.

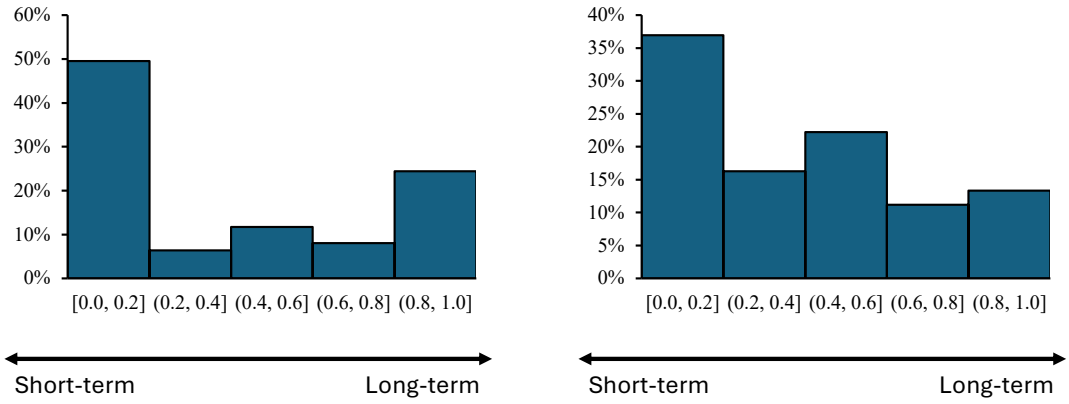
To further verify the existence of short-term and long-term preferences in session-aware rec-

ommendation systems, we conduct an empirical analysis on the public Diginetica[1] and RetailRocket[2] datasets. Here, we particularly analyze the recommendation performance of two candidate models: one that only models long-term preferences without short-term preferences, and the other vice versa. In particular, we use two representative models, GRU4REC [51], which models short-term preferences through sequential session interactions, and BPR [114], which captures long-term user preferences through user-item interactions. More details on the datasets are provided in subsection 5.3.1. In these datasets, user interactions occur in sessions, where each session consists of a sequence of item interactions within a given time window, and users may have multiple sessions over time. We hypothesize that there exist both short-term and long-term preferences in session-aware recommendations. To validate this hypothesis, we evaluate the recommendation effectiveness of BPR and GRU4REC using the same training data but formatted to suit each model's input requirements. BPR processes a user-item interaction matrix without sequential information, while GRU4REC takes a sequence of recently interacted items as input. Both models generate ranked recommendation lists as next item prediction. To determine whether an item aligns more with short-term or long-term preferences, we compare the ranking of the ground truth item in the test set. If BPR ranks the item higher than GRU4REC, we classify it as a long-term preference (bit = 1). Conversely, if GRU4REC ranks it higher, we classify it as a short-term preference (bit = 0). In cases where both models rank the item equally, we assign a value of 0.5.

The histograms in Figure 5.3 visualize the distribution of long-term bits (as a fraction) across the test datasets. While some users have extreme long-term preferences, with an average long-term bit value of 1, others do not have any long-term interest with a value of 0. The average values are 0.39 and 0.38 for Diginetica and RetailRocket datasets respectively, indicating that short-term preferences are slightly more dominant.

Through this empirical analysis, we observe that there exists both short-term and long-term

[1] https://competitions.codalab.org/competitions/11161
[2] https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset



(a) Diginetica (b) RetailRocket

Figure 5.3: Histogram of user preference values

preferences in personalized sequential recommendation, which can drive user behavior differently given their personal preferences. Effectively modeling and balancing these factors hold promise for enhancing recommendation quality, thereby improving user engagement.

5.2 Methodology

In this section, we discuss the use of variant experts for session-aware recommendations. The core idea is to leverage different types of both short-term and long-term preference models as experts, each providing unique insights into the data, that collectively combine the strengths of different experts to create a more effective recommendation model.

5.2.1 Preliminary

Here, we discuss background knowledge of well-known Mixture-of-Experts methods including continuous mixture of experts and sparse mixture of experts.

Mixture-of-Experts. (MoE) model [56] is designed to increase the model's capacity while maintaining low computational costs by using multiple "experts". This gating mechanism allocates input tokens to different experts, making each expert specified with distinct input data. There are several ways to implement the gating network $g(x)$ [18, 40, 160], but a simple and

commonly used approach is to apply a softmax function over the logits produced by a linear layer [124]:

$$g(x) := \text{Softmax}(x \cdot W_g) \tag{5.1}$$

where W_g represents the learnable parameters of the linear layer and Softmax formula is given as follows:

$$\text{Softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{5.2}$$

Continuous Mixtures of Experts. Mathematically, the continuous MoE model can be formulated as:

$$\sum_{i=1}^{n} g(x)_i f_i(x) \tag{5.3}$$

where $g(x)_i$ is the $i-$th component in the $n-$dimensional outputs, for the i-th expert $f_i(x)$. Here, each expert f_i receives the same input x but is weighted differently based on the gating output. The learnable gating function dynamically determines the contribution of each expert, assigning higher weights to more relevant experts. In this continuous MoE framework, all experts participate in the final aggregation, with their influence proportional to their assigned weights.

Sparse Mixtures of Experts (SMoE) [124]. SMoE is a variant of MoE that selects only the top-K experts based on the gating function, reducing computational costs by routing the input through a subset of experts. The SMoE gating mechanism is defined as:

$$G(x) := \text{Softmax}(\text{TopK}(x \cdot W_g)) \tag{5.4}$$

where $(\text{TopK}(l))_i := l_i$ if l_i is within the top-K values $l \in \mathbb{R}^n$ and $(\text{TopK}(l))_i := -\infty$ otherwise. This ensures that only the most relevant experts receive nonzero gating weights after applying the softmax function.

The final output of SMoE is then computed as:

$$\sum_{i=1}^{n} G(x)_i f_i(x) \tag{5.5}$$

Top-K experts selection in SMoE maintains model expressiveness while significantly improving efficiency.

5.2.2 Composition of Variant Experts via Hidden Factors (CoVE$_h$)

Here, we introduce the Composition of Variant Experts (CoVE), which differs from the traditional Mixture-of-Experts approaches in several key ways. First, instead of employing multiple instances of identical feed-forward network (FFN), CoVE leverages different types of experts. Particularly, these experts can include various recommendation models, such as collaborative filtering models (e.g., BPR [114]) for capturing user preferences and sequential models (e.g., GRU4Rec [51]) for modeling session-based interactions. Second, each expert interprets the same input data differently, offering diverse perspectives. For example, BPR treats input data as user-item interactions, while GRU4Rec models sequential behavior. Third, because each expert digests input data differently, during training, all experts are trained simultaneously to give all experts the ability to generalize; during inference, only the most relevant experts are activated. Fourth, we standardize the Compositions of Variant Experts so that each expert can determine its own input to the routing gate, which is then aggregated to produce the input to the gating mechanism. More details will be discussed in the next paragraphs.

Standardized Variant Experts. Various type of experts may produce various type of outputs from various type of inputs, providing unique insights into the data. For example, BPR learns from triplet user-positive-negative item interactions, optimizing rankings to favor positive items, providing robust user and item representations. In addition, GRU4Rec captures temporal user behaviors by predicting the next item in a sequence of interactions, relying sorely on item sequence for input and optimizing the ranking of the next item accordingly. By integrating

diverse expert models, we achieve a more comprehensive understanding of user behavior and preferences.

Despite their differences, we standardize the output of all experts including three components:

- Hidden context representation: $d-$dimensional vector that captures the user u context up to time T

$$h_i(u, T) = \Phi_i\left(u, C^u_{1:T}\right) \in \mathbb{R}^d \tag{5.6}$$

- Item embeddings: $\Psi(p) \in \mathbb{R}^d$ is a $d-$dimensional vector representation of item p
- Item bias: $\beta(p) \in \mathbb{R}$ is a scalar representing the bias of item p

Each expert serves as a sub-model. Using a gating mechanism, the three components from all experts are aggregated to form the final representations. We named this approach CoVE$_h$ as the aggregated components are hidden representation from variant of experts.

Gating Mechanism. The gating mechanism determines the relevance of each expert based on the input data, enabling the model to dynamically select the most appropriate experts for a given context. More concretely, the contribution of each expert is decided upon the model receiving the inputs via the gating mechanism. By learning the contribution of each expert, the model can adapt to diverse user behaviors and preferences, ultimately enhancing recommendation quality.

Each of the n experts determines its own $d-$dimensional input to the gating mechanism, denoted as $e_i(u, C^u_{1:T})$. These individual inputs are then aggregated to form an $n \times d-$dimensional input for the gate. This design allows for flexibility in selecting the most relevant features for the gating mechanism, tailored to each expert. For instance, given a specific input, user representations might be crucial for the gate in BPR, whereas GRU4Rec may utilize embedded input items as gate inputs. Once computed, all experts' gate inputs are aggregated and passed through the gating mechanism.

In this work, we employ the simple gate mechanism consisting of a fully connected layer

with a softmax activation function. This ensures that the gate assigns a weight to each expert while maintaining a total sum of 1. Particularly, the gating mechanism can be represented as:

$$g\left(u, C_{1:T}^{u}\right) := \text{Softmax}\left(W_g \left[e_i\left(u, C_{1:T}^{u}\right) \Big|_1^n\right]\right) \tag{5.7}$$

where $e_i\left(u, C_{1:T}^{u}\right)$ is the i-th expert's gate input, $\left[e_i\left(u, C_{1:T}^{u}\right) \Big|_1^n\right]$ denotes the concatenation of all gate inputs, and W_g is the trainable weights. The output of gate $g\left(u, C_{1:T}^{u}\right) \rightarrow \mathbb{R}^n$ is subsequently used to weigh the contributions of the experts' outputs to the final representation.

Training Variant Experts. Our CoVE framework enables each expert to process input data differently. During training, all experts are optimized simultaneously to collectively provide a comprehensive view of the data. This approach ensures that each expert learns distinct aspects of the data, contributing to the final recommendation.

The gating mechanism determines the contribution of each expert based on the input data. The final hidden representation, item embedding, and bias term are computed as weighted sums of the corresponding components from all experts, where the weights are determined by the gating function:

$$h(u) = \sum_{i=1}^{n} g\left(u, C_{1:T}^{u}\right)_i \times h_i(u, T) \tag{5.8}$$

$$\Psi(p) = \sum_{i=1}^{n} g\left(u, C_{1:T}^{u}\right)_i \times \Psi_i(p) \tag{5.9}$$

$$\beta(p) = \sum_{i=1}^{n} g\left(u, C_{1:T}^{u}\right)_i \times \beta_i(p) \tag{5.10}$$

where $g(u, C_{1:T}^{u})$ is derived from Equation 5.7, and $h_i(u, T), \Psi_i(p), \beta_i(p)$ represent the hidden context, item embedding, and item bias deriving from the $i-$th expert.

