Multi-Objective Personalization for Recommender Systems

Jie Wang

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Engineering
College of Science and Engineering
University of Glasgow

October 2025

Abstract

Modern recommender systems increasingly rely on deep neural architectures to learn user–item relationships from interaction logs. Sequential recommendation has become a prominent paradigm, where RNN-based models such as GRU4Rec and Transformer-based models such as SASRec/BERT4Rec achieve strong performance on accuracy-oriented metrics (e.g., Recall and NDCG). However, real-world deployments expose fundamental limitations that accuracy-centric formulations do not address: (i) ID-based representations are platform-specific and difficult to transfer across domains; (ii) optimizing only for relevance often produces homogeneous recommendation lists and fails to satisfy users' multifaceted needs for diversity, novelty, and serendipity; (iii) offline-learned policies degrade under distribution shift and face exploration risks in dynamic online environments; and (iv) black-box pipelines provide limited interpretability and offer little actionable value to stakeholders beyond end-users. This thesis studies these challenges under a unified theme of *multi-objective personalization* for sequential recommendation, and develops methods that improve transferability, controllability, deployability, and stakeholder-facing value.

First, to address the transferability bottleneck, we propose *TransRec*, which learns from mixture-of-modality (MoM) feedback by encoding items with content encoders (e.g., text and images) rather than categorical IDs. By learning directly from raw MoM features in an end-to-end manner, TransRec enables effective cross-domain transfer without requiring overlapped users or items, and yields significant gains in cold-start and cross-domain settings.

Second, to move beyond accuracy-centric optimization, we introduce two frameworks that reformulate recommendation as multi-objective sequential decision-making. *MODT4R* leverages return-conditioned Decision Transformers to integrate multiple objectives within a stable supervised learning pipeline, allowing flexible objective trade-offs via inference-time adjustment. Building on this, *HDT* employs a hierarchical architecture to capture long-term preferences across sessions and short-term intent within sessions, and uses hierarchical (expected and unexpected) returns to balance accuracy with diversity, novelty, and serendipity. Across multiple datasets, MODT4R and HDT achieve up to 16% improvement in diversity-related metrics while maintaining competitive accuracy.

Third, to bridge the offline-to-online gap for RL-based recommenders, we leverage Large Language Models (LLMs) as auxiliary components. We introduce *LE/LEA* to adapt LLMs

as state and reward models and to augment offline learning signals via action synthesis. Furthermore, *iALP* and its adaptive variant *A-iALP* use LLM-distilled preferences to warm-start policies offline and adapt them online through fine-tuning and exploration strategies, achieving up to 20% improvement in long-horizon cumulative rewards in online simulation and reducing convergence time.

Finally, to support multiple stakeholders, we propose *PDiT-GIM*, a two-stage diffusion framework that generates semantically meaningful preference representations and decodes them into interpretable, attribute-constrained textual and visual content, enabling actionable insights for retailers and designers in addition to end-user recommendation. Case studies report improved preference-aligned content generation and downstream engagement compared to generic baselines.

Overall, through extensive experiments spanning e-commerce, multimedia recommendation, and simulated online environments, this thesis demonstrates that multi-objective personalization can simultaneously improve beyond-accuracy objectives and long-term policy performance while maintaining strong accuracy. The thesis is presented in a thesis-by-publication format, with chapters organized around the above tasks and objectives.

Contents

List of Tables

List of Figures

Acknowledgements

The journey of completing this PhD would not have been possible without the support, guidance, and encouragement of many individuals and institutions. I would like to take this opportunity to express my sincere gratitude to them all.

First and foremost, I am profoundly grateful to my supervisor, Professor Joemon M. Jose, for his invaluable guidance, unwavering support, and insightful mentorship throughout this research. His expertise, patience, and constructive feedback have been instrumental in shaping this thesis and my growth as a researcher. I am especially thankful for the freedom he gave me to explore my own ideas.

I would like to extend my sincere gratitude to Dr. Alexandros Karatzoglou and Dr. Ioannis Arapakis, who have served as invaluable bridges between academic research and industrial applications. Their expertise, generous support with computational resources, and practical insights have been instrumental to my research journey.

I am grateful to my family members for their unconditional love, patience, and encouragement throughout this journey. Special thanks go to Yabin, who have been a constant source of support, understanding, and motivation.

Declaration

With the exception of Chapter 1 and Chapter 2, which contain introductory and background material, all work presented in this thesis was carried out by the author unless otherwise explicitly stated.

Chapter 1

Introduction

1.1 Problem Definition and Motivation

Recommender Systems (RS) have become indispensable components of the modern digital ecosystem, navigating vast streams of information to connect users with relevant content, products, and services. Platforms ranging from e-commerce giants and video streaming services to social networks leverage these systems to enhance user engagement and drive business value [1–3]. However, traditional approaches like Collaborative Filtering (CF) and Content-based Filtering often operate under a static model, typically abstracting the problem as matrix completion to predict missing user-item ratings [4, 5]. While effective at capturing long-term, stable user preferences by aggregating all past interactions into a single representation [6], this method has a critical flaw: it inherently overlooks the sequential order of those interactions. Consequently, these models lack the specific means to account for a user's short-term behavior or immediate intent, which is crucial in dynamic environments [7].

To address these limitations, the field has increasingly shifted towards Sequential Recommendation (SR), a paradigm that models the user's journey as an ordered sequence of events rather than an unordered set. This approach recognizes that user interests are not static; they evolve, shift, and are highly dependent on recent context. The core challenge of SR, therefore, is to accurately model these temporal dynamics to provide timely and contextually aware suggestions [8]. The primary goal becomes one of sequence modeling: capturing the dependencies between items in a user's history to predict their next action [9]. This allows for a more nuanced understanding of user behavior by distinguishing between short-term preferences (e.g., planning a vacation within a single browsing session) and long-term preferences that evolve gradually over time [8, 10]. By analyzing the order of interactions, SR systems can infer a user's evolving intent, representing a critical evolution from predicting what a user generally likes to predicting what a user wants right now.

While sequential modeling has advanced the state-of-the-art, the pursuit of more intelligent and versatile recommender systems reveals several fundamental challenges that limit their

real-world impact and future development. A primary issue is the silo effect, stemming from a lack of transferability and generalization. The dominant paradigm in recommender systems remains ID-centric collaborative filtering and sequential modeling [1, 11–14], where models learn embeddings for specific user and item IDs, rendering them highly specialized and non-transferable [6]. Consequently, each platform operates in isolation, unable to share knowledge with others (e.g., from TikTok to YouTube). This leads to significant practical challenges, including the persistent cold-start problem for new users and items [15], and the prohibitive cost of training bespoke models from scratch for each new task or platform [16].

Furthermore, the field often falls into the accuracy trap, neglecting long-term user satisfaction. While the ultimate goal of a recommender system is to enhance user experience, a singular focus on optimizing for immediate engagement metrics, such as click-through rates, often leads to monotonous and predictable recommendations. This can trap users in "filter bubbles" due to issues like popularity bias [17, 18]. A superior user experience requires a delicate balance between accuracy and these other critical objectives [19].

Beyond these challenges, paradigms like Reinforcement Learning (RL), while promising for optimizing long-term user rewards, face a significant deployment dilemma. Offline methods are hampered by noisy historical data, while online approaches risk user churn due to the critical cold-start problem, making real-world applications both costly and risky [20–22]. Finally, the field suffers from a stakeholder gap by designing systems almost exclusively for the end-user. This narrow focus overlooks the needs of businesses and creators, providing little feedback on why items are preferred and thus missing the opportunity to transform the recommender from a simple retrieval tool into a creative engine that generates value for the entire ecosystem.

Common recommendation setting. Although this thesis explores multiple modelling directions, all chapters target a common setting: *sequential recommendation from temporally ordered user–item interaction trajectories*. The chapters differ in the practical constraints imposed on this setting—domain transfer without shared IDs, multi-objective controllability, offline-to-online deployment under distribution shift and exploration risks, and stakeholder-facing interpretability/value generation. This shared setting provides the basis for a coherent conceptual integration across publication-based chapters.

Thesis viewpoint. This thesis argues that the next generation of recommender systems must evolve from isolated, accuracy-driven models into *multi-objective personalization* frameworks that are (i) *transferable across domains*, (ii) *controllable under competing objectives*, (iii) *deployable under offline-to-online constraints*, and (iv) *value-generative for multiple stakeholders*.

A composable conceptual framework. To make the integration explicit, we organize the thesis as a composable workflow (Figure 1.1): a transferable representation layer (TransRec) that learns beyond-ID content representations; a controllable decision layer (MODT4R/HDT)

Thesis overview diagram (conceptual framework).
Representation layer (TransRec) → Decision layer (MODT4R/HDT) → Deployability layer (LE/LEA, iALP/A-iALP) → Value-generation layer (PDiT-GIM).

Figure 1.1: Overview of the conceptual framework that unifies the publication-based chapters into a composable workflow.

that conditions recommendation on multi-objective returns; a deployability layer (LE/LEA and iALP/A-iALP) that improves learning signals and reduces online exploration risks; and a value-generation layer (PDiT-GIM) that decodes learned preferences into interpretable, attribute-compliant insights for business stakeholders.

This thesis argues that the next generation of recommender systems must evolve from isolated, accuracy-driven models into universal, multi-objective, and value-generative frameworks. To achieve this, we must address the foundational challenges of model transferability, expand the optimization scope beyond simple accuracy, overcome the practical hurdles of learning long-term policies, and create a multi-stakeholder ecosystem.

To address the challenge of model transferability, we first investigate a framework, TransRec, designed to learn transferable representations from mixture-of-modality feedback. By moving beyond ID dependency and focusing on the intrinsic content of items (e.g., text, images), we aim to build a foundational model for a general-purpose Recommendation System (gpRS), marking a pivotal step towards universal applicability.

To move beyond accuracy-centric optimization, this thesis expands its scope to the domain of Multi-Objective Sequential Recommendation (MOSR). We propose novel frameworks, including MODT4R and HDT, which reformulate the recommendation task as a sequence modeling problem using Decision Transformers. This approach allows the system to generate recommendations conditioned not only on user history but also on explicit, multi-objective goals, thereby enhancing overall user satisfaction.

To overcome the practical barriers of applying RL, and capitalizing on the recent breakthroughs in Large Language Models (LLMs), this thesis proposes novel methods to bridge the gap between offline training and online deployment. We first introduce an LLM-based Environment (LE) to serve as a high-fidelity state and reward model for augmenting offline RL. Subsequently, we present an Interaction-Augmented Learned Policy (iALP), which leverages LLMs to pre-train an effective policy offline, thereby mitigating the initial performance dip in online deployment.

To create a multi-stakeholder, value-generative ecosystem, this thesis culminates in a novel generative framework, PDiT-GIM, which shifts the role of the recommender from a simple retrieval system to a creative engine. It not only generates personalized items for users but also decodes latent user preferences into interpretable, attribute-based insights for business stakeholders [23], effectively closing the loop between consumer demand and product innovation.

Table 1.1: Roadmap of this thesis: task definition and objective of each publication-based chapter.

Axis	Chapter	Task / Setting	Objective (What this chapter achieves)
Transferability	Ch. 3	Cross-domain recommendation without overlapped user/item IDs using mixture-of-modality (MoM) feedback	Learn transferable user/item representations from content encoders rather than IDs; enable cross-domain and cross-modality transfer under MoM supervision
Controllability	Ch. 4	Multi-objective sequential recommendation (MOSR): accuracy–diversity–novelty trade-offs	Return-conditioned Decision Transformer (MODT4R) for controllable trade-offs via inference-time adjustment within a supervised pipeline
Hierarchical personalization	Ch. 5	Multi-objective personalized session-based recommendation (MOPSR): across/within-session dynamics + serendipity	Hierarchical Decision Transformers (HDT) with inter-/intra-session modelling and unexpected returns to balance long-/short-term objectives
Deployability	Chs. 6–7	RL-based recommendation under biased offline logs and risky online exploration	LLM-assisted learning: LE/LEA for state/reward modelling and offline augmentation; iALP/A-iALP for policy warm-start and safer online adaptation
Generation	Ch. 8	Preference generation and decoding for stakeholder-facing insights under attribute constraints	PDiT-GIM: diffusion-based preference generation + guided decoding to provide interpretable textual/visual insights

1.2 Thesis Roadmap: Tasks, Objectives, and Chapters

To address concerns about fragmentation and positioning, Table 1.1 explicitly summarizes the task formulation and objective of each publication-based chapter under the unified framework above.

1.3 Thesis-by-Publication Format and Experimental Alignment

This thesis is written in a thesis-by-publication format. Each technical chapter is based on a peer-reviewed paper (or an under-review manuscript), and is revised to improve coherence through unified definitions, consistent notation, and explicit task formulations. Since the chapters study different practical constraints of sequential recommendation (transfer, multi-objective control,

offline RL, online deployment, and preference decoding), datasets and baselines necessarily vary across chapters. To address examiner concerns about experimental rigour and consistency, we align experimental design across chapters through the following principles.

A unified evaluation backbone for sequential recommendation. Whenever the task is next-item recommendation from temporally ordered interactions, we report accuracy using standard ranking metrics such as Hit Ratio/Recall and NDCG (Chapters 3–7). For multi-objective settings, we additionally report objective-specific metrics for diversity and novelty, and for session-based personalization we include serendipity (Chapters 4–5). For online RL settings, we evaluate long-horizon performance using cumulative return and convergence/stability indicators (Chapter 7). For preference generation and decoding, we report both generation quality and controllability, including NLG metrics (BLEU/ROUGE/BERTScore) and attribute compliance, while still evaluating recommendation utility with HR/NDCG when applicable (Chapter 8).

Consistent preprocessing and splitting protocols, adapted to each setting. Across sequential datasets, we apply standard filtering to reduce noise (e.g., removing very short sequences and infrequent items), and use temporally meaningful splits. In MOSR (Chapter 4), we follow standard session-based preprocessing (e.g., removing sequences of length < 3; filtering low-frequency items) and evaluate on three public datasets (RC15, RetailRocket, LFM-1b), including a controlled subsampling protocol for RC15 (200k sessions) to ensure tractable yet representative evaluation. In MOPSR (Chapter 5), we follow established sessionization practice (60-minute threshold) and adopt a strictly chronological evaluation: the last session is held out for testing and earlier sessions form training, ensuring that cross-session preference modelling is evaluated under realistic temporal progression. For the online RL chapter (Chapter 7), we partition datasets into simulated training (80%) and testing (20%) environments to reflect the online learning/testing separation.

Baseline selection by task category, with shared backbones where possible. To ensure fair and interpretable comparisons, each chapter uses baselines that are representative for its task category, while reusing common backbones across chapters to isolate the effect of new components.

(i) Transferability (Chapter 3). We evaluate transfer learning from a large-scale proprietary source domain (QQBrowser news/video feed; MoM interactions) to multiple target domains (Tencent News variants and DouYin). The key baseline is a train-from-scratch counterpart (TFS) that uses the same architecture and modality inputs but without pre-training, isolating the benefit of transfer. We additionally include ID-based sequential baselines (GRU-, CNN-, and Transformer-based encoders) as references to contextualize the gap between ID-centric and content-centric transfer.

(ii) Controllable MOSR (Chapter 4). We compare against established sequential recommendation and RL-based baselines under consistent settings, including SASRec (accuracy-oriented backbone), SQN (self-supervised RL for SR), and SMORL (state-of-the-art MOSR). For methods not directly comparable due to missing or inconsistent reporting (e.g., SMONAC), we reimplement them under the same backbone (SASRec) to ensure fair comparison. We also include an internal variant (MRDT) to justify key design choices in return-state-action ordering.

(iii) Hierarchical multi-objective session personalization (Chapter 5). To test generality, we apply HDT to four widely used sequential backbones (GRU4Rec, Caser, NItNet, SASRec). We compare against: (a) the Normal training of each backbone (with session concatenation), (b) SMORL implemented under the same preprocessing, and (c) hierarchical PSR baselines (Hier and its extensions), including an InterDT variant to isolate the role of inter-session modelling.

(iv) Offline RL with LLM-assisted environment and augmentation (Chapter 6). We use two real-world datasets (LFM subset and Amazon Industry category) and compare against state-of-the-art RL frameworks (SNQN and SA2C) under two shared backbones (GRU4Rec and SASRec). We systematically ablate the contribution of the learned LLM environment by replacing state and/or reward signals in the RL frameworks (LE-derived state, LE-derived reward, both, and augmentation-only), enabling a controlled understanding of where LLM assistance improves learning.

(v) Online RL policy exploration (Chapter 7). We evaluate A-iALP on LFM, Industry, and Coat with an 80/20 simulated online split. Baselines include established online RL methods (DQN, policy gradient, A2C) and the frozen iALP warm-start agent. This isolates the effect of (a) LLM-distilled preference warm-start and (b) adaptive online exploration strategies on long-horizon return and convergence behaviour.

(vi) Preference generation and decoding (Chapter 8). We evaluate PDiT-GIM along three research questions: decoding quality (RQ1), recommendation utility of learned representations (RQ2), and guided controllability/compliance (RQ3). Baselines include a fine-tuned LLM (same base model as GIM with LoRA), diffusion variants (PDiTgen/PDiTtem), a conditional discrete diffusion text generator (DiffuSeq), and a recent diffusion recommender baseline (DreamRec). Attribute alignment is evaluated via compliance rates using an LLM judge, reflecting the absence of ground-truth labels for generated items under constraints.

Cross-chapter reuse to strengthen coherence. Where feasible, we reuse datasets and modelling backbones across chapters to improve coherence and comparability. Notably, LFM and Industry appear in Chapters 6–8, enabling controlled studies of how LLM-derived feedback improves RL training and how preference representations support both recommendation and decoding. GRU4Rec and SASRec recur as shared sequential backbones across multiple chapters, allowing improvements to be attributed to the proposed mechanisms rather than backbone changes. When chapters necessarily use different datasets (e.g., proprietary MoM transfer vs.

public MOSR/MOPSR vs. online simulation), we explicitly justify the choice based on the task definition and clarify the scope of conclusions.

1.4 Publications From This Research

The research presented in this thesis has led to the following publications:

- **Chapter 3:** Wang, Jie, Fajie Yuan, Mingyue Cheng, Joemon M. Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. "Transrec: Learning transferable recommendation from mixture-of-modality feedback." In Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data, pp. 193-208. Singapore: Springer Nature Singapore, 2024.

- **Chapter 4:** Wang, Jie, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xuri Ge. "Beyond Accuracy: Decision Transformers for Reward-driven Multi-objective Recommendations." IEEE Transactions on Knowledge and Data Engineering (2025).

- **Chapter 5:** Wang, Jie, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, Xuri Ge, and Joemon M. Jose. "Sparks of surprise: Multi-objective recommendations with hierarchical decision transformers for diversity, novelty, and serendipity." In Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, pp. 2358-2368. 2024.

- **Chapter 6:** Wang, Jie, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. "Reinforcement learning-based recommender systems with large language models for state reward and action modeling." In Proceedings of the 47th International ACM SIGIR conference on research and development in information retrieval, pp. 375-385. 2024.

- **Chapter 7:** Wang, Jie, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M. Jose. "Large Language Model driven Policy Exploration for Recommender Systems." In Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining, pp. 107-116. 2025.

- **Chapter 8:** Wang, Jie. PDiT-GIM: A Two-Stage Diffusion Framework for Personalized Preference Generation and Decoding. *Under review*.

1.5 Organization of the Thesis

The rest of this thesis is organized as follows:

In Chapter 2, a comprehensive literature review of modern recommender systems is provided to contextualize the contributions of this thesis. The review covers the progression from

traditional Sequential Recommendation (SR) to more complex Multi-Objective Sequential Recommendation (MOSR) frameworks. Furthermore, it delves into three advanced paradigms that are central to this work: the application of reinforcement learning, particularly the Decision Transformer; the use of pre-training to build general-purpose models; and the cutting-edge integration of LLMs in the recommendation domain.

In Chapter 3, a transfer learning approach for recommender systems is proposed to address the limitations of standard ID-based models. This chapter first discusses the challenge of applying the dominant pre-train and transfer paradigm to RS, as the identity features of users and items are generally not shareable across different platforms. To overcome this issue, a novel scenario is introduced where recommendation models are pre-trained on items with mixture-of-modality (MoM) features, such as text and images. Following this, TransRec is presented, which is designed to learn directly from the raw features of MoM items in an end-to-end manner, thus enabling effective transfer learning without relying on overlapping users or items. Finally, a detailed experimental evaluation studies TransRec's transferring ability and data scaling effects across four real-world settings, suggesting that learning from MoM feedback is a promising direction for universal recommender systems.

In Chapter 4, we address the problem of Multi-objective Sequential Recommendation (MOSR) and identify the limitations of accuracy-focused models and existing multi-objective solutions. A new framework named MODT4R is proposed, which reframes the MOSR task as a supervised sequence modeling problem. MODT4R utilizes a user trajectory concept to generate recommendations that balance accuracy with diversity and novelty based on specified multi-objective returns. A comprehensive empirical evaluation on three real-world datasets demonstrates that MODT4R significantly enhances non-accuracy metrics, such as diversity and novelty, without compromising predictive accuracy.

In Chapter 5, the problem of extending Personalized Session-based Recommendation (PSR) to multi-objective scenarios is addressed. The chapter begins by critiquing two primary limitations of existing PSR methods: the failure to capture dynamic user interests due to a static treatment of historical sessions, and an over-emphasis on accuracy that neglects crucial satisfaction aspects like diversity and novelty. To overcome these challenges, a new task named Multi-objective PSR (MOPSR) is defined, and a novel framework, the Hierarchical Decision Transformer (HDT), is proposed. For modeling dynamic user preferences, HDT uses a hierarchical structure comprising Inter- and Intra-session DTs to track long- and short-term interest transitions. For balancing multiple objectives, the concept of unexpected returns is proposed, which allows the model to trade off accuracy against diversity, novelty, and serendipity. Finally, HDT is applied to four state-of-the-art sequential recommendation models and evaluated on two public datasets.

In Chapter 6, we introduce LE Augmentation, a novel method for improving offline RL-based recommenders. The core of this approach is to address the challenge of modeling user

states and rewards by proposing a Language Environment (LE) powered by a LLM. The LE learns from limited data to synthesize high-quality user feedback, serving a dual role: it acts as both a state model to enrich user representations and a reward model to capture preferences, while also augmenting the training data by generating new positive actions. By leveraging these capabilities, LE Augmentation jointly optimizes the recommendation policy. Experiments on two public datasets demonstrate its effectiveness in enhancing state-of-the-art RL recommenders.

In Chapter 7, we propose a novel framework to tackle key challenges in online RL-based recommendation, such as distribution shift and the exploration-exploitation trade-off. Central to this framework is the Interaction-Augmented Learned Policy (iALP), which leverages an LLM to pre-train an RL policy by simulating user preferences offline. To ensure effective online deployment, we further develop an adaptive version, A-iALP. By incorporating strategies to fine-tune and adapt to dynamic user interactions, A-iALP successfully mitigates deployment risks and enhances exploration. Empirical evaluations across three simulated environments validate that our framework yields significant performance gains over baseline methods.

In Chapter 8, a novel framework named PDiT-GIM is proposed to transform recommendation systems into multi-party platforms that serve both consumers and business stakeholders. PDiT-GIM employs a generation-decoding architecture, where a Personalized Diffusion Transformer (PDiT) generates interpretable user preference representations, and a Guided Item Modeling (GIM) module decodes these representations into explicit, attribute-constrained business insights. Extensive empirical evaluation on real-world datasets demonstrates the framework's superior performance in generating high-quality, aligned, and compliant insights while maintaining recommendation effectiveness.

Chapter 2

Background

Modern recommender systems aim to anticipate user preferences by modeling the temporal dynamics of interactions [24, 25]. The most common form of achieving this is by capturing evolving user interests from historical engagement sequences to predict the next item of interest. The foundational approach to this task is Sequential Recommendation (SR), which utilizes various neural architectures to model these sequences. While early work employed Recurrent Neural Networks (RNNs) like GRU4Rec [26], the field has largely been shaped by the advent of self-attention mechanisms and Transformer-based models such as BERT4Rec [1] for their superior ability to model long-range dependencies. This chapter reviews the foundations in a way that explicitly supports the thesis roadmap introduced in Chapter 1: SR as the common setting, and four practical constraints that motivate the publication-based chapters—transferability (Chapter 3), controllable multi-objective optimization (Chapters 4–5), offline-to-online deployability (Chapters 6–7), and multi-stakeholder value generation (Chapter 8).

Traditional sequential recommendation systems, however, often consider a single primary objective: maximizing predictive accuracy. Today, researchers recognize that this narrow focus can lead to monotonous results and degrade long-term user experience [27]. The data and modeling capabilities available now provide not only information on what a user will click next, but also insights into maintaining user engagement through other objectives like diversity, novelty, and serendipity. With the help of these new objectives, it is possible to embed a more holistic view of user satisfaction into recommendation systems. One problem with this multi-objective approach, often termed Multi-Objective Sequential Recommendation (MOSR), is that these secondary goals are often in direct conflict with accuracy, creating a significant trade-off challenge that needs to be carefully managed. This thesis adopts *return-conditioned* sequence modeling (Decision Transformers) as a unifying mechanism to make such trade-offs explicit and controllable (Chapters 4–5).

Recently, methods in the literature started to consider entirely new paradigms beyond direct sequence modeling to increase recommendation quality and utility. One trend is to frame recommendation as a long-term optimization problem by applying Reinforcement Learning

(RL), modeling the process as a Markov Decision Process (MDP) to maximize cumulative rewards [28]. Another set of methods aims to address data sparsity and the cold-start problem by developing general-purpose recommendation systems through Pre-training. This is achieved via a pre-training and fine-tuning paradigm. Another idea is to leverage the reasoning and world knowledge of Large Language Models (LLMs), using them either in a text-to-text format like P5 [2] or as powerful components within existing frameworks [29]. These paradigms correspond to distinct thesis constraints: pre-training/content modeling for cross-domain transfer (Chapter 3); RL and Decision Transformers for controllable multi-objective decision-making (Chapters 4–5); and modular LLM integration to improve feedback quality and deployment robustness without incurring prohibitive inference cost (Chapters 6–7).

In Section 2.1, information on the fundamentals of Sequential Recommendation and its evolution into multi-objective optimization is explained. In Section 2.2, information on Reinforcement Learning for recommendation is given, including recent advances like the Decision Transformer [30]. In this section, we cover how RL is used to optimize for long-term user value. Afterwards in Section 2.3, the recommendation systems that use Pre-training paradigms to create generalizable and transferable models are explored. Lastly, in Section 2.4, related work about the cutting-edge integration of Large Language Models in the recommendation domain, both as standalone recommenders and as enhancement components, is presented. Sections 2.5–2.6 summarize dataset biases and evaluation metrics, and clarify how these considerations motivate the experimental protocols used throughout the thesis.

Key Concepts and Definitions

To address clarity concerns raised in the examination, we summarize key concepts used throughout the thesis.

Mixture-of-Modality (MoM) feedback. MoM feedback refers to sequential user interactions in which consumed items may come from different content modalities (e.g., text, image, video). The key implication is that user preference trajectories must be modeled from modality-specific content encoders rather than categorical item IDs. In Chapter 3, the MoM setting allows a user sequence to contain both text and image items (while each item may be single-modality), enabling transfer without overlapped IDs.

Beyond-accuracy objectives. Diversity measures dissimilarity within a recommended list (e.g., via intra-list similarity); novelty measures whether recommended items are new/unseen or less popular; serendipity measures whether items are both relevant and unexpectedly useful. Chapters 4–5 operationalize these objectives via explicit metrics and return-conditioned modeling.

Returns and controllability. Returns denote cumulative reward signals (possibly multi-objective) used as conditioning variables in Decision Transformer-style sequence modeling. Conditioning on returns enables explicit trade-offs at inference time without training separate models per objective (Chapters 4–5).

2.1 Sequential Recommendation

Sequential Recommendation (SR) forms the bedrock of modern recommender systems, focusing on modeling the temporal dynamics of user-item interactions to predict the next item a user is likely to engage with [24, 25]. The core challenge lies in capturing evolving user preferences from interaction sequences, which are typically ordered by timestamp. Early pioneering work in this area utilized Recurrent Neural Networks (RNNs), with Gated Recurrent Units (GRU) being a popular choice for modeling user sequences [26]. Following this, the field saw the adoption of more advanced architectures, including Convolutional Neural Networks (CNNs) for capturing local sequential patterns [31, 32] and, more recently, self-attention mechanisms and Transformers for their superior ability to model long-range dependencies [1, 33, 34]. A notable variant is Personalized Session-based Recommendation (PSR), which extends SR by modeling both intra-session dependencies (within the current interaction sequence) and inter-session dependencies (across a user's historical sessions). Hierarchical models, such as Hierarchical RNNs (HRNN) [35] and hierarchical CNNs [36], were developed to explicitly capture this long-term and short-term preference dynamic.

However, optimizing solely for prediction accuracy can lead to monotonous recommendations, potentially degrading long-term user experience. Consequently, the field has naturally evolved towards Multi-Objective Sequential Recommendation (MOSR), which aims to balance accuracy with other crucial objectives such as diversity, novelty, and serendipity [27, 37]. These secondary objectives often conflict with accuracy, presenting a significant trade-off challenge. Initial approaches addressed this through post-processing techniques, such as re-ranking a list of candidates generated by an accuracy-focused model [38–41]. Other methods integrated these objectives directly into the model training process, for instance, by designing joint loss functions [20, 21, 42–45] or learning dedicated representations for diversity and novelty [46, 47]. Pareto-based optimization has also been explored to find a set of optimal trade-off solutions rather than a single point [48–51]. More recently, reinforcement learning has been employed as a powerful framework for this multi-objective balancing act. A key example is the Sequential Multi-Objective Reinforcement Learning (SMORL) framework [52], which trains a separate Q-network for each objective alongside a supervised head, effectively promoting diverse and novel recommendations without compromising accuracy. This approach has been further extended with techniques like negative sampling to enhance performance [53]. In contrast to multi-head TD learning, this thesis studies MOSR via return-conditioned Transformers that make objec-

tive trade-offs explicit and controllable (Chapter 4), and extends controllability to cross-session personalization with hierarchical modeling and unexpected returns (Chapter 5).

2.2 Reinforcement Learning for Recommendation

The sequential nature of user interaction makes recommendation an ideal candidate for modeling as a Markov Decision Process (MDP), paving the way for the application of Reinforcement Learning (RL) to optimize for long-term user engagement and value [28, 54, 55]. In this paradigm, the recommender system acts as an agent that learns a policy to select items (actions) based on the user's current state, aiming to maximize a cumulative reward signal over time. Early applications of RL in recommendation often focused on offline settings, leveraging historical log data to train policies. For example, some work enriched the RL environment with external knowledge graphs to guide the learning process (KERL) [55], while others integrated novel data sources like wireless sensing to improve specific applications like music recommendation [56].

A significant advancement in this domain is Self-Supervised Reinforcement Learning (SSRL), which combines the predictive power of supervised learning with the long-term optimization of RL. The Sequential Q-Network (SQN) [57] and its Soft Actor-Critic (SAC) counterpart [58] are seminal works in this area. They integrate a Double Q-learning head with a supervised learning objective to enhance recommendation accuracy for metrics like clicks and purchases. These frameworks have been further improved with techniques such as negative sampling (SNQN and SA2C) [58] and state representation augmentation via contrastive learning [59]. Another line of research proposed shifting the focus from modeling per-step rewards to modeling desirable cumulative rewards to guide policy training [60].

Despite these advances, traditional offline RL algorithms based on temporal difference (TD) learning can suffer from bootstrapping errors and distribution drift when deployed online, as the learned policy may encounter states not seen in the static training data [30]. To address these limitations, a new paradigm has emerged that frames RL as a conditional sequence modeling problem. The Decision Transformer (DT) [30] exemplifies this approach. Instead of using dynamic programming, DT employs a causally masked Transformer architecture to predict actions conditioned on a trajectory of past states, actions, and desired cumulative rewards (returns-to-go). By modeling the entire trajectory $\tau = (R_1, s_1, a_1, R_2, s_2, a_2, \ldots)$, DT can generate actions in a supervised manner, making it particularly effective for offline RL settings with sparse rewards and long sequences, which are common in recommendation scenarios. This perspective underpins the return-conditioned modeling used for controllable MOSR/MOPSR (Chapters 4–5) and motivates the offline-to-online focus in Chapters 6–7, where the key bottleneck is the quality of state/reward feedback under biased logs and distribution shift.

2.3 Pre-training Paradigms for Generalizable Recommender

Inspired by the transformative success of large-scale foundation models in Natural Language Processing (NLP) and Computer Vision (CV) such as BERT [61], GPT-3 [62], and Vision Transformers (ViT) [63], the recommendation community has increasingly focused on developing general-purpose recommendation systems (gpRS) [64]. The goal is to build models with strong transferability that can mitigate challenges like data sparsity and the cold-start problem, and can be adapted to new domains or tasks with minimal effort. Early attempts in this direction used multi-task learning (MTL) to learn more general representations by training on several tasks simultaneously [65–67]. However, MTL models are typically limited to the tasks they were trained on and do not generalize well to entirely new scenarios.

A significant paradigm shift occurred with the introduction of self-supervised pre-training (SSP) followed by downstream fine-tuning, directly analogous to the approach in NLP. PeterRec [68] was a pioneering work that proposed a pre-training-then-fine-tuning paradigm for learning transferable user representations. This was followed by concepts like lifelong learning for recommendation models, as explored in Conure [69]. This line of work is closely related to cross-domain recommendation [70–73], but many of these early models rely on a shared user/item ID space across domains, a strict assumption that often does not hold in practice.

To overcome the ID-dependency limitation, recent efforts have focused on leveraging rich semantic information, particularly text, to build truly generalizable models [74–76]. Models like ZESRec [77], CLUE [75], and UnisRec [19] demonstrated that a recommender pre-trained on textual item sequences in one domain can be effectively transferred to other text-based recommendation scenarios. The most recent development in this area, P5 [2], unifies various recommendation-related tasks (e.g., rating prediction, explanation generation) into a single text-to-text format, pre-training a language model on this composite objective to create a highly versatile foundation model for recommendation.

The technical foundation for these models lies in transfer learning methodologies. Self-supervised pre-training can be broadly categorized into two types: generative pre-training, like the masked language modeling in BERT [61], and contrastive pre-training, like SimCLR [78]. Both approaches are well-suited for recommender systems, which have abundant implicit feedback for creating self-supervised signals. For downstream adaptation, various techniques are employed, including freezing parts of the model [79], full fine-tuning of all parameters [61], adapter tuning for parameter-efficient adaptation [80], and prompt-based tuning [62]. Chapter 3 connects this line to a practically relevant transfer setting by studying MoM feedback and end-to-end training with modality encoders, enabling transfer without overlapped user/item IDs.

2.4 Large Language Models in Recommendation

The unprecedented capabilities of Large Language Models (LLMs) [81–83] pre-trained on vast corpora of text have opened up a new frontier for recommender systems [29, 84, 85]. A primary line of research investigates using LLMs as standalone recommenders. This involves adapting an LLM to recommendation tasks through various strategies: comprehensive pre-training on recommendation-specific formats like in P5 [2]; fine-tuning on downstream datasets, often with parameter-efficient techniques like LoRA, as seen in TAllRec [86]; or using prompt-based methods to elicit recommendations in a zero- or few-shot manner [82, 87]. Another approach is to use LLMs to generate rich, semantic representations for items, which are then fed into traditional recommender architectures. For instance, item text descriptions can be encoded with models like BERT [19, 88] or Sentence-T5 [89] to create powerful semantic IDs [90]. However, deploying LLMs as full-fledged recommenders can be computationally expensive and may struggle to capture the fine-grained signals present in long user interaction histories.

Diverging from this paradigm, an alternative and highly promising direction is to leverage LLMs as enhancement components within existing recommendation frameworks, particularly those based on reinforcement learning. Instead of replacing the entire recommender, the LLM serves a specialized role. Inspired by recent successes in other domains where LLMs have excelled as reward models [91], this approach uses an LLM to provide dense, nuanced reward signals that can guide an RL agent more effectively than simple, sparse rewards like clicks. The LLM can interpret natural language descriptions of user goals or item attributes to generate rewards that align with complex objectives like user satisfaction or long-term engagement. This allows for the efficient adaptation of LLMs to augment the performance of existing, computationally leaner ID-based models. Furthermore, LLMs can also be adapted to function as other components in an RL loop, such as an environment simulator for offline policy evaluation or even as parts of the agent itself (e.g., actor, critic, or planner) for online recommendation [92], offering a powerful toolkit for building the next generation of intelligent and responsive recommender systems. This thesis primarily adopts the modular view: LLMs are used to improve *learning* (state/reward modeling and offline augmentation in Chapter 6) and *deployability* (policy warm-start and safer exploration in Chapter 7), while preserving inference efficiency of lean recommenders. Chapter 8 further leverages generative modeling to decode preferences into interpretable, stakeholder-facing artifacts.

2.5 Datasets and Biases

A handful of public benchmark datasets have become the de facto standard for evaluating sequential recommendation models. Popular choices include e-commerce datasets like Amazon Beauty and Yoochoose, as well as media consumption datasets such as MovieLens-1M and

Last.fm [1, 26, 93]. While these datasets have fueled significant research and enabled consistent benchmarking, they suffer from inherent biases that can limit the validity and generalizability of models trained on them.

A primary issue is selection bias, as these logs typically only contain records of interactions that occurred (i.e., positive feedback), with no explicit information on why users did not interact with other items [94]. This absence of negative feedback makes it challenging to distinguish between items a user genuinely dislikes and items they were simply never exposed to. Furthermore, these datasets are heavily influenced by exposure bias, as users could only interact with items presented to them by the pre-existing recommendation system that generated the logs. This creates a feedback loop where the data distribution is skewed towards items favored by the logging policy, rather than reflecting users' true, unbiased preferences [95]. For advanced paradigms like Reinforcement Learning (RL)-based recommendation, these biases are particularly problematic. In offline settings, where an agent learns from a static log of past interactions, the combination of selection and exposure bias leads to an impoverished and unreliable feedback signal. This can result in inaccurate value function estimation and flawed offline policy evaluation, causing the agent to learn a policy that merely exploits the biases of the previous system rather than discovering a truly optimal strategy for long-term user engagement [57, 59]. The resulting distribution shift between the logging policy and the learned policy can severely degrade online performance when the model is deployed.

To mitigate these issues, there has been a growing interest in developing methods and datasets that allow for less biased evaluation. One prominent approach is the collection of data using partially randomized exposure policies, such as in the KuaiRand dataset, which enables more robust offline policy evaluation and learning. Beyond data collection, novel modeling approaches are also required to either simulate more realistic user feedback from biased logs or to build models that are inherently less susceptible to these biases by leveraging rich, modality-based item content instead of platform-specific IDs [19, 88]. Addressing these data-centric challenges is crucial for advancing the next generation of intelligent and reliable recommender systems. These considerations motivate the dataset choices across the thesis: proprietary MoM transfer datasets (Chapter 3) are required to study realistic cross-platform transfer; public sequential benchmarks (RC15, RetailRocket, LFM-1b; Chapter 4) support reproducible MOSR evaluation; sessionized datasets (Album, Reddit; Chapter 5) are required to evaluate cross-session personalization; and LFM/Industry are reused across Chapters 6–8 to study how improved feedback modeling impacts offline RL, online adaptation, and preference decoding under consistent item textual attributes.

2.6 Evaluation Metrics

The evolution of recommender systems is mirrored in the paradigms used for their evaluation. The choice of metrics is not merely a technical decision; it profoundly influences which aspects of a system are optimized and reflects the field's shifting focus from mere predictive accuracy to a more holistic view of user experience and long-term value.

Accuracy-Centric Evaluation. Historically, the evaluation of recommender systems has been dominated by metrics that measure predictive accuracy [27]. This paradigm, central to traditional Sequential Recommendation (SR), assesses the quality of a ranked list of recommendations based on its ability to predict a user's immediate next interaction. Standard metrics such as NDCG@K (Normalized Discounted Cumulative Gain) [96] and Recall@K (also known as Hit-Rate@K) [26] have become de facto standards. These metrics reward models for placing ground-truth items higher in the top-K recommendation list, effectively quantifying the model's short-term relevance prediction capabilities. While crucial, a singular focus on accuracy has been shown to be insufficient for capturing the full spectrum of a positive user experience [37].

Beyond-Accuracy Evaluation. Recognizing the limitations of accuracy-only optimization, which can lead to monotonous recommendations and "filter bubbles," researchers have developed metrics to assess other desirable qualities. This paradigm is central to the field of Multi-Objective Sequential Recommendation (MOSR), a core topic of this thesis. These metrics often address objectives that are in direct tension with accuracy, creating a challenging trade-off space. Diversity metrics (e.g., intra-list similarity) assess the dissimilarity of items within a recommendation list [37]. Novelty measures whether recommended items are new or unknown to the user, while serendipity evaluates the extent to which recommendations are both surprising and useful [97]. Furthermore, Catalog Coverage measures the proportion of the total item catalog that a model recommends over time, indicating its ability to avoid popularity bias and leverage long-tail items [98]. In this thesis, diversity/novelty are evaluated in MOSR (Chapter 4), while serendipity is emphasized in cross-session personalization (Chapter 5) where "unexpected returns" are introduced to model preference shifts beyond immediate relevance.

Long-Term Value Evaluation. With the increasing application of Reinforcement Learning (RL) to recommendation, the evaluation paradigm has shifted from immediate, per-interaction accuracy to the maximization of long-term cumulative rewards [54, 55]. In this framework, the recommender is viewed as an agent whose goal is to learn a policy that optimizes for sustained user engagement and satisfaction over an extended period. The reward function can be defined as a sophisticated combination of various signals, such as user clicks, purchases, dwell time, and overall session duration. This approach directly aligns with the objective of fostering long-term user value rather than optimizing for single, myopic interactions. Chapter 7 follows this paradigm by evaluating online policies via cumulative return and convergence stability under simulated environments, while Chapters 6–7 study how improved state/reward feedback (from an LLM-based environment) affects both offline learning and online adaptation.

Generative and controllability evaluation. When recommendation is extended to preference generation and decoding (Chapter 8), evaluation additionally requires generation-quality and controllability metrics. We therefore report NLG metrics (e.g., BLEU/ROUGE/BERTScore) for decoded textual tokens, attribute compliance rates under constraints, and still evaluate recommendation utility of learned representations via HR/NDCG when applicable.

This evolution in evaluation, from myopic accuracy to holistic long-term value, sets the stage for the contributions presented in this thesis, which aims to advance the state-of-the-art in creating more intelligent, experience-driven recommender systems. Importantly, the thesis uses these metric families consistently to match each task setting: HR/NDCG for next-item recommendation (Chs. 3–6), beyond-accuracy metrics for MOSR/MOPSR (Chs. 4–5), return for online adaptation (Ch. 7), and NLG/compliance metrics for preference decoding (Ch. 8).

Chapter 3

Learning Transferable Recommendation from Mixture-of-Modality Feedback

Note. This chapter is based on our APWeb/WAIM 2024 paper on TransRec, and is revised in this thesis to align terminology/notation with the unified workflow, clarify the MoM (sequence-level mixture of modality behavior) setting and the non-overlapped transfer assumption, and connect the chapter outcomes to later decision/deployment layers.

3.1 Introduction

The mainstream RS typically model domain-specific user behaviors and then generate item recommendations only for the same platform. Such specialized RS has been well-established in literature [13, 14, 26, 32, 33, 99], yet they routinely suffer from some intrinsic limitations, such as low-accuracy problems for the cold-start setting [68], heavy manual work, and high cost for training from scratch per-task models [69]. Hence, developing general-purpose recommendation models to be useful to many systems has significant practical value. These kinds of models are popular in computer vision [63, 100] and natural language processing [61, 62] literatures, and they are recently referred to as the foundation models (FM) [64]. Despite their remarkable progress, there has yet to be a recognized learning paradigm for building general-purpose models for recommender systems (gpRS).

One principal reason is that ID-based collaborative filtering (CF) techniques nearly dominate existing RS models. Thereby, well-trained RS models can only be used to serve the current system because neither users nor items are easily shared across different private systems, e.g., from TikTok[1] to YouTube[2]. Even in some special cases, where userIDs or itemIDs in two platforms can be shared, it is still less prone to realizing the desired transferability since users/items on the two platforms also have limited overlapping situations. Less user and item overlapping

[1] https://www.tiktok.com/

[2] https://www.youtube.com/

will lead to limited transferring effects. Recent attempts such as PeterRec [68], lifelong Conure model [69] and STAR [101] fall exactly into this category.

To study the transferability of RS, we attempt to explore item modality[3]- or content-based recommendation where a modality encoder represents items (e.g. BERT [61] and ResNet [100]) rather than the ID embedding. By modeling modality features intuitively, recommendation models have the potential to achieve domain transferring for this modality in a broader sense — i.e. no longer relying on overlapping and shared-ID information. Moreover, the latest revolution of robust encoder networks in NLP and CV is also potentially beneficial to modality-based item recommendation and might even bring about a paradigm shift for RS from ID-based CF back to content-based recommendation.

To this end, we study a common yet unexplored recommendation scenario where user behaviors are composed of items with mixed-modality features — e.g., these interacted items by a user can be all texts or images or both. However, for simplicity, each item is restricted to only one modality for this study. Such a scenario is prevalent in many practical recommender systems such as feed recommendations, where recommended feeds can be a piece of news, an image, or a micro-video. In this chapter, we claim that developing recommendation models based on user feedback with mixture-of-modality (MoM) items is a vital way towards transferable and general-purpose recommendation. To verify our claim, we design TransRec, a recommendation framework for modeling user MoM feedback. In essence, TransRec is model-agnostic and can be a direct modification on most ID-based recommendation models. To eliminate the non-transferable ID features, we encode an item by a modality encoder rather than the itemID embedding. We encode users by a sequence of items (again, each item here is also encoded by their respective MoM encoder) rather than the userID embedding, as shown in Figure 3.1. We evaluate TransRec on the simple yet most widely adopted two-tower-based DSSM model [102], where one tower represents users, and the other represents items. We train TransRec, including both user and item encoders, by an end-to-end manner rather than using frozen features pre-extracted from modality encoders.

More importantly, we perform extensive empirical studies on TransRec — the first RS regime enabling effective transfer across modalities & domains. Specifically, we first train TransRec on a large-scale source dataset collected from a commercial website, where a user's feedback contains either textual or visual modality or both. Then we evaluate the pre-trained TransRec on the first target dataset collected from a different platform but has similar user feedback formats. Second, we evaluate TransRec on the second target dataset, where items have only one modality. Third, we evaluate TransRec with still one modality, but along with additional user/item features to verify its flexibility. At last, we evaluate TransRec on another target dataset where items look very different from the source domain to demonstrate its generality.

[3]modality used in this chapter mainly refers to some multimedia modalities, such as text, image, audio or videos, excluding the categorical IDs.

Beyond this, we empirically examine the performance of TransRec with varying scaling strategies on both the source and target datasets. Our results confirm that TransRec effectively learns from MoM feedback for various transfer learning tasks and benefits from large-scale source and small-scale target datasets. To some extent, TransRec is by far probably the closest model toward the goal of gpRS. We believe its broad transferability will point out a new way toward the foundation models in the RS domain.

To summarize, the contributions of this chapter can be described as: (1) We identify an essential fact that learning from MoM feedback has the potential to realize the goal of gpRS. To the best of our knowledge, we are one of the few works of learning a universal recommendation model from over one modality of feedback. (2) We present TransRec, the first recommendation model that realizes both cross-modality and cross-domain recommendation. (3) We study the transferability of TransRec across four types of recommendation scenarios. (4) We study the effects of TransRec by scaling the data and provide some valuable insights.

Table 3.1: Comparison of the transferability of existing recommendation methods. 'O' in the source domain indicates the modality type used for pre-training and in the target domain indicates that the pre-trained model can be used to serve recommendations with items of this modality, otherwise denoted by 'X'.

Methods	Source domain			Target domain		
	Text	Image	Mixed	Text	Image	Mixed
PeterRec [68]	X	X	X	X	X	X
ZESRec [77]	O	X	X	O	X	X
UnisRec [19]	O	X	X	O	X	X
CLUE [75]	O	X	X	O	X	X
TransRec (Ours)	O	O	O	O	O	O

3.2 Related Work

The pursuit of general-purpose recommender systems (gpRS), inspired by the success of foundation models in NLP and CV [61, 64, 100], has challenged the dominant paradigm of non-transferable, ID-based collaborative filtering. Initial attempts at creating transferable models, such as PeterRec [68], Conure [69], and STAR [101], focused on cross-domain scenarios but were fundamentally limited by their reliance on overlapping user or item IDs. This strong assumption restricts their applicability in heterogeneous ecosystems. A more recent and promising line of research has shifted towards content-based transferability, primarily leveraging textual information to build models free from ID dependencies. Works like ZESRec [77], CLUE [75], and UnisRec [19] have shown that pre-training on text-rich sequences can effectively transfer to new

text-based recommendation tasks.

Despite this progress, existing content-based approaches are predominantly limited to a single modality, failing to capture the diverse nature of modern platforms where users interact with a mix of content like articles, images, and videos. While multimodal recommendation is a related field [79, 103], it typically addresses items that inherently contain multiple modalities and often relies on using modality encoders as fixed feature extractors rather than fine-tuning them end-to-end. Our work, TransRec, addresses this crucial gap by introducing a novel Mixture-of-Modality (MoM) learning paradigm. We are among the first to propose pre-training a universal recommender on user histories composed of items with varied single modalities (e.g., text and images). This MoM pre-training, coupled with an end-to-end learning strategy, enables TransRec to develop a generalized understanding of user preferences that transcends specific modalities, allowing for unprecedented transferability to downstream tasks with any combination of the source modalities.

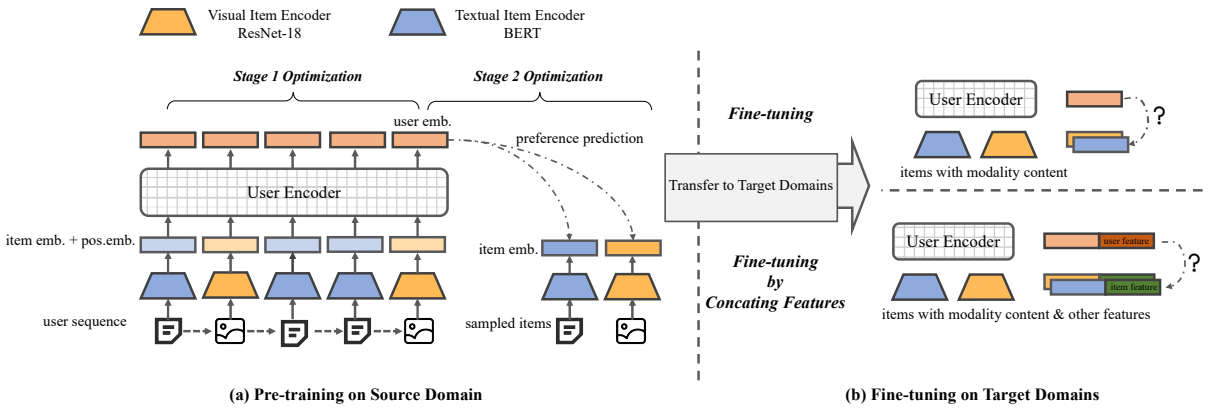


Figure 3.1: Illustration of the training process of TransRec. Here, the inner product is employed to compute the preference between users and candidate items.

3.3 The TransRec Framework

In this section, we first formulate the recommendation tasks with mixture-modality feedback and introduce some notations used in this chapter. Then, we introduce the TransRec framework in detail.

3.3.1 Problem Definition

Assume that we are given two categories of domains: source domain S and target domains $T = \{T_1, T_2, ..., T_N\}$. In source (target) domain, suppose that there exist user set U_s (U_t) and item set V_s (V_t), involving $|U_s|$ ($|U_t|$) users and $|V_s|$ ($|V_t|$) items. In both domains, the content feature of items is recorded with modality set $\mathscr{M} = \{a, b\}$, containing textual and visual

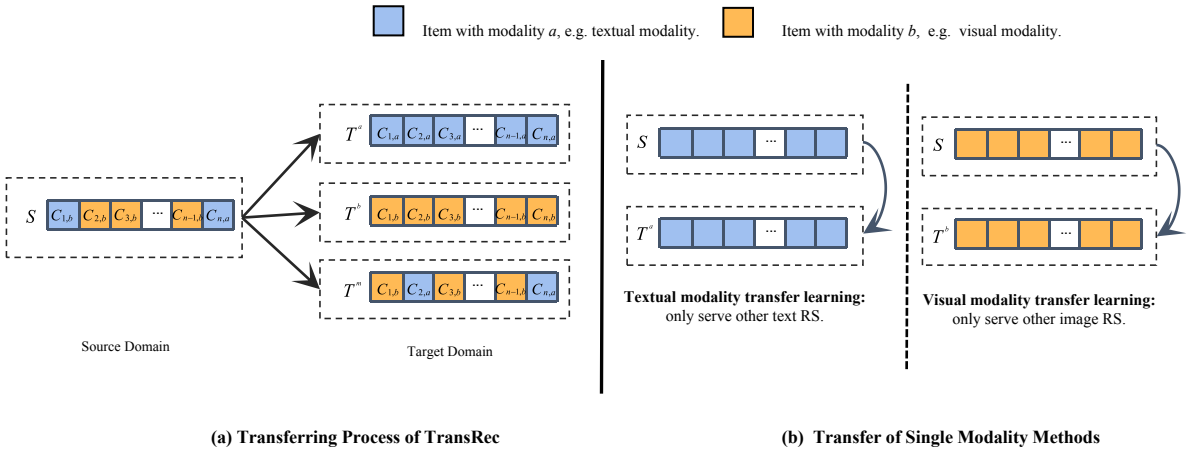


Figure 3.2: Schematic of learning from MoM. TransRec first pre-trains a unified recommendation model with MoM feedback in the source domain and then serves any target domain as long as the item's modality type is contained in the MoM feedback.

modalities, denoted as a and b, respectively. Following the setting of item-based collaborative filtering [68], users can be represented with the sequence of their historical interaction records $C_u = \{c_{1,m}, ..., c_{n,m}\}$. Here, n indicates the sequence length while $m \in \mathscr{M}$. This work aims to learn a generic recommender from source domain S, which can be transferred to N target domains T, involving non-overlapping of userIDs or itemIDs.

As illustrated in Figure 3.2, by learning from M, the trained model can be applied to the following domains, including single-modality domain $C = \{c_{1,a}, ..., c_{n,a}\}$ ($c_{i,a} \in V_t^a$) in domain T^a, single-modality domain $C = \{c_{1,b}, ..., c_{n,b}\}$ ($c_{i,b} \in V_t^b$) in domain T^b, and mixed-modality domain $C = \{c_{1,m}, ..., c_{n,m}\}$ ($c_{i,m} \in V_t^m$, $m \in M$) in domain T^m. Suppose that we extend the source domain with four modalities $M = \{a, b, c, d\} = \{\text{text}, \text{image}, \text{audio}, \text{video}\}$. The target domain can be served with 15 types of modalities, i.e. $\{a\}$, $\{b\}$, $\{c\}$, $\{a, b\}$, $\{a, b, c\}$...$\{a, b, c, d\}$ that covers a majority of existing multimedia modalities. In other words, learning from MoM feedback could easily cover downstream recommendation scenarios with items in the form of any single or combined modalities used in the source domain, which differs from traditional recommendations where user feedback contains modality features exactly the same as the per-interacted item.

3.3.2 TransRec Architecture

To verify our claim, we develop TransRec on the most popular two-tower-based recommendation architecture, a.k.a. DSSM [102]. To eliminate ID features, we represent both users and items with item modality contents. That is, the item tower of DSSM is represented by an item modality encoder (e.g. BERT for textual items and ResNet for visual items), and the user tower is represented by a user encoder that directly models an ordered collection of item interactions. For this study, we chose the self-attention-based Transformer blocks as the user encoder, given

their popularity and outstanding performance in modeling sequence data. But we emphasize that TransRec is not limited to the basic Transformer block — i.e. any deep neural network module that can model ID embedding sequences, such as the pure MLP network [104], GNN [105], RNN [26], TCN [32] and more advanced Transformers [106, 107] could be employed here (see Figure 3.1 for the TransRec architecture).

Formally, given a user interaction sequence C from an MoM scenario, TransRec divides it into two sub-sequence C^u and C^e, their relationship is denoted as:

$$C = C^u \cup C^e, \tag{3.1}$$

where $C = \{c_{1,m}, ..., c_{n,m}, c_{n+1,m}, ..., c_{n+l,m}\}$, with length of $n+l$, which is divided into $C^u = \{c_{1,m}, ..., c_{n,m}\}$, with the front ordered n interactions, and $C^e = \{c_{n+1,m}, ..., c_{n+l,m}\}$, with last l interactions. TransRec takes them as inputs to the item encoder E_i. Then, we obtain Z^u and Z^e, which are item representations for C^u and C^e, respectively. The item representations in Z^u are then fed into the user encoder E_u to achieve user representation U^u. U^u and Z^e are used to compute their relevance scores $R_{u,e}$. The process can be formulated as:

$$Z^e = E_i(C^e), \;\; Z^u = E_i(C^u), \tag{3.2}$$

$$U^u = E_u(Z^u), \;\; R_{u,e} = U^u \cdot Z^e, \tag{3.3}$$

where $Z^u = \{z_1, ..., z_n\}$ and $Z^e = \{z_{n+1}, ..., z_{n+l}\}$, z_t is the item representation of t-th item in the user sequence C. And $R_{u,e} = \{r_{u,1}, r_{u,2}, ..., r_{u,l}\}$, $r_{u,t}$ denotes the relevance score between U^u and t-th item of the sub-sequence C^e. $R_{u,e}$ demonstrates the relation between the user and his next interaction sequence. Next, we describe the two components of our model in detail.

Item Encoder. Given the MoM scenario, each item encoder of TransRec takes the item's (single) modality as input. We consider at most two types of modalities, i.e. textual tokens and image pixels, for each user in this chapter. Attempts for more modality scenarios are interesting for future work. For an item with textual modality (i.e. $c_{i,a}$), we adopt BERT-base [61] to model its token content $c_{i,a} = [t_1, t_2, ..., t_k]$. Then we apply a commonly used attention network to compute the relations of each token, similar to [11], and average pooling is used to obtain the final textual representation Z_i:

$$Z_i = \text{Avg}(\text{SelfAtt}(\text{BERT}(c_{i,a}))). \tag{3.4}$$

Likewise, we apply the ResNet-18 [100] to encode an item with visual modality shown by an image $c_{i,b}$. Then we perform average pooling to produce the final visual representation Z_i,

followed by a standard MLP layer, which is given:

$$Z_i = \text{Avg}(\text{MLP}(\text{ResNet}(c_{i,b}))). \tag{3.5}$$

User Encoder. For user encoder, we propose using the Transformer (denoted as Trsf_u) block architecture, where each token embedding is the representation from an item encoder rather than the original word ID embedding. Position embedding $P = \{p_1, ..., p_n\}$ is added to model the sequential patterns of user behaviors. The specific process is formulated as follows:

$$S^u = Z^u + P^u, \tag{3.6}$$

$$U^u = E_u(S^u) = \text{Last}(\text{Trsf}_\text{u}(S^u)), \tag{3.7}$$

where the user representation U^u is represented by the hidden state at the last position of the user encoder.

3.3.3 Optimization

Inspired by the pre-training and fine-tuning paradigm, we apply a similar training regime for TransRec: first pre-training the user encoder network, and then training the whole framework of TransRec.

Stage 1: User Encoder Pre-training. We perform pre-training for the user encoder network in a self-supervised manner. Specifically, we apply the left-to-right style generative pre-training to predict the next item in the interaction sequence, similar to SASRec and NextItNet [32, 33]. The way we choose unidirectional pre-training rather than BERT-style (bidirectional) (i.e. Eq.3.7) is simply because unidirectional pre-training converges much faster but without obvious loss of precision. We use the softmax cross-entropy loss as the objective function:

$$\tilde{\mathbf{y}}_t = \text{softmax}(\mathbf{z}_t), \quad \mathbf{z}_t = \mathbf{S}'_t \mathbf{W}^U + \mathbf{b}^U, \tag{3.8}$$

$$\mathscr{L}_{\text{UEP}} = -\sum_{u \in U} \sum_{t=1}^{n_u - 1} \mathbf{y}_{t+1}^{(u)\top} \log \tilde{\mathbf{y}}_t^{(u)} = -\sum_{u \in U} \sum_{t=1}^{n_u - 1} \log \tilde{\mathbf{y}}_t^{(u)}[x_{t+1}^{(u)}], \tag{3.9}$$

where W^U, b^U are the projection matrix & bias terms, S'_t is the representation of last hidden layer.

Stage 2: End-to-End Training. TransRec is trained in an end-to-end manner by fine-tuning model parameters of both user and item encoders. This is vastly different from much multimodal recommendation literature where they first pre-extract offline features by modality encoder and then treat them as fixed features for a recommendation network [12, 15, 79]. End-to-end training

enables a better adaption of textual and visual features to the current recommendation domain. Specifically, we propose to use the Contrastive Predictive Coding (CPC) [16] learning method. Given a sequence of user interactions C, we divide the sequence into sub-sequence C^u and C^e to encode the relationship between them. The binary cross-entropy loss function is as follows:

$$L_{\text{CPC}} = -\sum_{u \in U} \left[\sum_{t=n+1}^{n+l} \log\left(\sigma\left(r_{u,t}\right)\right) + \sum_{g=1}^{j} \log\left(1 - \sigma\left(r_{u,g}\right)\right) \right], \tag{3.10}$$

where $_g$ is randomly sampled negative items [108, 109] during model training. For each user sequence, we choose 4 un-interactive items from the whole recommendation pool as negatives.

3.3.4 Transfer to Downstream Tasks

After the training process of TransRec, the user & item representations and their matching relationship in the source domain can be well learned, which can then be used to improve various downstream recommendation tasks by performing a direct fine-tuning on the corresponding task.

The fine-tuning process of TransRec in the target domain follows the same end-to-end training procedure (i.e. Stage 2 of Section 3.3.3) as used in the source domain. The key difference is that (i) TransRec could achieve faster and better convergence by fine-tuning on the target domain because of its well pre-trained parameters; (ii) TransRec could use fewer training examples to achieve the same performance compared to itself without pre-training. Because of the flexible architecture of DSSM, TransRec can also incorporate additional user and item features like many standard ID-based models. That is, we can fuse user/item features by simply concatenating them with user/item representation generated by the user/item encoder network (also see Figure 3.1 (b)).

During the online inference stage, TransRec is as efficient as ID-based models since the modality representations of all items can be calculated in advance by an offline manner.

3.4 Empirical Study

To verify the effectiveness of our proposed TransRec, we conduct empirical experiments and evaluate pre-trained recommenders on four types of downstream tasks.

3.4.1 Experiments for The Source Domain

Datasets. The source data is the news recommendation data collected from QQBrowser[4] from 14th to 17th, December 2020. We collect around 25 million user-item interaction behaviors, involving about 1 million randomly sampled users and 133, 000 interacted items. Each interaction

[4]https://browser.qq.com/

denotes observed feedback at a certain time, including full play and clicks. The users with less than 20 interactions are removed and the average length of user sequences is 25.

Table 3.2: Characteristics of the source dataset. 'Form' indicates the item category. 'All' means all users in this dataset, including the above three types. For example, the first line denotes that there are 765,895 users whose interacted items always have two-modal (i.e. textual and visual) features.

Form	Modality	User	Item	Interaction
Mixed	Text + Image	765,895	133,107	19,233,882
Article	Text	133,107	62,837	3,327,463
Video	Image	123,897	70,270	2,996,048
All	Text + Image	1,022,899	133,107	25,557,393

We construct the sequence behaviors for each user using his recent 32 ordered interactions. Beyond the ID features, our datasets represent each item with its raw modality features. More accurately, items in a user session can be videos-only, news-only, or both. However, each item is either a video or a piece of news, i.e. containing only one modality. We adopt item titles to represent news, item thumbnails to represent videos. The statistics are in Table 3.2.

Evaluation Metrics. We use the typical *leave-one-out* strategy [14] for evaluation, where the last item of the user interaction sequence is denoted as test data, and the item before the last one is used as validation data. The remaining sequence is used as the training data, in which the latest five interactions are used to predict given all previous interactions (see Figure 3.1). We pad zero at the end of the user sub-sequences C^u if their lengths are smaller than 25. Unlike many previous methods that employ a small scope of randomly sampled items for evaluation, which may lead to inconsistencies with the non-sampled version [110], we rank the full item set without using the inaccurate sampling measures. We apply Hit Ratio (HR) [68] and Normalized Discounted Cumulative Gain (NDCG) [32] to measure the performance of each method. Our evaluation methods are consistent for both the source and downstream tasks.

Implement Details. Hyper-parameters are searched on the validation set to ensure a fair evaluation. We notice that the Adam [111] optimizer performs the best on all validation sets. Then we set the learning rate to $1e^{-4}$, embedding size to 256 and batch size to 512 after empirical searching and training TransRec on 4 NVIDIA A100s (40G memory). We train all models until it converges and saves parameters when they reach the highest accuracy on the validation set. We set the layer number of the user encoder to 4 and the head number of multi-head attention to 4. The input size of an image is 3×224×224, and the token length of an article description is set to 32 maximum.

Results. Before evaluating TransRec on the downstream datasets, we first examine its performance in the source domain. The purpose is not to demonstrate that TransRec can achieve state-of-the-art recommendation accuracy. Since it depends on both user and item encoder net-

works — i.e. with a more expressive user and item encoders, TransRec is likely to be more powerful. Instead, we hope to investigate via fair comparison, whether learning from modality contents has advantages over traditional ID-based methods i.e. IDRec. Throughout this chapter, we use IDRec to denote the counterpart of TransRec with a similar recommendation architecture with the same sampling and optimization method, but instead with an ID embedding item encoder.

Table 3.3: Results on the source dataset. The terms below have the same meaning as in Table 3.2. TransRec- denotes TransRec without first stage user encoder pre-training.

Method	Modality	HR@5	NDCG@5	HR@10	NDCG@10
IDRec	ID	0.0141	0.0089	0.0230	0.0118
TransRec-	Image	0.0337	0.0216	0.0540	0.0281
	Text	0.0330	0.0214	0.0536	0.0280
	Mixed	0.0326	0.0210	0.0530	0.0275
	All	0.0327	0.0211	0.0532	0.0276
TransRec	Image	0.0696	0.0414	0.1128	0.0553
	Text	0.0325	0.0206	0.0582	0.0272
	Mixed	0.0390	0.0233	0.0679	0.0326
	All	0.0403	0.0239	0.0699	0.0334

Table 3.3 shows the results of IDRec and TransRec regarding HR@{5, 10} and NDCG@{5, 10}. Please note 'TransRec-' denotes TransRec without the first stage user encoder pre-training. Two important observations can be made: (1) TransRec- and TransRec largely outperform IDRec (e.g. 0.0699 vs. 0.0230, and 0.0532 vs. 0.0230 on HR@10), demonstrating strong potential of learning from modality content data. The higher results of TransRec are presumably attributed to three key factors: large-scale training data, powerful item encoder and user encoder networks, and an end-to-end training fashion. (2) TransRec exceeds TransRec- with all types of modality settings (e.g. 0.0699 vs. 0.0532, and 0.0679 vs. 0.0530 on HR@10), which evidences the effectiveness of user encoder pre-training (see Section 3.3.3). The results of (1) further motivate us to develop modality content-based recommendation models for downstream tasks.

Training Comparison. In Table 3.4, we present TransRec *vs.* IDRec in terms of FLOPs (the number of floating-point operations). As can be seen, TransRec requires almost 2000× larger computing cost than IDRec alongside up to 5× more training time. TransRec, with the standard training method, achieves better accuracy than IDRec, but with the cost of much higher computing and training time. This result is not surprising since pre-training foundation models in other fields, such as BERT-large, GPT-3, and various ViT require huge computing power. We believe the results here will inspire more research to study model optimization and compression techniques for large gpRS — e.g. fine-tuning only the top few layers or applying more advanced fine-tuning techniques, such as adapter [68, 80] and prompt tuning [112].

Table 3.4: Comparison of TransRec (full parameter fine-tuning) and IDRec in terms of FLOPs and training time per batch. TransRec is trained on a user sequence with 13 images and 12 news articles where images have a size of $3\times224\times224$ and text length is 32.

Model	FLOPs	s/batch
IDRec	32.973M	0.7s
TransRec	78.305G	3.2s

3.4.2 Experiments for The Target Domains

All the target datasets below are from other recommender systems.

TN-mixed: It was collected from Tencent News (TN)[5], where an interacted item can be either a News article or a video thumbnail. Similar to the source domain, the interacted item set of a user contains mixture-of-modality features, i.e. both text and images.

Table 3.5: Datasets for downstream recommendation tasks. Except for DouYin, interactions in other datasets are mainly about users' clicking or watching behaviours.

Domain	Modality	User	Item	Interaction
TN-mixed	Text+Image	49,639	48,383	870,894
TN-video	Image	47,004	50,053	809,483
TN-text	Text	49,033	49,142	903,543
DouYin	Image	100,000	66,228	1,688,944

TN-video and TN-text: The two datasets only contain items with a single modality. For example, users' interactions in TN-video include only videos, while TN-text includes only textual items, i.e. news. They are used to evaluate TransRec's generality for single-modal item recommendation. Given that users and items in real-world recommender systems have various additional features, we introduce two types of user features (gender and age) and one item feature (category) for TN-text.

DouYin: It was collected from DouYin[6] (the Chinese version of TikTok), a well-known short video recommendation application. Unlike all previous datasets, the positive user feedback in DouYin only contains comment behaviors. In addition, video genres and cover image size in DouYin are vastly different from the source domain. Table 3.5 summarizes the statistics of downstream datasets.

Baselines. The key baseline used to compare with TransRec is TFS, which is used to verify whether TransRec pre-trained in the source domain can effectively improve the performance of the downstream tasks. Thereby, TFS adopts the same architecture, hyper-parameters and modality inputs as TransRec, but is trained from scratch with randomly initialized parameters for the key architecture (i.e. the user encoder). Note that its item encoder still adopts the pre-trained

[5]https://news.qq.com/

[6]https://www.douyin.com/

Table 3.6: Comparison of recommendation results on four downstream domains. TFS denotes training target datasets with random parameters as initialization. It shares the same network architecture and hyper-parameters as TransRec. The best results are bolded.

Domain	Modality	Metric	IDRNN	IDCNN	IDRec	TFS	TransRec	Improv.
TN-mixed	Mixed	HR@5	0.0109	0.0112	0.0117	0.0249	**0.0285**	14.46%
		NDCG@5	0.0063	0.0067	0.0068	0.0160	**0.0177**	10.63%
		HR@10	0.0129	0.0195	0.0210	0.0428	**0.0478**	11.68%
		NDCG@10	0.0062	0.0094	0.0100	0.0213	**0.0239**	12.21%
TN-video	Image	HR@5	0.0134	0.0159	0.0153	0.0208	**0.0271**	30.29%
		NDCG@5	0.0093	0.0098	0.0092	0.0131	**0.0173**	30.06%
		HR@10	0.0201	0.0265	0.0267	0.0336	**0.0424**	26.19%
		NDCG@10	0.0114	0.0133	0.0125	0.0173	**0.0221**	27.75%
TN-text	Text	HR@5	0.0105	0.0123	0.0105	0.0303	**0.0358**	18.15%
		NDCG@5	0.0063	0.0078	0.0062	0.0192	**0.0227**	18.23%
		HR@10	0.0189	0.0220	0.0192	0.0500	**0.0597**	19.40%
		NDCG@10	0.0089	0.0109	0.0090	0.0255	**0.0303**	18.82%
DouYin	Image	HR@5	0.0059	0.0057	0.0023	0.0115	**0.0146**	26.96%
		NDCG@5	0.0037	0.0035	0.0014	0.0073	**0.0090**	23.29%
		HR@10	0.0096	0.0100	0.0035	0.0205	**0.0259**	26.34%
		NDCG@10	0.0049	0.0049	0.0018	0.0101	**0.0126**	24.75%

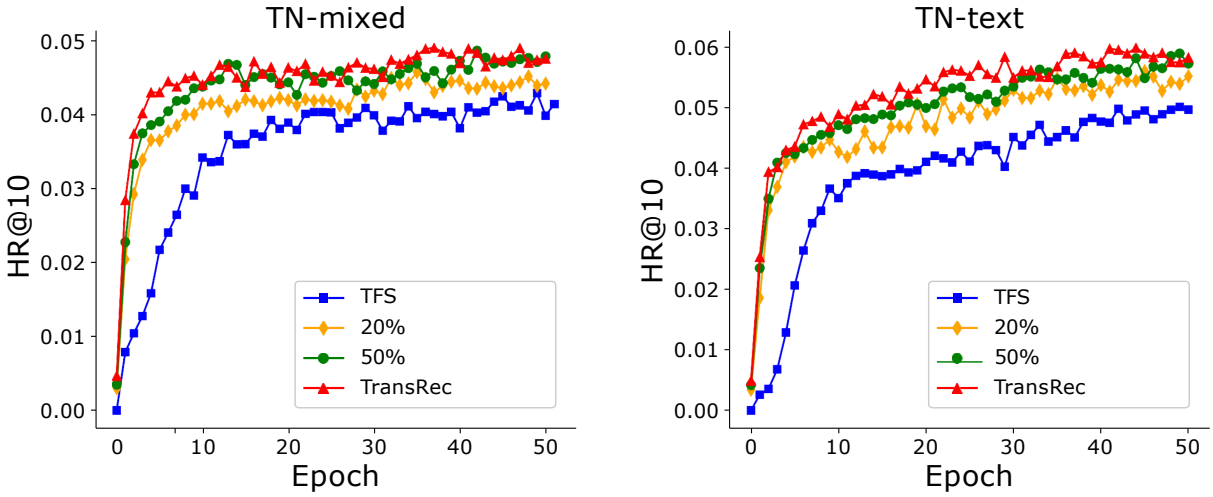


Figure 3.3: Convergence trend by scaling the source data.

BERT and ResNet. In addition, we compare TransRec with several ID-based baselines for reference, including IDRNN [26], IDCNN [32] and IDRec. IDRNN uses the GRU to encode user sequence, while IDCNN and IDRec use the temporal CNN (TCN) and Transformer architecture to encode user sequence.

For a fair comparison, we adopt IDRNN, IDCNN and IDRec with exactly the same training objective function, negative sampling method and optimization strategy as the fine-tuned TransRec (see Figure 3.1). All the batch size of target domains is set to 128 and the learning rate of TN-mixed, TN-video, TN-text and Douyin is set to $1e^{-5}$, $1e^{-5}$, $1e^{-5}$, and $5e^{-5}$, respectively. Despite that, we emphasize again that the purpose of this study is neither to propose a more advanced neural recommendation architecture nor to pursue some state-of-the-art results. The key purpose of this study is to indicate that: (1) learning from modality content features instead of ID features achieves the goal of transferable recommendations across different domains; (2) learning from MoM feedback rather than single-modal or typical multimodal feedback reaches the goal of generic recommendations across different modalities.

Results. The overall results are shown in Table 3.6, which includes four recommendation scenarios. Two important observations can be made: (1) TransRec performs consistently better than its training-from-scratch version, i.e. TFS; (2) TransRec performs better than ID-based methods as well. The results suggest that training the source domain brings much better results for TransRec on all target datasets. By analyzing these scenarios, we can conclude that TransRec — learning from MoM feedback — can be broadly transferred to various recommendation scenarios, including the source-like mixed-modal scenario (i.e. TN-mixed), the single-modal scenario (TN-video), the scenario with more additional features (TN-text), and the scenario with very different modality content (DouYin).

3.4.3 Scaling Effects

Scaling effects of the source dataset. Table 3.7 shows the model performance in the four downstream recommendation tasks. First, it can be seen that the recommendation accuracy of TransRec is improved by scaling up the source training data. For example, HR@10 in the TN-mixed dataset grows from 0.0428 to 0.0448 with 20% of the source data, and then it grows from 0.0448 to 0.0474 with 50% of the source data. Such a property of TransRec is desired since it implies that scaling up the source dataset is an effective way to improve downstream tasks. We also plot the convergence behavior in Figure 3.3, which shows consistent improvements.

Scaling effects of the target dataset. We study the effects of TransRec by scaling down the target data, aiming to verify whether TransRec can alleviate the insufficient data issue. Specifically, we decrease the size of these target datasets by using 20% and 60% of the original data. Results are shown in Table 3.8 and Figure 3.4. It can be seen that (1) TransRec's accuracy increases with more training data; (2) more accuracy gains are achieved with less training data. This suggests that when a recommender system lacks training data, transferring the (user-item)

Table 3.7: Results of TransRec by scaling up the source corpus.

Domain	Metric	TFS	20%	50%	TransRec
TN-mixed	HR@10	0.0428	0.0448	0.0474	0.0478
	NDCG@10	0.0213	0.0227	0.0237	0.0239
TN-video	HR@10	0.0336	0.0400	0.0417	0.0424
	NDCG@10	0.0173	0.0209	0.0214	0.0221
TN-text	HR@10	0.0500	0.0543	0.0581	0.0597
	NDCG@10	0.0255	0.0281	0.0292	0.0303
DouYin	HR@10	0.0205	0.0233	0.0254	0.0259
	NDCG@10	0.0101	0.0113	0.0120	0.0126

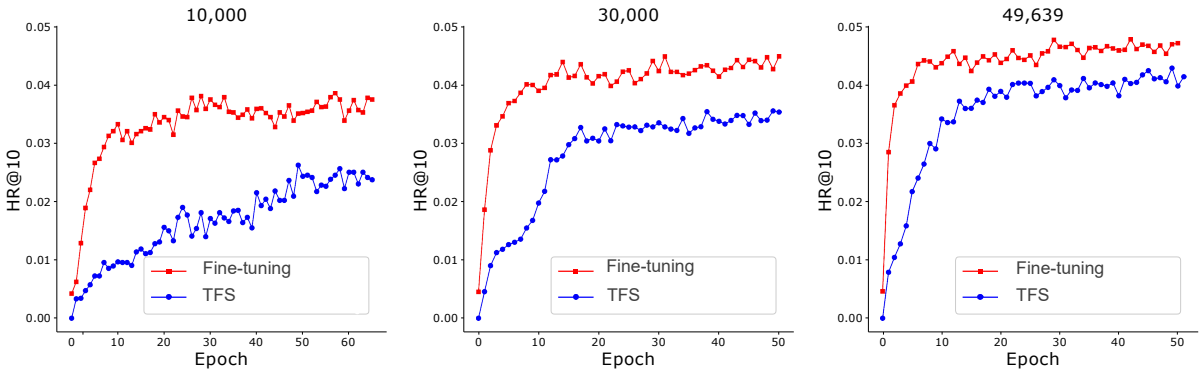


Figure 3.4: Comparison of convergence by scaling TN-mixed dataset.

matching relationship from a large source dataset is helpful.

Table 3.8: Comparison of relative performance improvement on downstream tasks with varied target data size. 'Num. Sample' denotes the number of user behavior sequences for training. 'Improv.' indicates the relative performance improvement of TransRec compared with TFS.

Domain	Num. Sample	HR@10		Improv.	NDCG@10		Improv.
		TFS	TransRec		TFS	TransRec	
TN-mixed	10,000	0.0261	0.0385	41.51%	0.0126	0.0193	53.17%
	30,000	0.0354	0.0448	26.55%	0.0176	0.0223	26.70%
	49,639	0.0428	0.0478	11.68%	0.0213	0.0239	12.21%
TN-text	10,000	0.0393	0.0597	51.91%	0.0201	0.0262	30.35%
	30,000	0.0453	0.0549	21.19%	0.0230	0.0283	23.04%
	49,033	0.0500	0.0597	19.40%	0.0255	0.0303	18.82%

Table 3.9: End-to-end training vs. frozen features.

Method	Manner	TN-mixed		TN-text		TN-video	
		HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
TFS	Frozen	0.0334	0.0167	0.0350	0.0176	0.0037	0.0017
	End2end	0.0428	0.0213	0.0500	0.0255	0.0336	0.0173
TransRec	Frozen	0.0359	0.0181	0.0411	0.0206	0.0040	0.0023
	End2end	0.0478	0.0239	0.0597	0.0303	0.0424	0.0221

3.4.4 End-to-End Training v.s. Frozen Features.

Traditional multimodal and multimedia recommendations have been well studied [79]. While due to high computing resources and a less powerful text/image encoder network, prior art tends to extract frozen modality features and feed them into a CTR or recommendation model. Such practice is prevalent for industrial applications given billions of training examples [13, 99]. However, we want to explore whether end-to-end learning is superior to learning from frozen features. The results are in Table 3.9. We can achieve consistent improvements through end-to-end learning; however, finetuning BERT and ResNet is more computationally expensive than using pre-extracted features. Notably, we notice that frozen textual features yield worse results than visual features. This may imply that The textual features generated by BERT are more general than the visual features generated by ResNet. This finding is also aligned with findings in NLP and CV fields — finetuning all parameters is generally better than finetuning only the classification layer (with the backbone network frozen).

3.5 Conclusion

In this chapter, we study a novel recommendation scenario where user feedback contains items with the mixture-of-modality features. We develop TransRec, the first recommendation model learning from MoM feedback in an end-to-end manner with the goal of learning general-purpose models for recommender systems. To show its transferring ability, we conduct an empirical study in four types of downstream recommendation tasks. Our results verify that TransRec is a generic model that can be broadly transferred to improve many recommendation tasks as long as the modality has been trained in the source domain. Our work has significant practical implications towards universal recommender systems to realize 'One Model to Serve All' [69, 101].

One limitation is that we only examine TransRec with two types of modality features (image and text). As a result, it can only serve three scenarios: image-only, text-only, and image-text. Since both video and audio data can be represented by images [113, 114], intuitively, TransRec can be extended to scenarios where items involve more modalities. For example, suppose four distinct modalities (image, text, audio and video) are available in user feedback. TransRec can potentially serve at most 15 types of scenarios, covering most modalities for multimedia data. This is an interesting future direction to realize a more general recommender system. A second limitation is the high training cost of TransRec because of the end-to-end learning paradigm. Although TransRec enables effective transfer learning for many downstream recommendation tasks, its huge computation cost and training time should receive more future research attention.

3.5.1 Summary and Link to Next Chapters

This chapter establishes the representation layer of the thesis workflow by learning transferable user/item encoders from MoM content rather than platform-specific IDs. The empirical results demonstrate that MoM-based pretraining improves cross-domain performance under non-overlapped identities. Building on this transferable foundation, Chapters 4–5 move to the decision layer, where recommendation is conditioned on multi-objective returns to enable controllable trade-offs beyond accuracy.

Chapter 4

Decision Transformers for Reward-driven Multi-objective Recommendations

Note. This chapter is based on our TKDE 2025 paper on MODT4R, and is revised in this thesis to align the MOSR task definition and notation with the thesis-wide framework, strengthen motivation for return-conditioned decision making, and standardize evaluation reporting for multi-objective trade-offs.

4.1 Introduction

In e-commerce[88] and video platform[115], it is common to recommend the next relevant item to users based on the interacted items within an active session. Such Sequential Recommendation methods [24–26] are usually trained to produce recommendations given user's interest from sequential user-item interactions, where the next item is the ground truth label for self-supervised learning. Hidasi et al. [26] first proposed modeling user sequences based on gated recurrent units (GRU) for recommending next items. Since then, sequential models, such as convolutional neural networks (CNN) [116] and Transformer [33], have been adapted to handle the SR task.

Many prior works [6, 57, 60] focused on optimizing the objective of recommending accurate items for users, which refers to the efficacy of ranking relevant items and evaluated by HR and NDCG metrics. In reality, a strictly relevance-oriented RS may not guarantee a good user experience in terms of diversity and novelty, which have emerged as pivotal elements in boosting user engagement, since a diverse and novel list of relevant items is apt to meet the fluctuating requirements of users. This holds particularly true for e-commerce, music, and video streaming services, where if users are confined to a limited segment of the broad item space [117], the value of the recommendations decreases due to repetitive exposure to analogous content. Therefore, Multi-objective Sequential Recommendation (MOSR) [52] is proposed to offer recommendation lists that balance accuracy with diversity and novelty to enhance user satisfaction.

However, how to integrate multiple objectives into a single model to efficiently accomplish the tasks of the MOSR remains an unresolved issue. In recommendation methods outside the SR domain, Multi-objective optimization (MOO) approaches that follow Pareto principles [118, 119] require selection methods to identify the best solution from a set of optimal solutions. MOO methods have been applied to some multi-objective recommendation areas to address the challenges associated with balancing multiple objectives, such as averaging multiple objectives [48, 120] or determining the optimal solution via a single objective [121], yielding sub-optimal outcomes due to the conflicting objectives in such settings. While evolutionary algorithms [122, 123] can effectively discover optimal solutions, this efficacy may be countered by implementation inefficiencies or increased diversity of solutions. Scalarization methods [124] that recast the problem into a single-objective problem have been used as a means of achieving simplicity and efficiency with large datasets. Previous studies [48, 50] also aim to address the challenge of weight assignment and Pareto optimal solution selection through reinforcement learning. Nonetheless, these methods require training multiple models for multiple objectives and applying ensemble methods, making them inefficient and complex.

Moreover, integrating multiple objectives with the user state to effectively generate desired items through RL agent still poses significant challenges. Motivated by Self-supervised Reinforcement Learning-based Sequential Recommendation (SRLSR) [57, 125] that improve recommendation accuracy in various user behaviors, e.g., purchases and clicks, [52] first proposed Sequential Multi-Objective Reinforcement Learning (SMORL) to balance recommendation performance of accuracy, novelty, and diversity. SMORL trained the RL agent through multiple rewards to actions based on Q-learning in the offline setting, as shown in Figure 4.1a. Specifically, a user sequence is treated as the state, and the next interacted item is considered the action. The agents are trained to interact with the environment (the user) by taking actions (recommending items) and obtain multiple rewards responded by the user. However, this decision process requires the update of multiple agent heads, each tailored to optimize a different objective. Managing these multiple heads can complicate the training process, as each must be synchronized and effectively integrated to ensure that the overall system remains balanced and effective across all targeted metrics. Furthermore, the single-step optimization approach used in SMORL fails to capture the full scope of user state transitions and tends to oversimplify the nuanced progression of user interests and behaviors over time. This can lead to a skewed model learning, predominantly favoring frequently recommended items, and potentially overlooking less obvious but equally valuable recommendations.

To solve the above issues, we propose Multi-objective Decision Transformer for Reward-driven Recommendation (MODT4R) to trade-off recommendation accuracy, diversity and novelty in sequential recommendation. We formulate multi-objective sequential recommendation as an offline RL problem, which is taken as as sequence modeling and solved via supervised learning [30]. First, we propose multi-objective returns infused with user state to capture the user's

signals of preferences on accurate, novel and diverse items. The items are predicted conditioned on the state and returns to balance the recommendation results, as shown in Figure 4.1b and Figure 4.2. Specifically, MODT4R contains two layers: User Transformer (UserTrans) maintains effective user states within a sequence by the ordered interacted items; Multi-objective Transformer (MoTrans) models the user trajectory consisting of states multi-objective returns, and previous actions to predict next items.

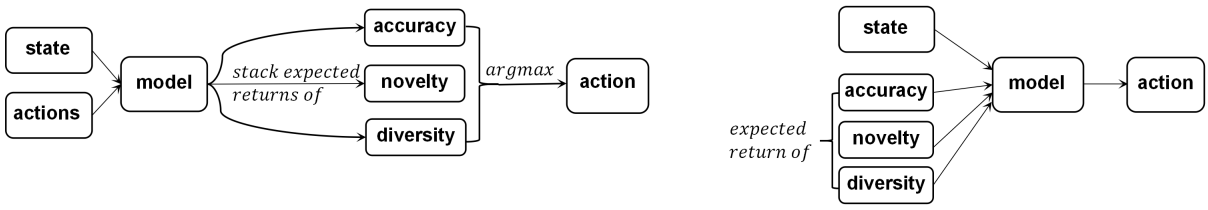
In the training stage, MODT4R relies only on the offline sequential data. The user trajectory is generated by the sequences, and error-prone explorations are not required to avoid huge costs. Both UserTrans and MoTrans are trained via supervised loss, where actions (items) are recommended conditioned on all sequential states and returns. We co-train the supervised loss with multi-objective return prediction loss in MoTrans to enhance recommendation stability in balancing multiple objectives. During inference, a score function is proposed to adjust the ranking of items considering different objectives: recommendations are determined by both predictions from UserTrans and MoTrans. The former provides foundational recommendation accuracy, while increasing the weight of the latter can enhance novelty and diversity performance with minimal impact on accuracy, and vice versa. Additionally, MODT4R allows adjusting diversity returns at inference time to regulate objectives. Extensive experimental results show the superior performance of MODT4R on the MOSR task.

Our contributions can be summarised as follows:

- We introduce MODT4R, a novel framework designed to effectively address the balance between accuracy, diversity, and novelty for Multi-objective Sequential Recommendation.
- We propose a novel user trajectory, consisting of user state transitions and multi-objective returns within a sequence, for MODT4R to effectively predict the next items through simple supervised learning.
- We propose a flexible scoring function that adeptly adjusts and balances accuracy and diversity during the inference stage, allowing for tailored recommendation strategies.
- We evaluate MODT4R on real-world datasets to demonstrate that our method can achieve substantial improvement in diversity and novelty, and maintaining or improving accuracy compared to the state-of-the-art method.

4.2 Related Work

Research in Sequential Recommendation has evolved from early recurrent neural networks [26] to Transformer-based models [33], with a long-standing focus on enhancing recommendation



(a) MORL (b) MODT

Figure 4.1: (a) Traditional RL-based MOSR methods predict the multiple expected returns (i.e., cumulative rewards) to select action and update the recommender via the highest return. MODT (b) directly predicts action given the state and multi-objective expected returns.

accuracy. However, an overemphasis on accuracy often compromises the diversity and novelty of recommendation lists, thereby impairing user experience and giving rise to the Multi-objective Sequential Recommendation (MOSR) task [52]. Existing approaches primarily fall into two categories. The first involves Multi-Objective Optimization (MOO) strategies like Pareto optimization [48, 49], which often lead to sub-optimal solutions or are complex to implement. The second category is based on RL. While RL has been successfully applied in SR to improve accuracy [57, 58], state-of-the-art models in MOSR, such as SMORL [52] and its variant SMONAC [53], exhibit significant limitations. These methods typically rely on single-step Q-learning, fail to capture the long-term dynamics of user state evolution, and require training separate RL agents for each objective, making them complex, inefficient, and difficult to scale.

To address these challenges, this chapter draws inspiration from the DT [30], a novel paradigm that recasts offline RL as a conditional sequence modeling problem. By taking trajectories of desired cumulative rewards (returns), states, and actions as input, DT leverages a Transformer architecture to predict actions through supervised learning, effectively circumventing the bootstrapping errors and instability of traditional temporal difference learning. Inspired by this, our work introduces the DT framework to the MOSR domain for the first time. Our proposed MODT4R model constructs user trajectories that incorporate both user state transitions and multi-objective returns—corresponding to accuracy, diversity, and novelty—enabling the recommendation process to be conditioned on desired multifaceted goals. This approach not only simplifies training via simple supervised learning but also more comprehensively utilizes sequential information to balance conflicting objectives, directly addressing the shortcomings of existing methods.