The final item score for a user u and item p is computed by taking the dot product between

the hidden representation and item embedding, and then adding the item bias:

$$\hat{r}_{u,p} = h(u) \cdot \Psi(p) + \beta(p) \tag{5.11}$$

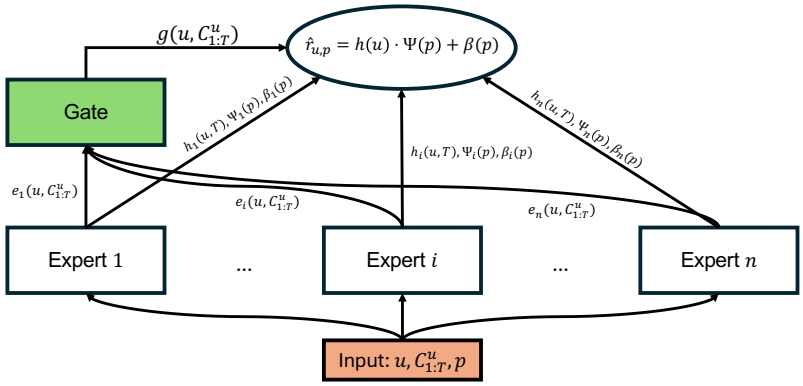


Figure 5.4: Overall flow of CoVE$_h$ variant. All experts receive user identity u with their historical interactions $C^u_{1:T}$ and candidate item p. Each expert i then returns the current context representation $h_i(u,T)$, item embeddings $\Psi_i(p)$, and item biases $\beta_i(p)$.

Learning objective. The general learning objective is to enhance ranking performance between positive and negative items in the candidate set. Its primary objective is to maximize the probability that positive items are ranked higher than negative ones. This can be achieved by minimizing the negative log-likelihood:

$$\ell = -\frac{1}{N_I} \log \sigma(\hat{r}_i - \hat{r}_j) \tag{5.12}$$

where $\hat{r}_i$ and $\hat{r}_j$ denote the predicted scores for a positive item (a preferred item) and a negative item (an unobserved or less preferred item), respectively, and N_I is the total number of item pairs. This formulation treats all negative items equally, ensuring uniform optimization across the training samples.

In practice, however, the learning dynamics differ between popular and long-tail items. Popular items are frequently sampled and updated during training, while long-tail items (those with fewer interactions) receive less attention. As a result, the prediction scores for long-tail items

tend to remain low, making them "easy negatives" that do not significantly impact the optimization process. By contrast, distinguishing between positive items and popular negative items is more challenging.

To address this imbalance, BPR-max loss incorporates the softmax score corresponding to each negative item in the candidate set so that it assigns lower weights to these easy negatives, allowing the model to focus more on harder negatives, minimizing the following loss:

$$\ell = -\log \sum_{i} \sum_{j=1}^{N_I} s_j \sigma(\hat{r}_i - \hat{r}_j) \tag{5.13}$$

where $s_j = \frac{\exp(\hat{r}_j)}{\sum_{k=1}^{N_I} \exp(\hat{r}_k)}$ is the normalization score of the negative item j via the Softmax function. By prioritizing these challenging comparisons, the model promotes more effective learning and improves its ability to rank positive items accurately across a diverse set of candidates.

Inference. In our unique framework, each expert observes input data differently, so in the training phase, we employed the continuous gate mechanism. During inference, we derive the final prediction using sparse gate mechanism, where only the most relevant experts are activated based on the input data. This routing mechanism allows the model to dynamically select the most appropriate experts for the given context, reducing the computational cost since there only k experts were selected. We entrust the routing mechanism to choose the best experts. Following Equation 5.5, the hidden representation during inference is computed as:

$$h(u) = \sum_{i=1}^{n} G(u, C^u_{1:T})_i f_i(u, C^u_{1:T}) \tag{5.14}$$

where $G(u, C^u_{1:T})$ is the sparse gate defined in Equation 5.4.

5.2.3 Composition of Variant Experts via Scoring Function (CoVE$_s$)

We propose another approach for aggregation that offers greater flexibility but shallower expert connections. Unlike CoVE$_h$ which experts are combined at *hidden* representation level, this

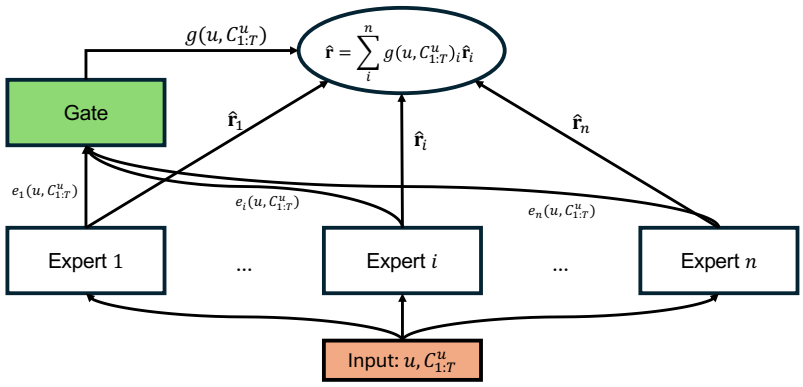


Figure 5.5: Overall flow of CoVE$_s$ variant. All experts receive user identity u with their historical interactions $C^u_{1:T}$ and candidate item p. Here, each expert i just returns its predicted scores for the set of candidate items $\hat{\mathbf{r}}_i$.

approach, referred to as CoVE$_s$, gathers item *scores* predictions from individual experts.

Specifically, each expert generates predicted item scores directly, which are subsequently combined using a weighted sum via the same routing mechanism. This method offers greater flexibility compared to the CoVE$_h$ approach, as it does not require experts to share the same hidden dimensions, provided that an expert is capable of providing scores over the item set. This approach bears resemblance to a learnable ensemble method, yet it possesses the advantage of trainable aggregation weights that can adapt dynamically to varied contexts. Consequently, the application of the same set of experts may result in varying weights, depending on the specific context.

The final ranking scores for all items are formally computed using the gating aggregation as follows:

$$\hat{\mathbf{r}} = \sum_{i=1}^{N} g\left(u, C^u_{1:T}\right)_i \hat{\mathbf{r}}_i \tag{5.15}$$

where $\hat{\mathbf{r}}_i \in \mathbb{R}^N$ denotes the scores of all items provided by the i-th expert.

5.3 Experiments

In this section, we evaluate our proposed CoVE in the task of session-aware recommendation, mainly on two publicly available benchmark datasets. The improvement in ranking performance validates the effectiveness of our solution.

5.3.1 Experimental Setup

Datasets. Recent studies have identified several widespread flaws in recommendation systems, including dataset-task mismatch and negative sampling during testing, which hinder accurate evaluation [49, 64, 129]. In this chapter, we strive to follow best practices to enhance the effectiveness of model evaluation.

To address the dataset-task mismatch, we utilize session-aware datasets[3]. In these datasets, each user interacts with multiple items within a browsing *session*, and may have several such sessions recorded. Specifically, our primary experiments are conducted on two datasets, Diginetica and RetailRocket, as detailed in section 5.1.

Furthermore, during the evaluation phase, we score and rank items across the entire item set rather than relying on candidate sampling. This approach mitigates biases introduced by the selection of candidate sets [10, 20, 64], ensuring fairer comparisons and improved reproducibility.

For statistical sufficiency, we retain users with at least three sessions and items with at least five interactions. The last two sessions of each user are reserved for validation and testing, while the remaining sessions are used for training. These two reserved sessions are randomly assigned to validation and test sets. The dataset statistics are presented in Table 5.2. Code and data for reproducibility are publicly available[4].

Evaluation metrics. To evaluate the effectiveness of each model, we let each model produce

[3] In contrast, previous studies converted user ratings into session data for evaluation. However, since these rating-based datasets were not inherently session-specific and may exhibit weaker sequential signals [129], they were excluded from this study.

[4] https://github.com/PreferredAI/CoVE.git

Table 5.2: Data statistics

	Diginetica	RetailRocket
#interactions	12,146	230,817
#users	571	4,249
#sessions	2,670	24,732
#items	6,008	36,658
#sessions per user	4.68	5.82
#interactions per item	2.02	6.30
#interactions per session	4.55	9.33
density	0.354%	0.148%

a ranked list of recommended items for every user. We quantitatively evaluate ranking performance using various well-known ranking metrics, including Area Under the ROC Curve (AUC), Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG), and Recall. For these metrics, higher values indicate better quality. Comparisons between methods are tested with one-tailed paired-sample Student's t-test at 0.05 level (i.e., p-value < 0.05).

Expert Selection. Sequential recommendation models may not achieve their full replicability and reproducibility due to flaws in implementation and hyper-parameter tuning [48, 99, 106, 125]; while simple yet effective models such as BPR [114] and GRU4Rec [50, 51] can achieve competitive results with proper training settings [23, 62, 106]. Furthermore, [67, 86] have demonstrated that shallow models can outperform deep models in various settings. In our experiments, we not only focus on deep models but also on popular and effective shallow models as experts, including two short-term preference models and two long-term preference models:

- Short-term preference models as expert:
  - **GRU4Rec** [50, 51][5]: GRU4Rec is a session-based recommendation model that employs a Gated Recurrent Unit (GRU) to capture sequential user behavior. Benchmarks indicate that GRU4Rec outperforms more recent session-based recommendation models.

[5] In our experiment, we use the improved GRU4Rec model proposed in [50].

- **SASRec++**: SASRec [61] is a session-based recommendation model that leverages self-attention mechanisms to capture user preferences. The SASRec+ [62] variant using Cross-Entropy loss can outperform SASRec with Binary Cross-Entropy loss and the allegedly superior BERT4Rec [128]. In this paper, we employ the BPR-max loss for SASRec, named SASRec++, which yields the best performance.

- Long-term preference models as expert:
  - **BPR** [114]: BPR is a collaborative filtering model that optimizes the ranking of positive items over negative ones. BPR is a simple yet effective model widely used in recommendation systems.
  - **FPMC [116]**: Factorized Personalized Markov Chains is a sequential recommendation model that captures user preferences by modeling item transitions within a session.