4.3 Preliminary

4.3.1 Task Definition

The sequential recommendation or next item recommendation task aims to recommend next item x_{t+1} from the entire item set I that aligns with user's interest, given a recent sequence of user-item interactions $x_{1:t} = \{x_1, x_2, \ldots, x_{t-1}, x_t\}$, where $x_i \in I (0 < i \leq t)$ is the index of the interactions ordered by timestamp. Traditional SR [24, 25] primarily focus on recommendation accuracy, we extend it to Multi-objective Sequential Recommendation (MOSR) [52] that balances accuracy with multiple objectives to improve user experience. Specifically, RSs recommend the most relevant items to a user's historical interactions that are often influenced by popularity, thereby neglecting long-tail (new and unknown) items and reducing diversity and novelty of recommendation results. Therefore, the goal of MOSR is to maintain or sacrifice minimal accuracy while recommending novel and diverse items for users.

4.3.2 Markov Decision Process Formulation of MOSR

The MOSR can be seen as a Markov Decision Process (MDP) [52] that can be described by tuple $(\mathscr{S}, \mathscr{A}, \mathbf{P}, \mathbf{R})$. Specifically, $\mathscr{S}$ is the user state, $\mathscr{A}$ is the action space (candidate items), $\mathbf{R}$ is the reward function, and $\mathbf{P}$ is the state transition probability. In recommendation, users are seen as the environment $\mathscr{E}$ and the recommender is the agent that is trained to interact with users by taking action a_t (recommending item) at timestamp t based on the state s_t from environment. In traditional offline RL methods, the environment continually generates new states s_{t+1} and feedback (reward) r_t for the agent as users respond to the recommendations. The process generates the user trajectory $\tau = (s_1, a_1, r_1, s_2, a_2, r_2, ...)$. Traditional Q-learning based methods [57] aim to learn a policy $\pi_\theta(a \mid \mathrm{s})$ that can obtain the maximum expected cumulative rewards $\mathbb{E}\left[\sum_{t=1}^{T} r_t\right]$ by mapping the state $s \in \mathscr{S}$ into candidate items (actions) $a \in \mathscr{A}$ based on the Bellman Equation.

4.3.3 Research Objective

To this end, this chapter aims to reformulate the process of MOMDP for the Multi-objective Sequential Recommendation task and to better trade-off accuracy, diversity and novelty. We then present our proposed MODT4R to a multi-objective recommendation setting by proposing multiple desirable rewards and constructing the corresponding trajectories.

4.3.4 Notations

This chapter contains numerous notations. To facilitate understanding reading, we illustrate the important notations and corresponding descriptions in Table 4.1.

Table 4.1: An illustration of important notations in this chapter.

Notation	Description
I	The set of candidate items.
x_t	The user-item interaction at timestamp t.
$x_{1:T}$	A user-item sequence with T interactions, $x_{1:T} = \{x_1, ..., x_T\}$.
s_t	The user state at timestamp t.
$s_{1:T}$	The user state transitions of a user-item sequence, $s_{1:T} = \{s_1, ..., s_T\}$.
a_t	The action taken at timestamp t, it's the next interaction in a user-item sequence.
$r_{t,acc}$	The accuracy reward by taking action a_t at state s_t.
$R_{t,acc}$	The cumulative rewards of accuracy (accuracy return) from timestamp t for a sequence with T actions.
$r_{t,div}$	The diversity reward by taking action a_t at state s_t.
$R_{t,div}$	The cumulative rewards of diversity (diversity return) from timestamp t for a sequence with T actions.
$r_{t,nov}$	The novelty reward by taking action a_t at state s_t.
$R_{t,nov}$	The cumulative rewards of novelty (novelty return) from timestamp t for a sequence with T actions.
$R_{1:T,o}$	The cumulative rewards(return) of objective o at each timestamp t for a sequence with T actions, where $o \in \{acc, nov, div\}$.
H_t	The user trajectory at each timestamp t, $H_t = \langle s_t, R_{t,acc}, R_{t,nov}, R_{t,div}, a_t \rangle$.
H	The user trajectory of a sequence with T actions, $H = \{H_1, ..., H_T\}$.

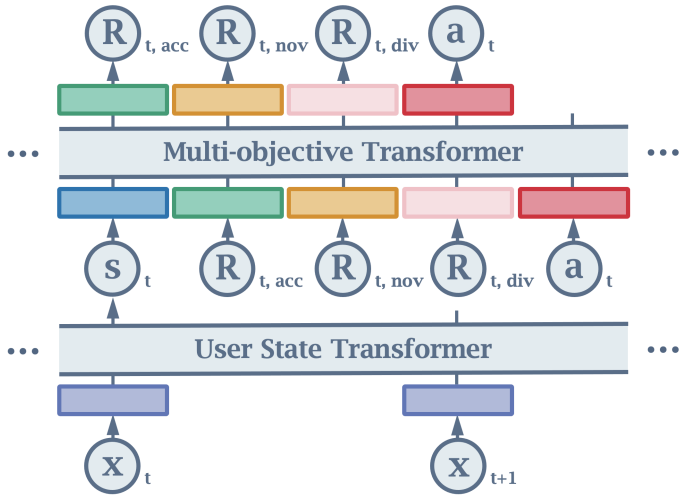


Figure 4.2: Our proposed Multi-objective Decision Transformer for Recommendation (MODT4R). MODT4R is the multi-objective sequential recommendation framework, which consists of a User State Transformer (UserTrans) and a Multi-objective Transformer (MoTrans). UserTrans encodes user sequences and explicitly generates user states to help sequence modeling in Multi-objective Transformers. MoTrans considers multi-objective returns $R_{t,acc}$, $R_{t,nov}$, and $R_{t,div}$ in the same trajectory. The user trajectory is formulated as $\langle \ldots, s_t, R_{t,acc}, R_{t,nov}, R_{t,div}, a_t, \ldots \rangle$.

4.4 Method

Traditional offline RL-based recommender mimics the online process by observing the rewards w.r.t specific behaviors from users and is trained to maximize the expected cumulative rewards (expected returns) [52, 57]. To address the complexity in a simple and effective manner, we propose the Multi-objective Decision Transformer for Reward-driven Recommendation (MODT4R) to address the MOSR task. MODT4R recasts the offline RL as sequence modeling, predicting actions in a supervised manner conditioned on the multi-objective returns and states.

4.4.1 Overall Framework

Figure 4.2 illustrates our proposed MODT4R, which consists of two layers: the User State Transformer (UserTrans) generates dynamic user state as the sequence evolves; the Multi-objective Transformer (MoTrans) incorporates reward-driven properties/objectives (i.e., accuracy, novelty, and diversity) represented by multi-objective returns into the trajectory to predicts items. The actions (items) are generated under the guidance of objective-oriented returns to balance accuracy with diversity, novelty objectives. The trajectory is formed as:

$$H = \left\langle \ldots, s_t, R_{t,acc}, R_{t,nov}, R_{t,div}, a_t, \ldots \right\rangle. \tag{4.1}$$

The details of the components are:

- State space $\mathscr{S}$: $s_t \in \mathscr{S}$ is the user state at timestamp t generated by the user-item interactions $x_{1:t}$ in sequential recommendation.
- Accuracy return $\mathscr{R}^{acc}$: $R_{t,acc} \in \mathscr{R}^{acc}$ represents the number of interactions contributed by remaining items from timestamp t in a sequence.
- Novelty return $\mathscr{R}^{nov}$: $R_{t,nov} \in \mathscr{R}^{nov}$ represents the novelty achieved by the remaining user-item interactions from timestamp t in sequence, which reflects a user's preference for novel items in the system.
- Diversity return $\mathscr{R}^{div}$: $R_{t,div} \in \mathscr{R}^{nz}$ denotes the diversity achieved by remaining interactions from step t in the session C_c, which reflects a user's preference for diverse items in the system.
- Action space $\mathscr{A}$: $\mathscr{A}$ is the discrete action set consisting of candidate items. In the offline recommendation, the action $a \in A$ at time step t in a sequence is the next interacted item, i.e., $a_t = x_{t+1}$.

We can see the overall training process from Figure 4.2. Initially, each item x_i transformed into the embedding $x_i \in R^d$, where d is the embedding size. Then, the user state s_t at timestep t is generated by mapping the sequence $x_{1:t}$ into the state encoder, which is the UserTrans.

Subsequently, the state s_t, along with objective returns $R_{t,acc}, R_{t,nov}, R_{t,div}$ and action a_t form the input trajectory H. MoTrans, a causally masked Transformer, takes as input the user trajectory H to predict next item (action) a_t in the sequence conditioned on state and multi-objective returns.

In the next section, the details of modeling user state, design of multi-objective returns, and action prediction are introduced.

4.4.2 User State

The UserTrans, acting as a state encoder, models user-item interactions $x_{1:T} = \{x_1, ..., x_T\}$ to produce the user state $s_{1:T} = \{s_1, ..., s_T\}$ for the input trajectory of MoTrans:

$$s_{1:T} = UserTrans(x_{1:T}), \tag{4.2}$$

where δ is the activation function.

4.4.3 Multi-objective Returns

Multi-objective return functions are proposed for each recommendation objective individually. Specifically, accuracy aims to recommend relevant items; novelty improves displaying of items that are rarely seen; diversity is to present as many items as possible from the entire system. The following introduction of each return function is based on a user sequence $\mathbf{x}_{1:T} = \{x_1, \ldots, x_T\}$, where $\{x_1, \ldots, x_T\}$ is the user sate with T steps, actions at each timestamp correspond to $\{a_1, \ldots, a_T\} = \{x_2, \ldots, x_{T+1}\}$. In addition, the process to generate multiple returns for the offline training data is shown in Algorithm 1.

Accuracy Return.

The accuracy return $R_{t,acc}$ at time step t is defined as the cumulative rewards $r_{t,acc}$ of actions from step t to T in the sequence, which is set to 1 for all types of interactions, i.e. clicks, and purchases. The accuracy return function is defined as:

$$R_{t,acc} = \sum_{t'=t}^{T} r_{t',acc} = T - t + 1, \tag{4.3}$$

where T denotes the total steps of a user-item interaction sequence. The setting of accuracy return can mitigate the bias caused by the significant difference in popularity between items, ensuring that each item is treated equitably.

Novelty Return.

The novelty returns $R_{t,nov}$ presents user's future interest in novel/new items in a sequence with T steps, which is the cumulative novelty reward $r_{t,nov}$ of the remaining interactions from time

step t to T. The novelty return function is formulated as:

$$R_{t,nov} = \sum_{t'=t}^{T} r_{t',nov}, \tag{4.4}$$

where $r_{t',nov}$=1 if action $a_{t'}$ is in the less popular set I_{nov}. Otherwise, $r_{t',nov}$=0. The setting is based on the assumption that items rarely interacted with are generally novel to users. We obtain the popularity of items based on their interacted frequency in the training data. The items in the bottom 90% are considered as the less popular item set I_{nov}, as these items constitute the long tail in all datasets.

Diversity Return.

The diversity return $R_{t,div}$ captures the extent to which a user explores items that are distinct from their historical interactions. Thus, we employ the difference in similarity between sequence $x_{1:t}$ and sequence $x_{1:t+1}$, changed by taking action a_t at step t, as its diversity reward $r_{t,div}$:

$$r_{t,div} = Sim(x_1, ..., x_t) - Sim(x_1, ..., x_t, a_t), \tag{4.5}$$

where $a_t = x_{t+1}$. The similarity of a sequence with $t > 1$ interactions is the average cosine similarity of the interacted items [1]. $Sim(x_1)$ is set to 1 when computing the $r_{1,div}$. Thereby the diversity return $R_{t,div}$ is computed as the cumulative diversity rewards starting from the tiemstamp t to T :

$$R_{t,div} = \sum_{t'=t}^{T} r_{t',div} = Sim(x_1, ..., x_t) - Sim(x_1, ..., x_T, a_T) = 1 + Sim(x_1, ..., x_t), \tag{4.6}$$

where $Sim(x_1, ..., x_T, a_T)$ equals to $Sim(x_1, ..., x_T, x_{T+1})$, which is set to the minimum similarity value -1 to ensure the the sequence have maximum diversity (dissimilarity).

4.4.4 Action Prediction

MoTrans models the sequence of user states, multi-objective returns, and actions to predict the next item. [2] Since all the returns are discretized, we map them into the same dimension as the state and action embeddings by a trainable embedding $E_o \in \mathbb{R}^d$, i.e., $R_{t,o} = R_{t,o} \cdot E_o$. We also add the positional embeddings $P \in \mathbb{R}^{T \times d}$ to the input tokens for each timestep. The input embeddings at timestep t are formulated as:

$$s_t = s_t + P_t, \quad R_{t,o} = R_{t,o} + P_t, \quad a_t = a_t + P_t, \tag{4.7}$$

[1] $Sim(x_1, ..., x_t) = \frac{2}{t(t+1)} \sum_{t_1=1}^{t} \sum_{t_2=t_1+1}^{t} \cos(x_{t_1}, x_{t_2})$

[2] The architecture of MoTrans is GPT [126].

where $t \in \{1,...,T\}$, $o \in O = \{acc, nov, div\}$. $E_o \in \mathbb{R}^d$ is the trainable embeddings for multi-objective returns, respectively. We stack the tokens at each step to obtain the input trajectory representations $H = \{H_1,...,H_T\}$ as described in Eq. 4.1, which are then fed into the MoTrans to output the sequential representations $\tilde{H} = \{\tilde{H}_1,...,\tilde{H}_T\}$. The process is formulated as:

$$H_t = stack(s_t, R_{t,acc}, R_{t,nov}, R_{t,div}, a_t), \tag{4.8}$$

$$\tilde{H} = MoTrans(H), \tag{4.9}$$

$$\tilde{R}_{t,acc}, \tilde{R}_{t,nov}, \tilde{R}_{t,div}, \tilde{a}_t = unstack(\tilde{H}_t), \tag{4.10}$$

where $t = \{1,...,T\}$. The final representations of returns and actions at each step t are extracted from $\tilde{H}_t$.

We map the multi-objective return output $\tilde{R}_{t,o}$ to the predicted returns using a linear layer. Additionally, the final state representations are fed into a fully connected layer to produce the logits for the item set:

$$\hat{R}_{t,o} = W_o \tilde{R}_{t,o} + b_o, \tag{4.11}$$

$$\hat{A}_t = \delta(W_a \tilde{a}_t + b_a), \tag{4.12}$$

where $\hat{A}_t = [a_1, a_2, \ldots, a_n]$ denotes the n classification logits at each timestep t on n candidate actions, δ denotes the activation function, $W_o \in \{W_{acc}, W_{div}, W_{nov}\} \in \mathbb{R}^{d\times 1}$ and $W_a \in \mathbb{R}^{d\times N}$ are trainable parameters, $b_o \in \{b_{acc}, b_{div}, b_{nov}\} \in \mathbb{R}^1$ and $b_a \in \mathbb{R}^N$ are bias vectors.

4.4.5 Training Objective

The overall learning optimization process of MODT4R consists of an item prediction loss in UserTrans, an action loss in MoTrans, and Multi-objective return oredcition loss in MoTrans.

UserTrans is trained to generate next items of a sequence via mapping the state s_t into n classification logits on the candidate items, which is optimized via the cross-entropy loss:

$$\hat{Y}_t = \delta(W_u s_t + b_u), \tag{4.13}$$

$$\mathscr{L}_S = -\sum_{i=1}^{n} Y_i \log(p_i), \tag{4.14}$$

where $\hat{Y}_t = [y_1,...,y_n]$, n is the number of candidate items, y_i is the logit of i-th items. $Y_i = 1/0$ denotes if the i-th item is interacted in the next step or not. $W_u \in \mathbb{R}^{d\times N}$ is trainable parameters, $b_u \in \mathbb{R}^N$ are bias vector. $p_i = e^{y_i}/\sum_{i'=1}^{n} e^{y_{i'}}$.

In MoTrans, the returns are predicted by the Mean Squared Error (MSE) loss function $\mathscr{L}_R$ defined as :

$$\mathscr{L}_R = 1/T \sum_{o\in O} \sum_{t=1}^{T} (R_{t,o} - \hat{R}_{t,o})^2, \tag{4.15}$$

where $R_{t,o}$ is the target returns at timestep t. The multi-return loss $\mathscr{L}_R$ acts as an additional learning task to enhance stability in balancing the recommendation performance between accuracy, diversity, and novelty.

In addition, the actions for the input sequence are predicted via cross-entropy $\mathscr{L}_a$:

$$\mathscr{L}_a = 1/T \sum_{t=1}^{T} \sum_{i=1}^{n} A_{t,i} \log(\hat{a}_{t,i}), \tag{4.16}$$

where $\hat{a}_{t,i} = e^{a_{t,i}} / \sum_{i'=1}^{n} e^{a_{t,i'}}$. $A_{t,i} = 1/0$ denotes if the i-th item is interacted or not. Combining with $\mathscr{L}_S$ in Section 4.4.2, the final loss function of MODT4R is formulated as:

$$\mathscr{L} = \mathscr{L}_S + \mathscr{L}_a + \beta \mathscr{L}_R, \tag{4.17}$$

where β controls the importance of return prediction loss $\mathscr{L}_R$ in the decision process.

Algorithm 1 Multi-return generation and training procedure of MODT4R.

Input: training set $\mathscr{D}_{ts}$, model MODT4R
Output: parameters in the learning space θ

Initialize trainable parameters
repeat
  Sample a user sequence $x_{1:T}, x_{2:T+1}$ from $\mathscr{D}_{ts}$
  Compute $R_{1:T,acc}$, $R_{1:T,nov}$, $R_{1:T,div}$ for each sequence via Eq. 4.3, Eq. 4.4, Eq. 4.6, respectively.
  Compute $s_{1:T}, R_{1:T,acc}, R_{1:T,nov}, R_{1:T,div}, a_{1:T}$ via Eq. 4.7
  Compute $\tilde{H}_{1:T}$ via Eq. 4.8 - Eq. 4.9
  Compute $\tilde{R}_{1:T,acc}, \tilde{R}_{1:T,nov}, \tilde{R}_{1:T,div}, \tilde{a}_{1:T}$ via Eq. 4.10
  Compute loss $\mathscr{L}_S$ according to Eq. 4.13
  Compute loss $\mathscr{L}_R$ via Eq. 4.15
  Compute loss $\mathscr{L}_a$ via Eq. 4.16
  Compute overall loss $\mathscr{L}$ via Eq. 4.17
  Update parameters by $\nabla_\theta \mathscr{L}$
until converge
return parameters in θ

4.4.6 Inference Setting

In this section, we first outline how to establish the sequence returns for each objective during the inference phase. Subsequently, we present the function for obtaining the final scores that consider multiple objectives across all items.

Inference Returns.

The sequential multi-objective returns are generated using offline data for training. However, during the inference stage, we need to provide the model with a desirable return at time step

t for the input user sequence $x_{1:p} = \{x_1, ..., x_p\}$ to predict the next action $a_p = x_{p+1}$. The provided returns determine how the model will perform, and we demonstrate the return generation strategy in the following:

- Accuracy return: produced by Eq. 4.3, where $T = p + 1$.
- Novelty return: we set the novelty reward at the predicted position p according to previous interactions:

$$r_{p,nov} = \begin{cases} 1.0 & \text{if mean reward of previous } p-1 \text{ steps} \geq \varepsilon \\ 0.0 & \text{otherwise} \end{cases} \tag{4.18}$$

  where ε is the novelty threshold that determines whether the predicted item is novel or not. The sequential novelty cumulative rewards are generated by the strategy in Eq. 4.4.
- Diversity return: produced by Eq. 4.6. To flexibly change diversity objective, we use a hyperparameter λ to control the importance of diversity return, i.e., $R_{t,div} = \lambda R_{t,div}$.

To be clear, we outline the process of return generation for a single input sequence to predict the next item in Algorithm 2, which is employed to all test sequences during the inference stage. The multi-objective returns and the user-item interactions form the complete inputs to recommended items for users.

Next Item Recommendation.

We combine both UserTrans and MoTrans to produce recommendation scores for testing. Their predicted scores on actions (i.e., next items) are added to rank all the candidate items:

$$Score_p = \hat{A}_p + \alpha \hat{Y}_p, \tag{4.19}$$

where $\hat{A}_t = [a_1, ..., a_n]$ is the score list on all n actions (items) predicted by the MoTrans shown in Eq. 4.12, $\hat{Y}_t = [y_1, ..., y_n]$ is the score list on all n items predicted by the UserTrans shown in Eq. 4.2. α controls the influence of the user state encoder. Therefore, $Score_p = [a_1 + \alpha y_1, ..., a_n + \alpha y_n]$ denotes the final scores predicted on n items at step p. The inference stage is illustrated in Algorithm 2. In Section 4.5.4, we will discuss the effect of α on the performance of distinct objectives.

4.5 Experiments

To extensively verify the MODT4R model, we conduct experiments by the following questions:

RQ1: How does MODT4R trade-off the recommendation performance of accuracy, novelty and diversity compared with previous methods?

Algorithm 2: Multi-objective return generation and next item prediction at inference stage.

Input: a sequence $x_{1:p}$, x_{p+1} is the ground-truth item, novel item set I_{nov}, trained recommendation model MODT4R

Output: prediction scores of the next item on all candidate items for the testing sequence $x_{1:p}$

Compute $R_{1:p,acc}$ via Eq. 4.3
Compute $r_{p,nov}$ via Eq. 4.18
Compute $R_{1:p,nov}$ via Eq. 4.4
if $p > 1$ **then**
    Sample an interaction sequence $x_{1:p-1}$ from $x_{1:p}$
    Compute $R_{1:p-1,div}$ via Eq. 4.6
end if
if $p = 1$ **then**
    Set $R_{1,div}$ to 1
else
    Set $R_{p,div}$ to $R_{p-1,div}$
end if
return $(R_{1:p,acc}, R_{1:p,nov}, R_{1:p,div}, x_{1:p}, x_{p+1})$
Compute $\hat{Y}_p$ via Eq. 4.2
Compute $\hat{A}_p$ via Eq. 4.12
Compute $Score_p$ via Eq. 4.19
return $Score_p$

RQ2: How effective is UserTrans in modeling the user state for the MoTrans?

RQ3: How does the performance vary the hyperparameters change in the training stage & how to control the weights of objectives at the inference stage?

RQ4: How do the returns of accuracy, diversity, and novelty affect the performance of multiple objectives?

Table 4.2: Dataset summary.

	RC15	RetailRocket	LFM-1b
#sequences	200,000	195,523	66,293
#items	26,702	70,852	352,602
#interactions	1,234,309	1,154,911	4,864,064

4.5.1 Experimental Setup

We introduce the real-world datasets, evaluated metrics for multi-objectives, baselines and implementation details in this section.

Datasets

The methods are verified on three publicly available datasets: RC15[3], RetailRocket[4] and LFM-1b [93]. The former two are from e-commerce websites, and the latter one is from a music recommendation website.

RC15 is collected from RecSys Challenge 2015, including sequential user-item interactions of clicks and purchases behaviors sorted by timestamp. We eliminate sequences with lengths less than three and randomly sample 200k sessions as the final dataset, containing a total of 1,234,309 interactions over 26702 items.

RetailRocket is from an e-commerce website and contains user sequential behaviors of viewing and adding to the cart. After removing items that interacted less than three times and sequences with lengths less than three, we obtained 195,523 sessions with a total of 1,154,911 interactions on 70,852 items.

LFM-1b is collected from Last.fm[5], a music streaming platform, that contains listening events of tracks created by users. We sample events of the first week of June 2012 and sort the sequence by timestamp. The tracks/items played less than three times and users/sequences with less than three events are removed. The final dataset contains 4,864,064 listening events from 66,293 user sequences on 352,602 tracks.

Detailed statistics of the three datasets are presented in Table4.2. In all cases, we randomly sample 80%, 10%, and 10% of the data for training, validation, and testing. At the inference stage, we evaluate the interacted items of a test sequence one by one. We perform each experiment five times and report the average performance to ensure consistency and reliability in the results.

Metrics

The performance of all the models are evaluated by corresponding metrics, which are illustrated as follows:

Accuracy. We use two commonly applied metrics: Hit Ratio (HR) [57] and Normalized Discounted Cumulative Gain (NDCG) [52] to evaluate the recommendation accuracy. HR@k, $k \in \{10, 20\}$ evaluates if the ground-truth item appears in the top-k positions of the recommendations. NDCG@k, $k \in \{10, 20\}$ measures the rank of the ground-truth item in the top-k recommendation list. The average results of overall test interactions are reported.

Diversity & Novelty. Coverage (CV) [98] metric is commonly used to evaluate the diversity and novelty performance at the system level, examining the extent to which the system suggests a diverse and novel set of items. To evaluate diversity, we calculate $\text{CV}_{div}@k$, $k \in \{5, 10, 20\}$ as a percentage of all candidate items covered by top-k recommended items of all sessions. To

[3]https://recsys.acm.org/recsys15/challenge/

[4]https://www.kaggle.com/retailrocket/ecommerce-dataset

[5]http://www.last.fm/api

assess the novelty, we compare $\text{CV}_{nov}@k$, $k \in \{5, 10, 20\}$ as a percentage of less popular(long-tail) items included within all top-k recommended items. By comparing the coverage of the whole item pool and less interacted items, we can obtain insights into the system's ability to provide both diverse and novel recommendations, promoting exploration and variety for users.

Repetitiveness. An RS may trap users in a filter bubble with narrow information, limiting their exposure to diverse and novel items. We additionally adopt Repetitiveness (R)[52] to evaluate the usefulness of recommendation results. $\text{R}@k$, $k \in \{5, 10, 20\}$ measures the repetitions within the top-k recommended items of each position in total N sessions, indicating the level an RS creates filter bubbles. The definition is as follows:

$$R@k = 1/N \sum_{i=1}^{N} \#\text{repetitions of recommended items in top-}k\text{ list for session } i\ . \tag{4.20}$$

For accuracy, diversity and novelty metrics, the higher the better, while for repetitiveness, the lower the better.

Baselines

We compare MODT4R with four state-of-the-art methods and a variant of our MODT4R according to the recent research [52, 53]:

SASRec [33] is a well-established baseline, which is based on self-attention to encode sequential items in a session. It is similar to the UserTrans in our MODT4R that can be seen as adapting SASRec to realize MOSR task.

SQN [57] improves sequential recommendation model by training another RL agent to optimize recommendations by user feedback. It first formulated the traditional sequential recommendation task as RL problem and realize superior performance on accuracy metrics. The method is integrated with multiple sequential models, e.g., SASRec, to further improve performance.

SMORL [52] is the state-of-the-art MOSR model that balances accuracy, diversity, and novelty based on Q-learning to reward each interaction. It first proposed the MOSR task and realized state-of-the-art performance. Same as SQN, SMORL can realize the best performance when integrated with SASRec compared with other sequential models.

SMONAC [53] is the recent MOSR model that extends SMORL by sampling negative items to predict actions. However, it does not implement experiments and provide results on par with the SASRec. Thus this method is not directly comparable. To ensure fair comparison, we re-implement it based on SASRec and compare the results with our method and other baselines presented in this chapter.

MRDT is a adaptation of our proposed MODT4R that models the same sequence of the order of return-state-action in DT rather than state-return-action in MODT4R.

Implementation

The length of sequences for all three datasets is set to 10. A padding token is added to shorter sequences to ensure the same length. For SASRec, SQN and SMORL methods, the experimental results on RC15 and RetailRocket datasets are from the previous research paper [52]. The Adam optimizer [111] is set to all models and their variations. We choose the learning rate from {0.001, 0.005, 0.01, 0.05, 0.1}. The mini-batch size is 256 for RC15 and RetailRocket, and 512 for LFM-1b. The validation process is conducted every 5,000, 10,000, and 10,000 iterations of updates on RC15, RetailRocket and LFM-1b, respectively. To be fair, the item embedding size, state embedding size, return embedding size, and action embedding size are all set to 64 for all models. The head in self-attention of UserTrans is 1, the same as in SASRec [33]. We apply the GPT [126] architecture to MoTrans, which uses causal masking for autoregressive generation. For the four baselines, we set the reward to 1 for all interactions in SQN regardless of the specific behaviors, while other models follow the original settings.

4.5.2 Overall Performance (RQ1)

Table 4.3 illustrates the experimental results of the top-k multi-objective recommendations on RC15, RetailRocket and LFM-1b. Overall, our MODT4R achieves superior performance on all metrics compared to the state-of-the-art, indicating its ability to balance recommendation accuracy with the conflicting objectives, i.e., diversity and novelty. On RC15 dataset, MODT4R outperforms all benchmark models on all metrics. For example, MODT4R exhibits a remarkable increase (22%) in coverage metrics and a modest improvement (2%) in HR and NDCG compared to SMORL. Similarly, MODT4R exhibits a significant improvement of up to 16% on diversity and novelty compared with the second-best baseline on the RetailRocket dataset,, while still maintaining a slight increase (1.3%) in accuracy compared to other baselines. On the LFM-1b, we obtain similar results to RetailRocket, with the coverage metrics improving by up to 18% compared to the second-best method, while accuracy is maintained. This indicates that the prediction of actions conditioned on the returns and states by our MODT4R can stably balance multiple objectives compared to traditional RL methods (SQN, SMORL and SMONAC) that predict items based solely on state.

In addition, for the performance of MODT4R, compared to the MRDT model, MODT4R gains improved performance with the multi-objective returns learning and adapted trajectory order, which verifies the importance of the additional training signal for MODT4R. Compared with the strong baseline of traditional RL models, MRDT achieve outstanding performance. This verifies the effectiveness of the straightforward application of the original DT model. However, MRDT cannot consistently bring significant improvements across different datasets. In contrast, our MODT4R can reliably provide substantial enhancements for various datasets. In conclusion, MODT4R demonstrates the ability to significantly enhance the metrics of diversity and novelty,

Table 4.3: Recommendation performance on RC15, RetailRocket and LFM-1b, respectively. CV represents Coverage. Boldface denotes the highest scores and the second-best scores are marked with __. $*$ means the significance p-value < 0.05 compared with SMORL.

	Objective	Metric	Models					
			SASRec	SQN	SMONAC	SMORL	MRDT	MODT4R
RC15	Accuracy	HR@10	0.4257	0.4288	0.4302	0.4315	0.4386	**0.4417**
		NDCG@10	0.2599	0.2630	0.2635	0.2651	0.2687	**0.2715***
		HR@20	0.5053	0.5073	0.5063	0.5104	0.5168	**0.5201**
		NDCG@20	0.2801	0.2829	0.2821	0.2851	0.2885	**0.2914***
	Diversity	CV_{div}@5	0.5208	0.4527	0.5875	0.5755	0.5827	**0.6445***
		CV_{div}@10	0.6046	0.5194	0.6628	0.6508	0.6642	**0.7243***
		CV_{div}@20	0.6792	0.5755	0.7274	0.7158	0.7314	**0.7706***
	Novelty	CV_{nov}@5	0.4679	0.3922	0.5357	0.5285	0.5678	**0.6325***
		CV_{nov}@10	0.5607	0.4660	0.6393	0.6120	0.6632	**0.7205***
		CV_{nov}@20	0.6436	0.5283	0.7263	0.6842	0.7418	**0.7876***
	Repetitiveness	R@5	10.62	10.94	10.32	10.38	10.57	**10.30**
		R@10	23.24	23.85	22.31	22.79	23.28	**22.69**
		R@20	49.28	50.79	46.23	48.48	49.44	**43.37**
RetailRocket	Accuracy	HR@10	0.3085	0.3302	0.3473	0.3521	0.3496	**0.3565***
		NDCG@10	0.2107	0.2279	0.2472	0.2477	0.2398	**0.2478**
		HR@20	0.3572	0.3803	0.3922	0.4028	0.4043	**0.4083***
		NDCG@20	0.2227	0.2406	0.2523	0.2605	0.2537	**0.2608**
	Diversity	CV_{div}@5	0.5305	0.4617	0.5775	0.5724	0.5798	**0.6545***
		CV_{div}@10	0.6300	0.5490	0.6765	0.6672	0.6795	**0.7633***
		CV_{div}@20	0.7149	0.6254	0.7612	0.7476	0.7635	**0.7961***
	Novelty	CV_{nov}@5	0.4806	0.4040	0.5329	0.5261	0.5345	**0.6021***
		CV_{nov}@10	0.5899	0.5001	0.6398	0.6311	0.6446	**0.7244***
		CV_{nov}@20	0.6838	0.5847	0.7387	0.7202	0.7377	**0.7961***
	Repetitiveness	R@5	15.67	15.60	12.32	12.58	13.16	**12.16**
		R@10	32.27	32.20	26.01	26.69	27.45	**25.54***
		R@20	66.07	66.10	55.86	56.14	56.83	**53.11***
LFM-1b	Accuracy	HR@10	0.2857	0.2905	0.2874	0.2911	0.2935	**0.2955**
		NDCG@10	0.2394	0.2407	0.2399	0.2405	0.2407	**0.2451**
		HR@20	0.3080	0.3128	0.3102	0.3152	0.3186	**0.3192**
		NDCG@20	0.2450	0.2464	0.2448	0.2465	0.2471	**0.2508***
	Diversity	CV_{div}@5	0.5637	0.5542	0.5273	0.5167	0.6677	**0.7124***
		CV_{div}@10	0.6621	0.6451	0.6242	0.6046	0.7667	**0.8148***
		CV_{div}@20	0.7504	0.7251	0.7162	0.7184	0.8443	**0.8748***
	Novelty	CV_{nov}@5	0.5275	0.5163	0.5028	0.4934	0.6397	**0.6903***
		CV_{nov}@10	0.6330	0.6141	0.6023	0.6045	0.7468	**0.7999***
		CV_{nov}@20	0.7286	0.7008	0.6837	0.6958	0.8308	**0.8644***
	Repetitiveness	R@5	161.85	178.19	169.43	178.34	173.90	**159.69***
		R@10	336.59	374.69	348.53	375.64	366.28	**332.97***
		R@20	691.47	774.70	723.19	776.98	761.72	**684.70***

while simultaneously maintaining a high level of accuracy for the MOSR task.

4.5.3 Discussion of User State (RQ2)

In our MODT4R, UserTrans plays an important role in generating effective state for the MoTrans to finish the decision processing. To explore the impact of user state encoder and how the representation of state affect recommendation performance, we compare MODT4R with its two state settings: MODT4R$_i$ and MODT4R$_{imp}$. As shown in Figure 4.3a), MODT4R$_i$ directly removes user state encoder and replaces this input with item embedding at that step to evaluate the function of an individual state encoder; and MODT4R$_{imp}$ in Figure 4.3b takes the implicit positional embeddings i_t as states in a sequence to show the effectiveness of explicitly encoded state.

The experimental results are detailed in Table 4.4. We can observe that on three datasets, compared with MODT4R$_i$, MODT4R$_{imp}$ achieves a substantial improvement in the coverage metrics (up to 10%) while there is only a slight fluctuation in the HR and NDCG metrics. These indicate that the explicit representation of user state has a significant impact on enhancing diversity and novelty performance. Furthermore, compared to the basic method SASRec in Tables 4.3, both MODT4R$_{imp}$ and MODT4R$_i$ improve the accuracy results on RetailRocket and LFM-1b, except for a slight decrease on the RC15 dataset. Additionally, the accuracy does not improve while novelty and diversity are substantially increased compared with SQN and SMORL. This suggests that the multi-objective decision process incorporating implicit positional states can significantly enhance diversity and novelty without incurring a substantial loss of accuracy.

Comparing the results of MODT4R$_{imp}$ and MODT4R, we can observe that on all three datasets, MODT4R demonstrates a marked improvement in HR and NDCG (up to 10%) metrics. The and maintains high coverage metrics. Furthermore, The comparison highlights the importance of the state representation explicitly generated by the user state encoder UserTrans. By effectively modeling and utilizing user states, MODT4R can better balance the recommendation performance of accuracy, novelty, and diversity. In conclusion, our findings underscore the critical role of effective user state representation in enhancing the performance for MOSR.

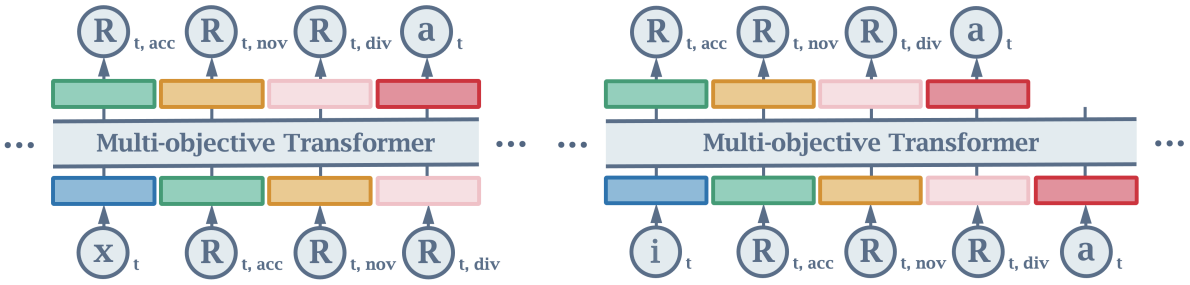
4.5.4 Ablation Study (RQ3)

Effect of the Weighted Prediction Score.

According to Eq. 4.19, the ranking scores for candidate items of MODT4R are dominated by both action and state predictions. In this part, we investigate how the weight α of $Score_{i,S}$ at the inference stage affects the performance of distinct recommendation objectives. Figure 4.4 shows the behavior of MODT4R on RC15 and RetailRocket in terms of HR@20 and CV$_{div}$@20 metrics as α changes. On all datasets, the accuracy performance improves as α increases from 0 to

Table 4.4: Performance comparison of varying the user states. The structure of MODT4R$_i$ and MODT4R$_{imp}$ can be seen in Figure 4.3a and Figure 4.3b. NG represents NDCG. Boldface denotes the highest scores.

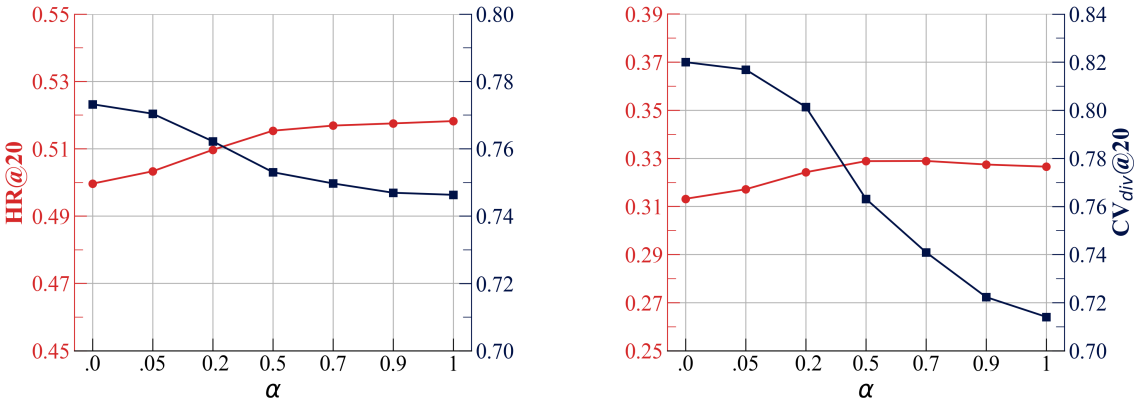
	Methods	Accuracy				Diversity		Novelty		Repetitiveness	
		HR@10	NG@10	HR@20	NG@20	CV$_{div}$@5	CV$_{div}$@10	CV$_{nov}$@5	CV$_{nov}$@10	R@5	R@10
RC15	MODT4R$_{imp}$	0.4063	0.2430	0.4856	0.2631	0.6275	0.7082	0.6203	0.7164	10.75	23.47
	MODT4R$_i$	0.4163	0.2521	0.4953	0.2722	0.5709	0.6606	0.5540	0.6590	11.52	24.94
	MODT4R	**0.4417**	**0.2715**	**0.5201**	**0.2914**	**0.6445**	**0.7243**	**0.6325**	**0.7205**	**10.30**	**22.69**
Retail Rocket	MODT4R$_{imp}$	0.3129	0.2180	0.3623	0.2305	0.6365	0.7453	0.5975	0.7175	**11.84**	**24.61**
	MODT4R$_i$	0.3223	0.2255	0.3711	0.2379	0.6051	0.7168	0.5627	0.6860	12.74	26.53
	MODT4R	**0.3565**	**0.2478**	**0.4083**	**0.2608**	**0.6545**	**0.7633**	**0.6021**	**0.7244**	12.16	25.54
LFM-1b	MODT4R$_{imp}$	0.2898	0.2374	0.3140	0.2435	0.7092	0.8055	0.6869	0.7897	177.02	368.06
	MODT4R$_i$	0.2839	0.2278	0.3119	0.2349	0.6068	0.7198	0.5767	0.6973	189.76	397.47
	MODT4R	**0.2955**	**0.2451**	**0.3192**	**0.2508**	**0.7124**	**0.8148**	**0.6903**	**0.7999**	**159.69**	**332.97**



(a) MODT4R$_i$ (b) MODT4R$_{imp}$

Figure 4.3: (a) MODT4R$_i$ is a variant of our proposed MODT4R that removes the state modelling step, and replaces the state with the item embedding at that position. (b) MODT4R$_{imp}$ is the variant of our proposed MODT4R that replaces the generated state with implicit positional state embedding and removes the state encoder.

1, while the coverage metric has the opposite results. It suggests that prediction from UserTrans part is vital for maintaining high accuracy and that MoTrans mainly contributes to providing novel and diverse items for users. Moreover, we adjust the model to focus on recommendations for different sets of objectives by controlling this weight. Therefore, the value of the α is determined by recommendation requirements, i.e., user preference for accuracy or diversity. The optimal values of α are set to 1, 1, and 0.2 for RC15, RetailRocket and LFM-1b, respectively.



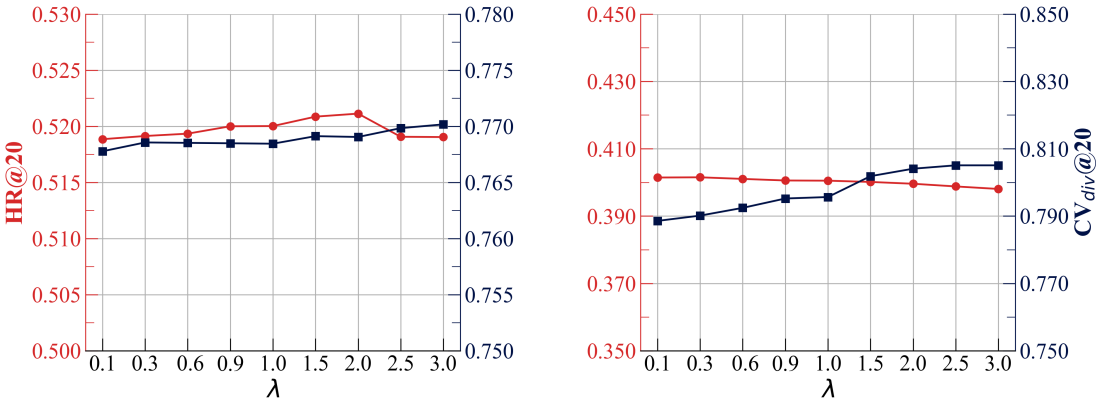
(a) RC15 (b) LFM-1b

Figure 4.4: MODT4R with different weights of scores from user state encoder represented by Eq. 4.19 during inference.

Investigation of Diversity Return at Inference.

According to Section 4.4.6, λ determines the weight of the diversity objective at inference stage. We investigate how the settings of diversity return in the inference stage affect the model's performance. Figure 4.5 shows that as λ increases, HR@20 decreases slowly, while CV$_{div}$@20 increases continuously. The insights from these findings are twofold. First, by adjusting λ, we can tailor the recommender to prioritize either diversity or accuracy objective, depending on

the specific needs of the application or user preferences. Second, the continuous increase in CV_{div}@20 with higher λ suggests that the model effectively expands its recommendation pool, potentially introducing users to a broader array of items.



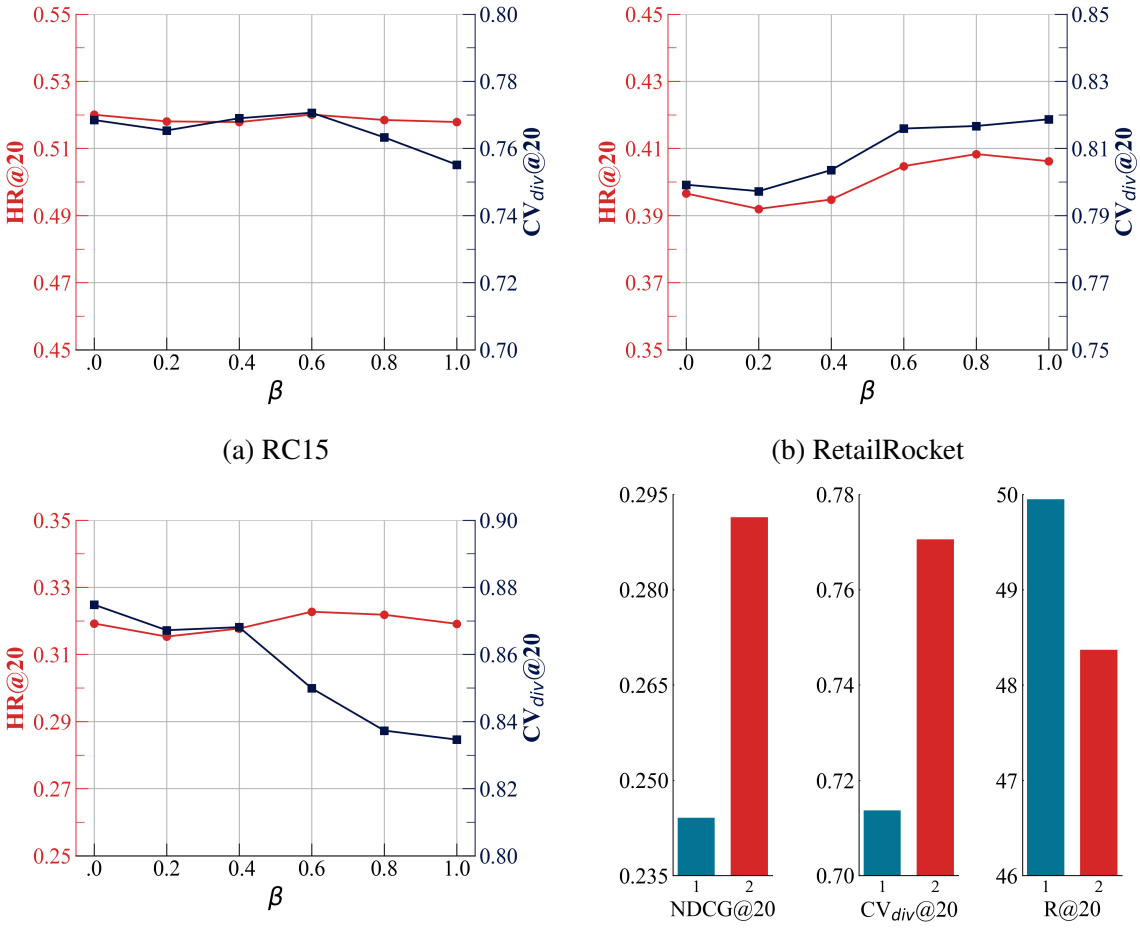
(a) RC15 (b) RetailRocket

Figure 4.5: Performance of MODT4R with different weights of diversity returns illustrated in Section 4.4.6 during inference.

Effect of the Return Prediction Loss.

We report the interaction between the weight β of multi-objective return prediction loss $\mathscr{L}_R$ from Eq.4.17 and the recommendation performance. Figure 4.6a-4.6c indicate the behavior of MODT4R with respect to HR@20 and CV_{div}@20 on diversity when varying β from 0 to 1. We observe that, for all datasets, the value of HR@20 fluctuates slightly as the hyperparameter changes, while CV_{div}@20 shows distinct trends among datasets. When increasing from 0.6 to 1, the result of RC15 changes from a small fluctuation at the beginning to a slight decrease. For RetailRocket, the overall trend is upward, and the subsequent fluctuations are small. On the contrary, the return loss brings a slight reduction of diversity to LFM-1b. These observations suggest that the return loss plays an important role in maintaining a stable balance between the three conflicting objectives. The optimal values of β are 0.6, 0.8 and 0 for RC15, ReatailRocket and LFM-1b, respectively.

We further explore the impact of loss $\mathscr{L}_S$ (Eq.4.13) by comparing the results of MODT4R with the variant without it. Figure 4.6(d) shows the comparisons on RC15, and RetailRocket and LFM-1b exhibit similar behavior. It is clear that the performance of MODT4R falls significantly on all metrics without $\mathscr{L}_S$, indicating that the inclusion of an item prediction loss is essential to maintain a reasonable level of recommendation accuracy.



(a) RC15

(b) RetailRocket

(c) LFM-1b

(d) Effect of loss L_S on RC15

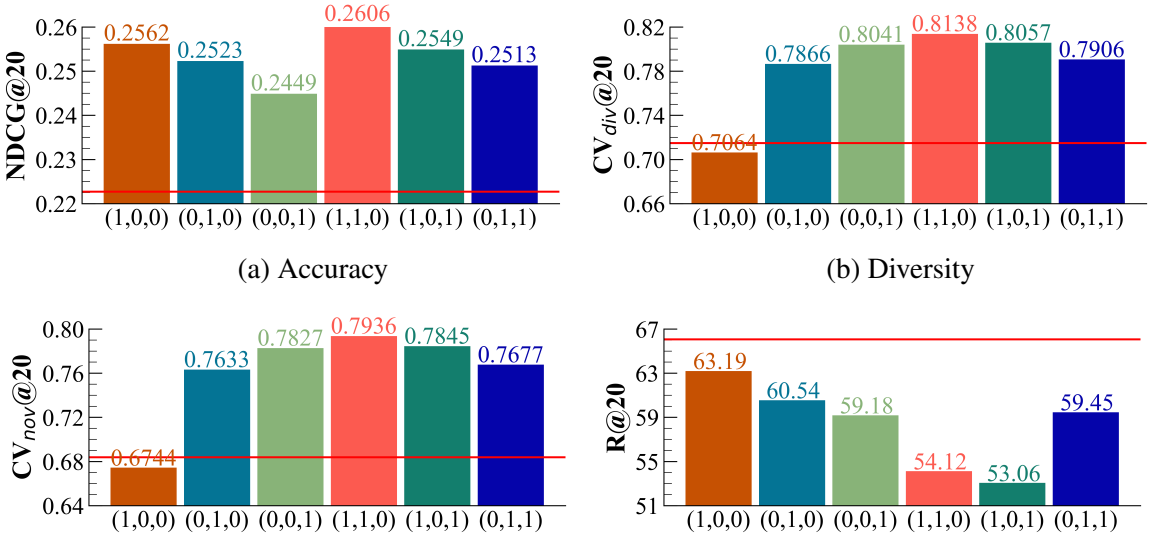
Figure 4.6: Figure(a), (b) and (c) show the performance of MODT4R with different weights of loss $\mathscr{L}_R$ on all datasets. Figure(d) shows the comparison of MODT4R without(blue bar) and with(red bar) $\mathscr{L}_S$ on RC15, CV_{div}@20 here is for diversity, and novelty metric shows the similar trend.

4.5.5 Objective-oriented Reinforcement (RQ4)

To explore the capability of distinct reinforcement objectives and their interrelationships in MODT4R, we conduct experiments with different objective combination subsets by varying the multi-returns. We use 1 and 0 to denote whether or not the relevant returns/objectives are involved. For example, $(O_{acc}, O_{div}, O_{nov}) = (1,0,0)$ indicates that only the accuracy return is considered, while the diversity and novelty objectives are discounted. In total, we conduct experiments under the following 6 settings:

$$\begin{gathered}(O_{acc}, O_{div}, O_{nov}) \in O_{sets}, \\ O_{sets} = \{(1,0,0),(0,1,0),(0,0,1),(1,1,0),(1,0,1),(0,1,1)\},\end{gathered} \tag{4.21}$$

Different from the previous study [52] that controls the weight of objective RL loss, we define the variant methods by changing the input of objective returns. The comparison results on the above subsets for RetailRocket are shown in Figure 4.7, while RC15 and LFM-1b are omitted due to similar trends. Specifically, Figure 4.7a demonstrates the comparison of NDCG@20 on the accuracy objective. It is evident that the methods reinforcing accuracy perform better in comparison to those that do not. When only the objectives of diversity or/and novelty are emphasized, NDCG@20 is decreased to varying degrees. We obtain a phenomenon that when combining the accuracy objective with either the diversity or novelty, both of them show a significant improvement in relevant metrics. From Figure 4.7b and 4.7c, we observe that the methods that are limited to reinforcing accuracy alone result in the poorest performance on the other two objectives.



(a) Accuracy (b) Diversity

(c) Novelty (d) Repetitiveness

Figure 4.7: Performance when integrating different subsets of objectives on RetailRocket dataset, the settings are represented by Eq. 4.21. Red lines denote metrics of the baseline model SASRec .

4.6 Conclusion

In this chapter, we presented the Multi-Objective Decision Transformer for Reward-driven Recommendation (MODT4R), a novel framework to tackle the problem of multi-objective recommendation for sequential recommendation. MODT4R effectively conceptualizes MOSR as a sequence modeling challenge resolved through simple supervised learning, avoiding the complexities and instabilities associated with traditional reinforcement learning approaches. Our approach captures the dynamics of user interests in terms of accuracy, diversity, and novelty by transforming them into sequential returns. We developed a novel user trajectory representation that connects user state transitions with multi-objective returns, all within a unified sequence modeled by our proposed Multi-Objective Transformer (MoTrans). The User Transformer (UserTrans) complements this by explicitly modeling user states based on historical interactions.

We have shown that MODT4R can leverage the strengths of the Transformer architecture, notably its ability to take actions based on the entire sequence of past states and returns. By co-training with a Mean Squared Error (MSE) loss to predict subsequent returns in MoTrans, we enhance the model's ability to maintain stability while balancing multiple objectives. Empirical results from extensive experiments on real-world datasets underscore MODT4R's capacity to improve or maintain accuracy while making significant strides in diversity and novelty in recommendations. This positions MODT4R as a robust and adaptable solution, particularly suited for real-world applications where balancing various recommendation objectives is crucial. In future work, we expect to incorporate more personalized reward functions that adapt to individual user preferences over time and could make recommendations more user-centric.

4.6.1 Summary and Link to Next Chapters

This chapter establishes the first part of the decision layer by reformulating MOSR as return-conditioned sequence modeling with Decision Transformers, enabling controllable accuracy–diversity–novelty trade-offs via inference-time adjustment. The results demonstrate that return conditioning provides a stable and flexible mechanism for multi-objective personalization beyond accuracy-centric ranking. Chapter 5 extends this decision layer to hierarchical personalization across sessions and introduces explicit serendipity modeling under the MOPSR setting.

Chapter 5

Multi-objective Recommendations with Hierarchical Decision Transformers for Diversity, Novelty, and Serendipity

Note. This chapter is based on our CIKM 2024 paper on HDT, and is revised in this thesis to clarify the MOPSR setting and the operational definitions of unexpectedness/serendipity, and connect hierarchical decision-making to the deployability challenges addressed in later chapters.

5.1 Introduction

Personalized Session-based Recommendation (PSR) extends the traditional sequential recommendation models—which typically recommends the next item based on a recent active session—to leverage historical sessions of a user for short-term recommendations in the current session. However, existing PSR methods face two limitations: (1) treating offline sessions uniformly as static data and relying on user embeddings to represent personalized information overlook the dynamic evolution of interests over time, which can change significantly as sessions progress in practical application. (2) focusing on accuracy, i.e., recommending items relevant to recent interactions, ignores the balance of multi-faceted requirements for user satisfaction, i.e., diversity, novelty, and serendipity.

Therefore, we introduce Multi-objective PSR (MOPSR) task and propose Hierarchical Decision Transformers (HDT) framework, which models strictly sequential preference transitions of users across and within sessions to balance recommendation accuracy with the mentioned objectives. To address the first problem, Inter-session DT dynamically tracks the user's long-term preference across sessions by maintaining a goal state. This goal state serves as personalized information to collaboratively make recommendations with short-term state via the Intra-session DT. To tackle the second limitation, we propose inter-session and intra-session unexpected returns to trade off relevant recommendations and user preferences on diversity,

novelty, and serendipity. The hierarchical returns help the recommender accurately identify signals of the user's expectations and changes in multi-objective preferences. To verify the effectiveness of our method on the MOPSR, we apply HDT to four state-of-the-art sequential recommendation models and conduct experiments on two publicly available datasets. Experimental results demonstrate that (1) HDT can widely generalize sequential models to solve the MOPSR task in scenarios with incrementally generated sessions, and (2) our method can balance multi-objectives by maintaining and even enhancing accuracy while effectively improving the diversity, novelty, and serendipity objectives.

Recommender Systems (RS) help users discover content from large-scale data streams and promote engaging content [1, 116, 127] for users. In recent years, next-item recommendation systems have been at the forefront of research, e.g., in entertainment (e.g., video streaming) and e-commerce, with a strong focus on recommending the next relevant item based on user-item interactions within one recent active session. For the most part, sequential or next item recommendation methods [26, 31–33] attempt to model a short-term sequence of user-item interactions within one recent active session to predict the next recommendations for users. However, in many scenarios, recommendations are determined by both user's long-term and short-term interests. For example, users may prefer revisiting past topics in music application [93] and forums [35]. Therefore, Personalized Session-based Recommendation [35, 128–130] is proposed to recommend the next item by considering both user's past sessions and current session.

However, how to effectively model dynamic changes of user preference through historical sessions and short-term interest in the current session remains a challenging task. Some research treats all historical sessions as equivalent interactions and adapts the Graph Neural Network (GNN)-based methods [31, 128–130] extract either global or local graph relations from entire offline static data in advance to model items/users. However, they ignore the personalized information from the dynamic changes of user preference over time in a real-world scenario, where user session proceeds sequentially. Even user embeddings are trained to present personalized, they are ineffective due to insufficient training when user sessions are sparse. Consequently, GNN-based approaches fall outside the scope of our research. To be practical, we focus on PSR scenarios where user interactions are strictly sequential across and within sessions.

Moreover, Multi-objective PSR (MOPSR) that realizes trade-offs between accuracy, diversity, novelty, and serendipity remain an unsolved task. Existing PSR methods still align with most traditional recommendation models [6, 26, 57, 60] to optimize the metrics of accuracy, referring to recommend items that are relevant to recently interacted items, e.g., HR [26] and NDCG [96] metrics. However, a strictly relevance-oriented RS overlooks the intrinsic recommendation qualities, affecting user experience in terms of diversity (recommended items are dissimilar to the user's consumption patterns), novelty (recommended items are new/unseen), and serendipity (recommendations are relevant to users and beyond expectations). Recently, a multi-objective recommender system [131] considering the diversity and novelty is proposed for

the sequential recommendation by the offline reinforcement learning (RL) process. It trains multiple RL agents to balance accuracy with the two objectives in recommended items (the taken actions) by maximizing the expected cumulative multi-objective rewards (returns). However, traditional RL approaches, e.g., Q-learning [57, 58] cannot be directly applied to MOPSR since they tend to prioritize immediate rewards of action. This concentration on short-term gains in the current session poses a challenge in capturing multiple long-term preferences in past sessions. Furthermore, the training becomes complex due to setting different agent heads for each objective. Additionally, the limited offline user data cannot provide sufficient feedback for each agent, making it challenging to accurately tune and optimize for specific objectives.

To address the above issues, we introduce Multi-objective Personalized Session-based Recommendation (MOPSR) task to realize trade-offs between accuracy and diversity, novelty and serendipity objectives. First, the Hierarchical Decision Transformers (HDT) framework is proposed to personalize sequential models by concurrently modeling the user's long-term preferences derived from historical sessions, and their short-term interests within the current session. Moreover, we propose inter-session and intra-session unexpected returns infused with HDT to capture the user's signals of preferences on unexpected items to improve the balance between the mentioned objectives. Specifically, HDT contains two layers: Inter-session DT (InterDT) maintains user state transitions across sessions to generate goal states conditioned on the user's future expectations; Intra-session DT (IntraDT) models state transitions by the user-item interactions within the current session, and the recommendation process is guided by the goal state from the InterDT with an extra intra-session novelty return.

The Hierarchical Decision Transformer (HDT) is trained on offline user data in a supervised manner, mitigating the limitations associated with Q-value estimation [57], Multi-agents [52], and high variances in traditional RL. During the inference, given the user state from the previous session and the expectations we want to obtain for the next recommendations, HDT can directly take actions (recommend items) considering long-term and short-term objectives, e.g., serendipity and novelty. To verify the effectiveness of our approach, we apply HDT to four state-of-the-art sequential recommendation models and conduct experiments on two publicly available datasets. Experimental results are compared by specific objective metrics for accuracy, diversity, novelty and serendipity, which demonstrate the superior performance of our method in balancing those objectives.

Our contributions can be summarized as follows:

- We introduce Multi-objective Personalized Session-based Recommendation (MOPSR) task and propose a Hierarchical Decision Transformers (HDT) framework to recommend next items by leveraging user preference transitions across historical sessions.
- We infuse HDT with hierarchical unexpected returns in InterDT and IntraDT to comprehensively model long-term and short-term changes in user preferences for unexpected items.

- We establish specific metrics for multi-objectives to evaluate the trade-off capabilities of HDT. To validate the effectiveness of our method, HDT is applied to four state-of-the-art sequential recommendation models and experiments are conducted on two public datasets.
- The experimental results show that our method can balance multi-objectives by maintaining and even enhancing accuracy while effectively improving the diversity, novelty, serendipity.

5.2 Related Work

The work in this chapter is positioned at the intersection of three key research areas. First, Personalized Session-based Recommendation (PSR), which extends sequential recommendation by modeling dependencies both within a session (intra-session) and across a user's historical sessions (inter-session) [35]. Our research focuses on hierarchical methods that respect the strict chronological order of sessions, as this aligns with real-world scenarios where user data is generated incrementally. Second, multi-objective recommendation, which aims to improve user satisfaction beyond accuracy by considering objectives like diversity, novelty, and serendipity. Existing approaches often rely on joint loss functions or post-processing techniques, but there is a lack of work that systematically addresses these objectives within the PSR context. Third, Reinforcement Learning (RL) for recommendation, which trains an agent to recommend items by maximizing cumulative rewards. For multi-objective tasks, this can involve training multiple RL agents, one for each objective [52].

While these fields offer valuable contributions, they present limitations when combined. Existing PSR methods primarily optimize for accuracy, neglecting other important objectives. Concurrently, traditional RL approaches, such as multi-agent Q-learning, are not only complex to train for multiple objectives but also struggle to capture the long-term, cross-session user interests central to PSR. To address these gaps, we are inspired by the Decision Transformer (DT) [30], which recasts offline RL as a sequence modeling problem. We propose the Hierarchical Decision Transformer (HDT) for the Multi-objective PSR (MOPSR) task. HDT adopts a hierarchical structure to model users' preference transitions across and within sessions, and leverages the DT paradigm to condition on hierarchical unexpected returns. This allows our model to effectively balance accuracy with diversity, novelty, and serendipity in a supervised manner, directly tackling the identified challenges.

5.3 Methodology

We first illustrate the Multi-objective Personalized Session-based Recommendation (MOPSR) task.Next, we introduce the overall framework of the proposed Hierarchical Decision Trans-

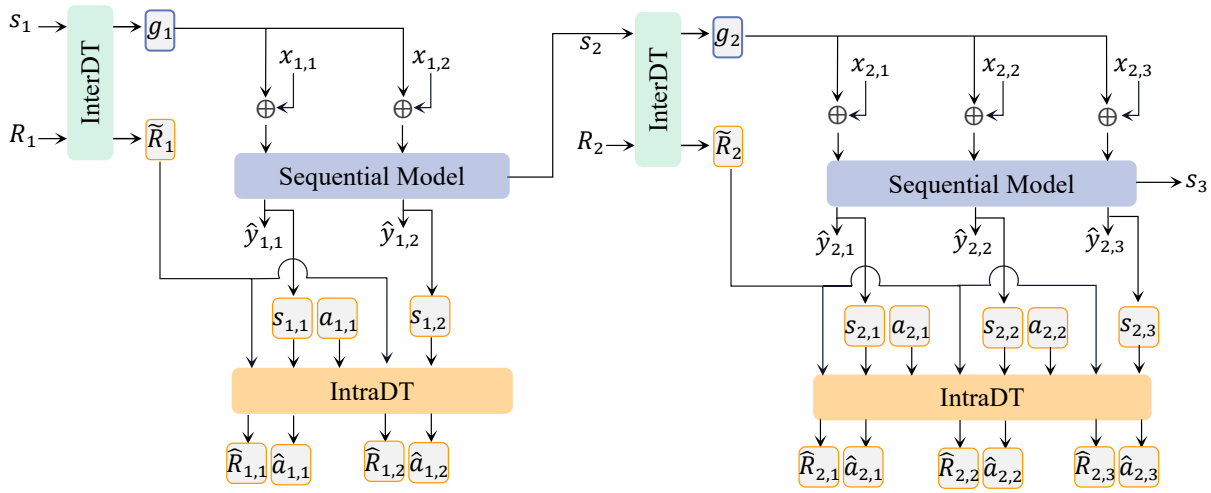


Figure 5.1: Example of personalizing sequential models by Hierarchical Decision Transformers (HDT). The InterDT observes the inter-session state s_c and inter-session unexpected return R_c, where $c \in \{1,2,...\}$, to produce the goal state g_c and goal return $\tilde{R}_c$ for session C_c. s_1 is initialized by zeros. The IntraDT observes the intra-session state $s_{c,t}$, goal state g_c, goal return $\tilde{R}_c$, where $t \in \{1,2,3,...\}$, to predict the intra-session unexpected return $R_{c,t}$ and the action a_t in session c.

former (HDT) for MOPSR. Finally, we illustrate the modeling details.

5.3.1 Problem Formulation

Task Definition. Different from Sequential Recommendation [1, 31, 33, 34, 132] or Session-based Recommendation [26, 32, 133] that recommend next items based on ordered/unordered interactions within a recent active session, Personalized Session-based Recommendation (PSR) [35] aims to recommend the next items based on users' current session and historical sessions. PSR models[35] typically leverage the ordered information inside and across sessions to model the evolution of user preference for recommendations. Therefore, it is of great significance for platforms where user session data is incrementally generated. Let I denote the set of items, $C = \{C_1, \ldots, C_h\}$ denotes a user's sequence of sessions in chronological order, $C_i = \{x_{i,1}, \ldots, x_{i,T}\} \in C (0 < i \leq h)$ denote the user-item interactions in each session, where $x_{i,j} \in I (0 < j \leq T)$ is the index of the interacted item ordered by a timestamp. Given historical sessions $C = \{C_1, \ldots, C_{c-1}\}$ and the current session $C_c = \{x_{c,1}, \ldots, x_{c,t}\}$, the goal is to recommend next item $x_{c,t+1}$ from the whole item set I that the user might be interested in.

Trade-off of Multi-Objectives. We extend the PSR to the Multi-objective Personalized Session-based Recommendation (MOPSR) to balance recommendation accuracy with multiple objectives that affect user experience. First, general RSs aim to recommend the most relevant items to historical sessions that are often influenced by popularity, leading to recommendations that are frequently seen while neglecting long-tail (new and unknown) ones. This reduces diversity and novelty [20, 52] in recommendations. Second, recommending relevant and unexpected items can provide users with serendipity [97]. However, considering past historical interests may bias

recommendations towards items the already been seen or similar items to a user's historical interactions for a PSR model, so that reduces unexpectedness. Conversely, overly recommending dissimilar items can reduce relevance. Therefore, the goal of our MOPSR is to maintain or sacrifice minimal accuracy while providing items that offer novelty, diversity, and serendipity to users.

MOMDP Formulation of MOPSR. We formulate MOPSR as a Multi-Objective Markov Decision Process (MOMDP) [52] and the Hierarchical RL (HRL) is introduced to solve the offline problem. The recommender is the agent, and the users are the environment. The agent (recommender) interacts with the environment (users), taking actions (recommending items) based on the state of the environment, and is updated by the feedback (reward) from the environment. Continually, the environment generates a new state for the agent as users respond to the actions. In HRL-based MOPSR, users provide inter-session and intra-session states, where the former guides the agent considering the dynamic preferences across sessions to predict items with the intra-session state.

Notation. To facilitate clarity for future reading, we provide a reference Table 5.1 illustrating the important ones throughout this chapter.

Table 5.1: An illustration of notations.

	Description
I	The set of items
C_c	c-th session or the current session of a user
$x_{c,t}$	t-th user-item interaction in C_c, i.e., $x_{c,t} \in I$
s_c	Inter-session state session C_c
R_c	Inter-session unexpected return for session C_c
g_c	Goal state for session C_c
$\tilde{R}_c$	Goal return for session C_c.
$s_{c,t}$	Intra-session state at timestamp t in C_c
$R_{c,t}$	Intra-session unexpected return at timestamp t in C_c
$R^n_{c,t}$	Intra-session novelty return at timestamp t in C_c
$a_{c,t}$	Action at timestamp t in C_c, i.e., $a_{c,t} = x_{c,t+1}$

5.3.2 Overall Framework

Traditional offline RL-based recommender mimics the online process by observing the rewards w.r.t specific behaviors from users and is trained to maximize the expected cumulative rewards (expected returns) [52, 57]. However, this process becomes very complex when handling multi-objective tasks for offline data and HRL problems due to adding more agent/value heads [52, 134]. Following the Decision Transformer (DT) [30] that recast the offline RL as sequence modeling to predict actions in a supervised manner conditioned on the expected returns and states, we solve the HRL problem by the proposed **H**ierarchical **D**ecision **T**ransformers (**HDT**)

framework as illustrated in Figure 5.1. HDT consists of two layers: 1) the Inter-session DT (InterDT) models dynamic long-term interest as the session evolves, and 2) the Intra-session DT (IntraDT) models short-term interests within the current session.

The MOPSR is naturally empowered by expected returns and user states at both inter- and intra-session levels. On the one hand, actions (items) are generated under the guidance of designed rewards and user states to balance accuracy with diversity, novelty, and serendipity objectives. On the other hand, to personalize the recommendation with users' long-term interests, the goal state and reward of inter-session are conveyed to the intra-session state to jointly recommend the next items. Formally, the trajectory the HDT aims to model is:

$$\tau = (..., R_c, s_c, g_c, \tilde{R}_c, R^n_{c,1}, s_{c,1}, a_{c,1}, ..., R^n_{c,T}, s_{c,T}, a_{c,T}, ...). \tag{5.1}$$

The components of the trajectory are:

- Inter-session state space $\mathscr{S}^u$: $s_c \in \mathscr{S}^u$ is a user's inter-session state at session C_c, providing representation of long-term preferences evolved across sessions. The initial state $s_1 \in \mathscr{S}^u$ for all users is initialized to zeros.

- Inter-session unexpected returns $\mathscr{R}^u$: $R_c \in \mathscr{R}^u$ represents the expected unexpectedness contributed by the interactions from session C_c to the final session for a user.

- Goal state space $\mathscr{G}$: $g_c \in \mathscr{G}$ is generated by the inter-session state s_c and the inter-session unexpected return R_c from the InterDT at session C_c. g_c will affect the intra-session recommendations by conveying a user's long-term interest in unexpected items to the intra-session state.

- Goal return space $\tilde{\mathscr{R}}^u$: $\tilde{R}_c \in \tilde{\mathscr{R}}^u$ is also generated by the inter-session state s_c and the inter-session unexpected return R_c from the InterDT at session C_c. Differently, $\tilde{R}_c$ is the output at the return position that will control the intra-session recommendations by producing the intra-session unexpected return.

- Intra-session state space $\mathscr{S}^z$: $s_{c,t} \in \mathscr{S}^z$ is a user's intra-session state at timestamp t within the session C_c, which represents the short-term interest of a user. $s_{c,t}$ is generated by a sequential model $\mathscr{E}$ that encodes the sequence $x_{c,1:t} = \{x_{c,1}, ..., x_{c,t}\} \in C_c$ and the goal state g_c. When recommending items in session C_c, we also call it the current session C_c.

- Intra-session unexpected return $\mathscr{R}^z$: $R_{c,t} \in \mathscr{R}^z$ represents the expected unexpectedness achieved by the remaining interactions from timestamp t in session C_c, which reflects a user's short-term preference for unexpected items. Instead of comprising the input trajectory, $R_{c,t}$ is taken as the target unexpectedness in the training process to balance multiple objectives.

- Intra-session novelty return $\mathscr{R}^{nz}$: $R^n_{c,t} \in \mathscr{R}^{nz}$ denotes the novelty achieved by remaining interactions from step t in the session C_c, which reflects a user's short-term preference for novel items in the system.
- Action space $\mathscr{A}$: $\mathscr{A}$ is the discrete action set consisting of candidate items. In the offline setting, the action $a \in A$ at time step t in the session c is the next item, i.e., $a_{c,t} = x_{c,t+1}$.

We can see the overall training process from Figure 5.1: at the beginning of a session C_c, the InterDT observes the inter-session state s_c and the inter-session unexpected return R_c to produce the goal state g_c and goal return $\tilde{R}_c$ for intra-session recommendations. Then, the IntraDT incorporates the intra-session state $s_{c,t}$ (generated by user-item sequence $X_{c,1:t}$ and the goal state g_c), goal return $\tilde{R}_c$, and the intra-session novelty return $R^n_{c,t}$ to generate action (recommend item) a_t at timestamp t. Next, we detail designing returns and modeling trajectories in Inter-session and Intra-session DTs.

5.3.3 Inter-session Modeling

Inter-session Unexpected Return

To provide users with diverse recommendations that touch upon a broader selection of items as possible, we design an inter-session unexpected return R_c to quantify the maximum unexpectedness that future sessions will bring to the user starting from session C_c.

$$R_c = dis_{max} - dis(C_1, ..., C_c), \tag{5.2}$$

where $dis\,(C_1, ..., C_c)$ is the dissimilarity of previous c sessions, calculated as the average dissimilarity (1 minus average cosine similarity) among all interacted items' embeddings in the past c sessions. dis_{max} is the maximum dissimilarity the user can obtain by items in the system. R_c reflects users' propensity to explore diverse, novel, and unexpected items across sessions.

InterDT

Inter-session DT models R_c and s_c to generate the goal state and goal return for intra-session recommendation. R_c is mapped by a trainable embedding to form its representation with the same dimension as s_c. Then the inputs for InterDT at session C_c are formulated as:

$$H_c = stack(R_c, s_c) \in \mathbb{R}^{2\times d}, \tag{5.3}$$

where d is the embedding size. The first state s_1 is initialized by zeros for each user. When session C_c ends, the s_{c+1} for the next session C_{c+1} is updated by the last intra-session state from session C_c, we illustrate the intra-session state in the following part. The goal return $\tilde{R}_c$ and goal

state g_c are formulated as:

$$\begin{aligned} \tilde{H}_c &= \mathbf{T}_1(H_c), \\ \tilde{R}_c, g_c &= unstack(\tilde{H}_c), \end{aligned} \tag{5.4}$$

where $\mathbf{T}_1$ is a self-attention layer with residual connection [135].

5.3.4 Intra-session Modeling

Intra-session State

Given goal state g_c that represents long-term preference on diverse and unexpected items and the user-item interactions $x_{c,1:t}$ within the session C_c, the intra-session state $s_{c,t}$ is the hidden state from the sequential model [26, 33] G:

$$x_{c,t} = W_s[x_{c,t}, g_c] + b_s, \tag{5.5}$$

$$s_{c,t} = G(x_{c,1:t}), \tag{5.6}$$

Intra-session Novelty Returns

We introduce a novelty return $R^n_{c,t}$ to promote the recommendation of unknown/novel items [52, 136] that belongs to the less frequently interacted item set I_{nov} (within the bottom 90%) in the training data. In this way, recommending more unpopular items can lead to an increase in the system's diversity and novelty. In session C_c with T steps, $R^n_{c,t}$ is calculated as:

$$R^n_{c,t} = \#\text{ items} \in I_{nov} \text{ from step } t \text{ to } T \tag{5.7}$$

We expect to achieve maximum novelty and diversity, so the predicted item is treated as novel during inference to calculate the $R^n_{c,t}$, which is mapped by a trainable embedding to form its representation with the same dimension as $s_{c,t}$.

Intra-session Unexpected Return

$R_{c,t}$ aims to represent the unexpectedness achieved by the remaining interactions from timestamp t to the end of session C_c:

$$R_{c,t} = dis(x_{c,1}, ..., x_{c,T}) - dis(x_{c,1}, ..., x_{c,t}), \tag{5.8}$$

where T is the length of session c. $dis\,(x_{c,1}, ..., x_{c,t})$ is the average dissimilarity (1 minus average cosine similarity) of previous ts interacted items' embeddings in session c. During the training phase, we treat $R_{c,t}$ as the prediction target rather than an input, with the aim of eliminating the need for searching for optimal settings during the inference stage.

IntraDT

Given the goal return $\tilde{R}_c$, intra-session novelty return $R^n_{c,t}$, intra-session state $s_{c,t}$, and action a_t (next item), intra-session DT models $(\tilde{R}_c, R^n_{c,t}, s_{c,t}, a_{c,t})$ sequentially. Specifically, the input at timestamp t is constructed as:

$$Z_{c,t} = stack(\tilde{R}_c, R^n_{c,t}, s_{c,t}, a_{c,t}) \in \mathbb{R}^{4\times d}, \tag{5.9}$$

Therefore, the complete input sequence of Z_c for InterDT is:

$$Z_c = \{Z_{c,1}, \ldots, Z_{c,T}\}, \tag{5.10}$$

$$\tilde{Z}_c = \mathbf{T}_2(Z_c), \tag{5.11}$$

where $\tilde{Z} = \{\tilde{Z}_{c,1}, ..., \tilde{Z}_{c,T}\}$. $\mathbf{T}_2$ is the GPT [126] architecture to autoregressively predict actions conditioned on returns and states. The hidden state $\tilde{R}_{c,t}$ and $\tilde{s}_{c,t}$ at each step t are then unstacked from $\tilde{Z}_t$:

$$\tilde{R}_{c,t}, \_, \tilde{s}_{c,t}, \_ = unstack(\tilde{Z}_{c,t}), \tag{5.12}$$

where $\tilde{R}_{c,t}$ is mapped by an MLP layer to produce the predicted intra-session unexpected return $R_{c,t}$:

$$\hat{R}_{c,t} = W_r \tilde{R}_{c,t} + b_r, \tag{5.13}$$

where $W_r \in \mathbb{R}^{d\times 1}$ are trainable parameters, $b_r \in \mathbb{R}^1$ are bias vectors. The actions (next items) are subsequently predicted based on the hidden state $\tilde{s}_{c,t}$:

$$\hat{a}_{c,i} = softmax(\delta(W_a \tilde{s}_{c,t} + b_a)), \tag{5.14}$$

where δ denotes the activation function, $\hat{a}_{c,i}$ is the predicted score on i-th candidate action, $W_a \in \mathbb{R}^{d\times N}$ are trainable parameters, and $b_a \in \mathbb{R}^N$ are bias vectors.

5.3.5 Overall Learning

Training Objective

The overall learning optimization process of HDT consists of a goal loss in InterDT, a sequential loss, a return loss, and an action loss in IntraDT.

In InterDT, g_c is used to predict the first action (item) in the current session C_c via the goal

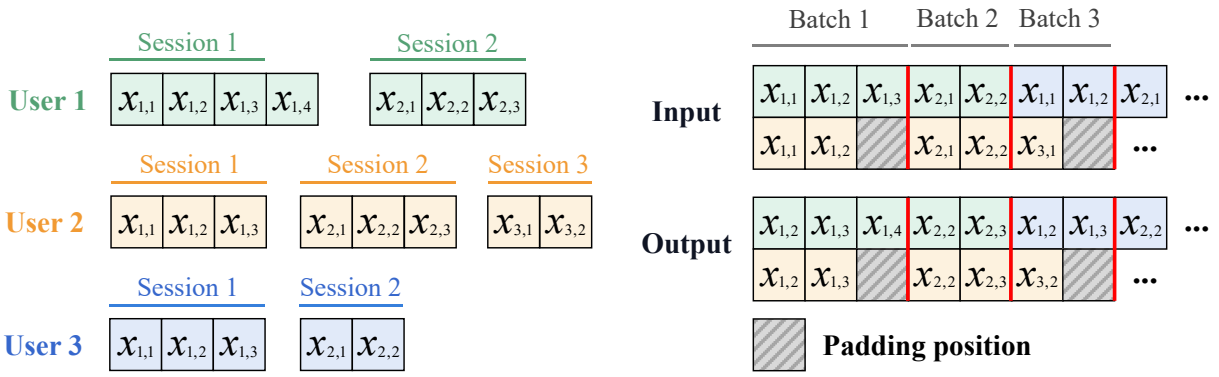


Figure 5.2: *Left*: user data. *Right*: user-parallel mini-batches for batch size 2.

loss:

$$\hat{y}_i = softmax(\delta(W_u g_c + b_u)), \tag{5.15}$$

$$\mathscr{L}_g = -\sum_{i=1}^{n} y_i \log(\hat{y}_i), \tag{5.16}$$

where δ is the activation function, $W_u \in \mathbb{R}^{d \times N}$ are trainable parameters, $b_u \in \mathbb{R}^N$ is the bias vector. $\hat{y}_i$ is the probability that the user interacted with i-th item at the first step of session C_c. $y_i = 1$ if the session starts with the i-th item, otherwise $y_i = 0$.

The sequential model G is optimized by a sequential loss to predict next items in a session:

$$\hat{y}_{c,i} = softmax(\delta(W_s s_{c,t} + b_s)), \tag{5.17}$$

$$\mathscr{L}_S = -\sum_{i=1}^{n} y_{c,i} \log(\hat{y}_{c,i}), \tag{5.18}$$

where $s_{c,t}$ is mapped into n classification logits on the candidate items by a fully connected layer with softmax function. $W_s \in \mathbb{R}^{d \times N}$ are trainable parameters, and $b_s \in \mathbb{R}^N$ is the bias vector. $y_{c,i}$ is the logit of i-th action. $W_s \in \mathbb{R}^{d \times N}$ are trainable parameters, and $b_s \in \mathbb{R}^N$ is the bias vector. $y_{c,i} = 1$ if the user interacted with the i-th item at the next time step in the current session c. Otherwise, $y_{c,i} = 0$.

In IntraDT, to enhance the stability of the balance between accuracy and diversity, novelty, and serendipity objectives, we also introduce the learning of the intra-session unexpected return by the Mean Squared Error (MSE) loss:

$$\mathscr{L}_R = (1/T)\sum_{t=1}^{T}(R_{c,t} - \hat{R}_{c,t})^2, \tag{5.19}$$

where $R_{c,t}$ is the target intra-session unexpected return, $\hat{R}_{c,t}$ is the predicted value.

Furthermore, an action loss in a cross-entropy manner is used to predict the actions (next

items):

$$\mathscr{L}_a = -(1/T)\sum_{t=1}^{T}\sum_{i=1}^{n} a_{c,t}\log(\hat{a}_{c,i}), \tag{5.20}$$

$a_{c,i} = 1$ if the user interacted with the i-th item at the next time step in session c. Otherwise, $a_{c,i} = 0$.

Finally, we jointly train all the loss on the training data to update all components of HDT:

$$\mathscr{L} = \mathscr{L}_S + \alpha\mathscr{L}_R + \beta\mathscr{L}_a + \lambda\mathscr{L}_g, \tag{5.21}$$

where α and β are hyperparameters that control the weights of intra-session unexpected return prediction loss and intra-session action prediction loss. λ is the weight of the first action prediction loss in the inter-session layer. In the inference, items are ranked by the scores from the sequential model and the interDT. The impact of these hyperparameters will be discussed in section 5.4.6.

Efficient Offline Training

Algorithm 3 Offline training set generation.

Input: user-item interaction sessions set $\mathscr{D}_{ts}$, novel item set I_{nov}
Output: multi-objective training set $\mathscr{D}_{tr}$

repeat
  Sample a user with sequential sessions $C = \{C_1, \ldots, C_m\}$
  from $\mathscr{D}_{ts}$, where $C_c = \{x_{c,1}, \ldots, x_{c,T}\}, c = 1, \ldots, m$
  for c=1 : m **do**
    Compute R_c according to Eq. 5.2
    for i=1 : T **do**
      Compute $R_{c,1:T}$ according to Eq. 5.8
      Compute $R^n_{c,1:T}$ according to Eq. 5.7
    end for
  end for
  $\mathscr{D}_{tr}$.append$(C, R_{1:m}, R_{1:m,1:T}, R^n_{1:m,1:T})$
  $\mathscr{D}_{ts}$.remove(C)
until $\mathscr{D}_{ts} = \emptyset$
return $\mathscr{D}_{tr}$

Algorithm 3 outlines the procedure for generating the offline training set $\mathscr{D}_{tr}$ from training data $\mathscr{D}_{ts}$. Specifically, the HDT framework is trained in an end-to-end manner, while the InterDT is updated between sessions. The InterDT captures the evolution of user preferences across sessions sequentially, which personalizes the sequential model and IntraDT then generates recommendations. To improve training efficiency, we adapt the user-parallel mini-batch training strategy [35] to HDT, where sessions are grouped by users and sorted by timestamp

(see Figure 5.2). In the iteration, the initial session with sequence $\{x_{1,1},...,x_{1,T}\}$ of the first B users are fed into HDT; the sequence $\{x_{1,2},...,x_{1,T+1}\}$ is used to construct the sequence of next items/actions for the intra-session part. In the next iteration, the second session of the first B users continue the process, and so on. Sessions of users are processed in chronological order. Once a user has been completely processed, the inter-session state s_1 to InterDT is reset and the next new user takes its place in the mini-batch. With both generative training and user-parallel mini-batches, we can train HDT efficiently over users with different numbers of sessions.

Table 5.2: Dataset statistics († denotes mean).

Dataset	Album	Reddit
Users	14,341	11,160
Items	129,273	21,583
Sessions	115,225	321,878
Events	1,051,698	1,234,442
Events per item†	8.13	57.19
Events per session†	9.12	3.83
Sessions per user†	8.03	28.84
Training - Events	915,195	1,194,739
Training - Sessions	100,890	310,719
Test - Events	136,503	39,703
Test - Sessions	14,335	11,159

5.4 Experiments

In this chapter, we aim to address the following research questions:

RQ1: How does HDT architecture personalize the sequential methods to improve user satisfaction across sessions, focusing on the objectives of diversity, novelty, and serendipity without sacrificing the accuracy performance?

RQ2: How do the intra-session unexpected and novelty returns affect the recommendation in current session?

RQ3: How to select weights of training objectives to balance accuracy, diversity, novelty, and serendipity?

5.4.1 Datasets

We perform experiments on two real-world datasets: **Album** [93] and **Reddit** [35]. The detailed statistics are summarized in Table 5.2. The Album dataset contains one-week listening events of users on albums, collected by the music streaming platform Last.fm[1]. The second Reddit

[1]http://www.last.fm/api

dataset came from the Reddit social media platform and includes interactions made by users over a five-day period.

To reduce the noise in the Album dataset, we remove consecutive repeated interactions while retaining the first timestamp. Both datasets are preprocessed following the approach described in [35], and user interactions are divided into sessions using a 60-minute threshold. More precisely, we filter items with less than three interactions. Then, sessions with less than three interactions are excluded. Finally, users with at least five sessions are retained to ensure sufficient cross-session information. For hyperparameter tuning, we further partition the training set using the same procedure to create a validation dataset. We conduct each experiment[2] five times and report the average results.

Same as [35], the testing set consists of all interactions of the last session of each user, and the sessions before the test (last) sessions form the training set. For example, for a user with sessions $C = \{C_1, C_2, C_3\}$, the training set are the $C_1 = \{x_{1,1}, x_{1,2}\}$ and $C_2 = \{x_{2,1}, x_{2,2}, x_{2,3}\}$, and the last session $C_3 = \{x_{3,1}, x_{3,2}, x_{3,3}\}$ is the testing set. During the inference stage, historical sessions, current session, and truth next item are built as $\{\{x_{1,1}, x_{1,2}\}, \{x_{2,1}, x_{2,2}, x_{2,3}\}\}$, $\{x_{3,1}\}$, and $x_{3,2}$; $\{\{x_{1,1}, x_{1,2}\}, \{x_{2,1}, x_{2,2}, x_{2,3}\}\}$, $\{x_{3,1}, x_{3,2}\}$, and $x_{3,3}$.

5.4.2 Baselines and Settings

Our proposed HDT can personalize many sequential models to recommend the next items by strictly adhering to the sequential dependencies across sessions and order information within a recent session. To verify its generalization, we integrate HDT with four well-known sequential recommendation models:

- GRU4Rec [26] that employs a GRU layer to capture user interactions within a session.
- Caser [31] that utilizes convolution operations to process previous item embeddings based on CNN.
- NItNet [32] that improves the receptive field via dilated CNN operations and deepens the network by residual connections.
- SASRec [33] that is based on self-attention to encode sequential items in a session.

Each sequential recommendation model is adapted and compared with the following approaches:

- Normal: Sequential recommendation models Trained by the normal cross-entropy loss. Sessions from the same user are concatenated into a single session to form the training data.

[2] https://drive.google.com/drive/folders/1AnN7dyf3_-i79ZPfvQe-cSjlJqcujvBU?usp=sharing

- SMORL [52]: A state-of-the-art RL-based multi-objective sequential recommendation model that balances accuracy, diversity, and novelty based on Q-learning to reward each interaction. We implement it with the same setting as the Normal method.
- Hier [35, 36]: The only hierarchical structure-based PSR models that recommend items following strictly order information across and within sessions. Besides the original RNN [35] and CNN-based [36] methods, we extend Hier to SASRec for comprehensive comparison.
- InterDT: Hier methods that incorporate inter-session DT. Item prediction is done by the intra-session state and goal state.
- HDT: Our proposed Hierarchical Decision Transformers.

The length of user sessions is limited to 20, and longer sessions are truncated to retain only the most recent interactions. Padding tokens are added to shorter sessions in the end to ensure the same length. The user mini-batch size is set to 100 for the Album dataset and 200 for the Reddit dataset. To ensure a consistent and fair comparison, all models and variations are trained using the Adam optimizer [52], and the learning rate is searched from [0.005, 0.001, 0.0005, 0.0001]. Evaluation of the validation set is performed every 1,000 batches of updates. The embedding sizes for item, state, return, and action are all set to 64. For GRU, the hidden state size is also set to 64. Caser uses 1 vertical convolution filter and 16 horizontal filters with heights set from 2,3,4. NItNet follows the parameters described in the original paper. For SASRec, the number of heads in self-attention is set to 1, and the dropout ratio is set to 0.1, as described in [33].

5.4.3 Metrics

We apply the following metrics to comprehensively evaluate the performance of HDT in balancing multiple objectives.