Comparative Methods. To sufficiently evaluate the effectiveness of our proposed method, we compare the proposed CoVE$_h$ and CoVE$_s$ models with individual selected short-term and long-term preference models as well as:

- **BERT4Rec** [128] is a session-based recommendation model that employs a Transformer architecture to capture sequential user behavior. BERT4Rec is included as a baseline due to its popularity and effectiveness in session-based recommendation.
- **LightGCN** [44] is a graph-based recommendation model that captures user-item interactions through a lightweight Graph Convolutional Network (GCN). LightGCN is included as a baseline due to its effectiveness in capturing user-item relationships. LightGCN will also be used as a long-term preference model in our investigation of CoVE$_s$ with graph-based experts.
- **HGRU4Rec** [109] is a hierarchical GRU-based method for session-aware recommendation which take long-term user preferences into account. HGRU4Rec is included as a baseline

due to its similarity in modeling both short- and long-term preferences.

Hyper-parameter Tuning. For all baselines, we perform a grid search with various hyper-parameters to find the best configuration. For CoVE$_h$ and CoVE$_s$, we also conduct a grid search to determine the optimal configuration. We use the validation set to tune the hyper-parameters to their best in term of NDCG@20 and the test set to evaluate model performance. The main hyper-parameters include:

- **Loss Function.** We consider various loss functions, including BPR [114], BPR-max [50], binary cross-entropy loss [61], cross-entropy [128], top-1 [51]. Generally, BPR and BPR-max losses perform the best based on their ability to distinguish positive from negative samples.
- **Learning Rate.** We consider learning rates in the range of $[0.005, 0.1]$.
- **Batch Size.** We consider batch sizes in the range of $[32, 512]$. Different loss functions may require different batch sizes. For example, BPR-max loss fits well with a smaller batch size of around $[32, 128]$, while BPR-loss can speed up with a larger batch size of 256 or 512 without trade-off in performance.

Experts' Gate Input. As described in Section 5.2, we give the freedom of deciding gate input to the experts. Individual experts can use their own unique features as gate input by a standardized function. In the experiments for CoVE$_h$ and CoVE$_s$, we follow common practice in NLP research [58], where the input to the gate is item embeddings.

5.3.2 Overall Performance

Here, we investigate whether using a mixture of experts enhances recommendation effectiveness compared to individual experts and baseline models.

Results reported in Table 5.3 consistently show that mixtures of experts outperform the baselines across datasets in terms of MRR, NDCG, and Recall significantly. The AUC metric does not fully reflect recommendation performance, measuring the proportion of pairwise ranking cor-

Table 5.3: Overall ranking performance: Comparison of CoVE against baselines across datasets

	Model	AUC	MRR	NDCG@k		Recall@k	
				k=10	k=20	k=10	k=20
Diginetica	GRU4Rec	<u>0.7771</u>	<u>0.2979</u>	0.3171	0.3233	0.3888	0.4133
	SASRec	0.7666	0.2943	<u>0.3301</u>	<u>0.3343</u>	<u>0.4466</u>	<u>0.4641</u>
	BERT4Rec	0.7177	0.3426	0.3560	0.3575	0.4011	0.4063
	BPR	0.6334	0.0748	0.0854	0.0961	0.1361	0.1775
	FPMC	0.6609	0.0945	0.1107	0.1205	0.1805	0.2189
	LightGCN	0.7547	0.2120	0.2488	0.2601	0.3835	0.4273
	HGRU4Rec	0.5606	0.1188	0.1333	0.1392	0.1893	0.2130
	CoVE$_s$	0.7854	**0.4112**§	**0.4379**§	**0.4416**§	**0.5254**§	**0.5394**§
	CoVE$_h$	**0.8007**§	0.3756§	0.4003§	0.4060§	0.4869§	0.5096§
	Improvement %	3.04%	38.03%	32.66%	32.10%	17.64%	16.22%
RetailRocket	GRU4Rec	<u>0.8772</u>	<u>0.2687</u>	<u>0.3035</u>	<u>0.3175</u>	<u>0.4356</u>	<u>0.4909</u>
	SASRec	0.6606	0.0962	0.1334	0.1412	0.2594	0.2892
	BERT4Rec	0.7827	0.2278	0.2551	0.2596	0.3476	0.3653
	BPR	0.8035	0.2161	0.2451	0.2528	0.3493	0.3794
	FPMC	0.8393	0.2437	0.2744	0.2848	0.3879	0.4290
	LightGCN	0.8044	0.1065	0.1238	0.1333	0.1970	0.2346
	HGRU4Rec	0.8692	0.2327	0.2612	0.2731	0.3735	0.4203
	CoVE$_s$	0.8767	0.3365§	0.3675§	0.3783§	0.4813§	0.5239§
	CoVE$_h$	**0.8798**§	**0.3406**§	**0.3732**§	**0.3831**§	**0.4928**§	**0.5319**§
	Improvement %	0.30%	26.76%	22.97%	20.66%	13.13%	8.35%

$^{\S}p$-value < 0.05. The statistical test is performed against the best-performing baseline. The best values are **bolded**. The best values among baselines are <u>underlined</u>.

rectly. CoVE$_h$ and CoVE$_s$ maintain high AUC results, close to the best reported AUC yet still statistically significant. Similarly, CoVE$_s$ also achieves the best performance across datasets. Among the selected experts, GRU4Rec remains the best-performing model overall, showing its robustness in the sequential recommendation task. The session-aware baseline HGRU4Rec does not outperform GRU4Rec, which has also been reported in some benchmark [67].

FPMC itself can be interpreted as a combination of Matrix Factorization (MF) and Factorization Markov Chains (FMC), which is similar to a composition of one long-term (MF) and one

Table 5.4: Gating effectiveness evaluation: Ranking performance of both variants of CoVE with and without gating mechanism

Dataset	Model	Gate	AUC	MRR	NDCG@k		Recall@k	
					k=10	k=20	k=10	k=20
Diginetica	CoVE$_s$	✓	**0.7854**	**0.4112**§	**0.4379**§	**0.4416**§	**0.5254**§	**0.5394**§
		–	0.7694	0.3860	0.4071	0.4119	0.4816	0.5009
	CoVE$_h$	✓	0.8007	**0.3756**	**0.4003**	**0.4060**	**0.4869**§	**0.5096**
		–	**0.8022**	0.3694	0.3939	0.3997	0.4799	0.5026
RetailRocket	CoVE$_s$	✓	**0.8767**§	**0.3365**§	**0.3675**§	**0.3783**§	**0.4813**	0.5239
		–	0.8726	0.2847	0.3279	0.3395	0.4794	**0.5251**
	CoVE$_h$	✓	0.8826	**0.3419**§	**0.3735**§	**0.3851**§	**0.4902**§	**0.5357**
		–	**0.8836**	0.3216	0.3549	0.3670	0.4782	0.5265

§ p-value < 0.05. The best values are **bolded**.

short-term (FMC) expert. However, each component contributes equally towards final optimization objective.

5.3.3 Ablation Study

In this section, we systematically study the contribution of different components to our CoVE. First, we investigate the effectiveness of the proposed gating mechanism. Then, we analyze various compositions of experts to gain insight in selecting the optimal combination of variant experts.

Gating effectiveness. To further emphasize the contribution of the proposed gating mechanism, we evaluate the performance of both variants of CoVE without the learnable weights, the gate values are always uniform values, i.e., $g(x)_i = \frac{1}{K}$, where K is the number of experts, which assuming all experts contribute equally towards the overall optimization objective. Table 5.4 reports the ranking performance of CoVE$_h$ and CoVE$_s$ with and without the learnable weights from the proposed gating mechanism. Without the flexibility of controlling how much each expert contributes, we observed a degradation in the recommendation performance for both CoVE$_s$

and CoVE$_h$ in term of ranking performance. CoVE$_h$ without the learnable gating achieves a bit higher AUC measure in both datasets, however, it is not statistically significant different from CoVE$_h$ with learnable gating. Thus, balancing the contribution among variant experts is not a trivial task.

Same-Type Experts vs. Varying-Type Experts. The results from section 5.1 indicate the presence of both short-term and long-term preferences within the session-aware recommendation context. We categorize the compositions of experts into two sub-categories: same-type experts and varying-type experts. The same-type expert mixture includes only short-term (or long-term) experts, while the varying-type expert mixture comprises both short-term and long-term experts. We further explore different compositions of variant experts, including combinations of 2, 3, and 4 experts with varying types (which include both short-term and long-term experts). The results for these compositions on the Diginetica dataset are presented in Table 5.5, while compositions for the RetailRocket dataset are shown in Table 5.6.

As demonstrated in Table 5.5, compositions of the same-type experts for CoVE$_h$ result in a performance degradation compared to individual experts, whereas compositions of varying types lead to significant improvements. For CoVE$_s$, it appears that it is easier to achieve better results compared to individual experts. The key factor is identifying the optimal expert composition; in this case, the selection of the optimal composition depends on the respecting metric.

The results in Table 5.6 show a similar pattern, where the best compositions consistently include cross-type experts (GRU4Rec+BPR) for both CoVE$_h$ and CoVE$_s$.

5.3.4 CoVE$_h$ With Graph-Based Expert

We further investigate the impact of incorporating a graph-based expert, LightGCN, as a long-term preference model within the CoVE$_h$ framework. Table 5.7 summarizes the experimental results on both Diginetica and RetailRocket datasets, comparing individual experts as well as CoVE$_h$ based on BPR model.

Table 5.5: Ranking performance with varying different experts on Diginetica dataset. GRU and SAS are GRU4Rec and SASRec++ respectively.