Accuracy. Hit Ratio (HR@k) [52] and Normalized Discounted Cumulative Gain (NDCG@k) [96] are applied to measure the relevance of recommended items, where $k \in \{10, 20\}$. We report the average results over all interactions of the test sessions.

Diversity & Novelty. We measure the aggregate diversity and novelty at the system level by the commonly applied Coverage (CV) [98] metrics. $\text{CV}_d@k$, $k \in \{5, 10\}$ for diversity is calculated as a percentage of the whole item set covered by all top-k recommended items for all events in test sessions. $\text{CV}_n@k$ for novelty is calculated based on the novel items.

Unexpectedness & Serendipity. Unexpectedness (UP@k) metric, where $k \in \{5, 10\}$, evaluates the ability of an RS to recommend items that are unexpected for a user given his historical sessions. In this chapter, UP@k for a user is formulated as the disparity between the top-k recommended items for current session and the interacted items in the past sessions, i.e., $\text{UP}@k = \frac{1}{U}\sum_{u=1}^{U} UNSEEN_u$, where $UNSEEN_u$ is the count of items in the top-k recommenda-

Table 5.3: Results on Album dataset. NG denotes NDCG. Unexp. denotes Unexpectedness. Boldface denotes highest scores. Second-best scores are marked with __. $*$ denotes the significance p-value < 0.05 compared to best baseline of first three models.

Models	Accuracy				Diversity		Novelty		Unexp.		Serendipity	
	HR@10	NG@10	HR@20	NG@20	CV_d@5	CV_d@10	CV_n@5	CV_n@10	UP@5	UP@10	HR_{ser}@10	NG_{ser}@10
GRU4Rec	0.0761	0.0485	0.0994	0.0543	0.4151	0.5285	0.3551	0.4701	2.524	3.462	0.0678	0.0463
GRU-SMORL	0.0765	0.0484	0.0983	0.0563	0.4739	0.5909	0.4102	0.5472	2.932	4.202	0.0842	0.0573
Hier-GRU	0.0814	0.0568	0.1004	0.0622	0.4119	0.5199	0.3602	0.4799	2.885	4.112	0.0743	0.0469
InterDT-GRU	0.0845	0.0574	0.1086	0.0629	0.5411	0.6601	0.4691	0.6069	3.034	4.216	0.0862	0.0591
HDT-GRU	**0.0896***	**0.0628***	**0.1107***	**0.0681***	**0.5846***	**0.6948***	**0.5395***	**0.6611***	**3.122***	**4.301***	**0.1023***	**0.0769***
Caser	0.0637	0.0291	0.0932	0.0365	0.2617	0.3428	0.1906	0.2752	2.811	3.826	0.0655	0.0337
Caser-SMORL	0.0643	0.0299	0.0968	0.0385	0.2844	0.3758	0.2356	0.3245	2.967	3.967	0.0795	0.0519
Hier-Caser	0.0634	0.0298	0.0945	0.0376	0.2301	0.2984	0.1567	0.2253	2.535	3.531	0.0764	0.0507
InterDT-Caser	0.0661	0.0313	0.0972	0.0392	0.2748	0.3632	0.2305	0.3199	2.667	3.713	0.0792	0.0511
HDT-Caser	**0.0783***	**0.0435***	**0.1084***	**0.0453***	**0.3024***	**0.4122***	**0.2458***	**0.3515***	**3.103***	**4.355***	**0.0843***	**0.0553***
NItNet	0.0803	0.0548	0.1007	0.0603	0.5402	0.6468	0.5234	0.6424	2.955	3.873	0.1005	0.0632
NItNet-SMORL	0.0813	0.0552	0.1009	0.0612	0.5834	0.6956	0.5432	0.6601	3.462	4.474	0.1019	0.0628
Hier-NItNet	0.0862	0.0614	0.1041	0.0658	0.5515	0.6757	0.5081	0.6427	3.314	4.366	0.1093	0.0711
InterDT-NItNet	0.0920	0.0666	0.1091	0.0708	0.5936	0.7126	0.5546	0.6856	3.514	4.576	0.1181	0.0799
HDT-NItNet	**0.0946***	**0.0673***	**0.1127***	**0.0719***	**0.6241***	**0.7368***	**0.5837***	**0.7078***	**3.622***	**4.782***	**0.1219***	**0.0809***
SASRec	0.1061	0.0742	0.1282	0.0798	0.4728	0.5743	0.4186	0.5291	3.495	4.563	0.1292	0.0813
SAS-SMORL	0.1084	0.0753	0.1301	0.0808	0.5144	0.6186	0.4642	0.5914	4.092	5.364	0.1486	0.0912
Hier-SAS	0.1163	0.0786	0.1431	0.0853	0.4809	0.5851	0.4276	0.5411	3.957	5.253	0.1445	0.0882
InterDT-SAS	0.1178	0.0802	0.1451	0.0871	0.4923	0.5951	0.4152	0.5476	4.014	5.327	0.1467	0.0903
HDT-SAS	**0.1202***	**0.0816***	**0.1462***	**0.0879***	**0.5742***	**0.6891***	**0.5285***	**0.6548***	**4.195***	**5.525***	**0.1503***	**0.0924***

tion list that are not interacted by user u before. The unexpectedness is reported as the average value of all users in the test set.

To evaluate serendipity, i.e., unexpected and relevant items for a user, we adopt HR$ser@k$ [137], where $k \in \{10\}$, to measure if a serendipity item (ground truth and unexpected item for the user) is recommended in top-k list. Similar to accuracy metrics, the Serendipity-Based Normalized Discounted Cumulative Gain (NDCG$ser@k$) [137] is also used to calculate the rank of the serendipity items. The results are reported as the average value over all serendipity interactions of the test sessions.

5.4.4 Performance Comparison (RQ1)

In this section, we analyze the performance of various recommendation methods across multiple objectives using the Album and Reddit datasets, as detailed in Tables 5.3 and 5.4. Our HD method outperforms other techniques, achieving the best balance across all evaluated metrics. For instance, when employing the Caser-based approaches on the Album dataset, HDT-Caser enhances performance by 18.4% in HR@10, 13.4% in CV_d@10, 17.2% in UP@10, and 6.4% in HR_{ser}@10, all relative to InterDT-Caser. HDT not only sets new benchmarks in these metrics but also uniquely improves both accuracy and unexpectedness, a combination where conventional sequential methods typically show high unexpectedness but lower accuracy. While GRU-SMORL and Caser-SMORL on the Reddit dataset achieve marginally better results in diversity and novelty, they fall short of HDT's performance across other metrics.

Additionally, Hierarchical (Hier) method exhibits strong improvements in accuracy-related metrics; for example, Hier-NItNet on the Album dataset reports up to a 12% increase in NDCG@10. This underscores the significant influence of modeling the evolution of user preferences across sessions. However, it is notable that the improvements in accuracy by Hier- models do not extend equally to diversity and serendipity. For instance, while Hier-GRU on the Album dataset shows a 6.96% improvement in HR@10 over standard GRU, it suffers from noticeable declines in diversity and novelty. This trend is consistent across other hierarchical implementations like Hier-Caser and Hier-NItNet on the Reddit dataset when compared with baseline methods like Caser and NItNet.

InterDT models are more effective compared with Hier methods, particularly InterDT-GRU and InterDT-NitNet, which enhance accuracy, diversity, and serendipity concurrently.

Specifically, InterDT-Caser on the Album dataset shows increases of about 4.2% in HR@10, 41.9% in CV_d@10, 5.2% in UP@10, and 3.6% in HR_{ser}@10. These gains are attributed to the model's capability to track the user's goal state across multiple objectives effectively.

In conclusion, while hierarchical models excel in capturing user interest evolution to recommend more accurate items, they do not equally enhance other objectives like diversity and serendipity. While HDT achieves a superior balance across all metrics, setting a new standard for multi-objective recommendation systems.

Table 5.4: Results on Reddit dataset. NG denotes NDCG. Unexp. denotes Unexpectedness. Boldface denotes highest scores. Second-best scores are marked with __. $*$ denotes the significance p-value < 0.05 compared to best baseline of first three models.

Models	Accuracy				Diversity		Novelty		Unexp.		Serendipity	
	HR@10	NG@10	HR@20	NG@20	CV_d@5	CV_d@10	CV_n@5	CV_n@10	UP@5	UP@10	HR_{ser}@10	NG_{ser}@10
GRU4Rec	0.2653	0.1615	0.3448	0.1816	0.3695	0.4843	0.3042	0.4278	3.058	4.755	0.2855	0.1821
GRU-SMORL	0.2948	0.1892	0.3858	0.2013	**0.3954**	**0.5045**	0.3423	**0.4544**	3.145	4.996	0.3171	0.2252
Hier-GRU	0.3711	0.2205	0.4769	0.2473	0.2778	0.3673	0.2161	0.3069	3.099	5.037	0.3052	0.2131
InterDT-GRU	0.3675	0.2156	0.4703	0.2484	0.2949	0.3783	0.2302	0.3171	3.181	5.121	0.3101	0.2169
HDT-GRU	**0.3754**	**0.2245**	**0.4829***	**0.2517***	0.3869	0.4871	**0.3581***	0.4454	**3.257***	**5.186***	**0.3361***	**0.2355***
Caser	0.2489	0.1165	0.3482	0.1416	0.2001	0.2344	0.1143	0.1556	2.882	4.754	0.2391	0.1092
Caser-SMORL	0.2632	0.1376	0.3685	0.1725	**0.2265**	**0.2644**	**0.1378**	0.1694	3.240	5.145	0.2412	0.1122
Hier-Caser	0.3159	0.1699	0.4192	0.1961	0.1638	0.1957	0.0798	0.1088	3.845	5.703	0.2377	0.1117
InterDT-Caser	0.3109	0.1723	0.4082	0.1998	0.1663	0.1968	0.0873	0.1149	3.878	5.722	0.2481	0.1177
HDT-Caser	**0.3212***	**0.1745***	**0.4239***	**0.2005***	0.2067	0.2541	0.1368	**0.1796***	**3.928***	**5.837***	**0.2937**	**0.1632***
NItNet	0.2421	0.1437	0.3235	0.1643	0.2496	0.3193	0.1858	0.2531	2.961	4.696	0.2612	0.1525
NItNet-SMORL	0.2741	0.1767	0.3623	0.1842	0.2606	0.3264	0.2081	0.2734	3.143	4.991	0.3047	0.1875
Hier-NItNet	0.3144	0.1845	0.4135	0.2096	0.2112	0.2793	0.1495	0.2128	3.132	4.926	0.3022	0.1793
InterDT-NItNet	0.3215	0.1917	0.4185	0.2162	0.2001	0.2608	0.1312	0.1879	3.214	5.066	0.3149	0.1997
HDT-NItNet	**0.3368***	**0.2072***	**0.4311***	**0.2312***	**0.2849***	**0.3599***	**0.2231***	**0.2976***	**3.443***	**5.288***	**0.3547***	**0.2204***
SASRec	0.2667	0.1504	0.3692	0.1743	0.2314	0.2869	0.2098	0.2571	2.861	4.542	0.2765	0.1552
SASRec-SMORL	0.2845	0.1856	0.3923	0.2054	0.2954	0.3867	0.2654	0.3462	3.634	5.584	0.3362	0.2363
Hier-SASRec	0.3937	0.2342	0.4969	0.2603	0.2891	0.3677	0.2475	0.3254	3.381	5.361	0.3854	0.2175
InterDT-SASRec	0.4032	0.2519	0.5017	0.2767	0.2681	0.3366	0.2278	0.2884	3.721	5.736	0.4182	0.2603
HDT-SASRec	**0.4073***	**0.2544***	**0.5092***	**0.2802***	**0.3445***	**0.4331***	**0.2972***	**0.3841***	**3.744***	**5.755***	**0.4354***	**0.2711***

Table 5.5: Effect of intra-session returns on Accuracy, Diversity, Novelty, Unexpectedness, Serendipity. uxp/nov denotes the variants of integrating either the intra-session unexpected or novelty return. Boldface denotes best results.

	Methods	Acc.	Div.	Nov.	UP.	Ser.
		HR@10	CV_d@5	CV_n@5	UP@5	HR_{ser}@10
Album	uxp	0.1182	0.5272	0.4768	4.042	0.1482
	nov	0.1197	0.5706	0.5264	4.001	0.1242
	HDT	**0.1202**	**0.5742**	**0.5285**	**4.195**	**0.1503**
Reddit	uxp	0.4036	0.3209	0.2575	3.721	0.4123
	nov	0.4021	0.3408	**0.2998**	3.311	0.4054
	HDT	**0.4073**	**0.3445**	0.2972	**3.744**	**0.4354**

5.4.5 Intra-session Return Investigation (RQ2)

We examine the impact of intra-session unexpected (uxp) and novelty (nov) returns by the IntraDT-SAS model, analyzing their influence on the performance of Album dataset. The results are presented in Table 5.5. Considering both nov and uxp, we note that they achieve comparable accuracy performance, with uxp achieving higher scores on serendipity and unexpectedness, while nov performs higher in diversity and novelty. This demonstrates that the intra-session unexpectedness return guides the model to recommend items that the user has not interacted with personally, while the novelty return encourages recommending unexplored/new items of the system. Moreover, HDT consistently outperforms other variants across all evaluated objectives, confirming its robustness in effectively integrating multiple objective returns to enhance the recommendation quality. On the Reddit, the performance of uxp and nov appears to be comparable, further underscoring the versatility of the HDT in different contexts. The findings highlight the utility of tailored intra-session returns in guiding the recommendation process toward achieving specific objectives and the superior capability of HDT to deliver multiple benefits.

5.4.6 Hyperparameter Study (RQ3)

We optimize three key hyperparameters: α, β, and λ, which control the weights of the return loss $\mathscr{L}_R$, action loss $\mathscr{L}_a$, and goal loss $\mathscr{L}_g$ in Eq.5.21, respectively, influencing the model's ability to balance between accuracy and other performance metrics such as unexpectedness and diversity. We optimize one weight at a time and individually observe the changes in accuracy and the corresponding objective metric to determine the optimal value that balances the two objectives. Due to space constraints, we present the results for the Reddit dataset on a GRU-based model.

Figure 5.3a illustrates the impact of α, which controls the influence of the return loss $\mathscr{L}_R$, on the metrics HR@20 and UP@10. We observe that both metrics improve as α increases from

0 to 0.2. While HR@20 begins to decline when α continues to increase beyond 0.2, UP@10 continues to show positive growth. Based on these trends, we set α to 0.5 to optimize HR@20 and UP@10 simultaneously.

Diversity and novelty are mainly affected by the action loss $\mathscr{L}_a$. Figure 5.3b shows that increasing β results in a steady increase in CV (we show CV_d@10) and a slow decrease in HR@20. We set β as 0.2 to maintain a reasonable balance between diversity, novelty and accuracy. We observe influence of λ on HR@20 and UP@10 by ensuring accuracy while reducing potential adverse effects of goal loss $\mathscr{L}_g$ on other objectives. We set λ to 0.1 which best supports accuracy without excessively compromising unexpectedness.

For the Album dataset, the optimal settings for α, β, and λ are found to be 0.5, 0.1, and 0.2 respectively. These values reflect a tailored approach to balancing the multiple objectives of our recommendation system, ensuring that accuracy is maintained while also enhancing other vital metrics like diversity, novelty and serendipity.

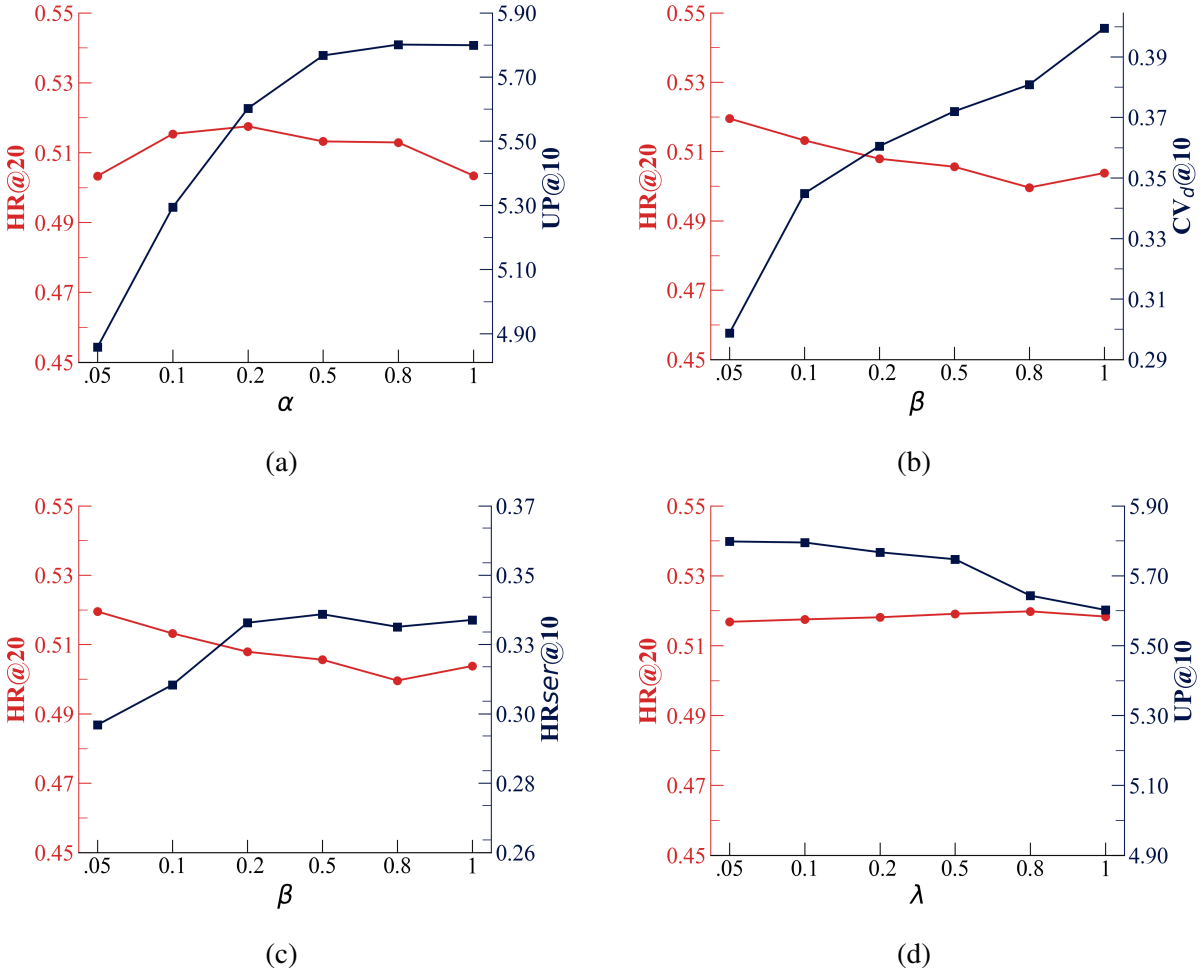


Figure 5.3: HDT with different weights of distinct objective loss on the Reddit dataset.

5.5 Conclusion

We proposed the novel Hierarchical Decision Transformers (HDT) framework for Personalized Session-based Recommendation (PSR). This framework models users' long-term interest changes across past sessions and their short-term preferences within the current session. To enhance user experience by addressing multiple objectives, we introduced the Multi-objective PSR (MOPSR) and integrated both inter-session and intra-session unexpected returns into the HDT. This integration aims to guide recommendations by balancing accuracy with diversity, novelty, and serendipity.

We established specific metrics for each objective and conducted comprehensive experiments on publicly available datasets. The results from these comparisons underscored the effectiveness of our method in managing the trade-offs between conflicting objectives in MOPSR. In the future, we plan to extend the proposed framework to encompass additional recommendation scenarios and balance more objectives, thereby broadening its applicability in real world.

5.5.1 Summary and Link to Next Chapters

This chapter completes the decision layer by extending multi-objective control to a hierarchical session-based setting (MOPSR), capturing both cross-session long-term preferences and within-session short-term intent. By introducing hierarchical (expected/unexpected) returns, HDT balances accuracy with diversity, novelty, and serendipity under realistic temporal progression. Chapters 6–7 then address the deployability layer, focusing on how to learn and deploy RL-based recommenders under biased offline logs, distribution shift, and online exploration risk.

Chapter 6

Reinforcement Learning-based Recommender Systems with Large Language Models for State Reward and Action Modeling

Note. This chapter is based on our SIGIR 2024 paper on LE/LEA, and is revised in this thesis to clarify the thesis stance that LLMs are used as auxiliary environment components (state/reward modeling and offline augmentation) rather than deployed recommenders, unify notation, and strengthen controlled ablations for where LLM assistance helps RL-based learning.

6.1 Introduction

RS has become an essential tool that navigates through extensive data to deliver relevant and engaging content to users [116, 127] in commercial platforms. Sequential or next-item recommendation [26, 31–33] have gained prominence, especially in music and video streaming RS, to recommend the next relevant item based on user-item interactions within a recent active session. Typically, sequential recommendation models [26], such as those based on gated recurrent units (GRU), convolutional neural networks [116], and Transformer [33], have been trained to predict the next interacted items based on historical user data in a self-supervised manner. To address suboptimal recommendations due to the dependence on supervised learning, self-supervised reinforcement learning (SSRL) has been proposed. This approach trains an RL agent to satisfy user expectations, e.g., the desire for diverse content, by employing sequential models with rewards tailored to accommodate various behaviours [52, 57]. However, recent efforts [57–59] to develop off-policy/offline RL policies trained on historical user data have been met with the challenge of constructing a high-quality environment that provides meaningful user feedback,

e.g., state representation and reward function.

LLMs with knowledge-transferring capabilities have recently received significant attention in RS [138–140]. In the context of sequential recommendation, LLMs have been shown to be acceptable zero-shot [86] or pre-trained [2] RS. These LLM-recommenders have been proven to perform on par with, or even outperform, conventional models. Recent research has heavily focused on fine-tuning/pre-training with user data to adapt LLM for recommendations. This results in large models with orders of magnitude more parameters than traditional recommendation models. Despite the adoption of efficient tuning methods [141, 142], the computational cost of inference with LLM models with billions of parameters remains very high, making serving LLM recommendation models impractical in real-world scenarios.

Motivated by their generative and language-understanding capabilities, we propose adapting LLM as an environment (LE) to model user behavior and return feedback for training the RL-based recommenders. More importantly, LE can help train leaner and more adaptable sequential recommendation models that outperform those trained with standard techniques without incurring additional computational costs at inference time. Specifically, we address the following limitations of the above-mentioned state and reward problems for RL-based RS: Recommendation approaches based on reinforcement learning typically utilize state representations derived from generative sequential models. These models, such as Transformers or RNNs, process user-logged actions to produce output or hidden states, which are then employed to generate recommendations. LLMs have been shown to be world models [143, 144], which may help generate more accurate representations of user state given the sequence of historical user actions. Additionally, in typical RL-based recommenders, rewards are usually predefined with respect to behavior categories [57], for example, purchase and click. However, a simple uniform reward setting might not accurately reflect user satisfaction across items, resulting in an agent that is unable to capture latent differences between actions. LLMs are capable of generating good reward estimates [91] that can be used to train RL algorithms. To this end, we capitalize on these powerful LLM properties to learn an environment (LE) that acts as state and reward model to return high-utility feedback for training RL-based recommendation models.

To construct the LE we fine-tune the LLM by introducing an item tokenization strategy, using autoregressive training. We first learn semantically-rich tokens by using the items' textual descriptions. We subsequently fine-tune the LLM through a small subset of user data and adapters to obtain the reward model (RM) and state model (SM). Specifically, we prompt the LLM with instructions based on user-item token interactions to output the scalar reward by a score head. We then learn effective state representations by contrasting the user-item tokens' interactions with the positive and negative actions. In our modular architecture, RM and SM constitute the LE that enables the acquisition of the state representations and scalar rewards for the RL-based recommendation model.

Moreover, in an offline setting, the agent is trained on fixed historical user-item interactions

without probing the environment. Thus, we further propose an LE Augmentation (LEA) method prompting the obtained LE to enrich the offline data for RL-based sequential recommendation. In particular, LE is tasked with selecting potentially positive feedback by prompting it with a combination of user historical behavior and a sampled list of items. This step aims at leveraging the predicted positive items as positive samples to augment the training of the supervised learning component, and as positive actions to reinforce the training of the RL agent.

Finally, we train the supervised loss and the RL loss on the original historical user data and the augmented positive samples. At the inference stage, only the sequential model with the supervised head is used for the evaluation of the top-k recommendation performance, to guarantee efficiency. To validate the effectiveness of our method, we compare LEA with two state-of-the-art Q-value-based RL frameworks, with two sequential models as the backbone. We also directly apply LE to the above frameworks by enhancing the state representation and reward function, and demonstrate significant performance gains on two publicly available datasets. Our contributions can be summarised as follows:

- We propose an LLM-based Environment, acting as the state model and reward function, to improve the performance of the offline RL-based recommender systems.
- We present an efficient fine-tuning method for adapting LLMs for LE using limited user data. Additionally, we propose an item-tokenization strategy to incorporate user data and improve training efficiency.
- We introduce a positive feedback augmentation approach LEA to enhance both supervised learning and Q-learning. The LE is utilized as the behaviour policy to infer positive signals from historical user data.
- We apply the environment LE and augmentation method LEA to two state-of-the-art RL-based sequential recommendation models. Experimental results on real-world datasets show a general improvement across recommendation performance metrics.

6.2 Related Work

Sequential recommendation has evolved from deep learning models like GRU [26] and Transformer-based architectures [31, 33] towards RL frameworks, such as SQN [57] and SA2C [58], which model the task as a Markov Decision Process [28, 55]. However, these RL methods are often limited by simplistic, pre-defined rewards and state representations derived directly from the sequential model itself. Concurrently, LLMs are being adapted as powerful standalone recommenders through methods like fine-tuning [83, 86] or pre-training [2]. While effective, this approach introduces significant computational overhead, especially during inference. Our work diverges from these trends by proposing a novel synthesis: we leverage an LLM not as the

recommender, but as a sophisticated offline environment to generate nuanced state representations and reward signals, thereby enhancing existing, efficient RL frameworks without adding inference-time cost.

6.3 Method

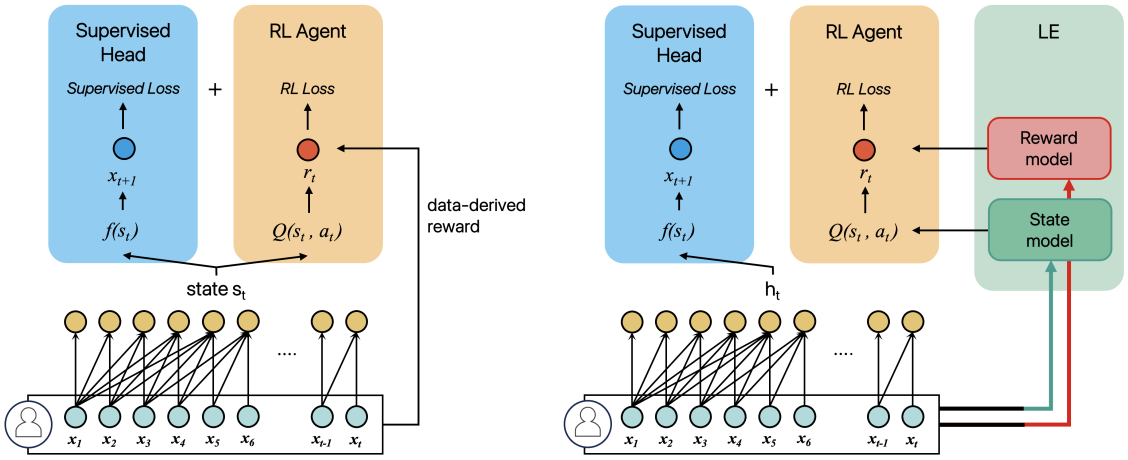
6.3.1 Task Formulation

Let I denote the set of items in the system, $x_{1:t} = \{x_1, \ldots, x_t\}$ denote the user-item interaction sequence, where $x_i \in I (0 < i \leq t)$ is the index of the interacted item ordered by timestamp. The goal is to recommend the next item x_{t+1} at timestamp $t+1$ for the user from the whole item set I, such that the user might be interested in given the previous interactions $x_{1:t}$. Sequential recommendation methods [57] have been proposed to tackle the task as a self-supervised learning problem and map the sequence $x_{1:t}$ into the hidden state h_t by a sequential model $G(\cdot)$, i.e., $h_t = G(x_{1:t})$. A fully connected layer then follows to map h_t to classification logits (ranking scores) on all candidate items at the next step. In RL-based recommendation, the task is further modeled as an MDP, where users are treated as the environment. As Figure 6.1a shows, the state from the environment is represented by the user's historical interactions, and an RL agent is additionally trained to interact with users by taking actions (i.e., recommending items) to maximize the cumulative rewards.

In previous methods, the user state is taken (re-used) from the hidden state h_t of the sequential model $G(\cdot)$, and the observed rewards are usually predefined w.r.t. specific behaviours, e.g., purchase=1, click=0.2. However, the uniform reward setting for all positive or negative actions fails to accurately capture the nuances of user preferences, resulting in an agent that cannot discern the latent differences between items for a user. Moreover, user states generated via $G(\cdot)$, based on implicit data comprised of item IDs, cannot reflect user behaviours on specific content. In offline training, the agent is usually trained on fixed historical user-item interactions without probing the environments due to the significant costs associated with error-prone explorations, which introduces the challenge of insufficient positive and negative signals. The proposed solution involves developing a state model to represent nuanced user states based on historical data, along with formulating an accurate reward model that assigns rewards contingent upon specific user states and actions. Additionally, we attempt to distill user's potential interests from the environment to augment the positive feedback for offline training, thereby reinforcing the exploration capabilities of RL-based RS.

6.3.2 Large Language Model as Environment

LLMs are considered as emergent world representations [143, 144] with transfer learning capabilities. They can adapt to specific downstream tasks with minimal data through fine-tuning



(a) Traditional SSRL4R (b) SSRL4R with state & reward models

Figure 6.1: Self-supervised Reinforcement Learning for Recommendation (SSRL4R). (a) shows the previous offline structure, where the state for the RL agent is the hidden state from the sequential model, and the reward value is a predefined scalar. (b) shows our proposed structure, where the state is generated from a separate state model, and the reward is from a reward model.

or prompt-based methods, e.g., optimization based on prompts and adapters that adjust a small number of parameters [80, 145] to generate question answers [146]. In this work, we leverage LLM as an offline user environment to return user feedback for RL-based RS attributes for three reasons. First, LLMs are ideal for creating environments that can simulate user queries/behaviours and feedback due to their ability to generate natural language [138, 147]. Second, environments created with LLMs significantly decrease the cost incurred by RL experiments in online settings, as error explorations could impair the user experience. Third, LLMs can generate feedback and shape rewards on various objectives, aiding RL-agents in learning complex signals that reflect user satisfaction. Therefore, we construct the LE based on the Transformer decoder-based LLM and use the latest Mistral 7B model for its simplicity and efficiency.

In summary, our objective is to efficiently tune LLMs with a small subset of offline user data, transforming them into a user environment (LE) capable of providing user feedback, including states, rewards, and predicted positive actions.

•**Item tokenization.** We first propose an initial step of tokenizing items based on their textual content, a preparation step for efficiently fine-tuning the LLM with user-item interaction data. Item tokenization addresses two issues. First, when representing items by textual content, e.g., descriptions of a product in Amazon, excessively long user-item interactions can lead to truncation and inefficiency [148], and the dilution of user signals for training an LE. Second, recent studies [90] represent semantic items via outputs of an encoder fed with textual content to support another recommendation model. However, such item embeddings are not suitable as the input of the LLM since they deviate from the LLM's token embedding distribution. Therefore, we aim to tokenize each item into the LLM's token embedding space $\mathscr{W}$ without losing its semantics and improve efficiency in learning an LE.

We achieve this goal by condensing the textual information of an item into a new item embedding space I^e belonging to $\mathscr{W}$. We refer to $i_i^e \in I^e$ as an item token, since it is not associated with an actual word, but is rather another representation of the items in $\mathscr{W}$. Given an item $i_i \in I$ with textual content, we build a sentence T and adopt an optimization-based approach that performs tokenization by updating the randomly initialized item token i_i^e for each item:

An example of T: S_* track is titled Live Forever from album Definitely Maybe, its artist is Oasis.

where S_* signifies a placeholder of i_i^e. T is fed to the decoder-only LLM. Figure 6.2a shows the process. The objective is to generate the next tokens of the sentence autoregressively, conditioned on the only-optimized item token. The final obtained item token encapsulates signals of descriptive content and serves as a semantic representation in LLM. We finally obtain the environment item token set I^e and the corresponding item ID embedding set I in the system, where their indices are aligned. The user-item interactions $x_{1:t}$ in the environment is denoted as $x_{1:t}^e$,where $x_i^e \in I^e (0 < i \leq t)$.

•Reward Model (RM). To learn a reward model from a small amount of historical data, we fine-tune the LLM to take input as a reward prompt concatenated by a user prompt p_t and an action prompt p_{a_t} to output a scalar reward:

Reward Prompt is the concatenated text of p_t and p_{a_t}: "$p_t p_{a_t}$", where $p_t \leftarrow$The user has listened to these tracks in chronological order: $x_{1:t}^e$ $p_{a_t} \leftarrow$Compute the likelihood that $item_X$ be the next track to be listened to based on the listening history.

where $item_X \in \{a_t^+, a_t^-\}$. a_t^+ is the truly interacted item x_{t+1}^e at next step and a_t^- is the negatively sampled one. As Figure 6.2b shows, the reward prompt p_t, p_{a_t} guides the LLM to generate the reward score based on the similarity between user-item token interaction and the specific action. A linear network ϕ is added as score head to generate rewards by the last hidden state. We use the Low-Rank Adaptation architecture (LoRA) [141] adapter ϕ for each Transformer block for computational efficiency[1]. The loss function for updating the RM is defined as:

$$\mathscr{L}_{rm}^e = -log[\sigma(r_{\theta+\phi}^e(p_t, p_{a_t^+}) - r_{\theta+\phi}^e(p_t, p_{a_t^-}))], \tag{6.1}$$

where $r^e \in [r_t^+, r_t^-]$ is the scalar reward conditioned on the reward prompt for action a_t^+ and a_t^- at timestamp t, while θ and ϕ represent the trained parameters of the score head and the adapter.

•State Model (SM). We learn an SM to refine state representations from historical interactions. As shown in Figure 6.2b, the user-item token interactions $x_{1:t}^e$ is fed to the LLM to generate the user state representation $s_t'^e$ at timestamp t, which is represented by the last hidden state of the outputs. Given that the state is modeled from a sequence of actions, where the difference between two consecutive states is the addition of the next interacted item in the input of the LE, there is a possibility that consecutive states could be quite similar. To ensure the model can

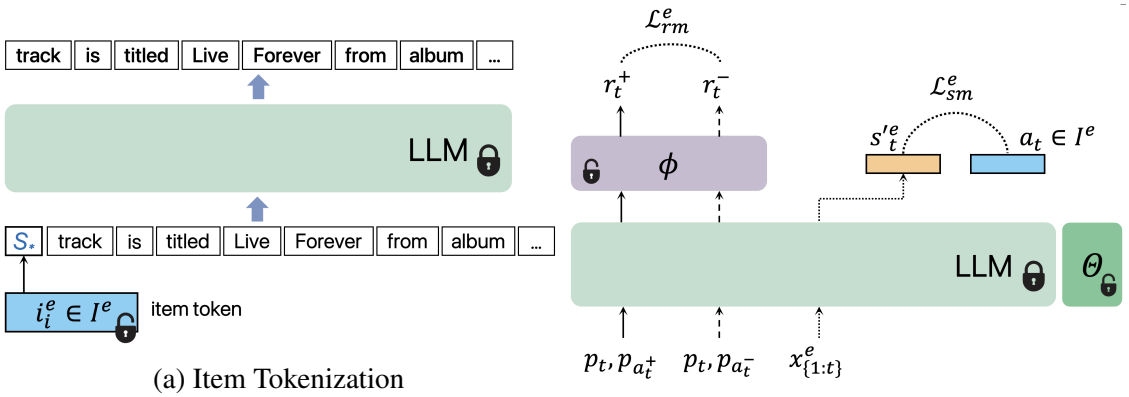
[1]We utilize the established prompt engineering [82, 147] and adapter techniques [86, 141].

differentiate states even when their interaction patterns are similar, we employ a contrastive loss:

$$\mathscr{L}^e_{sm} = -\frac{1}{B}\sum_{j=1}^{B} log(\sigma(\, s'^e_t \cdot a^+_t - \, s'^e_t \cdot a^j)), \tag{6.2}$$

where B is the batch size. For each sample, s'^e_t is the state at timestamp t generated by $x^e_{1:t}$, a^+_t is the next interacted item and a_j is the j-th action of the batch. Parameter ϕ is updated.

•Discussion. The SM loss computes the disparity between the user state and token embeddings of true and sampled actions. While the RM loss takes both user interaction and sampled actions as input, the score differences are generated by the reward values. We train RM and SM simultaneously using a shared adapter on a small amount of offline historical user data, e.g., 10% interactions (Figure 6.2b). After fine-tuning RM and SM, we obtain the LE that can generate user feedback for our RL-based recommendation tasks.



(a) Item Tokenization

(b) Prompt-based fine-tuning of LE

Figure 6.2: Our approach of adapting decoder-only LLM as Environment (LE). (a) we produce token $i^e_i \in I^e$ for item $i_i \in I$ by optimizing the objective of generating the next tokens of its textual content autoregressively. (b) we learn the LE by parameter-efficient adapters ϕ on a small subset of user data. User-item token interactions $x^e_{1:t}$, where $x^e_i \in I^e$, is the input to generate the state representation s^e_t. We enhance the state representation by comparing the similarity between the state and actions through loss $\mathscr{L}_{sm}$. Reward prompt p_t, p_{a_t} contains $x^e_{1:t}$ and action $a_t \in [a^+_t, a^-_t]$, where a^+_t is the positive action (next interacted item), and a^-_t is the negative action (sampled uninteracted item). The action-specific reward for a user is produced by a score head θ, and the LLM is trained by comparing user preferences for actions via loss $\mathscr{L}_{rm}$.

6.3.3 LE for RL-based Recommenders

We reframe the RL-based sequential recommendation as a novel LE-based Markov Decision Process (LEMDP), discussed in section 6.3.1. The process by which the agent interacts with the LE to obtain user feedback is shown in Figure 6.3, which can be represented by a tuple of $(\mathscr{S}^e, \mathscr{A}, \mathscr{P}, r^e, \gamma)$:

• State space $\mathscr{S}^e$ generated by the State Model (SM) $\mathscr{E}^e$ in LE. The set of states with the time

series, modeled by the user's historical interacted item tokens $x^e_{1:t}$ via the SM and the hidden state h_t from the sequential model:

$$\mathrm{s}'^e_t = \mathscr{E}^e(x^e_{1:t}), \tag{6.3}$$

$$\mathrm{s}^e_t = s'^e_t \parallel h_t, \text{where } h_t = G(x_{1:t}). \tag{6.4}$$

•Action space $\mathscr{A}$. The discrete action set is comprised of the candidate items. Taking action in LEMDP means recommending items. In the offline data, the action $a_t \in A$ at timestamp t is the interacted item in the next step, i.e., $a_t = x^e_{t+1}$.

•Reward Model (RM) r^e in LE. The reward function that returns immediate reward r_t as the state s^e_t and the action a_t taken by the agent at step t are observed:

$$r_t = r(s^e_t, a_t) = r^e(p_t, p_{a_t}), \tag{6.5}$$

where p_t, p_{a_t} is the reward prompt.

•State Transition Function $\mathscr{P}$. The transition function describes the next state from the environment given the observed action and the current state. When learning from offline data, only the positive actions affect the state.

• Discount factor γ. This defines the discount factor to the future rewards, where $\gamma \in [0,1]$.

RL aims to learn a target policy $\pi_\psi(a_t|s^e)$ that maps the state $s_e \in \mathscr{S}^e$ to an action distribution $a \in \mathscr{A}$ by maximizing the expected cumulative rewards (returns), where ψ denotes the parameters:

$$\max_{\pi_\psi} \mathbb{E}_{\tau\sim\pi_\psi}[R(\tau)], \text{ where } R(\tau) = \sum_{t=0}^{|\tau|} \gamma^t r(\mathrm{s}^e_t, a_t), \tag{6.6}$$

where τ denotes the trajectory of (s^e_t, a_t, s^e_{t+1}). We apply the LE to the value-based Q-learning algorithm [58] to train the target policy.

Following the objective in section 6.4.1, we improve the SNQN and SA2C [58] frameworks to combine supervised learning of the sequential model G and RL via the agent-LE interactions. More specifically, given the input item sequence $x_{1:t}$ and $G(\cdot)$ for self-supervised learning, the hidden representation is formulated as $h_t = G(x_{1:t})$. Then, h_t is used into a fully connected layer and a softmax function to output ranking scores over the candidate items:

$$\hat{Y}_i = softmax(\delta(W_u h_t + b_u)), \tag{6.7}$$

where $\hat{Y}_i = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_n] \in \mathbb{R}^n$. n is the number of items, δ is the activation function, $W_u \in \mathbb{R}^{d\times n}$ are trainable parameters, and $b_u \in \mathbb{R}^n$ is the bias vector. The self-supervised part is trained via the cross-entropy loss:

$$\mathscr{L}_h = -\sum_{i=1}^{n} y_i \log(\hat{y}_i), \tag{6.8}$$

$y_i = 1$ if the user interacted with the i-th item in the next timestamp. Otherwise, $y_i = 0$.

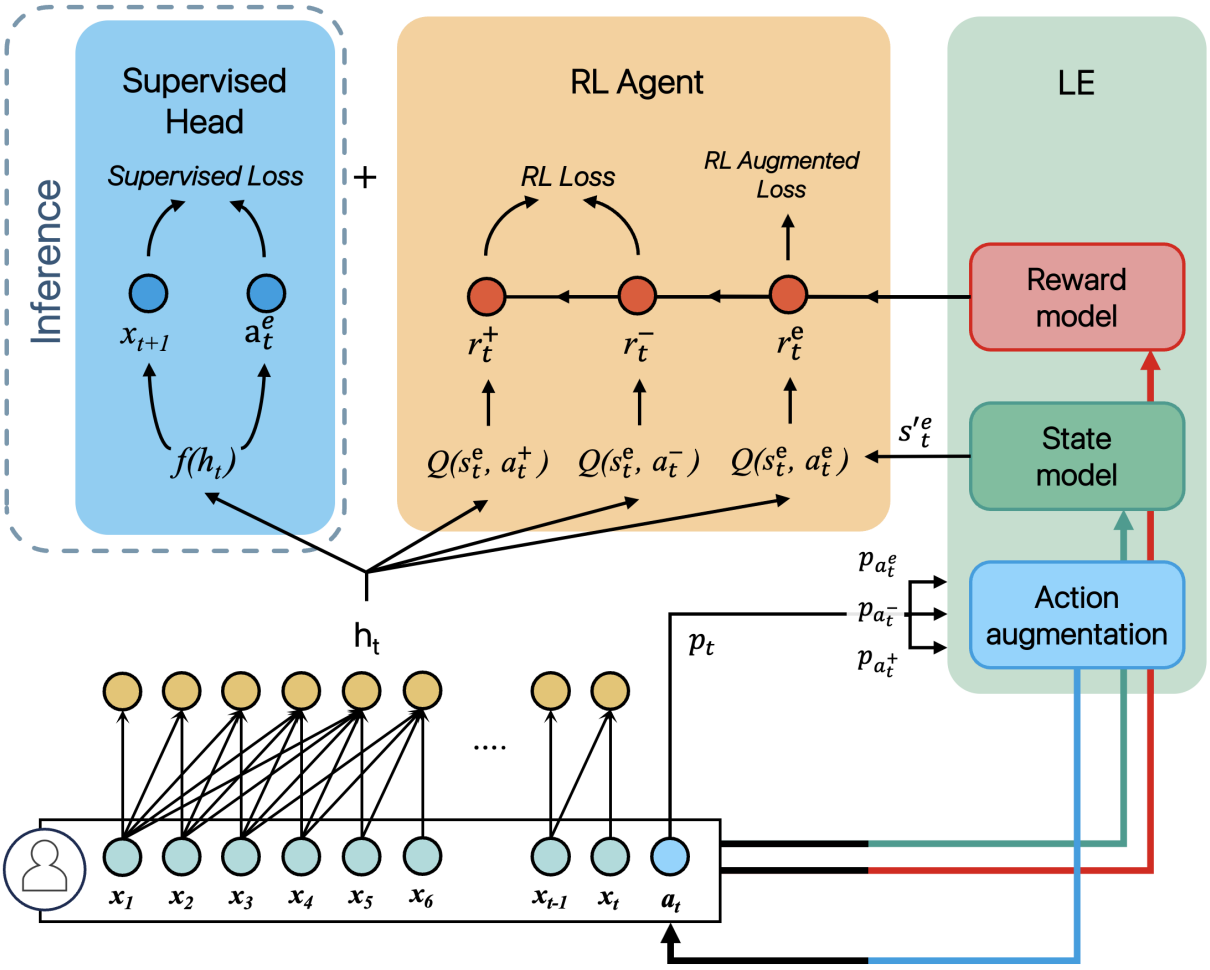


Figure 6.3: Structure of LEA. *Left*: the LE is applied to offline data. $(x_1^{(e)}, x_2^{(e)}, ..., x_t^{(e)})$ denotes the user-item interaction $x_{1:t}$ for the sequential model, where $x_i \in I$, and $x_{1:t}^e$ denotes the user-item token interaction for the LE, where $x_i^e \in I^e$. a_t^e is the positive action predicted by LE. *Right*: RL policy is trained via the original Q-loss and the augmented one $\mathscr{L}_{aq}$; the base sequential model is jointly trained through RL loss and the supervised loss over the original next item and the augmented one $\mathscr{L}_{ah}$ over a_t^e.

Regarding the Q-learning network, we obtain the state s_t^e by the SM in LE and compute the Q-value as follows:

$$Q(s_t^e, a_t) = \delta(W_q s_t^e + b_q), \tag{6.9}$$

where δ is the activation function, $W_q \in \mathbb{R}^{d \times n}$ are trainable parameters, and $b_q \in \mathbb{R}^n$ is the bias

vector. The one-step time difference (TD) Q-learning loss to improve SNQN is defined as:

$$\begin{aligned}\mathscr{L}_q = & \underbrace{(r^e(p_t, p_{a_t^+}) + \gamma \max_{a'} Q(\mathrm{s}_{t+1}^e, a') - Q(\mathrm{s}_t^e, a_t^+))^2}_{L_p\text{: positive TD error}} \\ & + \underbrace{(r^e(p_t, p_{a_t^-}) + \gamma \max_{a'} Q(\mathrm{s}_t^e, a') - Q(\mathrm{s}_t^e, a_t^-))^2}_{L_n\text{: negative TD error}},\end{aligned} \tag{6.10}$$

where a_t^+ and a_t^- are the positive action and sampled unobserved (negative) action at timestamp t, respectively. In our method, we only sample one negative action. We follow the assumption that taking negative actions will not affect the state. Therefore, the maximum operation is performed in $Q(\mathrm{s}^e{}_t, a')$ other than $Q(\mathrm{s}_{t+1}^e, a')$ in the negative TD error L_n. In SA2C, the advantage Q-value [60] is calculated to formulate the supervised loss:

$$A(\mathrm{s}_t^e, a_t^+) = (Q(\mathrm{s}_t^e, a^+) - Q(\mathrm{s}_t^e, a_t^-))/2. \tag{6.11}$$

Then the supervised loss $\mathscr{L}_h$ is formulated as $\mathscr{L}_h \leftarrow \mathscr{L}_h \cdot A(\mathrm{s}_t^e, a_t^+)$ for SA2C. The corresponding loss function used in the LE to directly improve the above RL frameworks is formulated as follows:

$$\mathscr{L}_c = \mathscr{L}_h + \mathscr{L}_q. \tag{6.12}$$

6.3.4 Augmentation via LE

Since the recommender is trained on historical data without online exploration, both the supervised model and the RL agent can only be trained on user-item interactions that exist in the offline training data. As a result, the models may fail to estimate the value functions for unseen user feedback. We propose an LE augmentation (LEA) method to augment the historical user data for offline training. We prompt the LE to predict items a_t^e the user is likely to interact with in the next timestep, these predicted positive actions are used for both supervised learning and Q-learning. We build an augmentation prompt template constructed by the user prompt p_t and a selection prompt p_{l_t} to generate feedback:

Augmentation Prompt is the text concatenation of p_t and p_{l_t}: "$p_t p_{l_t}$".
p_{l_t} ←In the list of the following 5 tracks:[$list_t$], based on the history, select the number of the track that he is most likely to continue to listen to.

where $list_t$ is the token sequence of the sampled items. After an early training stage, we sample the top-5 items from the supervised head to construct $list_t$. We feed the augmentation prompt into the LE and obtain the predefined item position classification (e.g., "`first`" for the first

Algorithm 4 Training procedure of LEA

Input: user-item interaction sequence set $\mathscr{X}$, recommendation model G, reinforcement head Q, supervised head, environment LE comprised of state model $\mathscr{E}^e$ and reward model r^e, item set I, item token set I^e

Output: all parameters in the learning space Θ

Initialize all trainable parameters
Create G' and Q' as copies of G and Q, respectively
repeat
    Draw a mini-batch of $(x_{1:t}, a_t)$ from $\mathscr{X}$, the corresponding user-item token interaction $x^e_{1:t}$, sample a negative action a_t^-, augmented action a^e_t
    $s^e{}_t = G(x_{1:t})||\mathscr{E}^e(x^e_{1:t})$, $s^e{}_{t+1} = G(x_{1:t+1})||\mathscr{E}^e(x^e_{1:t+1})$
    $s^{e\prime}{}_t = G'(x_{1:t})||\mathscr{E}^e(x^e_{1:t})$, $s^{e\prime}{}_{t+1} = G'(x_{1:t+1})||\mathscr{E}^e(x^e_{1:t+1})$
    Generate random variable $z \in (0,1)$ uniformly
    if $z \leq 0.5$ **then**
        $a^* = \text{argmax}_a\, Q(s^e{}_{t+1}, a)$
        Calculate $\mathscr{L}_q$ and $\mathscr{L}_{aq}$ by $Q(s^e{}_t, a_t^{(e)})$, $Q'(s^{e\prime}{}_{t+1}, a^*)$
        Calculate $\mathscr{L}_h$ and $\mathscr{L}_{ah}$ by G
    else
        $a^* = \text{argmax}_a\, Q'(s^{e\prime}{}_{t+1}, a)$
        Calculate $\mathscr{L}_q$ and $\mathscr{L}_{aq}$ by $Q'(s^{e\prime}{}_t, a_t^{(e)})$, $Q(s^e{}_{t+1}, a^*)$
        Calculate $\mathscr{L}_h$ and $\mathscr{L}_{ah}$ by G'
    end if
    Perform updates by $\nabla_\Theta \mathscr{L}$
until converge
Return all parameters in Θ

item in the list). Then we select the item with the highest label score as the predicted positive action. The predicted item will be used to augment both the supervised learning for the sequential model and the Q-learning for RL agent:

$$\mathscr{L}_{ah} = -\sum_{j=1}^{n} y_j^e \log\left(\hat{y}_j\right), \tag{6.13}$$

$$\mathscr{L}_{aq} = (r^e(p_t, a_t^e) + \gamma \max_{a'} Q(\, s_{t+1}^e, a') - Q(\, s_t^e, a_t^e))^2, \tag{6.14}$$

$y_j^e = 1$ if the LE predicts that the user will interact with the j-th item at the next timestamp. Otherwise, $y_j^e = 0$. The max operation of $\mathscr{L}_{aq}$ is performed in $Q(s_{t+1}^e, a')$, since the positive actions will transit the current state to the next state.

•Training. We jointly train the supervised and Q-learning loss on the original offline user data, with the augmented supervised and Q-learning loss on the predicted positive feedback:

$$\mathscr{L} = \mathscr{L}_c + w_{ah}\mathscr{L}_{ah} + w_{aq}\mathscr{L}_{aq}, \tag{6.15}$$

where w_{ah} and w_{aq} are the weights of augmented supervised learning loss and Q-learning loss, respectively.

Table 6.1: Dataset statistics. seq. denotes sequence, inter. denotes interaction.

Dataset	#item	#seq.	#inter.	Item content
LFM	18,927	11,073	146,255	title; album; artist
Industry	5,814	10,935	71,872	title; category; brand; description

•Inference. Our goal is to distill knowledge from LLMs into the RL-based RS. Previous methods train LLMs to act as recommenders, which are difficult to deploy in real world settings due to low inference speed. During the inference stage, we only use the sequential model with the supervised head to generate the top-k recommendations without compromising efficiency.

6.3.5 Discussion

We compare LEA with two SOTA RL frameworks: SNQN and SA2C [58]. The difference between LEA and the direct application of LE in existing RL frameworks is that it implements an augmentation method through the LE. We illustrate the training procedure of LEA utilizing LE as both state and reward models in Algorithm 4, where double Q-learning [125] is adopted to alternatively train two copies of trainable Q-networks. The function of LE as a state model or a reward model will be discussed in the experimental section by directly replacing the corresponding part of existing RL methods.

6.4 Experimental Setup

6.4.1 Research Questions

We detail the experimental setup for validation of our augmentation method via the LLM-based environment (LEA). We aim to answer the following research questions:
RQ1: How does LEA, integrated with feedback from LE, perform compared with the existing RL-based recommendation frameworks?
RQ2: Does action augmentation affect the performance of LEA?
RQ3: How does the user feedback from LE, i.e., rewards and state, affect the RL-based recommenders?
RQ4: Do various fine-tuning strategies for LE affect its performance, i.e., item tokenization, data scale, state model loss, the use of LLMs?

6.4.2 Datasets

We perform experiments on two real-world datasets (Table 6.1): **LFM** [93] and **Industry** [3]. The LFM dataset was collected from the music streaming platform Last.fm[2] and contains more than one billion listening events by 120k users. We sample a subset of 3-day listening events on tracks as experimental data. The textual description for each track (item) includes its title, album and the artist. We filter out sequences with fewer than three interactions and tracks listened to less than three times, obtaining a dataset comprising 18,927 items and 11,073 user sequences, with a total of 146,255 interactions. The second Industry dataset is from the publicly available Amazon review dataset[3] of the Industrial and Scientific category. The textual description for each product (item) includes the title, category, and product description. Similarly, we filter out sequences with fewer than three interactions and products reviewed less than three times, yielding a dataset containing 5,814 items and 10,93 sequences, with 71,872 interactions.

6.4.3 Baselines

We compare LEA with two state-of-the-art RL frameworks: **SNQN** and **SA2C** [57, 58] which are introduced in section 6.3.3, under two backbone sequential models: **GRU4Rec** [26], the first sequential recommendation RS based on RNN, and **SASRec** [33], a renowned sequential recommendation model based on self-attention. **Normal** denotes the original sequential models with normal supervised loss. By applying LEA method, we obtain the following strategies: (1) **LEAR** denotes training RL framework with the rewards from LE. (2) **LEAS** denotes that the state for the RL component is derived from LE. (3) **LEASR** denotes that both state and rewards are from the LE. In SA2C, the advantage calculation allows the training of the RL policy to affect

[2]http://www.last.fm/api
[3]https://nijianmo.github.io/amazon

the updates of the sequential model when only s'^{e}_{t} is used, therefore, we introduce an additional method (4) **LEAS′R** for SA2C that the state for RL agent is solely from LE, i.e., $s^{e}_{t} = s'^{e}_{t}$ in Eq. 6.3. In ablation studies, we replace each feedback in the compared RL framework with the corresponding one from LE to examine the LE environment: (1) **LER** and (2) **LES** denote the rewards and states in the baseline RL methods are generated by LE, respectively. Moreover, (3) **LEA** denotes only the augmented training strategy applied to the baseline models.

6.4.4 Metrics

We apply two commonly used metrics: Hit Ratio (HR@k) [58] and Normalized Discounted Cumulative Gain (NDCG@k) [96] to measure the relevance of recommended items for the evaluated sessions, where $k \in \{5, 10, 20\}$. HR@k evaluates whether the ground-truth item is in the top-k positions of the recommendation list. NDCG@k, $k \in \{5, 10, 20\}$ measures the rank of the ground-truth item in the top-k recommendation list. We report the average results over all interactions of the test sequences.

6.4.5 Implementation Details

For *training the LE*, we perform 300 iterations with a learning rate of $5e^{-3}$ for optimization-based item tokenization. The sequence length for the fine-tuning of LE for state generation is set to 10. The epochs for all datasets is 10 with a batch size of 20. We employ the Adam optimizer for fine-tuning. The batch size and epochs is 10 for all datasets. For efficient training, we only use a 10% subset of the original dataset to obtain the LE. For all *compared models and our LEA methods*, the length of sequences is set to 10, and a padding token is added to shorter sequences for all datasets. We train all models and variants with the Adam optimizer [52, 57]. The mini-batch size is 100 for LFM and Industry. The learning rates are $1e^{-3}$ for LFM, and $2e^{-3}$ for Industry. We evaluate the validation set every 1,000 and 500 steps of updates on LFM and Industry, respectively. All experiments are performed on one H100 GPU. To ensure a fair comparison, the item embedding size and hidden size are set to 64 for all models. Since there is only one kind of behaviour, we set the reward to 1 for all interactions in the original SNQN and SA2C [58].

6.5 Experimental Results

6.5.1 Performance Comparison (RQ1)

The performance comparison of the two public datasets is shown in Table 6.2. We observe the following: (1) In contrast to standard sequential models, our RL-based methods consistently outperform across all datasets. This demonstrates that the integration of augmented RL and

Table 6.2: Performance on LFM and Industry datasets. NG is short for NDCG. Boldface denotes the highest score and the second-best scores are marked with __. $*$ denotes the significance p-value < 0.01 compared with second best baseline.

Model	Method	LFM						Industry					
		HR@5	NG@5	HR@10	NG@10	HR@20	NG@20	HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
SASRec	Normal	0.2798	0.2324	0.3115	0.2646	0.3271	0.2685	0.0788	0.0684	0.1171	0.0909	0.1374	0.0961
	SNQN	0.3001	0.2602	0.3337	0.2828	0.3459	0.2855	0.0926	0.0738	0.1202	0.0913	0.1419	0.0968
	LEAR	0.3286	0.2879	0.3504	0.2951	0.3721	0.3005	0.1027	0.0864	0.1232	0.0929	0.1514	0.0983
	LEAS	0.3298	0.2905	0.3529	0.2981	**0.3773***	0.3039	0.1047	0.0856	**0.1281***	0.0931	**0.1581***	**0.1008***
	LEASR	**0.3323***	**0.2914***	**0.3533***	**0.2982***	0.3772	**0.3042***	**0.1059***	**0.0866***	0.1245	**0.0933***	0.1538	0.1001
	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885	0.0931	0.0735	0.1205	0.0912	0.1416	0.0966
	LEAR	0.3269	0.2573	0.3429	0.2942	0.3571	0.2977	0.1037	0.0861	0.1231	0.0922	0.1525	0.0996
	LEAS	0.3302	0.2907	0.3533	0.2982	0.3773	0.3043	**0.1055***	0.0868	**0.1278***	0.0938	0.1528	0.1002
	LEAS′R	0.3298	0.2853	0.3511	0.2931	0.3706	0.2981	0.1036	**0.0874***	0.1264	**0.0948***	0.1521	**0.1012***
	LEASR	**0.3325***	**0.2926***	**0.3572***	**0.3011***	**0.3801***	**0.3072***	0.1042	0.0862	0.1261	0.0932	**0.1533***	0.1001
GRU4Rec	Normal	0.2757	0.2536	0.2934	0.2594	0.3113	0.2641	0.0901	0.0751	0.1128	0.0824	0.1399	0.0893
	SNQN	0.2781	0.2526	0.2931	0.2575	0.3103	0.2619	0.0891	0.0727	0.1083	0.0896	0.1392	0.0931
	LEAR	0.3211	0.2878	0.3407	0.2941	0.3574	0.2983	0.0972	0.0887	0.1205	0.0907	0.1466	0.0975
	LEAS	0.2881	0.2596	0.3134	0.2679	0.3359	0.2735	0.0976	0.0826	0.1232	0.0909	0.1525	0.0982
	LEASR	**0.3234***	**0.2909***	**0.3464***	**0.2983***	**0.3667***	**0.3034***	**0.0978***	**0.0892***	**0.1238***	**0.0915***	**0.1557***	**0.0998***
	SA2C	0.2896	0.2743	0.3173	0.2797	0.3337	0.2838	0.0865	0.0705	0.1145	0.0842	0.1398	0.0913
	LEAR	0.3142	0.2782	0.3333	0.2844	0.3518	0.2891	0.0954	0.0816	0.1191	0.0883	0.1421	0.0948
	LEAS	0.3139	0.2846	0.3364	0.2921	0.3588	0.2977	0.0986	0.0814	0.1224	0.0891	0.1531	0.0968
	LEAS′R	0.3048	0.2795	0.3214	0.2849	0.3393	0.2894	0.0959	0.0818	0.1195	0.0889	0.1416	0.0953
	LEASR	**0.3281***	**0.2926***	**0.3502***	**0.2998***	**0.3711***	**0.3051***	**0.0981***	**0.0828***	**0.1234***	**0.0892***	**0.1559***	**0.0969***

supervised learning, with feedback from LE, contributes to enhancing recommendation performance. (2) LEA further improves the performance over the RL frameworks of SNQN and SA2C. This suggests that the augmentation method improves the learning performance on the supervised component, as well as the RL head. (3) On the LFM dataset, LEASR achieves the highest results on both HR and NDCG metrics, across two frameworks, outperforming other strategies. This indicates that the state and reward obtained from LE, when combined, can significantly improve the performance of RL-based recommenders. On the Industry dataset, the overall relative improvement of the RL method by the LEA approach is not as pronounced as on the LFM dataset. This could be attributed to the LLM potentially incorporating more knowledge about music, an artefact due to its exposure to such content during pre-training, compared to product review data from the Amazon dataset. In conclusion, the proposed LEA methods are effective in improving the recommendation performance of RL frameworks.

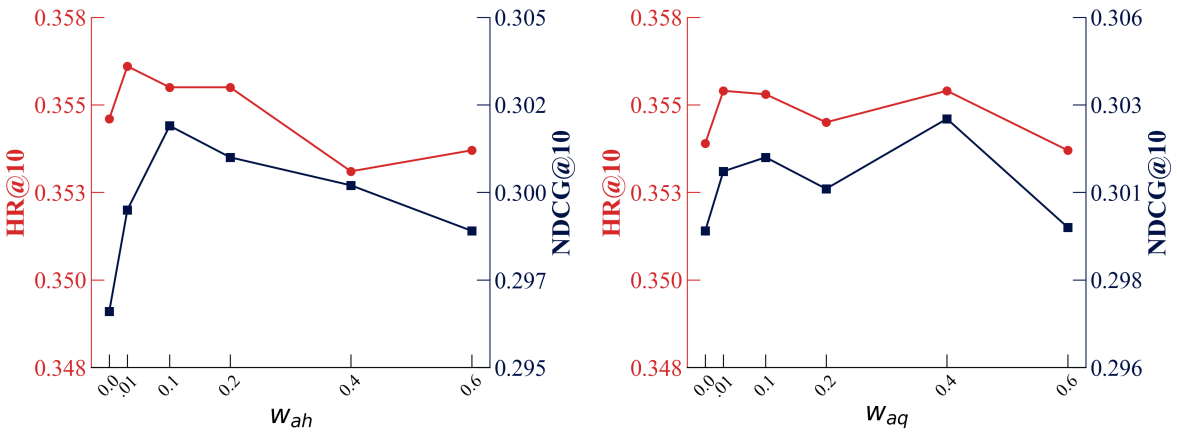
6.5.2 Effect of Action Augmentation (RQ2)

We assess the impact of the LE generated augmented positive actions. We first examine the influence of the feedback on supervised learning (sv) and Q-learning (q) separately. Here, sv+q denotes that both components are augmented. Results are reported on two datasets utilizing the SASRec as the backbone. As Table 6.3 shows, the first pair is the original sequential models (Normal) and its sv augmented version. We can see that when the augmented samples are taken as additional supervised labels, the recommendation performance increases in all cases and datasets. This confirms the LE's capability to generate high-quality positive samples for offline training. The second comparison is considering the SA2C framework and augmentation on sv, q, and sv+q. We observe that the sv augmentation produces the same trends as in the first pair we compare. Furthermore, the improved performance of q over SA2C indicates the effectiveness of augmenting the Q-loss with the positive feedback. Notably, when combining sv and q augmentations, sv+q on the Industry dataset achieves better performance than the individual methods, demonstrating the potential of joint augmentation. On the LFM dataset, sv+q surpasses the baseline and other augmentation methods in most cases, which suggests that the predicted positive samples may require specific reward configurations, distinct from the truth action, to fully realize the benefit.

Furthermore, we investigate the impact of w_{ah} and w_{aq} - the weights of augmentation $\mathscr{L}_{ah}$ and $\mathscr{L}_{aq}$ - in LEASR under the SA2C framework. The results on the LFM dataset are shown in Figures 6.4a and 6.4b, respectively. We can see that the performance initially improves with the increase of w_{ah}. The best performance is achieved when w_{ah} is 0.1. As w_{ah} constantly increases, the performance drops gradually. This suggests that over-weighting augmented actions incrementally diminishes the effect of the true labels, leading to a proportional decrease in performance. We achieve the best performance for $w_{aq} = 0.01$. As w_{aq} increases, we observe an unstable ascending trend in performance. When beyond a certain threshold, the incremental gains

Table 6.3: Effect of action augmentation on supervised learning (sv) and Q-learning (q). NG is short for NDCG. Boldface denotes the highest scores.

Model		LFM						Industry					
		HR@5	NG@5	HR@10	NG@10	HR@20	NG@20	HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
SASRec	Normal	0.2798	0.2324	0.3115	0.2646	0.3271	0.2685	0.0788	0.0684	0.1171	0.0909	0.1374	0.0961
	sv	**0.2899**	**0.2493**	**0.3263**	**0.2757**	**0.3408**	**0.2793**	**0.0928**	**0.0759**	**0.1202**	**0.0931**	**0.1418**	**0.0985**
	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885	0.0931	0.0735	0.1205	0.0912	0.1416	0.0966
	sv	0.3295	0.2898	0.3541	0.2978	**0.3755**	0.3032	0.1026	0.0847	0.1241	0.0916	0.1554	0.0995
	q	0.3314	0.2897	0.3535	0.2969	0.3735	0.3019	0.1022	0.0836	0.1234	0.0904	0.1535	0.0981
	sv+q	**0.3322**	**0.2915**	**0.3547**	**0.2987**	0.3754	**0.3041**	**0.1032**	**0.0859**	**0.1252**	**0.0929**	**0.1558**	**0.1006**



(a) (b)

Figure 6.4: LEASR with different weights of (a) supervised learning and (b) Q-learning augmentation loss on the LFM dataset.

drop. The optimal value of w_{aq} is 0.01. This shows that while the augmentation of Q-learning through predicted action is relatively stable, beyond a certain point further augmentation does not produce proportional performance benefits.

6.5.3 Effect of LE (RQ3)

Table 6.4: Effect of LE as reward function. NG is short for NDCG. Boldface denotes the highest score.

	SAS	HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
LFM	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885
	LER	**0.3252**	**0.2714**	**0.3401**	**0.2928**	**0.3539**	**0.2961**
	LEA	0.3296	0.2802	0.3405	0.2877	0.3548	0.2912
	LEAR	**0.3298**	**0.2833**	**0.3429**	**0.2942**	**0.3571**	**0.2977**
Industry	SA2C	0.0935	0.0775	0.1145	0.0842	0.1427	0.0913
	LER	**0.0994**	**0.0841**	**0.1219**	**0.0928**	**0.1432**	**0.0982**
	LEA	0.1032	0.0859	**0.1252**	**0.0929**	**0.1558**	**0.1006**
	LEAR	**0.1037**	**0.0861**	0.1231	0.0922	0.1525	0.0996

Effect of LE as reward model

We conduct experiments via the RL-based RS by SA2C framework using the SASRec backbone to see the effect of LE as a reward model (RM). The results on two datasets are reported in Table 6.4. LER method replaces the predefined reward of SA2C as the reward return from RM. We observe that LER outperforms SA2C across all cases, which substantiates the effectiveness of

Table 6.5: Effect of LE as state model. NG is short for NDCG. Boldface denotes the highest score.

Model		LFM					
		HR@5	NG@5	HR@10	NG@10	HR@20	NG@20
SAS	SA2C	0.2902	0.2501	0.3372	0.2851	0.3516	0.2885
	LES′	0.3202	0.2804	0.3504	0.2956	0.3711	0.3008
	LES	**0.3311**	**0.2913**	**0.3539**	**0.2988**	**0.3734**	**0.3037**
GRU	SA2C	0.2896	0.2743	0.3173	0.2797	0.3337	0.2838
	LES′	0.2973	0.2597	0.3141	0.2641	0.3345	0.2785
	LES	**0.3128**	**0.2818**	**0.3321**	**0.2891**	**0.3518**	**0.2935**

the RM. We further note that the smaller the k the greater the improvement in the top-k recommendation list, suggesting that a well-designed reward model can effectively prioritize correct actions to higher ranks. LEA incorporates the augmentation technique into the SA2C, whereas LEAR applies both the augmentation and the reward model. It is evident that LEAR generally outperforms LEA. This superior performance can be attributed to the common advantages gained from the augmentation feedback provided by the LE, which potentially obscures the enhancements derived from the RL components. However, the superiority of LEAR still holds the second phenomenon as described above.

Effect of LE as state model

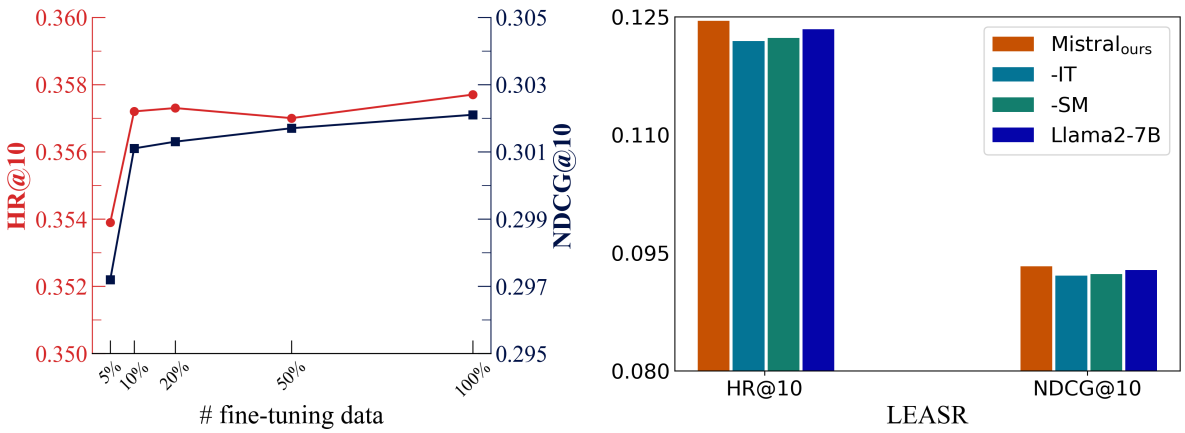
We conduct experiments within the SA2C framework on the LFM dataset between three methods: (1) the baseline SA2C, relying on the hidden state from the sequential model, i.e., h_t in Eq. 6.4, (2) LES′ that uses the state from SM, i.e., s'^{e}_{t} in Eq. 6.3, and (3) LES that leverages both the state from LE and hidden state from the sequential model i.e., s^{e}_{t} in Eq. 6.4. Table 6.5 indicates that LES has the highest performance across all backbones and metrics, while LES′ outperforms SA2C in most cases. This demonstrates that leveraging effectively all state representations can introduce performance gains in RL-based RS.

6.5.4 Effect of learning strategies for LE (RQ4)

Effect of data scale.

Despite optimizing the LE on a subset of the training data, we study the effect of data scaling to provide insights into how to more efficiently obtain LE in the future. We train the LE on a portion of the LFM dataset with varying portions: 5%, 10%, 20%, 50%, and 100%. The results on the LEASR method are shown in Figure 6.5a.

These indicate that as the data portion increases to 10%, we observe a notable enhancement in performance, demonstrating that a bigger dataset enables the LE to capture complex user-item



(a) (b)

Figure 6.5: Effect of scaling the training data for LE. The result of LEASR on the LFM dataset.

interactions and return effective user feedback. The scale we set is 10%. As the scale continuously increases, the effect of LE grows slowly and fluctuates within a range. This demonstrates that our method is capable of efficiently optimizing LLMs with a small data portion, producing a state, reward model, and augmented feedback that are scalable to the entire user data.

Effect of item tokenization

We compare the performance of LE utilizing item tokenization against representing items by their textual content (-IT), where user-item interactions are generated from sequences of item titles. We utilize LEASR to examine the impact on LE. Figure 6.5b shows the results, which suggest that applying item tokenization surpasses the contentenation method. This underscores that the items tokens, represent the semantic information in a concise way, and hence contribute to an effective learning of interaction sequences.

Effect of state loss

To observe the impact of the loss $\mathscr{L}^e_{sm}$ on state generation, we train LE without (-SM) contrastive learning and show the impact by the LES method. Figure 6.5b shows a decrease in performance when $\mathscr{L}^e_{sm}$ is omitted. This implies that the SM loss plays a critical role in refining the state representation, which is integral to the robustness and accuracy of the LE.

Effect of LLMs

We show the effect of different LLMs on LE by comparing Mistral with Llama2-7B, a model with a 7B parameter count similar to Mistral but launched earlier. Figure 6.5b shows that the performance of Mistral is slightly higher than Llama2-7B. We argue that as LLMs evolve, there is the potential for continuous improvement in building LE.

6.6 Conclusion

The use of LLM's as reward, state, and action models is shown to be a promising direction in the context of RL-driven recommender systems. In contrast to current adaptations of LLMs used as recommenders, the design of LLM as Environment (LE) enhances RL-based sequential models without requiring significant increases in inference computations; hence, our model is very easy to deploy in real-world recommendation settings. Future directions of research can include the shaping of different rewards and the inclusion of additional user behaviour data in the LLM to create better state representations. We also intend to investigate alternative fine-tuning methods for both reward state and action generation. Moreover, the use of more powerful LLM's can lead to even better results.

6.6.1 Summary and Link to Next Chapters

This chapter initiates the deployability layer by improving offline RL training signals through an LLM-based environment that provides richer state and reward modeling and supports offline augmentation (LEA). The experiments show consistent gains across RL frameworks and backbones, indicating that better feedback modeling can reduce offline learning brittleness. Chapter 7 builds on this by addressing online deployment explicitly: warming-starting policies and adapting safely under exploration–exploitation trade-offs without relying on LLMs at inference time.

Chapter 7

LLM driven Policy Exploration for Recommender Systems

Note. This chapter is based on our WSDM 2025 paper on iALP/A-iALP, and is revised in this thesis to to explicitly separate offline preference distillation from online adaptation, standardize notation for warm-start and adaptation variants, and strengthen the connection to the thesis deployability narrative.

7.1 Introduction

Modelling user states through interacted items to generate future recommendations has become a common paradigm for RS, particularly in session-based or sequential models [26, 31–33]. Traditionally, these methods implement offline training on historical user log data, using self-supervised techniques. To address long-term user benefits, recent research reformulated this process as a Markov Decision Process (MDP) [57–59], introducing RL to solve the associated decision-making challenges.

However, efforts to develop offline RL policies using historical user data [57–59] are vulnerable to distribution drift when deployed online [95, 149]. This can lead to erroneous actions in out-of-distribution states due to the absence of real-time user feedback from interactions with the environment. Furthermore, offline RL-based systems often suffer from insufficient exploration of the data space, resulting in sub-optimal model performance. These limitations are particularly pronounced in cold-start scenarios and during initial stages when training data is sparse, causing policies to perform poorly and fail to provide satisfactory recommendations. To address these challenges and improve long-term gains, effective methods that can thoroughly explore the data space while adapting to real-time user interactions are needed.

To this end, researchers have explored online RL methods for recommendation that learn from real-time user feedback, as illustrated in Figure 7.1. For example, DQN [150], A2C [151], and DDPG [152] have been used to optimize cumulative rewards, aiming to enhance long-

term user engagement. However, these methods typically require extensive training iterations before achieving optimal performance. Moreover, they overlook a crucial issue that can lead to user churn: if users receive unsatisfactory recommendations during their initial interactions with a system, they are likely to abandon it rather than stay and provide feedback for future improvements. This challenge is one of the main reasons [57] why online RL has not gained widespread adoption in RS.

LLMs with knowledge transfer capabilities have recently gained significant attention in RS [138–140]. In offline settings, LLMs have demonstrated potential as zero-shot [86] or pre-trained [2] RS, achieved through fine-tuning on user data. Moreover, LLMs have been shown to capture user objectives, learnt from expert demonstrations or preferences, through their intuitive use of language [153]. This suggests the potential to leverage LLMs to interpret user preferences and generate appropriate actions across various states. Such feedback can be utilised to pre-train a policy offline, enabling desirable initial behaviours for online recommendations.

To further enhance the performance of an RL-based RS, we introduce an Interaction-Augmented Learned Policy (iALP). We initially train the policy by exclusively interacting with an LLM. We first derive user preferences from an offline LLM, which are subsequently employed for online user interactions. To derive preferences for various items, we prompt the LLM with task-specific instructions to select potential actions. The LLM assigns rewards (1 or 0) for candidate items based on the prompted preferences. This process generates an interactive offline dataset, entirely generated by the LLM, which is used to continuously update the RL policy using an actor-critic framework [151].

For transitioning the policy to online, we propose an adaptive Interaction-Augmented Learned Policy (A-iALP) variant. Specifically, we introduce the direct fine-tuning strategy A-iALP$_{ft}$ to initialise and update the online agent with real feedback from the user environment. A-iALP$_{ft}$ can be deployed as an online agent to effectively address policy drift issues and limited exploration, two challenges with off-line RL. In addition, we propose a second adaptive approach, A-iALP$_{ap}$, which combines a frozen pretrained agent with an online policy to allow further learning. Specifically, the useful behaviours of iALP are retained to generate desirable sequences in the initial training stage, and in the later stage, the actions from the online policy are mainly used to interact with the environment. Furthermore, we validate the performance of A-iALP under different exploration strategies against established baselines. Experimental results on three simulated environments demonstrate the superior performance A-iALP in generating desirable initial and stable episodes. Our contributions can be summarised as follows:

- We propose to distill user preferences on items using LLMs and use these preferences to train an Interaction-Augmented Learned Policy (iALP).
- We introduce two adaptive methods for iALP-to-Online (A-iALP) RL-based recommendation, leading to faster and stable online policy convergence, to enhance long-term rewards (return) and alleviate gain loss for users at early steps.

- We conduct experiments on three recommendation datasets. Experimental results show significant improvements in all metrics.

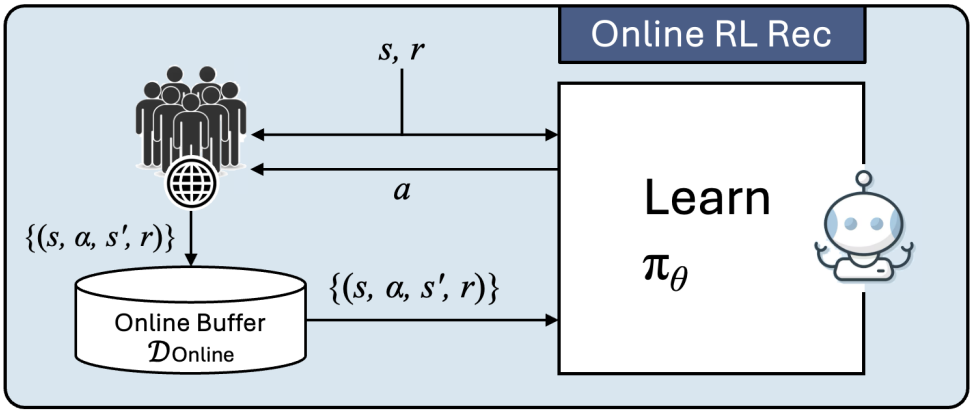


Figure 7.1: Process of online RL for recommendation.

Table 7.1: Prompt template to generate user preference from LLM.

	Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.
Instruction:	You are a user in a {**scenario**} platform now. The {**item**} is in the form of {**attribute1; attribute2; ...** }. Given a user's {**behavior**} history of {**item**}, and candidate {**item**} labelled by lowercase letter to be decided to recommend to the user, identify which {**item**} the user will mostly prefer to at next timestamp. Please judge by the user's preference on {**attribute1; attribute2; ...** }; if you think that none of the candidates will be selected by the user, please answer "None"
Input:	History: {**interacted items**}. Which one the user will mostly like at next timestamp in the following candidates? a. {**item1**} b. {**item2**} ... j. {**item10**} k. None
Response:	By analysing the user's preference, the user will select {**label from a-k**}

7.2 Related Work

Reinforcement Learning has been widely explored for optimizing long-term user engagement in recommender systems. However, a primary challenge divides current approaches. Offline RL methods train policies on historical data but suffer from distribution shift when deployed online, leading to sub-optimal performance [57, 95]. Conversely, online RL methods learn from real-time feedback but face a critical cold-start problem, where an initially untrained policy provides

poor recommendations, risking user churn before the model can converge [57, 154]. This paper directly addresses the challenge of bridging this gap: we aim to achieve the adaptability of online learning without suffering its detrimental initial performance.

Recent advancements have introduced LLMs into the recommendation landscape, leveraging their ability to comprehend user preferences [85]. Some studies have integrated LLMs directly into the online RL loop as agents or planners [92], but this often incurs significant computational latency. In contrast, our work proposes a more practical, two-stage approach. We utilize an LLM exclusively in an offline phase to distill user preferences and generate a rich interactive dataset. This dataset is then used to pre-train a lightweight RL policy (iALP), which serves as a high-quality "warm start" for subsequent online adaptation (A-iALP). This strategy effectively mitigates the cold-start problem of traditional online RL while maintaining the efficiency required for real-time recommendation.

7.3 Preliminary

Online recommender systems aim to retrieve items that can enhance the user experience and provide long-term engagement. RL-based RS for long-term user engagement/satisfaction is based on the principles of a Markov Decision Process (MDP) [57]. The agent (recommender) interacts with the environment (users), taking actions (recommending items) based on the state of the environment, which is represented by the interacted items of the user. The agent is then updated by the feedback (reward) from the user. Continually, the environment generates a new state for the agent as users respond to the actions. Figure 7.1 shows the process, represented by the tuple $(\mathscr{S}, \mathscr{A}, \mathscr{P}, r, \gamma)$:

- State space $\mathscr{S}$ represents the user's state. $s_t \in \mathscr{S}$ is the state at timestamp t, which is usually generated by mapping the sequence of interacted items (user history) into a sequential model/state encoder.
- Action space $\mathscr{A}$. The discrete action set is comprised of the candidate items. Taking action means recommending items. In the online setting, the action $a_t \in \mathscr{A}$ at timestamp t will be the interacted item in the next timestamp to construct the next state.
- Reward function r returns immediate reward r_t given state s_t and the action a_t taken by the agent at timestamp t, which reflects the user's feedback on current recommendation, e.g., likes, dislikes.
- State transition function $\mathscr{P}$ describes the next state s_{t+1} from the environment given the current state s_t and observed action a_t.
- Discount factor γ to the future rewards, where $\gamma \in [0, 1]$.

The RL-based recommendation aims to learn a target policy $\pi_\psi(a|s)$ that maps the state $s \in \mathscr{S}$ to an action distribution $a \in \mathscr{A}$ by maximizing the expected cumulative rewards (return) to realize long-term user engagement, where θ denotes the parameters:

$$\max_{\pi_\theta} \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)], \text{ where } R(\tau) = \sum_{t=0}^{|\tau|} \gamma^t r(s_t, a_t), \tag{7.1}$$

where τ denotes the trajectory of (s_t, a_t, s_{t+1}).

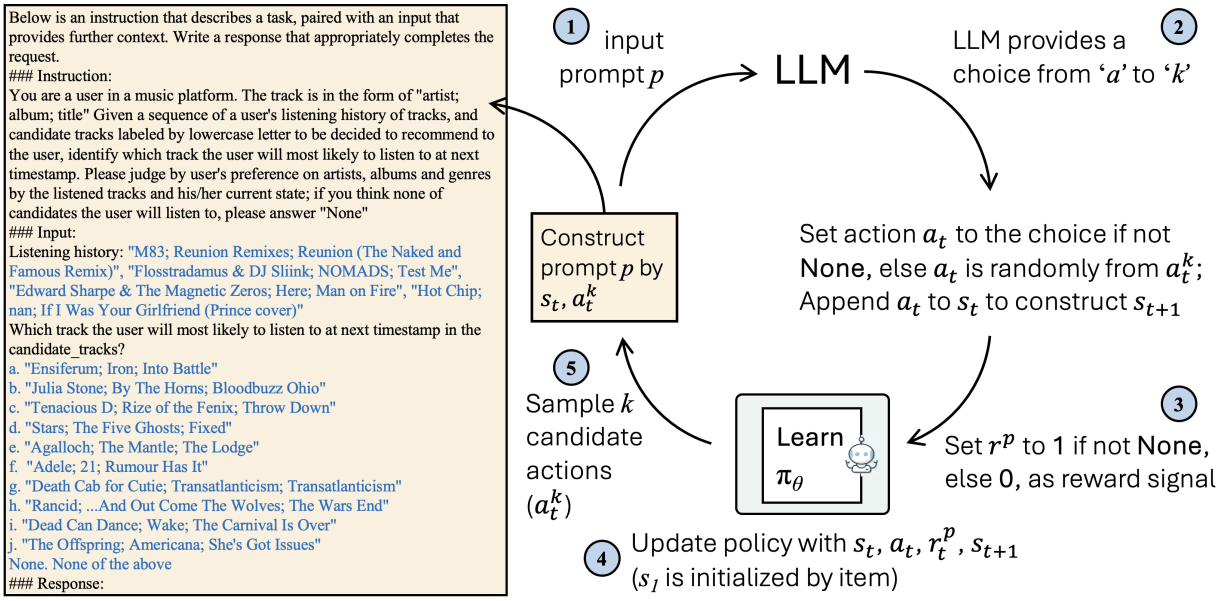
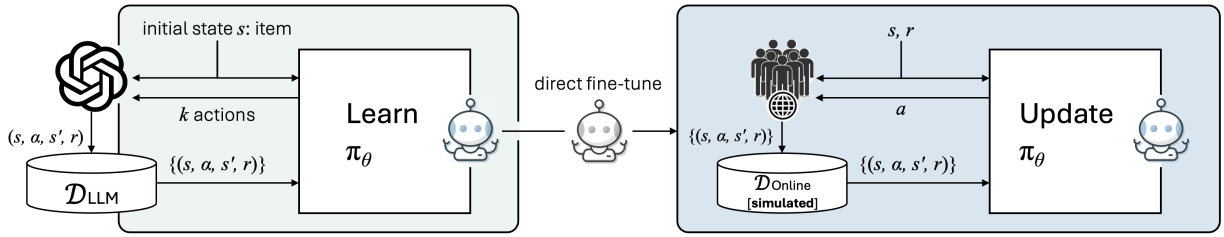
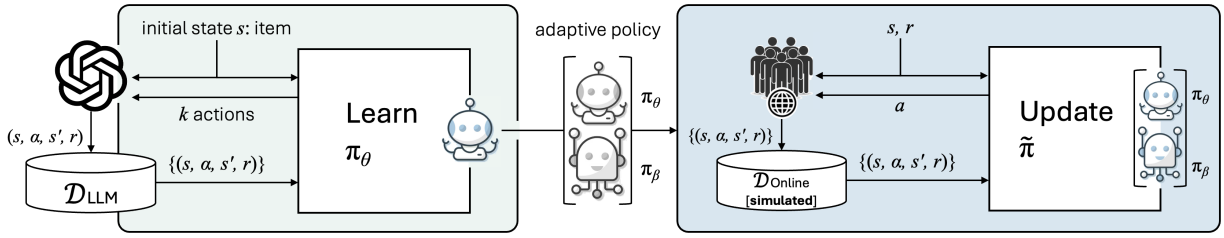


Figure 7.2: Generate user preference, e.g., action and reward, for offline pre-training.



(a) A-iALP$_{ft}$: Direct fine-tuning iALP θ



(b) A-iALP$_{ap}$: Adaptive policy combining θ and learnable policy β

Figure 7.3: Illustration of different adaptation schemes of iALP-to-online. D_{online} consists of generated actions a from policy π_θ and corresponding rewards from the reward model.

7.4 Method

We first introduce the interaction-Augmented Learned Policy (iALP), which is based on item preferences extracted from a Large Language Model. Following that, we outline strategies for adapting iALP (A-iALP) to an online RL-based recommendation system.

7.4.1 Recommendation Policy trained on Preferences from LLM

Preference Distillation

The principle of obtaining user preferences from an LLM involves using it as judge to evaluate which items the user is likely to prefer or dislike, based on their interaction with the LLM. We design a preference prompt $p(s, a^k)$ to obtain this knowledge. Table 7.1 presents the prompt template, which is guided by the description of the recommendation task for a specific scenario (e.g. scenario = track / video). The template uses the textual attributes of items (e.g., attribute1=title, attribute2=brand) to define the recommendation context. As illustrated in Figure 7.2, the prompt contains the state s and a list of k (k=10 in the prompt example) actions a^k, which are fed to an LLM to provide a response, a choice a from 'a' to 'k'. If the output is not None the corresponding action is selected for the user, otherwise the action is randomly selected from a^k. We can see this procedure as an LLM-based sub-policy to select action:

$$a \sim LLM[p(s, a^k)], \tag{7.2}$$

Next, the reward r^p for the action is designed as:

$$r^p = \begin{cases} 1.0 & \text{if the action is selected ,} \\ 0.0 & \text{if the response is None .} \end{cases} \tag{7.3}$$

Policy Pre-training

We utilize these responses as reward signals and training data to pre-train the recommendation policy offline. As the Figure 7.2 shows, once feedback from the LLM on actions is received, we train the interaction-Augmented Learned Policy (iALP) using the actor-critic (A2C) architecture [151]. This is combined with SASRec [33] as a state encoder $G(\cdot)$, a widely adopted approach in RL-based RS. More specifically, given the interacted items $x_{1:t}$, the state at timestamp t is:

$$s_t = G(x_{1:t}), \tag{7.4}$$

where $x_{1:t} = \{x_1, \ldots, x_t\}$ denote the interacted sequence of items, i.e., previous episode. I denote the set of items in the system, $x_i \in I (0 < i \leq t)$ is the index of the interacted item ordered by

timestamp. s_t is then used for the actor and critic networks:

$$a_t \sim \pi_\theta(\cdot|s_t), \tag{7.5}$$

The implementation of π_θ contains two steps. First, *Actor* network maps the state s_t into action distribution, and a list of k actions a_t^k is sampled (random or top-k actions), i.e., $a_t^k \sim Sample(Actor[s_t])$. Second, the preference distillation procedure in section 7.4.1 is executed to select action a_t and generate reward r_t^p, i.e. $a_t \sim LLM[p(s_t, a_t^k)]$. The LLM can be seen as a helper policy to select the best action for the recommendation agent that finishes decision-making. Accordingly, the actor loss is formulated as:

$$\mathscr{L}_A = -\log \pi_\theta(a_t|s_t), \tag{7.6}$$

Regarding the Q-learning network as critic network, we compute the Q-value as follows:

$$Q(s_t, a_t) = Critic[s_t], \tag{7.7}$$

The one-step time difference (TD) Q-learning loss as critic loss is defined as:

$$\mathscr{L}_Q = (r^p(s_t, a_t) + \gamma \max_{a'} Q(\, s_{t+1}, a') - Q(\, s_t, a_t))^2, \tag{7.8}$$

where a_t is appended to $x_{1:t}$ to generate next state s_{t+1} by $G(x_{1:t+1})$. The advantage Q-value is calculated as:

$$A(\, s_t, a_t) = r^p(s_t, a_t) + \gamma \max_{a'} Q(\, s_{t+1}, a') - Q(\, s_t, a_t), \tag{7.9}$$

The actor loss $\mathscr{L}_A$ is finally formulated as $\mathscr{L}_A \leftarrow \mathscr{L}_A \cdot A(\, s_t, a_t)$. In summary, the corresponding loss function for pre-training based on feedback from LLM is formulated as follows:

$$\mathscr{L} = \mathscr{L}_A + \mathscr{L}_Q. \tag{7.10}$$

7.4.2 Simulated Online Learning

We propose two strategies to adapt iALP (A-iALP) to online recommendation for better long-term gains and alleviation of reward loss in the initial steps. Note that while in the previous phase we used the LLM as the reward and action model, here we do not utilise any LLM model.