	Number of Experts	Expert model				Ranking metrics					
		Short-term		Long-term		AUC	MRR	NDCG@k		Recall@k	
		GRU	SAS	BPR	FPMC			k=10	k=20	k=10	k=20
CoVE$_h$	2	✓	✓			0.7287	0.1569	0.1785	0.1876	0.2627	0.2977
				✓	✓	0.7350	0.2368	0.2730	0.2781	0.3958	0.4168
		✓		✓		0.7940	0.3507	0.3720	0.3803	0.4518	0.4851
			✓	✓		0.7921	0.2380	0.2727	0.2811	0.3940	0.4273
		✓			✓	**0.8007**	**0.3756**	**0.4003**	**0.4060**	**0.4869**	**0.5096**
			✓		✓	0.6928	0.1321	0.1504	0.1562	0.2224	0.2452
	3	✓	✓		✓	0.7419	0.2189	0.2341	0.2444	0.2977	0.3380
		✓	✓	✓		0.7800	0.2871	0.3138	0.3191	0.4063	0.4273
			✓	✓	✓	0.7769	0.2377	0.2708	0.2785	0.3870	0.4168
		✓		✓	✓	0.7806	0.3355	0.3620	0.3675	0.4536	0.4746
	4	✓	✓	✓	✓	0.7744	0.3291	0.3541	0.3591	0.4413	0.4606
CoVE$_s$	2	✓	✓			0.7930	0.4036	0.4318	0.4359	0.5236	0.5394
				✓	✓	0.7423	0.2502	0.2844	0.2893	0.4046	0.4238
		✓		✓		**0.8119**	0.3674	0.3845	0.3968	0.4553	0.5044
			✓	✓		0.8059	0.3652	0.3998	0.4082	0.5166	0.5499
		✓			✓	0.8006	0.3521	0.3726	0.3801	0.4501	0.4799
			✓		✓	0.7943	0.3720	0.4045	0.4145	0.5166	**0.5569**
	3	✓	✓		✓	0.7860	**0.4098**	0.4362	0.4401	0.5219	0.5377
		✓	✓	✓		0.7833	0.4097	**0.4385**	**0.4428**	**0.5324**	0.5499
			✓	✓	✓	0.7919	0.3578	0.3980	0.4023	0.5289	0.5464
		✓		✓	✓	0.8035	0.3720	0.3897	0.4000	0.4606	0.5009
	4	✓	✓	✓	✓	0.7854	0.4112	0.4379	0.4416	0.5254	0.5394

The best values are **bolded**.

Table 5.6: Ranking performance with varying experts on RetailRocket dataset.

	Number of Experts	Expert model				Ranking metrics					
		Short-term		Long-term		AUC	MRR	NDCG@k		Recall@k	
		GRU	SAS	BPR	FPMC			k=10	k=20	k=10	k=20
CoVE$_h$	2	✓	✓			0.8782	0.3397	0.3712	0.3815	0.4869	0.5277
				✓	✓	0.7782	0.2221	0.2515	0.2595	0.3573	0.3886
		✓		✓		0.8798	**0.3406**	**0.3732**	**0.3831**	**0.4928**	**0.5319**
			✓	✓		0.8373	0.2224	0.2484	0.2593	0.3474	0.3904
		✓			✓	**0.8853**	0.3393	0.3716	0.3823	0.4898	0.5321
			✓		✓	0.8212	0.1791	0.1985	0.2082	0.2777	0.3163
	3	✓	✓		✓	0.8467	0.2942	0.3218	0.3316	0.4255	0.4639
		✓	✓	✓		0.8822	0.3337	0.3662	0.3769	0.4853	0.5277
			✓	✓	✓	0.7842	0.1600	0.1800	0.1880	0.2570	0.2885
		✓		✓	✓	0.8680	0.3207	0.3537	0.3644	0.4740	0.5159
	4	✓	✓	✓	✓	0.8644	0.3126	0.3439	0.3545	0.4582	0.5001
CoVE$_s$	2	✓	✓			0.8767	0.3365	0.3675	0.3783	0.4813	0.5239
				✓	✓	0.8150	0.2023	0.2338	0.2440	0.3490	0.3888
		✓		✓		**0.8898**	**0.3354**	**0.3694**	**0.3814**	**0.4952**	**0.5425**
			✓	✓		0.8283	0.1820	0.2404	0.2524	0.4354	0.4825
		✓			✓	0.8877	0.3087	0.3451	0.3577	0.4792	0.5288
			✓		✓	0.8269	0.1489	0.2036	0.2179	0.3921	0.4481
	3	✓	✓		✓	0.8038	0.2593	0.2824	0.2949	0.3742	0.4239
		✓	✓	✓		0.8235	0.2731	0.3070	0.3185	0.4297	0.4749
			✓	✓	✓	0.8129	0.1560	0.2129	0.2261	0.4048	0.4566
		✓		✓	✓	0.8828	0.3199	0.3546	0.3663	0.4815	0.5274
	4	✓	✓	✓	✓	0.8840	0.2379	0.2931	0.3067	0.4841	0.5382

The best values are **bolded**.

Table 5.7: CoVE$_h$ experimental results with graph-based expert

	Number of Experts	Expert model				Ranking metrics					
		Short-term		Long-term		AUC	MRR	NDCG@k		Recall@k	
		GRU	SAS	BPR	LightGCN			k=10	k=20	k=10	k=20
Diginetica	1	✓				0.7771	0.2979	0.3171	0.3233	0.3888	0.4133
			✓			0.7666	0.2943	0.3301	0.3343	0.4466	0.4641
				✓		0.6334	0.0748	0.0854	0.0961	0.1361	0.1775
					✓	0.7547	0.2120	0.2488	0.2601	0.3835	0.4273
	2	✓		✓		<u>0.7940</u>	**0.3507**	**0.3720**	**0.3803**	<u>0.4518</u>	<u>0.4851</u>
			✓	✓		0.7921	0.2380	0.2727	0.2811	0.3940	0.4273
		✓			✓	**0.8139**	0.2870	0.3156	0.3245	0.4203	0.4553
			✓		✓	0.7829	<u>0.3036</u>	<u>0.3429</u>	<u>0.3489</u>	**0.4711**	**0.4956**
RetailRocket	1	✓				<u>0.8772</u>	0.2687	0.3035	0.3175	0.4356	<u>0.4909</u>
			✓			0.6606	0.0962	0.1334	0.1412	0.2594	0.2892
				✓		0.8035	0.2161	0.2451	0.2528	0.3493	0.3794
					✓	0.8044	0.1065	0.1238	0.1333	0.1970	0.2346
	2	✓		✓		**0.8798**	**0.3406**	**0.3732**	**0.3831**	**0.4928**	**0.5319**
			✓	✓		0.8373	0.2224	0.2484	0.2593	0.3474	0.3904
		✓			✓	0.8344	0.2386	0.2640	0.2752	0.3634	0.4079
			✓		✓	0.8627	<u>0.3027</u>	<u>0.3318</u>	<u>0.3432</u>	<u>0.4425</u>	0.4872

The best values are **bolded** and second best values are <u>underlined</u>.

On the Diginetica dataset, integrating LightGCN with short-term experts leads to notable improvements. Specifically, the combination of GRU4Rec and LightGCN achieves the highest AUC score (0.8139), outperforming all other configurations. Additionally, the pairing of SASRec with LightGCN achieves the best Recall@10 (0.4711) and Recall@20 (0.4956). These results demonstrate that LightGCN as the graph-based long-term expert can complement short-term models and enhance overall recommendation.

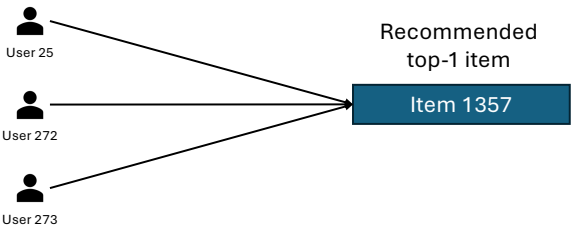
In contrast, on the RetailRocket dataset, LightGCN as an individual expert does not perform as well as BPR. Consequently, combinations involving LightGCN do not outperform those with BPR in most metrics. However, the combination of SASREC and LightGCN still achieves competitive results, ranking second-best in several Recall and NDCG metrics, despite SASREC and LightGCN performing poorly individually. This suggests that even when individual experts are weak, their combination can provide complementary benefits.

Overall, these findings highlight the potential of integrating graph-based long-term experts, such as LightGCN, within the COVE$_h$ framework, particularly when paired with strong short-term models. This integration can lead to improved recommendation performance, similar to the results achieved with long-term preference models like BPR.

5.3.5 Case Study

We investigate whether the same items can be recommended based on variant experts. For example, an item aligned with one user's long-term interests might simultaneously be recommended due to another user's short-term preferences.

In the Diginetica dataset (see Figure 5.6), item 1357 serves as a positive example, correctly recommended to three users: indices 25, 272, and 273. Interestingly, the underlying contributions for recommending item 1357 vary across these users. For user 25, the gate value contributions are (0.0138, 0.2435, 0.0262, 0.7166) for GRU4Rec, SASRec, BPR, and FPMC, respectively. For user 272, the gate values are (0.0163, 0.5362, 0.0264, 0.4212). For user 273, they are (0.0258,



	Gate values			
	GRU4Rec	SASRec	BPR	FPMC
User 25	0.0138	0.2435	0.0262	**0.7166**
User 272	0.0163	**0.5362**	0.0264	0.4212
User 273	0.0258	**0.5925**	0.0405	0.3411

Figure 5.6: An example of different gate values for different users being recommended the same item

0.5925, 0.0405, 0.3411).

For users 272 and 273, SASRec emerges as the dominant expert recommending item 1357, with slightly different weightings, suggesting that item 1357 aligns with their short-term preferences. In contrast, for user 25, FPMC is the primary contributor, indicating that item 1357 likely reflects this user's long-term preference.

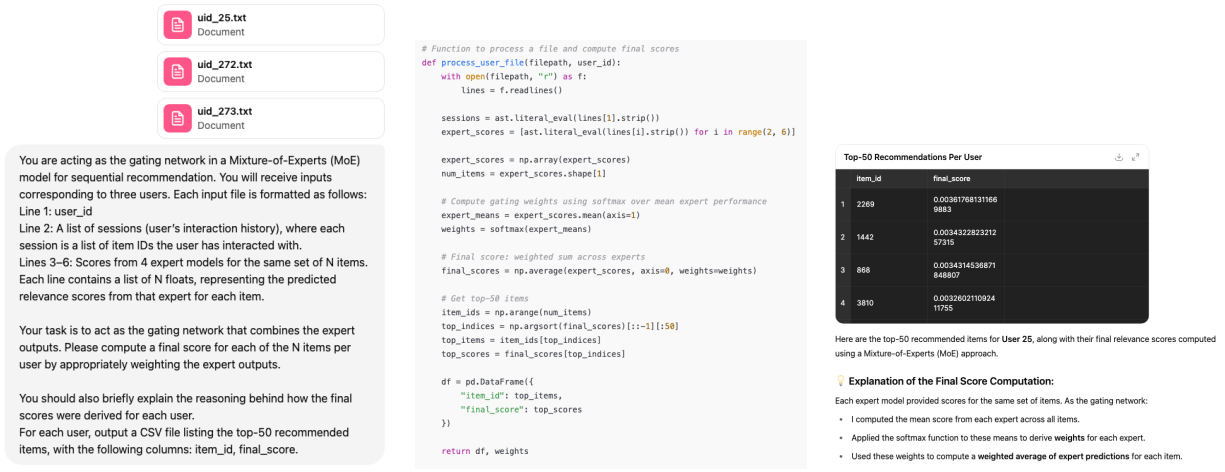
5.3.6 LLM as Gating Mechanism

In this section, we explore the potential of using Large Language Models (LLMs) as a gating mechanism for CoVE. The objective is to study if LLMs can dynamically select experts based on user and item information, potentially improving recommendation performance.