Online Environment Simulation

To provide immediate feedback and update online recommendation policies, we need to build RL environments. Due to the inaccessibility and high cost of real recommendation platforms, we

use a simulated environment, following the approach of Liu et al. [155] and Shi et al. [92]. This simulated environment returns rewards based on state and action and can be easily constructed using public datasets. We employ two different simulated environments tailored to specific experimental settings, generating states based on user history and providing rewards for the recommended items.

We perform experiments on three user environments from different recommendation scenarios: **LFM** [93], **Industry** [3] and **Coat**[94]. The LFM is a music streaming platform Last.fm[1] with textual content of title, album, and artist for each track (item). The Industry is the Industrial and Scientific category of Amazon review[2] with textual description of title, category and brand for each product (item). We simulate an online environment for each scenario using a publicly available sequential interaction dataset, which is summarized in Table 7.2.

Reward Model: For the Coat dataset, we follow the setting in [92], using Matrix Factorization (DeepFM)[156] to establish a user model, which acts as the reward model to simulate user feedback in training and testing environments. For the LFM and Industry datasets, we construct the reward model based on a sequential recommender method [155], which generates scores/rewards based on a user's state that consists of sequential items and the action. The Coat simulator focuses on user behaviors toward items, with the user as the initial state, while the latter two emphasizes the relationship between the recommended item and the user's interaction history, using an item as the initial state.

iALP-to-Online Fine-tuning

Although online recommendation benefits from real user feedback for performance improvement, it is less sample-efficient and experiences low returns at the beginning of training. In contrast, the pre-training RL policy is sample-efficient since no online interactions are required. Therefore, instead of treating online recommendation as an individual topic, we directly use the policy π_θ learned from LLM to generate the online policy and, in particular, we sample the action from policy π_θ:

$$a_t \sim \pi_\theta(\cdot|s_t) = Actor[s_t], \tag{7.11}$$

The use of the pre-trained policy π_θ essentially means that we deploy the pre-trained actor-critic network with the help of the simulator that provides the corresponding rewards r. The policy is then is fine-tuned based on the simulated real-time user reward r (feedback), we call this method

[1] http://www.last.fm/api
[2] https://nijianmo.github.io/amazon

A-iALP$_{ft}$. The online actor loss and critic loss are as follows:

$$\mathscr{L}_Q^{on} = (r(s_t, a_t) + \gamma \max_{a'} Q(\mathrm{s}_{t+1}, a') - Q(\mathrm{s}_t, a_t))^2, \tag{7.12}$$

$$\mathscr{L}_A^{on} = -\log \pi_\theta(a_t|s_t). \tag{7.13}$$

The key difference of A-iALP$_{ft}$ is that the same algorithm is further improved for online learning using the simulated environment reward r to replace the designed signal r_p in Eq. 7.8. The process is also illustrated in Figure 7.3a, which directly uses online feedback to continue to optimize π_θ.

Algorithm 5 Training iALP Phase

Input: Item set I (action space $\mathscr{A}$), LLM

Initialize replay buffer $\mathscr{D}_{\text{LLM}}$ by initial user sequence $x_1 \in I$
Initialize parameters of the agent θ: actor network $Actor$, critic network $Critic$, and the state encoder G
for each iteration **do**
    Obtain initial user sequence x_1 from $\mathscr{D}_{\text{LLM}}$
    for step $t = 1, \cdots, T$ **do**
        Generate state s_t by $G(x_{1:t})$
        Execute $Actor[s_t]$ to produce action distribution
        Sample a list of k actions a_t^k
        Construct prompt $p(s_t, a_t^k)$ based on template
        Act $LLM(p(s_t, a_t^k))$ to generate a_t(i.e., x_{t+1}), r_t
        Store transition $(x_{1:t}, a_t, r_t, x_{1:t+1})$ in $\mathscr{D}_{\text{LLM}}$
        Sample minibatch b from $\mathscr{D}_{\text{LLM}}$
        With b, calculate $\mathscr{L}_Q$ and $\mathscr{L}_A$
        Update θ
    end for
end for
Return Agent θ: G, $Actor$, and $Critic$

Adaptive Policy

Although direct fine-tuning is straightforward, it presents several challenges. For example, the pre-trained policy might be compromised or even deteriorate during the initial phase of online training, particularly due to distribution shifts between LLM-generated preferences and actual online user behavior [157]. Another concern is that exploration [158] may be overlooked as the iALP often prioritizes non-preferred actions, limiting the system's ability to discover newer and potentially better recommendations.

To solve the above issues, we propose an alternative scheme A-iALP$_{ap}$ (illustrated in Figure 7.3b) that combines the existing algorithm with the online policy to allow further learning. Given a policy π_θ obtained from pre-training phase, instead of directly fine-tuning the param-

eters, we freeze π_θ and propose another learnable policy π_β. Both are added into a policy set $\tilde{\pi} = [\pi_\theta, \pi_\beta]$, which is responsible for further performance improvement during online training. The final policy $\tilde{\pi}$ can be represented as follows:

$$a_t \sim \tilde{\pi}(\cdot|s_t) = (1-\alpha)\pi_\theta(\cdot|s_t) + \alpha\pi_\beta(\cdot|s_t), \tag{7.14}$$

$$\mathscr{L}_A^{on} = -\log \tilde{\pi}(a_t|s_t), \tag{7.15}$$

where α is the weight of taking actions from the learnable policy π_β, $1-\alpha$ indicates the probability of using the pre-trained policy π_θ. As the training steps increase, α increases to 1. In practice, the initial value of α depends on the specific recommendation scenario. It is intuitive to understand that the pre-trained policy π_θ mainly determines actions in the early stages of training, and as the new policy π_β learns more and more preferences, it will take over the decision-making process at the end of the fine-tuning process.

7.4.3 Discussion

Algorithms 5 and 6 illustrate the iALP pre-training and iALP-to-Online procedures, respectively. Note that the critic and state encoder networks of A-iALP$_{ap}$ are updated the same way as A-iALP$_{ft}$. The total loss of online recommendations is $\mathscr{L}^{on} = \mathscr{L}_A^{on} + \mathscr{L}_Q^{on}$. We can retain the useful behaviours of iALP to generate desirable sequences in the initial online training stage. The adaptive scheme enables learning new abilities by interact with the user environment.

Algorithm 6 A-iALP: iALP-to-Online Phase

Input: iALP θ, learnable agent β, scheme, weight α

Initialize empty online replay buffer $\mathscr{D}_{\text{Online}}$
Initialize parameters of the agent β with θ.
for each iteration **do**
    Obtain initial state s_0 from environment
    for step $t = 1, \cdots, T$ **do**
        if scheme is A-iALP$_{ft}$ **then**
            Act $a_t \sim \pi_\theta(\cdot|s_t)$
        else if scheme is A-iALP$_{ap}$ **then**
            Act $a_t \sim (1-\alpha)\pi_\theta(\cdot|s_t) + \alpha\pi_\beta(\cdot|s_t)$
        end if
        Obtain r_t, s_{t+1} from environment
        Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathscr{D}_{\text{Online}}$
        Sample minibatch b from $\mathscr{D}_{\text{Online}}$
        With b, calculate $\mathscr{L}_Q^{on}$ and $\mathscr{L}_A^{on}$
        Update β
    end for
end for

7.5 Experimental Setup

7.5.1 Research Questions

In this section, we describe the experimental setup used to validate our proposed A-iALP approach. We aim to answer the following research questions:
RQ1: How does iALP compare to traditional purely online RL-based recommendation methods in the initial stages of online recommendation?
RQ2: How effective is A-iALP at achieving long-term effectiveness in online recommendations?
RQ3: How does A-iALP compare to directly incorporating preference of LLMs in online training?
RQ4: How does A-iALP perform under various exploration-exploitation strategy?

Table 2 summarises the three data sets (section 4.2.1) used for the experiments. The datasets are partitioned to simulate online training (80% dataset) and testing (20% dataset) environments.

Table 7.2: Dataset statistics.

Dataset	#item	#seq./user	#inter.	Item attribute
LFM	18,297	11,073	146,255	title; album; artist
Industry	5,814	10,935	71,872	title; category; brand
Coat	290	300	11,600	jacket colour; type

7.5.2 Baselines

We compare the proposed A-iALP with several online RL methods for online recommendation.

- DQN [150]: A Q-learning based off-policy method learning from data produced by a policy different from the one currently being optimized.
- PG [159]: An on-policy gradient-based method that utilizes data generated by their current policy.
- A2C [151]: An on-policy gradient-based method with the actor-critic framework, which is the same as our proposed iALP in the pre-training stage.
- iALP: The pre-trained agent to take advantage of the preferences derived from LLM and optimised based on A2C. This model serves as a baseline, providing recommendations without further exploration or updates.

The two adaptation strategies of our A-iALP are described as A-iALP$_{ft}$ and A-iALP$_{ap}$, respectively, for clarity.

7.5.3 Metrics

We apply three metrics: Return, Length and Average Reward[154, 160] to evaluate the RL-based RSs. Return ($R = \sum_t r_t$) measures the cumulative rewards (return) of the recommended episode/sequence, which is applied in the RL methods [160] to evaluate long-term gains. The higher Return denotes better performance. In addition, the length (*Len*) of the sequence and the mean reward ($R_{avg} = R/Len$) of the actions are also used as a reference [92, 154] to analyse user engagement and performance of the immediate recommendation of the policy. To evaluate in early training steps, we incorporate $R@e$, $Len@e$ and $R_{avg}@e$, where $e \in \{0, 1, 2\}$, to show the results in epoch e.

7.5.4 Implementation Details

For the training stage of iALP, to generate the correct label format, we first tune the LLM with the commonly used LORA in the PEFT [141] strategy to learn the template prompt via 1000 randomly sampled interactions. The LLM we use is Mistral 7B [161]. iALP is an A2C framework trained with generated preferences (action, reward) from LLM at every step for 100 epochs. In the online stage, we use the same architecture for A-iALP. The difference in A-iALP$_{ap}$ is that it freezes the iALP policy network and initialises randomly another learnable policy with the same network. For online learning, all RL methods are trained with 50k steps and evaluated in the same test environment, and the most recent 20 episodes are used to generate the learning curve results.

7.6 Experimental Results

In this section, we present the experimental results to answer the research questions described in Section 7.5.1.

7.6.1 Initial Performance Comparison (RQ1)

Table 7.3 shows the performance of directly applying iALP to recommend items without learning from online environment, i.e., e=0. We see that iALP significantly outperforms traditional on-line RL methods that are randomly initialised in the context of cold-start online recommendation. The results demonstrate that iALP, which is trained on preferences distilled from LLMs, can provide a relatively desirable item sequence in terms of cumulative rewards and sequence length of the recommended items. Specifically, iALP manages to alleviate the common issue of poor quality recommendations during the initial phase of deployment, which is a typical problem for traditional RL methods trained from scratch. This advantage not only improves the user experience from the outset, but also accelerates the process of online updates and learning.

By leveraging the pre-trained knowledge, iALP is able to bypass the initial random exploration phase that purely online RL methods undergo, which often results in suboptimal recommendations and user dissatisfaction. Instead, iALP starts with a more informed and refined strategy, leading to higher initial user engagement and satisfaction. We also see in Table 7.4 that, after incorporating a small amount of user feedback (training 1 epoch), A-iALP maintains greater long-term returns compared to general methods. This demonstrates that A-iALP leverages effectively user feedback to enhance performance. This improved starting point facilitates gathering more relevant early feedback, enabling faster and more effective online learning and adaptation.

Table 7.3: The initial performance, i.e., epoch=0, on LFM and Industry environments for online RL-based recommendation. Boldface denotes the highest score.

Method	LFM			Industry		
	$R@0$	$Len@0$	$R_{avg}@0$	$R@0$	$Len@0$	$R_{avg}@0$
DQN	1.53	4.52	0.34	1.12	5.21	0.21
PG	1.73	4.73	0.37	1.06	5.15	0.22
A2C	1.96	5.15	0.38	2.43	5.92	0.41
iALP	**5.11**	**6.21**	**0.82**	**6.35**	**5.94**	**1.06**

Table 7.4: Performance of online RL-based recommendation on LFM and Industry environments while epoch=1. Boldface denotes the highest score.

Method	LFM			Industry		
	$R@1$	$Len@1$	$R_{avg}@1$	$R@1$	$Len@1$	$R_{avg}@1$
DQN	5.04	6.01	0.84	3.84	6.51	0.59
PG	4.85	6.32	0.77	3.32	5.96	0.56
A2C	6.12	6.84	0.89	7.32	6.41	**1.14**
A-iALP$_{ft}$	**8.83**	**8.01**	**1.11**	**9.28**	**8.62**	1.07

7.6.2 Long-term Performance Comparison (RQ2)

Figures 7.6 and Figure 7.7 illustrate the training curves between baseline and A-iALP w.r.t. returns, the sequence length on the LFM and Industry, respectively. The performance of all methods are presented on the LFM, while Industry focusses on the top three methods. We can see that in both LFM and Industry A-iALP$_{ap}$ converges quickly and demonstrates stability, which achieves best return at around 1,000 steps and 20,000 step, respectively. In Industry, A-iALP$_{ft}$ performs second best on Industry before 40,000 environmental steps, while A-iALP$_{ap}$ takes the first position as steps continue. In LFM, A-iALP$_{ft}$ increases stably though with slightly lower performance at the beginning. The performance in the test environment from is illustrated in Table 7.5. We can observe that A-iALP in general outperforms the training from scratch methods

(DQN, PG and A2C) in return and length. In the metric of length, there is no obvious differences on all methods, which might be attributed to the phenomenon that the length of recommended sequence might not reflect the user's long-term satisfaction. Specifically, A-iALP$_{ft}$ shows large improvements over iALP, implying the benefits brought about by additional online training over pure offline training based on LLM interactions. The proposed A-iALP$_{ap}$ outperforms baselines method overall. It shows stable improvements starting from a desirable performance brought about by the pre-training procedure.

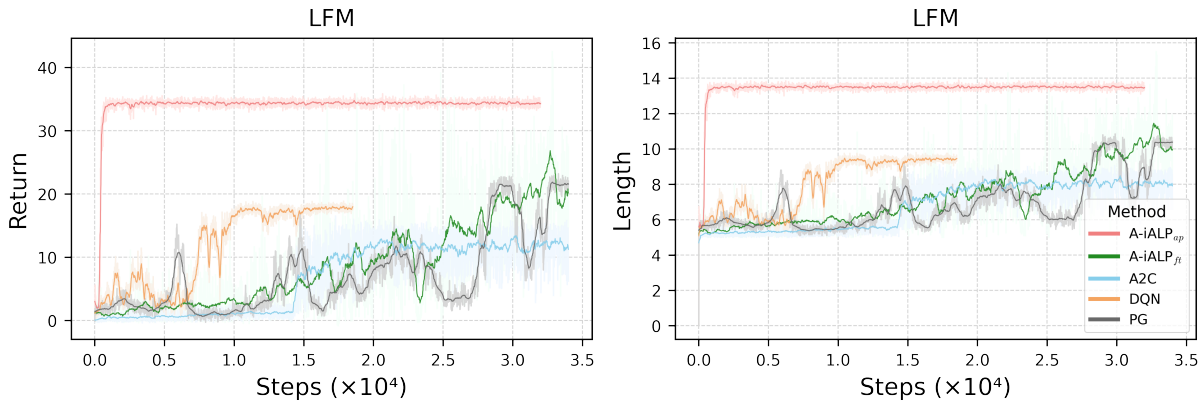


Figure 7.6: Learning curves of return (left) and length (right) between baselines and A-iALP on LFM.

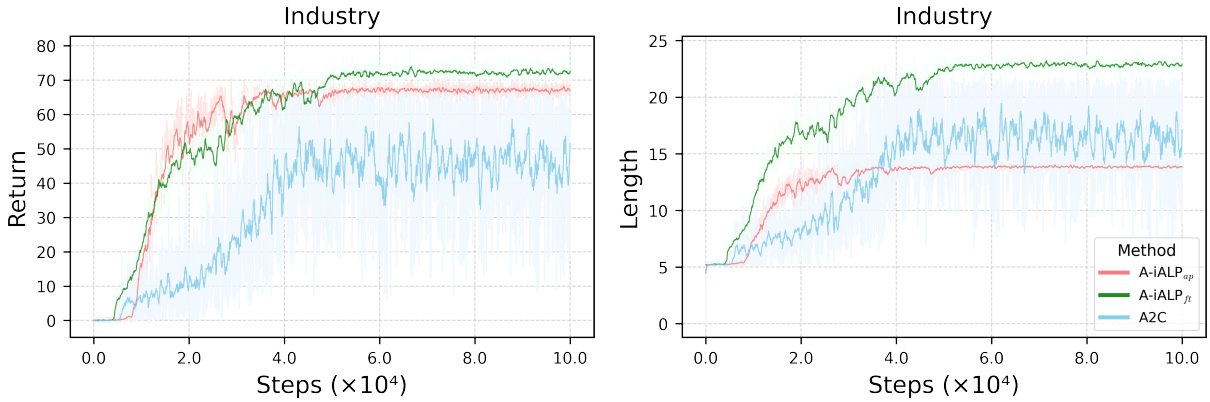


Figure 7.7: Learning curves of return (left) and length (right) between several baselines and A-iALP on Industry.

7.6.3 Utilisation of LLM (RQ3)

Here we compare A-iALP with LLMOnline, which directly uses LLM as a sub-agent in an online environment for taking actions. The goal is to verify the effectiveness of our iALP-to-Online recommendation procedure. LLMOnline is similar to pre-training iALP, but differs in that the agent is directly updated by user feedback in the online RL-based recommendation, rather than by a predefined reward function. The results on LFM (Figure 7.8) and Industry (Figure 7.9)

Table 7.5: Performance comparison of online RL methods for online recommendation. Boldface denotes the highest score, and the second-best results are underlined.

		DQN	PG	A2C	iALP	A-iALP$_{ft}$	A-iALP$_{ap}$
LFM	R	28.8	29.7	28.1	11.2	<u>31.5</u>	**33.1**
	Len	9.53	8.46	9.29	7.21	<u>10.1</u>	**10.2**
	R_{avg}	3.02	<u>3.51</u>	1.55	3.21	3.13	**3.22**
Industry	R	40.3	43.3	46.3	25.3	**52.4**	<u>51.8</u>
	Len	<u>11.3</u>	10.4	10.8	10.7	**11.5**	<u>11.3</u>
	R_{avg}	3.57	4.16	4.28	2.36	<u>4.55</u>	**4.58**
Coat	R	54.3	79.3	81.7	31.2	<u>83.2</u>	**84.4**
	Len	22.1	<u>29.8</u>	**29.9**	12.9	29.6	29.7
	R_{avg}	2.47	2.66	2.73	2.42	<u>2.81</u>	**2.84**

indicate that, in the initial stages, LLMOnline performs similarly to A-iALP. However, as the environment steps increase, A-iALP demonstrates superior performance. This can be attributed to the capacity of iALP to enable an effective exploration during the initial stage, while A-iALP refines its recommendations through continuous interaction with real-time simulated user feedback. In contrast, LLMOnline may be facing limitations as training progresses, with the model's reliance on LLM choices potentially diminishing its ability to adapt to preferences directly from the environment. Another key consideration is the time cost of online adaptation with LLMs. LLMOnline requires significant additional time for processing and generating recommendations through prompting, which can be less efficient in an online setting. In contrast, A-iALP maintains a time cost comparable to traditional RL methods, thanks to its independent pre-training phase.

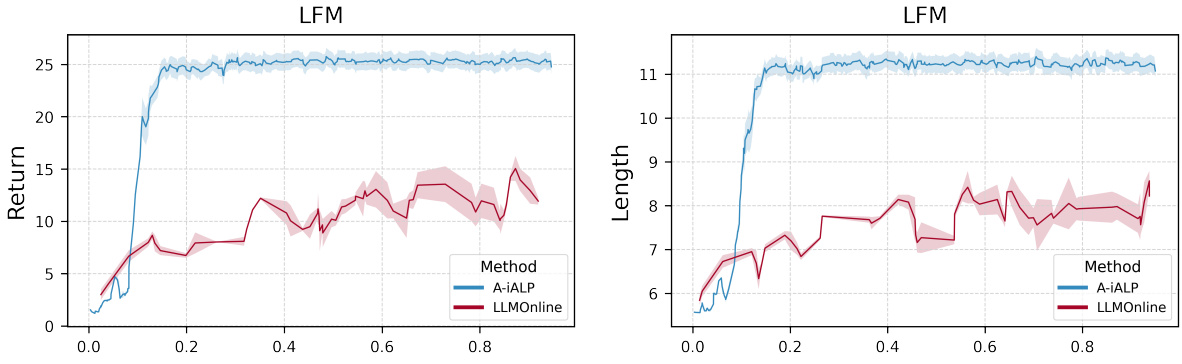


Figure 7.8: Comparison of return (left) and length (right) between using LLM online and A-iALP method on LFM.

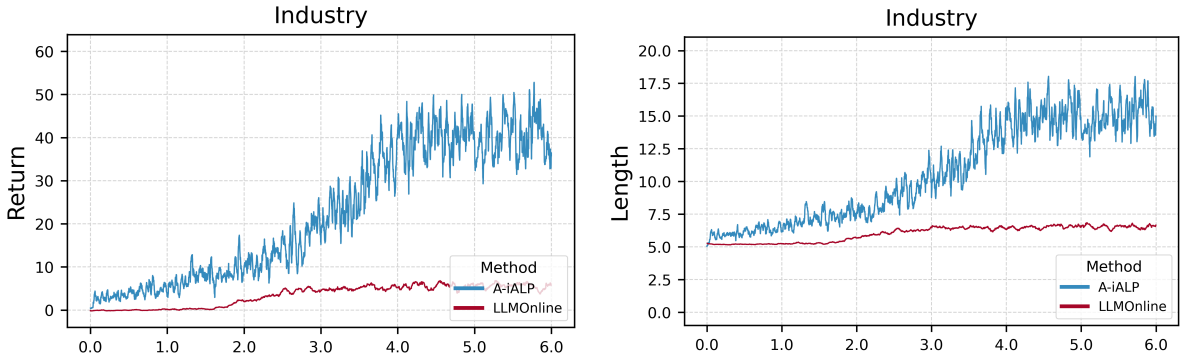
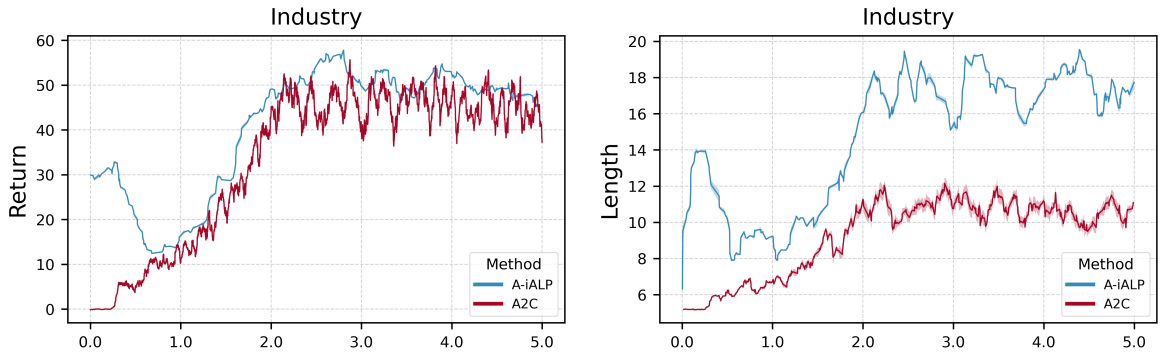
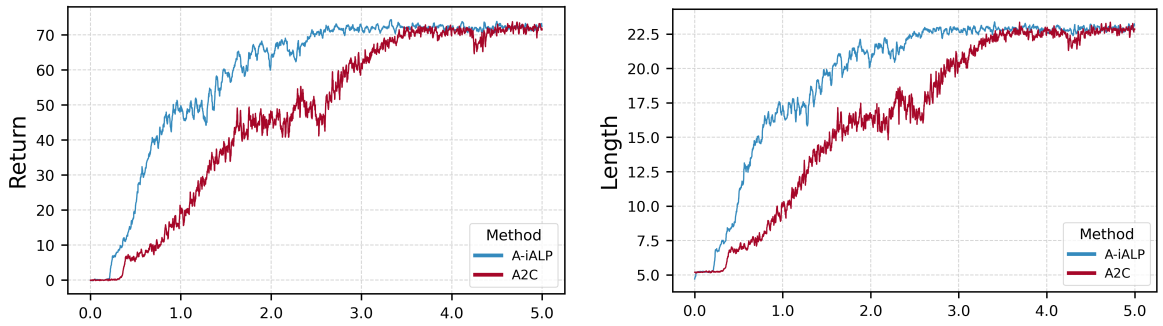


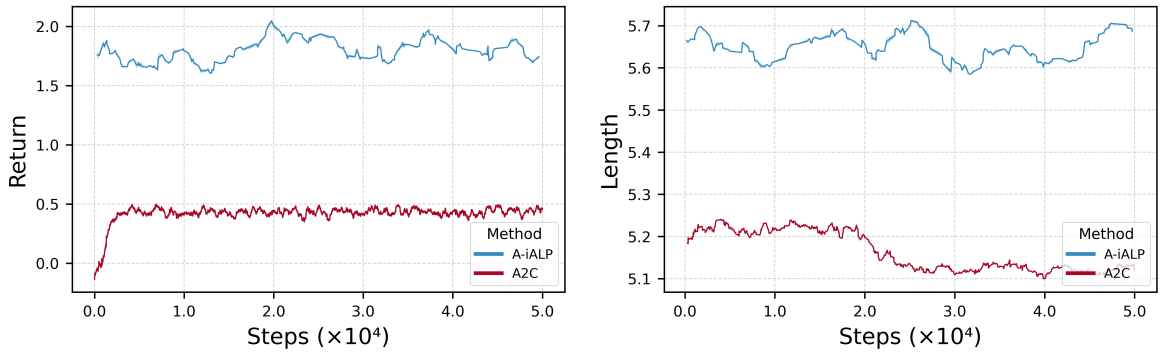
Figure 7.9: Comparison of return (left) and length (right) between using LLM online and A-iALP method on Industry.



(a) ε-greedy

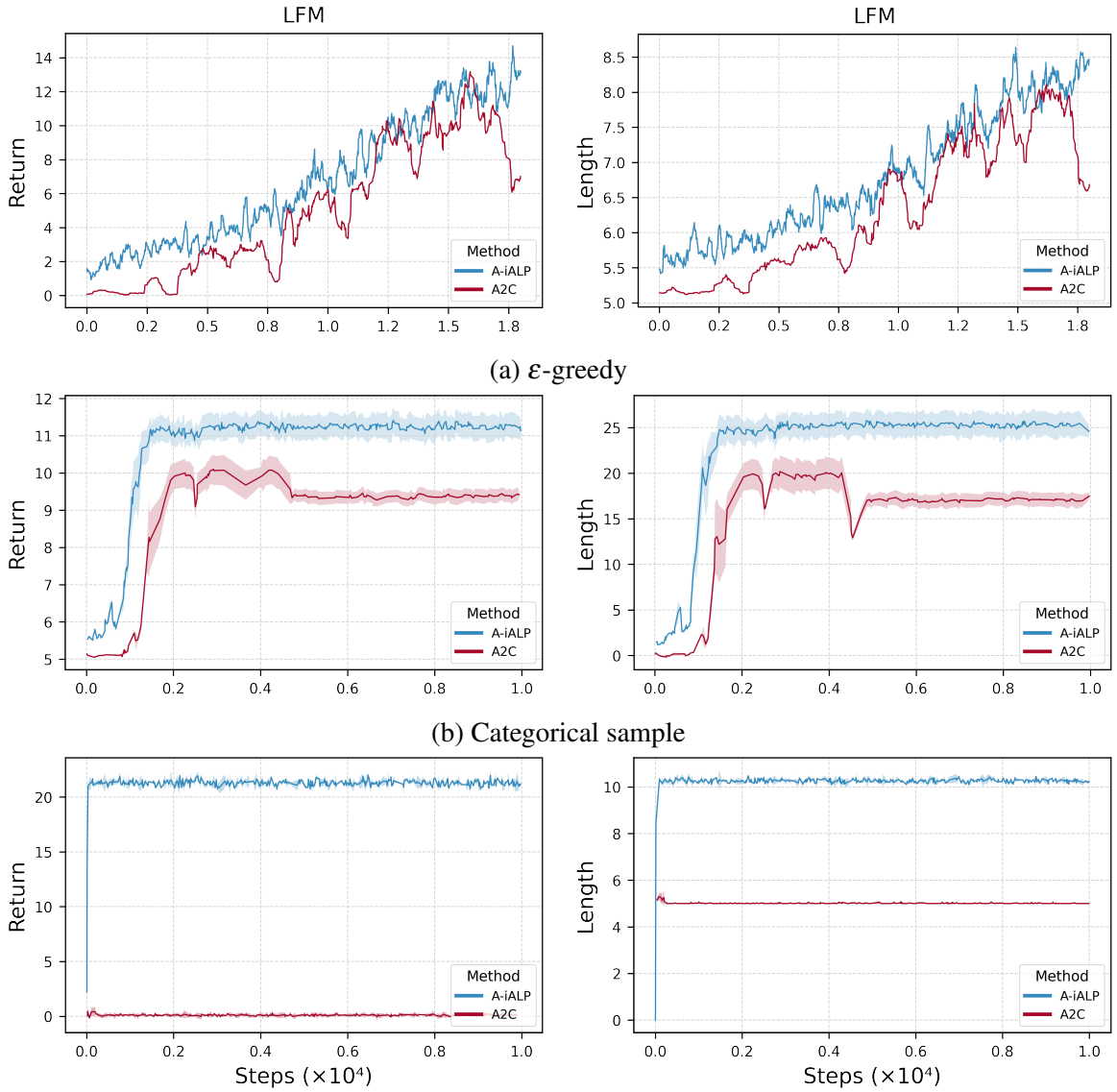


(b) Categorical sample



(c) Greedy

Figure 7.10: Performance comparison between A-iALP and the baseline under various exploration strategies on Industry environment.



(a) ϵ-greedy

(b) Categorical sample

(c) Greedy

Figure 7.11: Performance comparison between A-iALP and the baseline under various exploration strategies on LFM environment.

7.6.4 Effect of Exploration Strategy (RQ4)

Finally, we examine the impact of exploration-exploitation [162] strategies on A-iALP, including: (a) ε-greedy [163], which explores by a random action with probability ε and the greedy action with probability $1-\varepsilon$; (b) categorical sampling [155], which samples actions based on the probability distribution; (c) greedy selection, which is a baseline that exploits the action with the highest estimation without exploration and exploitation balance. Using LFM as a reference (Figure 7.11), A-iALP consistently outperforms the baseline A2C method across all exploration strategies. This demonstrates the effectiveness of integrating preferences distilled from LLMs at the pre-training phase, which provides a robust foundation for more efficient exploration. Specifically, with the ε-greedy method, A-iALP shows slightly better performance than A2C in the early training stages, with A-iALP achieving stable growth later, while A2C experiences significant fluctuations. In categorical sampling, both show stable growth, but A-iALP converges faster. Notably, under the greedy exploitation method, both maintain almost constant performance, but A-iALP's return and sequence length far exceed those of A2C. Similar results are shown in Figure 7.10 for the Industry scenario. These observations demonstrate A-iALP's advantages in convergence speed and stability, leading to greater long-term gains for users.

7.7 Conclusion

We addressed the challenges of offline and online distribution shift and the lack of data exploration in RL based recommender systems, which hinders online deployment of RL based systems. Our investigation focused on utilizing LLMs to pre-train the RL policy, enhancing both initial user gains and long-term effectiveness in recommendation scenarios. By distilling user preferences from LLMs, we created an offline user feedback to train our RL policy, which was then adapted online using our adaptive (A-iALP) methods. Our experiments on three simulated environments demonstrated that A-iALP enhances the quality of initial recommendations while maintaining stability over time. Future research directions include expanding the pre-training dataset with more diverse user interactions and integrating advanced reward mechanisms to further optimize user satisfaction and engagement.

7.7.1 Summary and Link to Next Chapters

This chapter completes the deployability by using LLM-distilled preferences to warm-start RL policies offline and by introducing adaptive strategies for safer and faster online convergence in simulated environments. The results show that warm-start mitigates early-stage performance drops and improves long-horizon return and stability under online learning. Chapter 8 then moves to the generation layer, where learned preferences are made interpretable and controllable via preference generation and decoding for stakeholder-facing insights.

Chapter 8

A Two-Stage Diffusion Framework for Personalized Preference Generation and Decoding

Note. This chapter is based on our research on PDiT-GIM, and is revised in this thesis to to align the generation/decoding task with the thesis workflow, clarify evaluation across decoding quality, ranking utility, and controllability/compliance, and emphasize the optional decoding pathway for interpretability and steering.

8.1 Introduction

Understanding user preferences is crucial for the success of e-commerce platforms, and existing recommender systems excel at suggesting relevant items to users. However, modern generative recommendation pipelines increasingly operate in implicit latent spaces, where learned preference signals can be effective for retrieval and ranking but remain difficult to interpret, audit, or steer toward explicit attribute-level intents. As a result, practitioners often lack an explicit interface to inspect what a model has captured about user preferences and to impose controllable constraints (e.g., material, color, restriction tags) at inference time. This motivates a framework that not only generates preference-conditioned item representations but also provides an optional pathway to decode them into human-readable, attribute-conditioned outputs for inspection and constraint satisfaction.

To address this limitation, we propose an augmented information flow for generative recommendation. As illustrated in Figure 8.1, we retain the standard retrieval/ranking pathway (orange arrows) while introducing an additional optional decoding pathway (green arrows). The model first captures user preference signals from interactions and generates personalized item representations; these latents can be used directly for ranking, or decoded into explicit, attribute-conditioned descriptions when interpretability, debugging, and constraint satisfaction

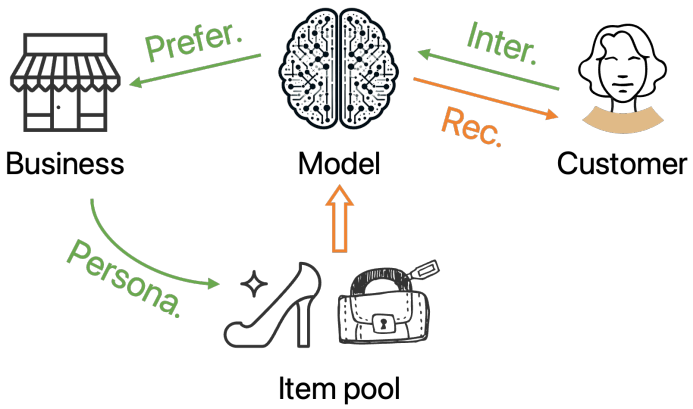


Figure 8.1: An augmented generative recommendation pipeline with an optional decoding pathway. Orange arrows: generating item representations for retrieval/ranking. Green arrows: decoding latent preferences into explicit, attribute-conditioned outputs for inspection and constraint satisfaction.

are required. This design provides a unified interface that preserves recommendation utility while exposing controllable, human-readable preference representations.

To this end, we propose PDiT-GIM, a two-stage *preference generation–decoding* framework that augments the generative recommendation process with an optional decoding pathway for interpretability and controllability. By introducing an independent decoding module, we extend conventional user-centric recommendation pipelines with an optional pathway that translates latent preference representations into explicit, attribute-conditioned outputs, while keeping the original ranking-oriented workflow intact. PDiT-GIM enables explicit inspection of preference factors regarding product features, design elements, and functional attributes, providing an auditable and controllable interface on top of latent generative recommendation. However, the generation and decoding process face challenges that have not been adequately addressed in existing literature.

First, although diffusion-based generative models [164, 165] and generative recommendation methods [166–169] provide a solid foundation, few approaches can generate flexible and interpretable preference representations in the first stage. While recently proposed diffusion-based recommendation methods [168, 170] generate item embeddings conditioned on user behavior to compute item similarity for ranking recommendations, they mainly target ranking performance in latent embedding spaces and seldom support interpretability or controllable decoding. These approaches suffer from a critical limitation: item embeddings generated using learned itemID embeddings as conditions lack semantic labels, making them difficult to decode into explicit textual or visual content. Consequently, the learned representations remain as latent vectors without readable outputs that can be inspected or audited by humans, limiting attribute-level analysis and controllable steering. Moreover, these approaches struggle to effectively incorporate temporal patterns [171] while generating personalized content, often treating user conditions as a single

encoded embedding rather than dynamic behavioral signals.

To address the above issues, we propose PDiT, a hierarchical Transformer architecture that captures both general and temporal user preferences and aims to generate comprehensive personalized item distributions in a text-encoder latent space. PDiT produces semantically grounded latent representations that can serve dual purposes: traditional recommendation ranking and optional preference decoding for interpretability and attribute steering.

Second, while diffusion models have shown success in text-to-image generation [164, 165], the task of decoding latent item representations into explicit, attribute-compliant textual descriptions for interpretability and controllable steering has not been systematically studied. In particular, we require a decoding mechanism that can (i) expose the semantics encoded in the generated latents, and (ii) incorporate explicit attribute constraints at inference time without retraining the diffusion generator.

To solve this problem, we propose Guided Item Modeling (GIM), a simple prompt-generator-based, attribute-guided decoding module that converts the latent features generated in Stage 1 into human-readable content under specified attribute constraints (e.g., textual descriptions, and optionally visual content). Notably, the first-stage generation and second-stage decoding processes are trained independently, leveraging a shared text encoder for item representation. The decoding step is invoked only at inference when explicit interpretation, auditing, or attribute-controlled generation is required.

Our contributions can be summarized as follows:

- We propose PDiT-GIM, a two-stage preference generation–decoding framework that augments generative recommendation with an optional decoding interface for interpretability, auditability, and controllability. The two stages are optimized independently for efficient training and modular deployment.

- We introduce the Personalized Diffusion Transformer (PDiT), which performs latent diffusion in a text-encoder representation space and conditions generation on both general and temporal user behaviors, producing semantically grounded item representations suitable for retrieval and ranking.

- We introduce Guided Item Modeling (GIM), an attribute-guided decoding mechanism that translates generated latents into human-readable outputs under explicit constraints, enabling attribute steering and compliance-aware generation without retraining the diffusion generator.

- We conduct extensive experiments on real-world datasets, demonstrating consistent improvements in decoding quality, preference alignment, and attribute compliance while maintaining competitive recommendation performance.

8.2 Related Work

8.2.1 Diffusion Models for Text Generation

Diffusion models have recently shown remarkable success in the domain of language generation. Initial explorations by [172, 173] demonstrated the feasibility of applying iterative denoising processes to discrete text generation. These approaches typically operate in continuous embedding spaces to enable gradual denoising. Subsequent innovations introduced latent space optimizations, notably through Latent Diffusion Models [164], which improve efficiency by operating in compressed representation spaces. The recent emergence of Diffusion Transformers [174] marked a paradigm shift by replacing convolutional architectures with transformer-based denoising networks, enabling superior modeling of long-range textual dependencies. While these advancements have demonstrated impressive capabilities in controlled text generation [173, 175] and multimodal synthesis [165], current literature predominantly focuses on general-purpose generation tasks. The specific challenges of personalized item generation for broader e-commerce scenarios – particularly the integration of dynamic user preferences with attribute constraints – remain largely unaddressed in existing diffusion frameworks.

8.2.2 Item Generation

Item generation refers to the process of creating items that satisfy specific requirements. The related concept is generative recommendation [166], which typically focuses on generating latent user or item representations for improved ranking performance. Traditional systems predominantly rely on retrieval-based methods, where items are selected from predefined catalogs. These methods, such as VAE-based approaches [176], generative adversarial recommenders [22], diffusion-based models [168, 177] aim to produce better latent embeddings for existing items rather than creating novel discrete item space under requirements. Recent work in controllable text generation [178] has demonstrated the feasibility of generating textual content with specific attributes, which provides technical foundations for our approach. However, these methods lack personalization mechanisms and are not designed for e-commerce user-specific scenarios. Recent developments integrate pretrained language models [61] for item generation, leveraging their deep semantic understanding. However, these methods typically lack mechanisms for precise attribute control and personalization. The critical challenge lies in balancing three competing objectives: generation quality that reflects nuanced user preferences, computational efficiency for real-world deployment, and explicit controllability over item attributes. Beyond accuracy-centric evaluation, recent studies emphasize that recommendation quality is shaped by data characteristics and may interact with fairness objectives across stakeholders [179, 180]. Our work complements this line by enabling explicit preference decoding for interpretability and controllable attribute steering.

8.3 Preliminary

8.3.1 Diffusion Models

Diffusion models [181, 182] are a class of generative models that learn to synthesize data by iteratively inverting a gradual corruption (noising) process. In practice, they are commonly trained via a denoising objective that predicts the injected noise at each diffusion step.

Latent diffusion. To improve computational efficiency, Latent Diffusion Models (LDMs) [164] perform diffusion in a compressed latent space. Given a data sample x_0, an encoder E maps it to a latent code $z_0 = E(x_0)$ (optionally decoded back via a decoder). The diffusion process is then defined over the latent trajectory $\{z_t\}_{t=1}^{T}$ in the encoder's representation space, rather than in the original data space (which corresponds to the pixel space in image diffusion).

Forward process. Let $\{\beta_t\}_{t=1}^{T}$ be a variance schedule, define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The forward noising process is:

$$q(z_t \mid z_0) = \mathcal{N}\left(z_t; \sqrt{\bar{\alpha}_t}\, z_0, (1 - \bar{\alpha}_t)\mathbf{I}\right), \tag{8.1}$$

which admits the reparameterized sampling form:

$$z_t = \sqrt{\bar{\alpha}_t}\, z_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \mathbf{I}), \quad t \in \{1, \ldots, T\}. \tag{8.2}$$

Reverse process. The generative model learns the reverse-time Markov transitions

$$p_\theta(z_{t-1} \mid z_t) = \mathcal{N}(z_{t-1}; \mu_\theta(z_t, t), \Sigma_\theta(z_t, t)), \tag{8.3}$$

where $\mu_\theta(\cdot)$ and $\Sigma_\theta(\cdot)$ are predicted by a neural denoiser. A standard parameterization predicts the noise $\varepsilon_\theta(z_t, t)$ and computes the mean as:

$$\mu_\theta(z_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(z_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}}\, \varepsilon_\theta(z_t, t) \right). \tag{8.4}$$

While early diffusion models often use U-Net denoisers [183], recent Diffusion Transformers (DiT) [174, 184, 185] replace convolutions with transformer blocks, improving the modeling of long-range dependencies; in our setting, the denoiser operates in the LDM latent space.

Training objective. A widely used objective trains ε_θ via noise prediction:

$$\mathcal{L}_{\text{simple}} = \mathbb{E}_{z_0 \sim q(z_0),\, \varepsilon \sim \mathcal{N}(0, \mathbf{I}),\, t} \left[\| \varepsilon - \varepsilon_\theta(z_t, t) \|_2^2 \right], \tag{8.5}$$

where z_t is obtained by Eq. (8.2) (typically with t sampled uniformly from $\{1,\dots,T\}$).

To learn a reverse-process covariance Σ_θ (e.g., learned variance) [186], one can additionally optimize the variational lower bound (up to constants):

$$\mathscr{L}_{\text{vlb}} = -\log p_\theta(z_0 \mid z_1) + \sum_{t=2}^{T} D_{\text{KL}}(q(z_{t-1} \mid z_t, z_0) \,\|\, p_\theta(z_{t-1} \mid z_t)), \tag{8.6}$$

and jointly train the model parameters to obtain improved likelihood and sampling quality.

8.3.2 Problem Definition

Personalized item generation aims to create items based on individual user preferences while satisfying specific requirements. The generated items could serve multiple purposes: informing future product design, facilitating user feedback collection and enhancing recommendation systems. In our context of language modeling and personalization, user preferences are captured through past textual interactions, while requirements are expressed as explicit attribute guidance.