We ask ChatGPT to give a list of top-50 items based on the scores from the pretrained experts, all user historical interactions[6]. Figure 5.7 shows the prompt we used to query ChatGPT-4o. The LLM is expected to analyze the scores from each expert and the user's historical interactions to make its recommendations.

Response from ChatGPT-4o is shown in Figure 5.7b and Figure 5.7c, where it computes the average score assigned by each expert and ranks the items accordingly. The top-50 items are then presented as a recommendation list. This corresponds to the ablation study of gating mechanism

[6] https://chatgpt.com/share/68511ca8-605c-8012-ac49-2a4aba7f8f66



(a) Prompt to ChatGPT-4o (b) ChatGPT's generated code (c) Top-50 items and reasoning

Figure 5.7: ChatGPT-4o as a gating mechanism for CoVE.

discussed in subsection 5.3.3. The results suggest that, at present, ChatGPT does not provide additional benefits in weighing the experts when acting independently. This leaves opportunities for further exploration in future research.

5.3.7 Discussion

In this section, we further discuss alternative ways to scale up our proposed CoVE using a larger model (such as increasing the number of experts) or a larger dataset. In addition, we enumerate a few limitations and potential direction for future improvements.

Increasing the number of experts by multiplying the same set of experts. Here we try another approach to increase the number of experts in CoVE, by multiplying the same set of experts multiple times. As reported in Table 5.8, it is not clear that increasing the number of the same experts multiple times helps in enhancing recommendation performance. However, this approach also increases the computational cost of training due to the larger number of experts being used. This further emphasizes that simply multiplying the same experts is not as effective as more diverse experts.

Table 5.8: Performance when increasing the number of experts: AUC and MRR

Multiply	Diginetica				RetailRocket			
	CoVE$_h$		CoVE$_s$		CoVE$_h$		CoVE$_s$	
	AUC	MRR	AUC	MRR	AUC	MRR	AUC	MRR
$\times 1$	0.8007	0.3756	0.7854	0.4112	0.8798	0.3406	0.8767	0.3365
$\times 2$	0.7982	0.3731	0.7964	0.4097	0.8826	0.3419	0.8755	0.3374
$\times 4$	0.7984	0.3842	0.7963	0.4125	0.8797	0.3406	0.8738	0.3383

Scale to larger dataset. Here we would like to assess whether the overall performance of the proposed CoVE is still maintained on a larger scale of data. For this experiment, we use Cosmetics dataset[7], containing more than 2.5 millions interactions (approximately 200 times bigger than Diginetica and 10 times bigger than RetailRocket dataset) with a total of $17,268$ users, $42,367$ items, and $172,242$ sessions. Due to the larger number of users and items, the model size of each individual expert increases, taking much more time to train, especially with the continuous gating. We can get rid of this problem as we perform offline training. During inference, we use top$k$ experts, which is a more efficient approach. Specifically, as reported in Table 5.9, the best performance model of CoVE is achieved using top-1 expert in inference. Overall, Both CoVE variants achieve significant better performance over the selected baselines in term of MRR, NDCG@k, and Recall@k. Although FPMC performs the best in term of AUC, its performance on MRR, NDCG@k, and Recall@k metrics are significantly worse than CoVE. This may be due to the main pairwise ranking objective, which is technically to optimize AUC.

CoVE$_h$ and CoVE$_s$ choices. The two proposed CoVE frameworks offer different levels of combining experts. CoVE$_h$ combines experts at the hidden layer, allowing for more complex interactions between expert outputs. This also limits the particular experts that can be used, as they must have the same output hidden dimension. In contrast, CoVE$_s$ combines experts at the output layer, which is simpler and more interpretable. We can utilize any experts with different output hidden dimensions, as long as they can output a score for the set of items. The gating

[7] https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop

Table 5.9: Performance on Cosmetics dataset with > 2.5 millions interactions

Model	AUC	MRR	NDCG@k		Recall@k	
			k=10	k=20	k=10	k=20
BPR	0.8054	0.0175	0.0189	0.0250	0.0396	0.0638
FPMC	**0.9248**	0.0985	0.1152	0.1319	0.2022	0.2684
GRU4REC	0.9147	0.1568	0.1842	0.2042	0.3069	0.3857
SASREC	0.9104	0.1568	0.1823	0.1999	0.2967	0.3661
HGRU4REC	0.8948	0.0764	0.0878	0.1020	0.1546	0.2113
COVE$_s$	0.9167§	0.1643§	0.1937§	0.2135§	0.3224§	0.4007§
COVE$_h$	0.9182§	**0.1647**§	**0.1940**§	**0.2139**§	**0.3229**§	**0.4016**§
Improvement %	-0.71%	5.04%	5.32%	4.75%	5.21%	4.12%

$^{\S}p$-value < 0.05. The best values are **bolded**. The best values among baselines are underlined. The statistical test is performed against the best-performing baseline.

mechanism in COVE$_s$ is also simpler, as it only needs to combine the output scores of the experts rather than their hidden states. However, this simplicity comes at the cost of potentially losing some complex interactions between expert outputs that COVE$_h$ can capture. The choice between these two variants depends on the specific use case and the desired complexity of the model. For example, COVE$_h$ may be more suitable for tasks requiring complex interactions between expert outputs, while COVE$_s$ may be preferred for tasks where interpretability is crucial.

Running time analysis. COVE models are mixtures of all individual experts, so the overall training time is approximately the sum of all experts' training time, overhead from the gating mechanism, and the time saved by requiring only a single backward pass for the entire model instead of separate backward passes for individual experts.

Table 5.10 reports the training time for the two COVE variants that incorporate all four experts (BPR, FPMC, GRU4REC, and SASREC), along with the training time for each expert on Diginetica dataset. All models were trained for 200 epochs with a batch size of 32, and running time was measured in seconds. Among the individual experts, BPR required the least time to train (7.5 minutes), while FPMC required the most (19.1 minutes). GRU4REC and SASREC took similar training times (11.3 and 11.8 minutes, respectively). The total training

Table 5.10: Training time for Diginetica dataset, same batch size (32) and number of epochs (200) for all models.

Model	BPR	FPMC	GRU4REC	SASREC	Expert total	CoVE$_h$	CoVE$_s$
Training time (s)	449.256	1146.133	676.165	705.373	2976.927	4677.913	3770.832

time for all four experts was 49.6 minutes. In comparison, CoVE$_h$ took 78 minutes, and CoVE$_s$ took 62.8 minutes, corresponding to approximately 1.6 and 1.3 times the combined training time of the individual experts, respectively. This increase is expected, as CoVE models incur additional computational costs from the gating mechanism and the aggregation of expert outputs.

In practice, the use of pretrained individual experts can substantially reduce the training time for CoVE variants. When pretrained experts are available, they can be directly incorporated into CoVE without training from scratch, enabling much faster convergence of the CoVE models.

Limitations and potential improvements. Although both of CoVE variants achieve consistently better performance, some limitations persist in the chosen approaches. Firstly, CoVE uses continuous mixture-of-experts training, where every single expert is activated. This may result in a longer and inefficient training process and difficulties in scalability. However, due to the uniqueness of how each expert views input data, continuous training seems necessary for the experts to capture complete user behaviors. Secondly, various experts may converge differently, which the proposed methods did not tackle and could lead to overfitting for individual experts when the overall objective is achieved. The Composition of Variant Experts framework is generalized and flexible in incorporating any type of experts; however, in this work, expert selection mainly focuses on a few well-known approaches. Furthermore, we presuppose the output hidden dimension of each expert in the proposed CoVE$_h$ variant to be the same. This limits the flexibility of each individual expert, as the best-performing parameters for one expert may differ from another in practice. Methods of combining experts with different hidden output dimensions could be considered.

5.4 Discussion

While users typically have a long-term preference profile built up over a long period of time, there are short-term effects that may affect preferences as well. In this work, we establish that different types of 'experts' that focus on either short- or long-term preference would work better together, as not only do both types of preferences matter, but the way they do so differs from user to user. Which model predominates also varies across datasets, emphasizing the proposed framework approach of CoVE, leveraging on multiple experts working in concert, while accommodating a gating mechanism that modulates the influence of each expert on recommendations.

Chapter 6

Discussion and Future Work

6.1 Summary of Contributions

This dissertation advances the modeling of multiple tasks within recommendation systems, moving beyond the limitations of traditional offline supervised learning. It investigate how "models" can be used effectively to address the complex and dynamic "tasks" in recommendation platforms. Multiple novel approaches have been presented, each designed to enhance the learning process within these frameworks

First, the dissertation addresses the "one model, multiple domains" problem through dual-target cross-domain recommendation, where two related tasks arise from distinct domains. The NO3 setting is introduced, characterized by no user overlap, no item overlap, and no side information. A multi-task learning framework is proposed, enabling the model to learn both domains simultaneously. Moreover, a methodology to "tie" similar users is presented, encouraging close representations and leveraging mediated latent user preferences to improve recommendation quality across both domains.

Second, the dissertation explores "one model, sequential tasks" in continual learning for recommendation systems. This framework focuses on how a recommender system can adapt to the arrival of new users, items, and interactions over time as a sequence of distinct tasks. A stan-

dardized framework for the training, testing, and evaluation of continual collaborative filtering is introduced. Additionally, a model-agnostic approach is proposed, which can seamlessly integrate with any traditional recommendation model. This approach aims to maximize transfer and mitigate interference among multiple tasks.

Third, the dissertation investigates "multiple models, one task" problem by integrating multiple specialized experts to learn complex user preferences within a single recommendation framework, allowing for the simultaneous modeling of short-term and long-term preferences. This is achieved through a mixture-of-experts framework, where each expert specializes in either short-term or long-term preferences. The proposed method is shown to outperform state-of-the-art models on various datasets, demonstrating its effectiveness in capturing user preferences.

6.2 Future Research Directions

For future research, several directions which involve "models" and "tasks" can be explored.

The rapid advancements in computer vision and natural language processing have been inspired by the success of transfer learning. Foundational studies and pretrained models (e.g., ResNet, EfficientNet in vision, and Transformer-based models in natural language processing) have demonstrated how knowledge learned from large-scale datasets can be transfered to a wide range of downstream tasks. These models provide a robust starting point, enabling researchers to quickly build upon generalized representations. For instance, in computer vision, models like ResNet and EfficientNet have been pretrained on large datasets such as ImageNet, allowing them to generalize well and extract general features from images into dense representations. These representations can then be fine-tuned for specific tasks, such as object detection or image segmentation, significantly reducing the need for extensive labeled data for each task. This "one model, any task" relationship has been well established in these domains. In recommendation systems, however, the development of such generalized, transferable models remains in its infancy. While many studies have leveraged pretrained language models, including LLMs, for

text-based recommendation, the field still lacks a powerful pretrained model that can serve as a foundation for general recommendation tasks. Future research could focus on developing such models, enabling transfer learning in this domain much like ResNet and EfficientNet in computer vision.

MoE-based recommendation frameworks have shown promise in capturing complex user preferences by integrating multiple experts. However, current approaches primarily focus on multi-task learning setting, where each expert is specialized in a different task. Future research could explore the potential of MoE-based frameworks in (i) continual learning settings, enabling the model to learn user long-term and task-specific preferences, and (ii) multi-interest recommendation, where the model can capture multiple user interests simultaneously.

Bibliography

[1] Abati, D.; Tomczak, J.; Blankevoort, T.; Calderara, S.; Cucchiara, R.; and Bejnordi, B. E. 2020. Conditional channel gated networks for task-aware continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3931–3940. 4.3.1

[2] Afsar, M. M.; Crump, T.; and Far, B. 2022. Reinforcement learning based recommender systems: A survey. *ACM Computing Surveys* 55(7):1–38. 2.1.4

[3] Altschuler, J.; Niles-Weed, J.; and Rigollet, P. 2017. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *Advances in neural information processing systems* 30. 3.2.2

[4] Bai, T.; Xiao, Y.; Wu, B.; Yang, G.; Yu, H.; and Nie, J.-Y. 2022. A contrastive sharing model for multi-task recommendation. In *Proceedings of the ACM Web Conference 2022*, 3239–3247. 2.1

[5] Ben-David, S.; Blitzer, J.; Crammer, K.; and Pereira, F. 2006. Analysis of representations for domain adaptation. *Advances in neural information processing systems* 19. 1

[6] Berkovsky, S.; Kuflik, T.; and Ricci, F. 2007. Cross-domain mediation in collaborative filtering. In *User Modeling 2007: 11th International Conference, UM 2007, Corfu, Greece, July 25-29, 2007. Proceedings 11*, 355–359. Springer. 2.3.1, 2.3.2

[7] Bhagat, R.; Muralidharan, S.; Lobzhanidze, A.; and Vishwanath, S. 2018. Buy it again: Modeling repeat purchase recommendations. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 62–70. 2.1.3

[8] Bostandjiev, S.; O'Donovan, J.; and Höllerer, T. 2012. Tasteweights: a visual interactive hybrid recommender system. In *Proceedings of the sixth ACM conference on Recommender systems*, 35–42. 2.1.4

[9] Burke, R. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12:331–370. 2.1.4

[10] Cañamares, R., and Castells, P. 2020. On target item sampling in offline recommender system evaluation. In *Proceedings of the 14th ACM Conference on Recommender Systems*, 259–268. 5.3.1

[11] Chen, X.; Chen, H.; Xu, H.; Zhang, Y.; Cao, Y.; Qin, Z.; and Zha, H. 2019a. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, 765–

–774. New York, NY, USA: Association for Computing Machinery. 2.1.4

[12] Chen, Z.; Wang, X.; Xie, X.; Wu, T.; Bu, G.; Wang, Y.; and Chen, E. 2019b. Co-attentive multi-task learning for explainable recommendation. In *IJCAI*, volume 2019, 2137–2143. 2.2

[13] Chen, L.; Wu, L.; Hong, R.; Zhang, K.; and Wang, M. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 27–34. 2.1.1

[14] Chen, L.; Gao, C.; Du, X.; Luo, H.; Jin, D.; Li, Y.; and Wang, M. 2024a. Enhancing id-based recommendation with large language models. *ACM Trans. Inf. Syst.* Just Accepted. 2.1.4

[15] Chen, X.; Wang, S.; McAuley, J.; Jannach, D.; and Yao, L. 2024b. On the opportunities and challenges of offline reinforcement learning for recommender systems. *ACM Transactions on Information Systems* 42(6):1–26. 2.1.4

[16] Chen, W.; Hsu, W.; and Lee, M. L. 2013. Making recommendations from multiple domains. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 892–900. 2.3.1, 2.3.2, 2.3.3

[17] Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. 5.1

[18] Clark, A.; de Las Casas, D.; Guy, A.; Mensch, A.; Paganini, M.; Hoffmann, J.; Damoc, B.; Hechtman, B.; Cai, T.; Borgeaud, S.; et al. 2022. Unified scaling laws for routed language models. In *International conference on machine learning*, 4057–4086. PMLR. 5.2.1

[19] Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems* 26. 3.2.2

[20] Dallmann, A.; Zoller, D.; and Hotho, A. 2021. A case study on sampling strategies for evaluating neural sequential item recommendation models. In *Proceedings of the 15th ACM Conference on Recommender Systems*, 505–514. 5.3.1

[21] De Campos, L. M.; Fernández-Luna, J. M.; Huete, J. F.; and Rueda-Morales, M. A. 2010. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International journal of approximate reasoning* 51(7):785–799. 2.1.4

[22] De Gemmis, M.; Lops, P.; Semeraro, G.; and Basile, P. 2008. Integrating tags in a semantic content-based recommender. In *Proceedings of the 2008 ACM conference on Recommender systems*, 163–170. 2.1, 2.1.2

[23] de Souza Pereira Moreira, G.; Rabhi, S.; Lee, J. M.; Ak, R.; and Oldridge, E. 2021. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM conference on recommender systems*, 143–153. 5.3.1

[24] Do, J. H., and Lauw, H. W. 2023. Continual collaborative filtering through gradient alignment. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1133–1138. 1, 2.1

[25] Do, J. H., and Lauw, H. W. 2025. Dual-target disjointed cross-domain recommendation mediated via latent user preferences. *ACM SIGKDD Explorations Newsletter* 27(1):52–61. 2.1

[26] Do, J. H.; Le, T.-H.; and Lauw, H. W. 2025. Compositions of variant experts for integrating short-term and long-term preferences. 2.1

[27] Dredze, M., and Crammer, K. 2008. Online methods for multi-domain learning and adaptation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 689–697. 1

[28] Elkahky, A. M.; Song, Y.; and He, X. 2015. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th international conference on world wide web*, 278–288. 2.3.1

[29] Fang, Z.; Gao, S.; Li, B.; Li, J.; and Liao, J. 2015. Cross-domain recommendation via tag matrix transfer. In *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, 1235–1240. IEEE. 2.3.1

[30] Fang, H.; Zhang, D.; Shu, Y.; and Guo, G. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* 39(1):1–42. 2.1.4

[31] Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, 1126–1135. PMLR. 1

[32] Finn, C.; Rajeswaran, A.; Kakade, S.; and Levine, S. 2019. Online meta-learning. In *International conference on machine learning*, 1920–1930. PMLR. 1

[33] Fu, W.; Peng, Z.; Wang, S.; Xu, Y.; and Li, J. 2019. Deeply fusing reviews and contents for cold start users in cross-domain recommendation systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 94–101. 2.3.1

[34] Gao, S.; Luo, H.; Chen, D.; Li, S.; Gallinari, P.; and Guo, J. 2013. Cross-domain recommendation via cluster-level latent factor model. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2013, Prague, Czech Republic, September 23-27, 2013, Proceedings, Part II 13*, 161–176. Springer. 2.1, 2.3.1, 2.3.2

[35] Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; and Zhang, Y. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM conference on recommender systems*, 299–315. 2.1.4

[36] Goodfellow, I. J.; Mirza, M.; Xiao, D.; Courville, A.; and Bengio, Y. 2013. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*. 1, 4.1

[37] Gunawardana, A., and Meek, C. 2009. A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, 117–124. 2.1.4

[38] Guo, L.; Tang, L.; Chen, T.; Zhu, L.; Nguyen, Q. V. H.; and Yin, H. 2021. Da-gcn: A domain-aware attentive graph convolution network for shared-account cross-domain sequential recommendation. *arXiv preprint arXiv:2105.03300*. 2.3.2

[39] Hadash, G.; Shalom, O. S.; and Osadchy, R. 2018. Rank and rate: multi-task learning for recommender systems. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 451–454. 2.1, 2.2

[40] Hazimeh, H.; Zhao, Z.; Chowdhery, A.; Sathiamoorthy, M.; Chen, Y.; Mazumder, R.; Hong, L.; and Chi, E. 2021. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems* 34:29335–29347. 5.2.1

[41] He, X.; Chen, T.; Kan, M.-Y.; and Chen, X. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, 1661–1670. New York, NY, USA: Association for Computing Machinery. 2.1.1

[42] He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; and Chua, T.-S. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, 173–182. 2.1, 2.1.1, 3.3, 4.3.1, 4.3.1

[43] He, M.; Zhang, J.; Yang, P.; and Yao, K. 2018. Robust transfer learning for cross-domain collaborative filtering using multiple rating patterns approximation. In *Proceedings of the eleventh ACM international conference on web search and data mining*, 225–233. 2.3.1

[44] He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 2020. 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*, 639–648. 2.1, 2.1.1, 5.3.1

[45] He, Y.; Feng, X.; Cheng, C.; Ji, G.; Guo, Y.; and Caverlee, J. 2022. Metabalance: improving multi-task recommendations via adapting gradient magnitudes of auxiliary tasks. In *Proceedings of the ACM Web Conference 2022*, 2205–2215. 2.1, 2.2

[46] He, R.; Kang, W.-C.; and McAuley, J. 2017. Translation-based recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, 161–169. New York, NY, USA: Association for Computing Machinery. 2.1.3

[47] Herlocker, J. L.; Konstan, J. A.; Borchers, A.; and Riedl, J. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, 230–237. 3.3, 4.3.1

[48] Hidasi, B., and Czapp, Á. T. 2023a. The effect of third party implementations on reproducibility. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 272–282. 5.3.1

[49] Hidasi, B., and Czapp, Á. T. 2023b. Widespread flaws in offline evaluation of recommender systems. In *Proceedings of the 17th acm conference on recommender systems*, 848–855. 5.3.1

[50] Hidasi, B., and Karatzoglou, A. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, 843–852. 2.1.3, 5.3.1, 5

[51] Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based recommendations with recurrent neural networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. 5.1, 5.2.2, 5.3.1

[52] Hoi, S. C.; Sahoo, D.; Lu, J.; and Zhao, P. 2021. Online learning: A comprehensive survey. *Neurocomputing* 459:249–289. 1

[53] Hou, Y.; He, Z.; McAuley, J.; and Zhao, W. X. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, 1162–1171. 2.1.4

[54] Hu, L.; Cao, J.; Xu, G.; Cao, L.; Gu, Z.; and Zhu, C. 2013. Personalized recommendation via cross-domain triadic factorization. In *Proceedings of the 22nd international conference on World Wide Web*, 595–606. 2.3.2

[55] Hu, Y.; Koren, Y.; and Volinsky, C. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*, 263–272. Ieee. 2.1.1

[56] Jacobs, R. A.; Jordan, M. I.; Nowlan, S. J.; and Hinton, G. E. 1991. Adaptive mixtures of local experts. *Neural computation* 3(1):79–87. 5.2.1

[57] Jendal, T. E.; Le, T.-H.; Lauw, H. W.; Lissandrini, M.; Dolog, P.; and Hose, K. 2024. Hypergraphs with attention on reviews for explainable recommendation. In Goharian, N.; Tonellotto, N.; He, Y.; Lipani, A.; McDonald, G.; Macdonald, C.; and Ounis, I., eds., *Advances in Information Retrieval*, 230–246. Cham: Springer Nature Switzerland. 2.1.1

[58] Jiang, A. Q.; Sablayrolles, A.; Roux, A.; Mensch, A.; Savary, B.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Hanna, E. B.; Bressand, F.; et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*. 5.3.1

[59] Jin, J.; Chen, X.; Zhang, W.; Chen, Y.; Jiang, Z.; Zhu, Z.; Su, Z.; and Yu, Y. 2022. Multi-scale user behavior network for entire space multi-task learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 874–883. 2.1

[60] Joshi, M.; Dredze, M.; Cohen, W.; and Rose, C. 2012. Multi-domain learning: when do domains matter? In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 1302–1312. 1

[61] Kang, W.-C., and McAuley, J. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, 197–206. IEEE. 2.1.3, 2.1.4, 5.3.1

[62] Klenitskiy, A., and Vasilev, A. 2023. Turning dross into gold loss: is bert4rec really better than sasrec? In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1120–1125. 5.3.1

[63] Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37. 2.1, 2.1.1, 3.3, 4.3.1

[64] Krichene, W., and Rendle, S. 2020. On sampled metrics for item recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 1748–1757. 5.3.1

[65] Kuhn, H. W. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly* 2(1-2):83–97. 3.2.1

[66] Kumar, A.; Kumar, N.; Hussain, M.; Chaudhury, S.; and Agarwal, S. 2014. Semantic clustering-based cross-domain recommendation. In *2014 IEEE Symposium on Computational*

Intelligence and Data Mining (CIDM), 137–141. IEEE. 2.3.3

[67] Latifi, S.; Mauro, N.; and Jannach, D. 2021. Session-aware recommendation: A surprising quest for the state-of-the-art. *Information Sciences* 573:291–315. 5.3.1, 5.3.2

[68] Le, T.-H., and Lauw, H. W. 2021. Explainable recommendation with comparative constraints on product aspects. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 967–975. 2.1, 2.2

[69] Le, T.-H., and Lauw, H. W. 2024. Question-attentive review-level explanation for neural rating regression. *ACM Trans. Intell. Syst. Technol.* 15(6). 2.1.4

[70] Le, D.-T.; Lauw, H. W.; and Fang, Y. 2017. Basket-sensitive personalized item recommendation. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 2060–2066. AAAI Press. 2.1.3

[71] LE, D. T.; LAUW, H. W.; and FANG, Y. 2018. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18): Stockholm, Sweden, July*, 13–19. 2.1.3

[72] Lee, D., and Seung, H. S. 2000. Algorithms for non-negative matrix factorization. *Advances in neural information processing systems* 13. 3.3

[73] Li, P., and Tuzhilin, A. 2020. Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, 331–339. 2.1, 2.3.1

[74] Li, H.; Wang, Y.; Lyu, Z.; and Shi, J. 2020. Multi-task learning for recommendation over heterogeneous information network. *IEEE Transactions on Knowledge and Data Engineering* 34(2):789–802. 2.1, 2.2

[75] Li, X.; Chin, J. Y.; Chen, Y.; and Cong, G. 2021. Sinkhorn collaborative filtering. In *Proceedings of the web conference 2021*, 582–592. 3.3

[76] Li, M.; Jullien, S.; Ariannezhad, M.; and de Rijke, M. 2023a. A next basket recommendation reality check. *ACM Trans. Inf. Syst.* 41(4). 2.1.3

[77] Li, R.; Deng, W.; Cheng, Y.; Yuan, Z.; Zhang, J.; and Yuan, F. 2023b. Exploring the upper limits of text-based collaborative filtering using large language models: Discoveries and insights. *arXiv preprint arXiv:2305.11700*. 2.1.4

[78] Li, B.; Yang, Q.; and Xue, X. 2009a. Can movies and books collaborate? cross-domain collaborative filtering for sparsity reduction. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 2052–2057. 2.3.1

[79] Li, B.; Yang, Q.; and Xue, X. 2009b. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th annual international conference on machine learning*, 617–624. 2.1, 2.3.1

[80] Liang, D.; Krishnan, R. G.; Hoffman, M. D.; and Jebara, T. 2018. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 world wide web conference*, 689–698. 2.1.1, 3.3

[81] Ling, G.; Yang, H.; King, I.; and Lyu, M. R. 2012. Online learning for collaborative filtering. In *The 2012 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE. 4

[82] Liu, M.; Li, J.; Li, G.; and Pan, P. 2020. Cross domain recommendation via bi-directional transfer graph collaborative filtering networks. In *Proceedings of the 29th ACM international conference on information & knowledge management*, 885–894. 2.3.1

[83] Liu, F.; Chen, H.; Cheng, Z.; Liu, A.; Nie, L.; and Kankanhalli, M. 2022. Disentangled multimodal representation learning for recommendation. *IEEE Transactions on Multimedia*. 2.1

[84] Liu, Y.-F.; Hsu, C.-Y.; and Wu, S.-H. 2015. Non-linear cross-domain collaborative filtering via hyper-structure transfer. In *International Conference on Machine Learning*, 1190–1198. PMLR. 2.3.2

[85] Lu, Y.-T.; Yu, S.-I.; Chang, T.-C.; and Hsu, J. Y.-j. 2009. A content-based method to enhance tag recommendation. In *Twenty-first international joint conference on artificial intelligence*. 2.1.2

[86] Ludewig, M.; Mauro, N.; Latifi, S.; and Jannach, D. 2019. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM conference on recommender systems*, 462–466. 5.3.1

[87] Luo, X.; Zhou, M.; Xia, Y.; and Zhu, Q. 2014. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. *IEEE Transactions on Industrial Informatics* 10(2):1273–1284. 3.3

[88] Luo, X.; Zhou, Y.; Liu, Z.; Hu, L.; and Zhou, M. 2021. Generalized nesterov's acceleration-incorporated, non-negative and adaptive latent factor analysis. *IEEE Transactions on Services Computing* 15(5):2809–2823. 3.3

[89] Ma, H.; Yang, H.; Lyu, M. R.; and King, I. 2008. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, 931–940. New York, NY, USA: Association for Computing Machinery. 2.1.4

[90] Ma, J.; Zhao, Z.; Yi, X.; Chen, J.; Hong, L.; and Chi, E. H. 2018a. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 1930–1939. 2.1, 2.5

[91] Ma, X.; Zhao, L.; Huang, G.; Wang, Z.; Hu, Z.; Zhu, X.; and Gai, K. 2018b. Entire space multi-task model: An effective approach for estimating post-click conversion rate. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 1137–1140. 2.1, 2.2

[92] Ma, J.; Zhao, Z.; Chen, J.; Li, A.; Hong, L.; and Chi, E. H. 2019a. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 216–223. 2.1, 2.5

[93] Ma, M.; Ren, P.; Lin, Y.; Chen, Z.; Ma, J.; and Rijke, M. d. 2019b. π-net: A parallel

information-sharing network for shared-account cross-domain sequential recommendations. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, 685–694. 2.3.2

[94] Ma, H.; King, I.; and Lyu, M. R. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 203–210. 3.3, 4.3.1

[95] Man, T.; Shen, H.; Jin, X.; and Cheng, X. 2017. Cross-domain recommendation: An embedding and mapping approach. In *IJCAI*, volume 17, 2464–2470. 2.3.2

[96] McCloskey, M., and Cohen, N. J. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24. Elsevier. 109–165. 4.1

[97] McLaughlin, M. R., and Herlocker, J. L. 2004. A collaborative filtering algorithm and evaluation metric that accurately model the user experience. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 329–336. 2.1.1

[98] Mi, F.; Lin, X.; and Faltings, B. 2020. Ader: Adaptively distilled exemplar replay towards continual learning for session-based recommendation. In *Fourteenth ACM Conference on Recommender Systems*, 408–413. 2.4

[99] Milogradskii, A.; Lashinin, O.; P, A.; Ananyeva, M.; and Kolesnikov, S. 2024. Revisiting bpr: A replicability study of a common recommender system baseline. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 267–277. 5.3.1

[100] Mnih, A., and Salakhutdinov, R. R. 2007. Probabilistic matrix factorization. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc. 2.1.1

[101] Moreno, O.; Shapira, B.; Rokach, L.; and Shani, G. 2012. Talmud: transfer learning for multiple domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 425–434. 2.3.1

[102] Nichol, A.; Achiam, J.; and Schulman, J. 2018. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*. 1, 4.3.1

[103] Ostapenko, O.; Puscas, M.; Klein, T.; Jahnichen, P.; and Nabi, M. 2019. Learning to remember: A synaptic plasticity driven framework for continual learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 11321–11329. 4.3.1

[104] Peng, D.; Pan, S. J.; Zhang, J.; and Zeng, A. 2021. Learning an adaptive meta model-generator for incrementally updating recommender systems. In *Proceedings of the 15th ACM Conference on Recommender Systems*, 411–421. 2.1, 2.4

[105] Petrov, A., and Macdonald, C. 2022a. Effective and efficient training for sequential recommendation using recency sampling. In *Sixteen ACM Conference on Recommender Systems*. 2.1.3

[106] Petrov, A., and Macdonald, C. 2022b. A systematic review and replicability study of bert4rec for sequential recommendation. In *Proceedings of the 16th ACM Conference on*

Recommender Systems, 436–447. 5.3.1

[107] Pham, Q.; Liu, C.; Sahoo, D.; and Steven, H. 2021. Contextual transformation networks for online continual learning. In *International Conference on Learning Representations*. 1

[108] Pham, Q.; Liu, C.; and Hoi, S. 2021. Dualnet: Continual learning, fast and slow. *Advances in Neural Information Processing Systems* 34:16131–16144. 1

[109] Quadrana, M.; Karatzoglou, A.; Hidasi, B.; and Cremonesi, P. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*, 130–137. 2.1.3, 5.1, 5.3.1

[110] Rafailidis, D., and Crestani, F. 2016. Top-n recommendation via joint cross-domain user clustering and similarity learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 426–441. Springer. 2.3.1, 2.3.2

[111] Ren, S.; Gao, S.; Liao, J.; and Guo, J. 2015. Improving cross-domain recommendation through probabilistic cluster-level latent factor model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29. 2.3.1

[112] Ren, P.; Chen, Z.; Li, J.; Ren, Z.; Ma, J.; and De Rijke, M. 2019. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, 4806–4813. 2.1.3

[113] Ren, X.; Wei, W.; Xia, L.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM Web Conference 2024*, 3464–3475. 2.1.4

[114] Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, 452–461. Arlington, Virginia, USA: AUAI Press. 2.1.1, 5.1, 5.2.2, 5.3.1

[115] Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2012. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*. 2.1.1

[116] Rendle, S.; Freudenthaler, C.; and Schmidt-Thieme, L. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th international conference on World wide web*, 811–820. 2.1.3, 5.3.1

[117] Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, 175–186. 2.1.1

[118] Riemer, M.; Cases, I.; Ajemian, R.; Liu, M.; Rish, I.; Tu, Y.; and Tesauro, G. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *In International Conference on Learning Representations (ICLR)*. 4.3.1, 4.3.1

[119] Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T. P.; and Wayne, G. 2019. Experience replay for continual learning. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 350–360. 4.3.1

[120] Saeed, W., and Omlin, C. 2023. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems* 263:110273. 2.1.4

[121] SAHOO, D.; PHAM, H. Q.; LU, J.; and HOI, S. C. Online deep learning: Learning deep neural networks on the fly.(2018). In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence IJCAI 2018, July 13-19, Stockholm*, 2660–2666. 1

[122] Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2000. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, 158–167. 2.1.1

[123] Sarwar, B.; Karypis, G.; Konstan, J.; and Riedl, J. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, 285–295. 2.1, 2.1.1

[124] Shazeer, N.; Mirhoseini, A.; Maziarz, K.; Davis, A.; Le, Q.; Hinton, G.; and Dean, J. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*. 5.2.1, 5.2.1

[125] Shehzad, F., and Jannach, D. 2023. Everyone's a winner! on hyperparameter tuning of recommendation models. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 652–657. 5.3.1

[126] Shu, K.; Wang, S.; Tang, J.; Wang, Y.; and Liu, H. 2018. Crossfire: Cross media joint friend and item recommendations. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, 522–530. 2.3.2, 2.3.3

[127] Sikka, R.; Dhankhar, A.; and Rana, C. 2012. A survey paper on e-learning recommender system. *International Journal of Computer Applications* 47(9):27–30. 4

[128] Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1441–1450. 2.1.4, 5.3.1

[129] Tang, J., and Wang, K. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, 565–573. 2.1.3, 5.3.1, 3

[130] Tang, H.; Liu, J.; Zhao, M.; and Gong, X. 2020. Progressive layered extraction (ple): A novel multi-task learning (mtl) model for personalized recommendations. In *Proceedings of the 14th ACM conference on recommender systems*, 269–278. 2.1, 2.5

[131] Tanjim, M. M.; Su, C.; Benjamin, E.; Hu, D.; Hong, L.; and McAuley, J. 2020. Attentive sequential models of latent intent for next item recommendation. In *Proceedings of The Web Conference 2020*, WWW '20, 2528–2534. New York, NY, USA: Association for Computing Machinery. 2.1.3

[132] Tay, Y.; Luu, A. T.; and Hui, S. C. 2018. Multi-pointer co-attention networks for recommendation. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2309–2318. 2.1.4

[133] Truong, Q.-T.; Salah, A.; and Lauw, H. W. 2021. Bilateral variational autoencoder for collaborative filtering. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, 292–300. 2.1.1

[134] Vasile, F.; Smirnova, E.; and Conneau, A. 2016. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM conference on recommender systems*, 225–232. 2.1, 2.1.2

[135] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc. 2.1.3

[136] Vitter, J. S. 1985. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)* 11(1):37–57. 4.2

[137] Wang, P.; Guo, J.; Lan, Y.; Xu, J.; Wan, S.; and Cheng, X. 2015. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, 403–412. New York, NY, USA: Association for Computing Machinery. 2.1.3

[138] Wang, N.; Wang, H.; Jia, Y.; and Yin, Y. 2018. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, 165–174. New York, NY, USA: Association for Computing Machinery. 2.1, 2.2

[139] Wang, J.; Ding, K.; Hong, L.; Liu, H.; and Caverlee, J. 2020. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, 1101–1110. New York, NY, USA: Association for Computing Machinery. 2.1.3

[140] Wang, J.; Ding, K.; Zhu, Z.; and Caverlee, J. 2021a. Session-based recommendation with hypergraph attention networks. In *Proceedings of the 2021 SIAM international conference on data mining (SDM)*, 82–90. SIAM. 2.1.3

[141] Wang, T.; Zhuang, F.; Zhang, Z.; Wang, D.; Zhou, J.; and He, Q. 2021b. Low-dimensional alignment for cross-domain recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 3508–3512. 2.3.2

[142] Wang, C.; Ma, W.; Chen, C.; Zhang, M.; Liu, Y.; and Ma, S. 2023a. Sequential recommendation with multiple contrast signals. *ACM Trans. Inf. Syst.* 41(1). 2.1.3

[143] Wang, Y.; Lam, H. T.; Wong, Y.; Liu, Z.; Zhao, X.; Wang, Y.; Chen, B.; Guo, H.; and Tang, R. 2023b. Multi-task deep recommender systems: A survey. *arXiv preprint arXiv:2302.03525*. 1

[144] Wang, H.; Chen, B.; and Li, W.-J. 2013. Collaborative topic regression with social regularization for tag recommendation. In *IJCAI*, volume 13, 2719–2725. 2.1.4

[145] Wang, J.; De Vries, A. P.; and Reinders, M. J. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, 501–508. 2.1.4

[146] Wu, X.; Magnani, A.; Chaidaroon, S.; Puthenputhussery, A.; Liao, C.; and Fang, Y. 2022. A multi-task learning framework for product ranking with bert. In *Proceedings of the ACM*

Web Conference 2022, 493–501. 2.1

[147] Xi, D.; Chen, Z.; Yan, P.; Zhang, Y.; Zhu, Y.; Zhuang, F.; and Chen, Y. 2021. Modeling the sequential dependence among audience multi-step conversions with multi-task learning in targeted display advertising. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 3745–3755. 2.2

[148] Yang, D.; He, J.; Qin, H.; Xiao, Y.; and Wang, W. 2015. A graph-based recommendation across heterogeneous domains. In *proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, 463–472. 2.3.2, 2.3.3

[149] Yang, Y.; Huang, C.; Xia, L.; Liang, Y.; Yu, Y.; and Li, C. 2022. Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, 2263–2274. New York, NY, USA: Association for Computing Machinery. 2.1.3

[150] Ying, H.; Zhuang, F.; Zhang, F.; Liu, Y.; Xu, G.; Xie, X.; Xiong, H.; and Wu, J. 2018. Sequential recommender system based on hierarchical attention network. In *IJCAI international joint conference on artificial intelligence*. 2.1.3

[151] You, K.; Long, M.; Cao, Z.; Wang, J.; and Jordan, M. I. 2019. Universal domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2720–2729. 1

[152] Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J. M.; and He, X. 2019. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, 582–590. New York, NY, USA: Association for Computing Machinery. 2.1.3

[153] Yuan, F.; Zhang, G.; Karatzoglou, A.; Jose, J.; Kong, B.; and Li, Y. 2021. One person, one model, one world: Learning continual user representation without forgetting. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 696–705. 2.1, 2.4

[154] Yuan, Z.; Yuan, F.; Song, Y.; Li, Y.; Fu, J.; Yang, F.; Pan, Y.; and Ni, Y. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2639–2649. 2.1.4

[155] Zhang, Y., and Yang, Q. 2021. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering* 34(12):5586–5609. 1

[156] Zhang, Z.; Jin, X.; Li, L.; Ding, G.; and Yang, Q. 2016. Multi-domain active learning for recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30. 2.3.2

[157] Zhang, Z.; Liu, S.; Yu, J.; Cai, Q.; Zhao, X.; Zhang, C.; Liu, Z.; Liu, Q.; Zhao, H.; Hu, L.; et al. 2024. M3oe: Multi-domain multi-task mixture-of experts recommendation framework. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 893–902. 2.5

[158] Zhang, Y.; Chen, X.; et al. 2020. Explainable recommendation: A survey and new

perspectives. *Foundations and Trends® in Information Retrieval* 14(1):1–101. 2.1.4

[159] Zheng, L.; Noroozi, V.; and Yu, P. S. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, 425–434. New York, NY, USA: Association for Computing Machinery. 2.1.4

[160] Zhou, Y.; Lei, T.; Liu, H.; Du, N.; Huang, Y.; Zhao, V.; Dai, A. M.; Le, Q. V.; Laudon, J.; et al. 2022. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems* 35:7103–7114. 5.2.1

[161] Zhu, F.; Chen, C.; Wang, Y.; Liu, G.; and Zheng, X. 2019. Dtcdr: A framework for dual-target cross-domain recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 1533–1542. 2.1, 2.3.1, 2.3.2

[162] Zhu, F.; Wang, Y.; Chen, C.; Liu, G.; and Zheng, X. 2020. A graphical and attentional framework for dual-target cross-domain recommendation. In *IJCAI*, volume 21, 39. 2.1, 2.3.1

[163] Zhu, Y.; Ge, K.; Zhuang, F.; Xie, R.; Xi, D.; Zhang, X.; Lin, L.; and He, Q. 2021. Transfer-meta framework for cross-domain recommendation to cold-start users. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 1813–1817. 2.3.2

[164] Zhu, Y.; Tang, Z.; Liu, Y.; Zhuang, F.; Xie, R.; Zhang, X.; Lin, L.; and He, Q. 2022. Personalized transfer of user preferences for cross-domain recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, 1507–1515. 2.3.2

[165] Zhuang, F.; Qi, Z.; Duan, K.; Xi, D.; Zhu, Y.; Zhu, H.; Xiong, H.; and He, Q. 2020. A comprehensive survey on transfer learning. *Proceedings of the IEEE* 109(1):43–76. 1