Formally, let $\mathscr{U} = \{v_1, v_2, ..., v_{n-1}\}$ denote a user's historical interactions, and each item v_i is represented as a sequence of words $v_i = [w_1, w_2, ..., w_m]$, where w_j is the j-th word in the textual description of item v_i, m is the token length. These historical interactions are concatenated to form user prompt $\mathscr{P}$. Given an attribute constraint $\mathscr{A}$ (e.g., "environmentally friendly" or "restricted items"), our goal is to generate a new item v_n that satisfies both the user features implied by $\mathscr{P}$ and the attribute constraint $\mathscr{A}$. This generation process can be formally expressed as: $v_n = \mathscr{F}(\mathscr{P}, \mathscr{A})$ where $\mathscr{F}$ represents the model that maps the input prompt and attribute constraint to a new item: $\mathscr{F} : \mathscr{P} \times \mathscr{A}$.

8.4 Methodology

We present the sampling/inference procedure of our framework in Figure 8.2. PDiT-GIM consists of two independently trained components: i) a diffusion network, Personalized Diffusion Transformer (PDiT), that generates items in the latent space based on user preferences (Figure 8.3), and ii) a Guided Item Modeling (GIM) module (Figure 8.4) that deciphers the latent features into discrete text. This design makes the diffusion process more efficient by operating in the compressed latent space. During sampling, we can optionally decode the generated latent conditioning on explicit attributes. PDiT achieves efficiency by performing its diffusion process in the compressed latent space. During inference, GIM can optionally decode these generated latent representations guided by explicit attributes.

Figure 8.3: Overview of the Personalized Diffusion Transformer (PDiT) architecture.

8.4.1 PDiT: Personalized Latent Representation Generation

We propose Personalized Diffusion Transformer (PDiT) for generating items grounded on personalization, a hierarchical framework that comprises two distinct Transformer blocks (Figure 8.3): General DiT (PDiT_{gen}) and Temporal DiT (PDiT_{tem}). The former captures general user preferences described in one sentence, and the latter captures dynamic preferences across temporal interactions. Each denoising step in PDiT consists of General Update on general user preference P via PDiT_{gen} and Temporal Update on temporal interactions U via PDiT_{tem}.

General DiT Suppose a user has bought products $[v_1, ..., v_{n-1}]$, e.g., v_1 is "*Oral-B Electric Toothbrush*",..., v_{n-1} is "*Colgate Optic White Toothpaste*", the general preference prompt P is "*A user has bought* $v_1, ..., v_{n-1}$", the next item he preferred is v_n. We choose Sentence-T5-xl [89] as text encoder E, which is widely used to understand and encode the input text prompts in generation tasks, to obtain latent representations of the general preference prompt P and each item v_i in U:

$$\begin{aligned} p &= E(P) \in \mathbb{R}^d, \\ h_i &= E(v_i) \in \mathbb{R}^d, \\ z &= E(v_n) \in \mathbb{R}^d, \end{aligned} \tag{8.7}$$

where d is the dimension, $i \in [1, ..., n-1]$, p is the general latent, h_i is the i_{th} temporal latent.

The noisy item latent is given as z_t according to Eq. 8.2. As shown on the left of Figure 8.3.1, PDiT_{gen} takes z_t and conditioning signals, i.e., general prompt latent p and diffusion timestep

t, as input to generate the item latent z_g at the global level. Specifically, z_t is reshaped into k-dimensional features $z_t \in \mathbb{R}^{k\times d}$ by an embedding layer for multi-head attention and a feed-forward layer. To integrate the condition information in the normalization step of PDiT$_{gen}$, i.e., Adaptive Layer Norm [174], we utilize the timestep embedding similar to baseline DiT [174], and add the mapped general latent p as the overall condition embedding. This embedding is utilized to calculate the scaling and shifting parameters.

$$z_g = \text{PDiT}_{gen}(z_t, p, t) \in \mathbb{R}^{k\times d}. \tag{8.8}$$

Temporal DiT Subsequently, the general latent z_g is taken as input for PDiT$_{tem}$ after being mapped by an embedding layer to generate temporal latent z_{tem}, where the condition signal becomes the temporal latents $[h_i, ..., h_{n-1}] \in \mathbb{R}^{(n-1)\times d}$. Different from the Adaptive Layer Norm used in PDiT$_{gen}$, we modify the transformer structure to combine the fine-grained temporal conditions with the diffusion model using multi-head cross-attention and a feedforward layer.The process of getting z_{tem} can be formally expressed as:

$$z_{tem} = \text{PDiT}_{tem}(z_g, [h_i, ..., h_{n-1}]) \in \mathbb{R}^{k\times d}. \tag{8.9}$$

After the Transformer backbone, we project the k item latent into predicted noise and predicted covariance, with dimensions matching the input latent $z \in \mathbb{R}^d$. Similar to [184], we employ a standard linear decoder and reshape operation to produce the outputs.

Training Objective. We train the PDiT by employing $\mathscr{L}_{PDiT} = \mathscr{L}_{simple} + \mathscr{L}_{vlb}$ in section 8.3.1, where the denoising loss follows standard DDPM formulation [182], the KL term aligns latent distributions with user preferences. $q(z)$ is the prior distribution of item embeddings.

8.4.2 GIM: Guided Decoding of Generated Latent Representations

GIM serves as an independent decoding module that translates the latent features generated by PDiT into textual content. This separation allows us to maintain the efficiency of latent-space diffusion while ensuring the generated items are explicit and meaningful. Moreover, the decoding can be guided by some explicit attributes to generate satisfied items. We employ prompt-tuning to train an Item Prompt Generator based on a pre-trained LLM, which translates PDiT's latent features into soft prompts. These prompts then guide the LLM to produce human-readable item descriptions that can be precisely controlled by specific attributes.

Item Prompt Generator Let $\mathscr{I} = \{i_1, i_2, ..., i_m\}$ denote all items in the dataset, and each item is represented as a sequence of words, e.g., "*Colgate Optic White Toothpaste*". Specifically, we use the same text encoder E to obtain latent item representations $x_j \in \mathbb{R}^d$ which are used for

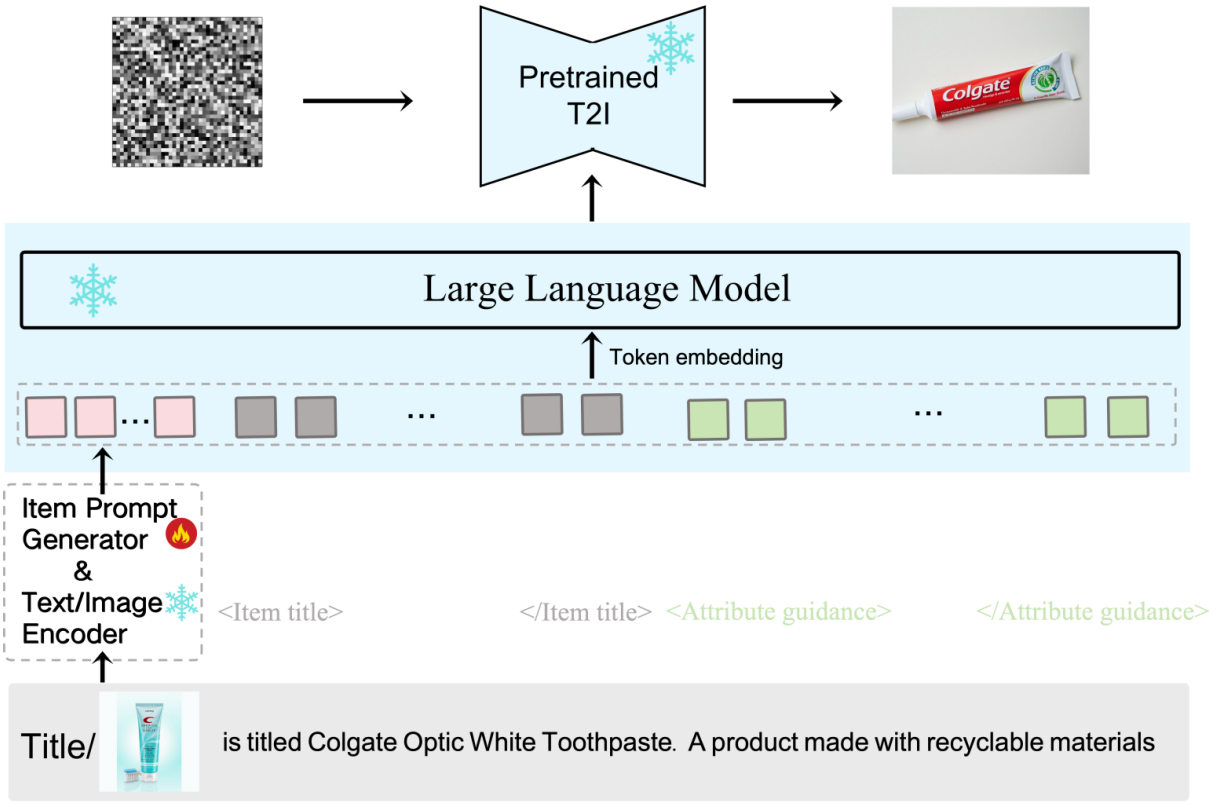


Figure 8.4: Architecture of the Guided Item Modeling (GIM) module.

training the Item Prompt Generator (IPG). We map x_j into $l \times (d_L)$ dimension via an MLP layer and reshape it into $x_j \in \mathbb{R}^{l \times (d_L)}$. A one-layer Transformer follows to yield L soft tokens, which prompt the LLM to reconstruct the item content i_j. The item modeling loss is:

$$\mathscr{L}_{im} = -\log(p_{\Phi,\Theta}(i_j|x_j)), \tag{8.10}$$

where Φ is the parameters updated during the training process. Θ is the frozen parameters of the LLM, we use Llama-3.1 8b [187].

Guided LLM-based Decoding To guide item generation with attribute $\mathscr{A}$ (e.g., "restricted purchasing", see Table 8.5), we construct the input for the LLM by concatenating the L soft prompt tokens, the attribute text (derived from $\mathscr{A}$), and user data P. This combined input conditionally prompts the LLM to reconstruct the item content i_j.

The guided item modeling loss is:

$$\mathscr{L}_{gim} = -\log(p_{\Phi,\Theta}(i_j|P,\mathscr{A},x_j)). \tag{8.11}$$

Weighted Training. We optimize the prompt generator by the weighted loss:

$$\mathscr{L}_g = \mathscr{L}_{im} + \alpha\mathscr{L}_{gim}, \tag{8.12}$$

Algorithm 7 Inference Process of PDiT-GIM

Input: User preference U and general prompt P, attribute set $\mathscr{A}$, diffusion steps T, scheduler $\{\beta_t\}_{t=1}^{T}$, IPG_Φ, LLM_Θ
Output: Generated item description i

Stage 1: PDiT Reverse Process

Precompute $\alpha_t = 1-\beta_t$, $\bar{\alpha}_t = \prod_{s=1}^{t}\alpha_s$, and $\tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$ for $t = 1,\ldots,T$
$z \sim \mathcal{N}(0,\mathbf{I})$ ▷ $z \equiv z_T$
for $t = T,\ldots,1$ **do**
  $(\varepsilon_\theta, v_\theta) \leftarrow \text{PDiT}(z,U,P,t)$ ▷ $v_\theta \in [-1,1]$ for variance
  $\mu_\theta \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(z - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\varepsilon_\theta\right)$
  if $t > 1$ **then**
    $\lambda \leftarrow \frac{v_\theta+1}{2}$ ▷ map to $[0,1]$
    $\log\sigma_t^2 \leftarrow \lambda\log\beta_t + (1-\lambda)\log\tilde{\beta}_t$ ▷ learned_range
    $\sigma_t \leftarrow \exp\left(\frac{1}{2}\log\sigma_t^2\right)$
    $\eta \sim \mathcal{N}(0,\mathbf{I})$
    $z \leftarrow \mu_\theta + \sigma_t\eta$ ▷ $z \equiv z_{t-1}$
  else
    $z_0 \leftarrow \mu_\theta$
  end if
end for

Stage 2: GIM Decoding Process

$s \leftarrow \text{IPG}_\Phi(z_0)$ ▷ Map to L soft tokens
prompt $\leftarrow$ Concatenate$(s,P,\mathscr{A})$ ▷ Add attribute guidance
$i \leftarrow \text{LLM}_\Theta(\text{prompt})$ **return** i

where α is the guidance weight that controls the influence of guidance information.

8.4.3 Inference

During inference, PDiT-GIM combines both components in a sequential manner to generate discrete items as shown in Algorithm 7. Initially, PDiT performs the reverse diffusion process following the DDPM [188] to generate personalized item representations conditioned on user preferences. The reverse process gradually denoises the latent features through T timesteps, where each step predicts the noise component using the personalized diffusion transformer. Specifically, starting from random Gaussian noise $z_T \sim \mathcal{N}(0,\mathbf{I})$, the model iteratively refines the latent representation by computing $p_\theta(z_{t-1}|z_t)$ until reaching z_0.

Subsequently, GIM decodes these latent features into explicit textual content while incorporating desired attribute guidance through cross-attention mechanisms. The final output is an interpretable item description that aligns with both user preferences and attribute constraints. This two-stage design offers several advantages: efficient training through independent optimization, reduced computational complexity by performing diffusion in latent space, and flexible attribute control during the decoding phase.

8.5 Experiments

In this section, we illustrate the datasets, tasks, metrics and baselines used to answer the following research questions.

RQ1: To what extent can the PDiT-GIM decoding pathway translate latent preference representations into interpretable and auditable item descriptions?

RQ2: How well does PDiT-GIM support personalized recommendation compared to existing diffusion-based recommendation approaches in terms of ranking performance?

RQ3: How effectively can the GIM module steer the decoded item generation to satisfy explicit attribute constraints (i.e., controllability and attribute compliance)?

8.5.1 Setup

Table 8.1: Examples of items and interaction data.

Industry Dataset Example	
Inter.1	"Oral-B Electric Toothbrush with Bluetooth Connectivity"
Inter.2	"Philips Sonicare Replacement Brush Heads, 4 Pack"
Inter.3	"Colgate Optic White Toothpaste, Fresh Mint"
Target	"Waterpik Water Flosser Electric Dental Countertop"
LFM Dataset Example	
Inter.1	"Rolling in the Deep by Adele from Album 21"
Inter.2	"Someone Like You by Adele from Album 21"
Inter.3	"Set Fire to the Rain by Adele from Album 21"
Target	"Skyfall by Adele from Album Skyfall"

Datasets. We evaluate PDiT-GIM on two real-world datasets from personalization and recommendation scenarios: LFM [93] and Industry [3](Table 8.2). The LFM dataset is collected from the Last.fm music streaming platform[1] and contains 146,255 listening behaviors from 11,073 users over 18,927 tracks (items). Each user has at least three listening events. The track metadata includes title, album, and artist. The Industry dataset is derived from the Industrial and Scientific category of the Amazon review dataset[2] and comprises 71,872 shopping behaviors from 10,093 users over 5,814 products (items). Each user has at least three reviewing interactions. The product metadata includes title and brand.

For both datasets, each item is represented by its corresponding textual tokens and encoded by the shared text encoder. The general user prompt is templated from a subset of the user's historical preferred items Given a user sequence with t timesteps, we use the first $t-1$ items as temporal preference context and treat the t-th item as the generative target. We split the datasets

[1] `http://www.last.fm/api`

[2] `https://nijianmo.github.io/amazon`

Table 8.2: Dataset statistics. seq. denotes sequence; inter. denotes interaction.

Dataset	#item	#seq.	#inter.	Item content
LFM	18,927	11,073	146,255	title; album; artist
Industry	5,814	10,935	71,872	title; brand

into training, validation, and test sets with an 8:1:1 ratio at the user level, i.e., all interactions of a user belong to exactly one split. Table 8.1 illustrates example user interactions for LFM and Industry, respectively.[3]

Tasks, Metrics, and Baselines. We verify the function of PDiT-GIM as the basic component for preference generation and decoding in three aspects. To assess the quality of decoding the personalized item latent (RQ1), we adopt natural language generation (NLG) metrics to evaluate the generated items tokens with respect to the ground truth item tokens using rule-based matching in terms of precision and recall, including BLEU [23], ROUGE [189], and BERTScore [190]. We compare our method with four baselines: a fine-tuned LLM (the same Llama-3.1 8b as GIM) model with LoRA adapter [141] trained on the same data. PDiT$_{gen}$, a variant that generates items only by the general DiT. PDiT$_{tem}$, a variant using the temporal DiT. Item descriptions of PDiT and variants are decoded by GIM. We also adapt DiffuSeq [172], a diffusion-based conditional discrete text generation model. We assess the utility of predicted user representations (RQ2) in recommendation tasks using Hit Ratio (HR@20) [14] measures the proportion of ground truth items appearing in top-k recommendations, and Normalized Discounted Cumulative Gain (NDCG@20, abbreviated to NG@20) [96] evaluates the ranking quality with position-aware weights. Besides the aforementioned LLM, PDiT$_{gen}$, and PDiT$_{tem}$, we also compare with DreamRec [177], a recent diffusion-based recommendation model using sequential item IDs as the condition. To evaluate the effectiveness of guided generation (RQ3), for the Industry, we guide the generation with product properties, e.g., recyclable materials (shorten as Recy.), restricted purchasing (shorten as Rest.), and measure the attribute alignment rate [191]. For the LFM, we control the music generation with genre constraints (e.g., Pop, Country). For guided item modelling, we compare GIM and Item Modeling (IM), a model trained without guided item mapping. Since ground truth labels for the generated items are unavailable, we employ a large language model (LLM) as an evaluator to assess whether the decoded items comply with the specified attributes. The LLM outputs binary judgments (yes/no) regarding attribute compliance.

[3]The general user prompt for PDiT$_{gen}$ is constructed as: "Ordered products: a product titled [inter.1], a product titled [inter.2], a product titled [inter.3]" or "Listened tracks: a track titled [inter.1], a track titled [inter.2], a track titled [inter.3]".

Implementation Details. All experiments are conducted on a single H100 GPU. We employ Sentence-T5-xl [89] as the text encoder. PDiT is a 2-layer hierarchical Transformer architecture featuring 768-dimensional embeddings and 12 attention heads for both general and temporal preference modeling. The model generates 8 latent item representations and is trained using AdamW [192] optimizer with a learning rate of 1e-4, batch size of 100, and 1000 diffusion steps following a linear noise schedule. GIM is based on Llama-3.1-8b [193] with 8-token soft prompts and trained by AdamW optimizer with a learning rate of 1e-5 and batch size of 10. The guidance weight α ranges from 0 to 20 for attribute control. We employ the standard DDPM sampler [182] with 1000 sampling steps and linear β schedule ranging from 0.0001 to 0.02.

Table 8.3 and 8.4 illustrate the parameter and training settings of PDiT and GIM, respectively. Table 8.5 illustrates the attribute prompt.

Table 8.3: Implementation details of PDiT

Text Encoder	Sentence-T5-xl [89]
Depth	2
Embedding Dimension (d)	768
Item Latent Num (k)	8
PDiT$_{gen}$ Self-Attention Heads	12
PDiT$_{tem}$ Cross-Attention Heads	12
Activation Function	GELU [194]
Normalization Layer	Adaptive Layer Norm [195]
Optimizer	AdamW [192]
Learning Rate	1e-4
Batch Size	100
Warmup Steps	1000
Learning Rate Schedule	Linear Warmup [135]
Weight Decay	0.01
Gradient Clipping	1.0
Diffusion Steps (T)	1000
Noise Schedule	Linear [182]
Most Training Steps	500,000

Setting of LLM-based Evaluation We call DeepSeek-chat/V3 [4] for attribute compliance evaluation. We instruct the model to answer "Yes" or "No" to assess whether each generated item satisfies the specified attribute constraints. For example, when evaluating if a product has purchasing restrictions, the following instruction is used:

Purchase restriction compliance. We ask the LLM to judge whether the generated description implies purchase restrictions (e.g., age verification, permit, prescription, hazardous shipping restrictions, region ban, or quota). The evaluator is prompted as follows:

[4] https://api-docs.deepseek.com/

Table 8.4: Implementation details of GIM

LLM	Llama-3.1-8b [193]
Text Encoder	Sentence-T5-xl [89]
Soft Prompt Length (L)	8
Hidden Dimension (d_L)	4096
Transformer Layer	1
Self-Attention Heads	12
Activation Function	GELU [194]
Normalization Layer	Layer Norm [196]
Max Sequence Length	500
Optimizer	AdamW [192]
Learning Rate	1e-5
Batch Size	10
Warmup Steps	500
Gradient Clipping	1.0
Guidance Weight (α)	[0, 20]
Most Training Epochs	50

Table 8.5: The prompt for attribute guidance. The content in '[]' is added as the items don't have the corresponding attribute.

Attribute Guidance Templates	
Industry-Recyclable	"A product made with [none-]recyclable materials"
Industry-Unrestricted	"A product without[with] purchase restrictions"
LFM-Pop	"A [none-]pop music track"
LFM-Country	"A [none-]country music track"

```
Does this product require purchase restrictions (age
verification / permit / prescription / hazardous shipping
restrictions / region ban / quota)? Output: Restricted:
Yes/No.
Product description: <generated item description>
```

Sampling Setting. Table 8.6 illustrates the detailed setting of sampling. We did not discuss the choice of sampler, but exploration of sampling configurations could potentially yield additional performance improvements for future research.

Table 8.6: Settings during sampling.

DDPM Sampler Configuration	
Sampler Type	DDPM [182]
Sampling Steps	1000
Noise Schedule	Linear β schedule
β_{start}	0.0001
β_{end}	0.02
Mean Type	epsilon
Variance Type	learned_range

8.5.2 Main Results

Table 8.7: Personalized item generation comparisons. Generation of PDiT and variants are decoded by GIM.

Method	LFM			Industry		
	BERTScore↑	BLEU↑	ROUGE↑	BERTScore↑	BLEU↑	ROUGE↑
LLM	0.5563	0.0902	0.2543	0.5832	0.0643	0.2401
DiffuSeq	0.5784	0.1024	0.3081	0.5932	0.0673	0.2543
PDiT$_{gen}$-GIM	0.6182	0.1649	0.4425	0.6179	0.0702	0.2776
PDiT$_{tem}$-GIM	0.6461	0.1905	0.5035	0.6219	0.0817	0.2848
PDiT-GIM	**0.6791**	**0.2289**	**0.5562**	**0.6262**	**0.0902**	**0.3003**

Decoding Performance (RQ1). Table 8.7 evaluates how effectively the PDiT-GIM decoding mechanism transforms latent user preferences into textual content. The results demonstrate that our two-stage approach significantly outperforms direct generation baselines. Compared to LLM approaches (which directly generate text from user prompts) and DiffuSeq (which generates discrete text without latent preference modeling), PDiT-GIM achieves superior text quality that better captures and communicates user preferences to business stakeholders.

Table 8.8: Recommendation performance compared with generative methods.

Model	LFM		Industry	
	HR@20↑	NG@20↑	HR@20↑	NG@20↑
LLM	0.0954	0.0631	0.0056	0.0022
DreamRec	0.1142	0.0372	0.0185	0.0073
PDiT$_{gen}$-GIM	0.1286	0.0374	0.0218	0.0087
PDiT$_{tem}$-GIM	0.1463	0.0486	0.0236	0.0093
PDiT-GIM	**0.1906**	**0.0647**	**0.0315**	**0.0119**

On the Industry dataset, the complete PDiT-GIM framework consistently outperforms baselines across all metrics, achieving substantial improvements of 28.5% on BLEU, 8.2% on ROUGE, and 1.3% on BERTScore compared to PDiT$_{gen}$ alone. The performance comparison between variants reveals that while PDiT$_{tem}$ shows advantages over PDiT$_{gen}$ with 16.4% higher BLEU score, the full PDiT framework with GIM decoding further enhances interpretability by effectively combining both general and temporal preference signals. This demonstrates that GIM successfully bridges the gap between abstract latent user preferences and concrete, readable content that business stakeholders can understand and act upon. The superior decoding quality indicates that our approach effectively addresses the critical challenge of making user preferences explicit and actionable for business decision-making, rather than remaining as latent embeddings of existing recommendation systems.

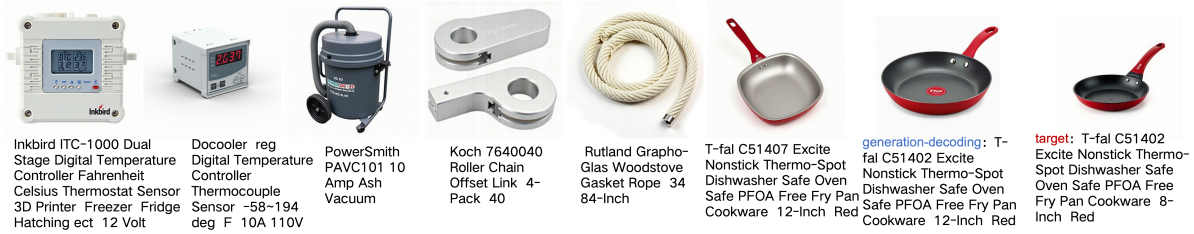


Figure 8.5: User case 1.

More examples of generated items can be seen from Figure 8.7 and 8.8.

Recommendation Performance (RQ2). Table 8.8 demonstrates that PDiT-GIM achieves superior recommendation performance compared to LLM and DreamRec baselines[5], validating our approach's effectiveness in predicting personalized items for generative recommendation. The results show that PDiT-GIM's diffusion-based preference modeling significantly outperforms existing diffusion-based recommendation approaches, with our method generating more accurate personalized item predictions that better align with user interests. Figure 8.5 and 8.6 illustrate two examples of generated personalized item alongside the user behaviours and

[5]For LLM and PDiT-GIM, the ranking scores of the generated item are computed as the cosine similarity between its embedding and all items, encoded by the text encoder.

Figure 8.6: User case 2.

Figure 8.7: User case 3.

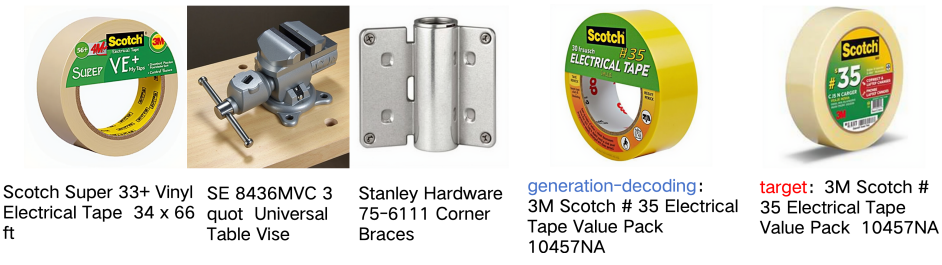
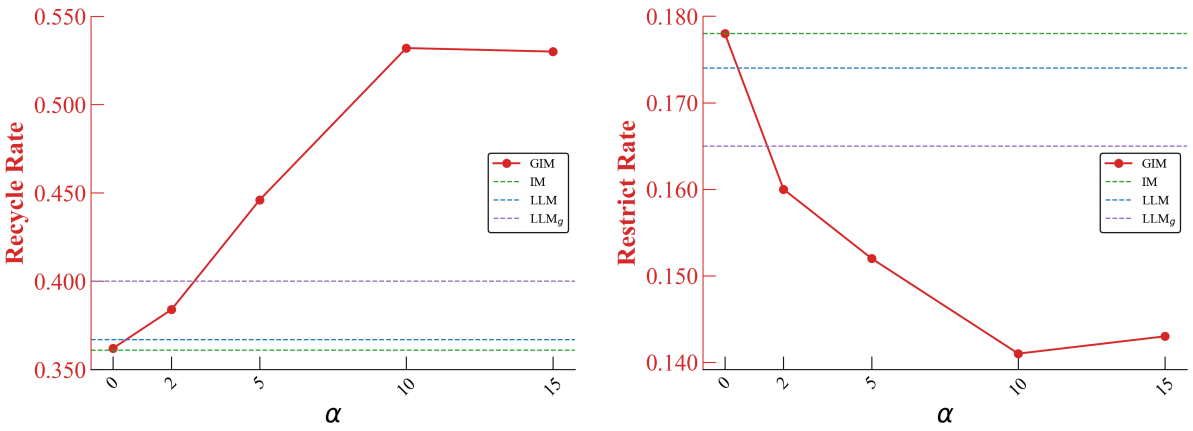


Figure 8.8: User case 4.

the target item, showcasing the quality of our prediction approach. Contrasting with existing diffusion-based methods that rely on learnable item embeddings, PDiT-GIM's semantic richness enables more effective similarity computation and ranking. The substantial performance gains also indicate that our text encoder-based latent space preserves meaningful semantic relationships that facilitate better user-item matching, while traditional ID-based embeddings lose this interpretable structure. The progressive improvements in HR@20 from variants (PDiT$_{gen}$-GIM: 0.0218, PDiT$_{tem}$-GIM: 0.0236) to the full PDiT-GIM framework (0.0315) demonstrate that our hierarchical modeling effectively captures user interests from both global and temporal perspectives, resulting in more accurate personalized item predictions compared to existing diffusion-based recommendation approaches.



(a) Recyclable Material Rate

(b) Restricted Purchase Rate

Figure 8.9: Impact of guidance weight α on Industry dataset.

Table 8.9: Attribute rate of generated items.

Model	LFM		Industry	
	Country↑	Pop↑	Recy.↑	Rest.↓
IM	0.105	0.242	0.362	0.178
GIM	0.127	0.264	0.441	0.153

Guidance Performance (RQ3). Table 8.9 demonstrates how effectively the GIM module guides item generation to meet target attribute requirements through its attribute-guided decoding mechanism. On the Industry dataset, GIM achieves substantial improvements in meeting target specifications: 21% higher compliance for recyclable material attributes and 14% reduction in restricted purchase violations compared to the IM baseline. For the LFM dataset, GIM demonstrates consistent guidance effectiveness with 20% improved compliance in country music generation and 9% better alignment with pop style requirements. These results validate that

GIM successfully transforms latent user preferences into attribute-compliant content that meets specific business requirements.

To analyze the guidance mechanism's controllability, we examine the impact of guidance weight α in Eq. 8.12 across 1,000 Industry samples. Figure 8.9 shows how attribute compliance rates vary as α ranges from 0 to 20, compared with IM and LLM baselines. The results reveal that GIM's guidance mechanism allows fine-grained control over target attribute adherence. LLM_g, which incorporates attribute guidance prompts during fine-tuning, shows higher attribute compliance than standard LLM, demonstrating the effectiveness of guidance-aware training in steering item generation toward desired specifications.

8.5.3 Ablation Study

Table 8.10: Ablation Analysis of PDiT-GIM on Industry dataset.

Model	GIM	HR@20↑	NG@20↑
PDiT_{tem}	w/o	0.0213	0.0083
	w	0.0236	0.0093
PDiT_{gen}	w/o	0.0185	0.0073
	w	0.0218	0.0087
PDiT	w/o	0.0282	0.0109
	w	0.0315	0.0119

Table 8.11: Effect of user data for GIM on Industry dataset.

Model	User data	BLEU↑	ROUGE↑	Recy.↑
IM	w/o	0.0703	0.2824	0.354
	w	0.0724	0.2895	0.362
GIM	w/o	0.0842	0.2901	0.452
	w	0.0902	0.3003	0.441

Ablation Analysis of PDiT-GIM We analyze the contribution of GIM to PDiT's performance in recommendation tasks. Table 8.10 shows the performance of PDiT with (w) and without (w/o) GIM module for decoding. Notably, PDiT w/ GIM outperforms PDiT w/o GIM by 11.7% in HR@20 and 9.2% in NG@20, indicating that decoded item representations provide more effective embeddings than direct latent features for recommendation ranking. Adding GIM consistently improves performance across all variants, with substantial gains of 7.9% and 9.0% in HR@20 for PDiT_{tem} and PDiT_{gen} respectively. The complete PDiT-GIM achieves the best performance, validating GIM's effectiveness in transforming latent preferences into semantic items for recommendation.

Prompt Analysis of Guided Item Modeling We analyze the effectiveness of different prompt configurations for training the GIM module through four variants: IM w/o (basic soft prompts from latent features), IM w (soft prompts with user history guidance), GIM w/o (soft prompts with attribute guidance), and GIM w (full version with soft prompts, user history, and attribute guidance). As shown in Table 8.11, incorporating user history (w methods) consistently improves generation quality with higher BLEU and ROUGE for variants of IM and GIM, indicating that user historical interactions effectively guide the decoding process. The full model achieves the best performance across all metrics, demonstrating that user history and attribute guidance provide complementary benefits for generating preference-aligned item descriptions.

8.5.4 Limitations

While PDiT-GIM demonstrates promising results in personalized item generation and attribute guidance, several limitations remain. First, in our current implementation, the general preference prompt P is solely based on users' historical interactions (e.g., "a user has bought/listened to items v_1, v_2"). Incorporating richer user profiles (e.g., demographics, explicit interests, shopping habits) may further improve preference modeling, but this is constrained by privacy considerations and data availability. Future work could explore ways to safely incorporate richer user information while maintaining generalizability across domains. Second, our attribute evaluation relies on an LLM-based judge, which may introduce assessment bias and may not fully align with domain-specific standards. For example, recyclability assessment can require specialized knowledge that general-purpose LLMs may lack. Future work should incorporate expert evaluation and develop domain-specific verification modules.

8.6 Conclusion

We presented PDiT-GIM, a two-stage preference generation–decoding framework for diffusion-based recommendation that augments latent generative modeling with an optional decoding interface for interpretability, auditability, and controllability. In Stage 1, PDiT performs diffusion in a text-encoder representation space conditioned on both general and temporal user behaviors, producing semantically grounded item latents that remain effective for retrieval and ranking. In Stage 2, GIM maps the generated latents into soft tokens and decodes them into human-readable outputs under explicit attribute guidance, enabling attribute steering and compliance-aware generation without retraining the diffusion generator. Experiments on two real-world datasets demonstrate consistent gains in decoding quality, preference alignment, and attribute compliance while maintaining competitive recommendation performance. Future work includes extending the decoding interface to richer modalities and more fine-grained constraints, improving evaluator reliability for automatic compliance assessment, and exploring more efficient sampling strategies for latency-sensitive deployment.

8.6.1 Summary and Link to Next Chapters

This chapter establishes the generation/value layer by introducing a two-stage preference generation–decoding framework that preserves ranking utility while enabling interpretable, attribute-constrained outputs for inspection, auditing, and stakeholder-facing decision support. The evaluation demonstrates improvements in decoding quality, preference alignment, and controllability/compliance, while maintaining competitive recommendation performance. The final chapter synthesizes the thesis contributions across the four layers (transferability, controllability, deployability, and value generation) and discusses limitations and future directions for multi-objective personalization in sequential recommendation.

Chapter 9

Conclusion and Future Work

9.1 Conclusions

The core contribution of this thesis lies in providing a comprehensive theoretical framework and technical pathway for building recommender systems that are more universal, useful, and value-generative. Through a series of interconnected studies, this work has demonstrated how the synergy between RL and LLMs can address fundamental limitations in the field. The specific conclusions are as follows:

First, concerning the transferability of recommendation models, we identified and addressed the limitations inherent in traditional ID-based collaborative filtering. The TransRec framework proposed in Chapter 3 demonstrates that by pre-training on Mixture-of-Modality (MoM) feedback, it is possible to learn general-purpose, transferable user and item representations. This model, which operates on the intrinsic content of items (e.g., text and images) rather than overlapping IDs, can be effectively transferred to various downstream recommendation tasks. This approach significantly mitigates the cold-start problem and lays a foundational step towards building a General-purpose Recommender System.

Second, in pursuit of enhancing user experience through multi-objective optimization, we moved beyond the conventional focus on accuracy. The MODT4R and HDT frameworks, presented in Chapters 4 and 5, innovatively reframe Multi-Objective Sequential Recommendation (MOSR) as a conditional sequence modeling problem using Decision Transformers. This paradigm enables the system to generate recommendations conditioned on explicit multi-objective returns (e.g., diversity, novelty, and serendipity), thereby significantly enhancing multiple dimensions of the user experience without compromising accuracy. Experiments confirm that this approach is more efficient and stable than traditional multi-agent RL methods.

Third, in leveraging LLMs to overcome RL challenges, we tackled critical bottlenecks in the application of RL to recommender systems. The LLM-based Environment (LE) framework in Chapter 6 successfully employs an LLM as a high-fidelity state and reward model, providing high-quality feedback signals for offline RL agents and improving policy performance through

data augmentation (LEA). Subsequently, the Interaction-Augmented Learned Policy (iALP) and its adaptive online version, A-iALP, introduced in Chapter 7, utilize an LLM to distill user preferences offline and pre-train a high-quality initial policy. This effectively resolves the critical cold-start and inefficient exploration problems of online RL, providing a viable pathway for its safe deployment in real-world recommendation scenarios.

Finally, in constructing a multi-stakeholder generative recommendation ecosystem, we proposed a paradigm shift. The PDiT-GIM framework in Chapter 8 elevates the role of the recommender from a simple retrieval system to a value-generative engine. Through its two-stage "preference generation-decoding" architecture, the framework not only generates novel, personalized items for consumers that satisfy their latent needs but also decodes these latent preferences into explicit, attribute-based insights for business stakeholders such as merchants and designers. This effectively closes the loop between consumer demand and product innovation, creating a symbiotic, multi-stakeholder value ecosystem.

In summary, the contributions of this thesis collectively delineate a developmental roadmap from foundational model building to advanced intelligent applications and, ultimately, to ecosystem innovation. They systematically showcase the immense potential of integrating RL and LLMs to drive recommender systems towards a future that is more universal, human-centric, and value-generative.

9.2 Future Work

Building upon the foundations laid in this thesis, future research will advance along two primary trajectories. The first path involves enhancing the core recommendation models by expanding their scope and efficiency. The work on the general-purpose recommender system, TransRec, can be extended by incorporating richer modalities such as video and audio to create a truly universal foundation model. This ambition, however, necessitates addressing the high computational costs through advanced parameter-efficient fine-tuning and model compression techniques. Concurrently, the reinforcement learning frameworks, including MODT4R and A-iALP, should be transitioned from simulated environments to real-world online deployments. This step will require tackling significant challenges in safety, scalability, and unbiased evaluation, while also exploring more sophisticated, personalized reward mechanisms that dynamically adapt to individual user needs and providing greater explainability for the agent's decisions.

The second trajectory focuses on expanding the novel, multi-stakeholder generative paradigm introduced by PDiT-GIM. This involves deepening the integration of Large Language Models to transform the recommendation ecosystem. The LLM's role can evolve from an offline enhancement tool into a fully interactive and dynamic user simulator, providing a robust testbed for RL policies before they encounter real users. Furthermore, the business insights generated by the framework can be made more accessible and actionable through a conversational interface,

allowing stakeholders to query user preferences in natural language. As these generative capabilities become more powerful, a critical and parallel line of inquiry must address the associated ethical considerations, ensuring that the generated content is fair, unbiased, and free from harm, thereby fostering a responsible and equitable recommendation environment for all parties.

Bibliography

[1] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[2] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315, 2022.

[3] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL `https://aclanthology.org/D19-1018/`.

[4] Wenhui Liao, Qian Zhang, Bo Yuan, Guangquan Zhang, and Jie Lu. Heterogeneous multidomain recommender system through adversarial learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(11):8965–8977, 2022.

[5] Dietmar Jannach, Pearl Pu, Francesco Ricci, and Markus Zanker. Recommender systems: Past, present, future. *Ai Magazine*, 42(3):3–6, 2021.

[6] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4425–4445, 2022.

[7] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM computing surveys (CSUR)*, 51(4):1–36, 2018.

[8] Xiaoyao Zheng, Xingwang Li, Zhenghua Chen, Liping Sun, Qingying Yu, Liangmin Guo, and Yonglong Luo. Enhanced self-attention mechanism for long and short term sequen-

tial recommendation models. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 8(3):2457–2466, 2024.

[9] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. Sequential recommender systems: Challenges, progress and prospects. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 6332–6338. International Joint Conferences on Artificial Intelligence Organization, 2019.

[10] Hongbo Wang, Yizhe Wang, and Yu Liu. A sequential recommendation model for balancing long-and short-term benefits. *International Journal of Computational Intelligence Systems*, 17(1):75, 2024.

[11] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Empowering news recommendation with pre-trained language models. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1652–1656, 2021.

[12] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. Mmgcn: multi-modal graph convolution network for personalized recommendation of micro-video. In *Proceedings of the 27th ACM international conference on multimedia*, pages 1437–1445, 2019.

[13] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.

[14] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017. doi: 10.1145/3038912.3052569.

[15] Zixuan Yi, Xi Wang, Iadh Ounis, and Craig Macdonald. Multi-modal graph contrastive learning for micro-video recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1807–1811, 2022.

[16] Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[17] Jiajin Wu, Bo Yang, Runze Mao, and Qing Li. Popularity-aware sequential recommendation with user desire. *Expert Systems with Applications*, 237:121429, 2024.

[18] Zihao Zhao, Jiawei Chen, Sheng Zhou, Xiangnan He, Xuezhi Cao, Fuzheng Zhang, and Wei Wu. Popularity bias is not always evil: Disentangling benign and harmful bias for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(10):9920–9931, 2022.

[19] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. Towards universal sequence representation learning for recommender systems. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 585–593, 2022.

[20] Kachun Lo and Tsukasa Ishigaki. Ppnw: personalized pairwise novelty loss weighting for novel recommendation. *Knowledge and Information Systems*, 63:1117–1148, 2021.

[21] Pan Li, Maofei Que, Zhichao Jiang, Yao Hu, and Alexander Tuzhilin. Purs: personalized unexpected recommender system for improving user satisfaction. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 279–288, 2020.

[22] Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. Generative adversarial user model for reinforcement learning based recommendation system. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1052–1061. PMLR, 2019. URL `https://proceedings.mlr.press/v97/chen19f.html`.

[23] Kishore Papineni et al. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002. URL `https://aclanthology.org/P02-1040.Pdf`.

[24] Shoujin Wang, Qi Zhang, Liang Hu, Xiuzhen Zhang, Yan Wang, and Charu Aggarwal. Sequential/session-based recommendations: Challenges, approaches, applications and opportunities. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3425–3428, 2022.

[25] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–42, 2020.

[26] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[27] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[28] Xiangyu Zhao, Liang Zhang, Zhuoye Ding, Long Xia, Jiliang Tang, and Dawei Yin. Recommendations with negative feedback via pairwise deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1040–1048, 2018.

[29] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. When large language models meet personalization: Perspectives of challenges and opportunities. *arXiv preprint arXiv:2307.16376*, 2023.

[30] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

[31] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[32] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 582–590, 2019.

[33] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[34] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902, 2020.

[35] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137, 2017.

[36] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. Hierarchical temporal convolutional networks for dynamic recommender systems. In *The world wide web conference*, pages 2236–2246, 2019.

[37] Rong Hu and Pearl Pu. Helping users perceive recommendation diversity. In *DiveRS@ RecSys*, pages 43–50, 2011.

[38] Azin Ashkan, Branislav Kveton, Shlomo Berkovsky, and Zheng Wen. Optimal greedy diversity for recommendation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[39] Lijing Qin and Xiaoyan Zhu. Promoting diversity in recommendation by entropy regularizer. In *Twenty-Third International Joint Conference on Artificial Intelligence*. Citeseer, 2013.

[40] Chaofeng Sha, Xiaowei Wu, and Junyu Niu. A framework for recommending relevant and diverse items. In *IJCAI*, volume 16, pages 3868–3874, 2016.

[41] Peizhe Cheng, Shuaiqiang Wang, Jun Ma, Jiankai Sun, and Hui Xiong. Learning to recommend accurate and diverse items. In *Proceedings of the 26th international conference on World Wide Web*, pages 183–192, 2017.

[42] Ruomu Zou. A deep, forgetful novelty-seeking movie recommender model. *arXiv preprint arXiv:1909.01811*, 2019.

[43] Yuanbo Xu, Yongjian Yang, En Wang, Jiayu Han, Fuzhen Zhuang, Zhiwen Yu, and Hui Xiong. Neural serendipity recommendation: Exploring the balance between accuracy and novelty with sparse explicit feedback. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(4):1–25, 2020.

[44] Mingwei Zhang, Yang Yang, Rizwan Abbas, Ke Deng, Jianxin Li, and Bin Zhang. Snpr: A serendipity-oriented next poi recommendation model. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2568–2577, 2021.

[45] Wanyu Chen, Pengjie Ren, Fei Cai, Fei Sun, and Maarten De Rijke. Multi-interest diversification for end-to-end sequential recommendation. *ACM Transactions on Information Systems (TOIS)*, 40(1):1–30, 2021.

[46] Younghoon Lee. Serendipity adjustable application recommendation via joint disentangled recurrent variational auto-encoder. *Electronic Commerce Research and Applications*, 44:101017, 2020.

[47] Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, Guojun Wang, and Kenli Li. Directional and explainable serendipity recommendation. In *Proceedings of The Web Conference 2020*, pages 122–132, 2020.

[48] Marco Tulio Ribeiro, Anisio Lacerda, Adriano Veloso, and Nivio Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, pages 19–26, 2012.

[49] Xiao Lin, Hongjie Chen, Changhua Pei, Fei Sun, Xuanji Xiao, Hanxiao Sun, Yongfeng Zhang, Wenwu Ou, and Peng Jiang. A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation. In *Proceedings of the 13th ACM Conference on recommender systems*, pages 20–28, 2019.

[50] Nikola Milojkovic, Diego Antognini, Giancarlo Bergamin, Boi Faltings, and Claudiu Musat. Multi-gradient descent for multi-objective recommender systems. *arXiv preprint arXiv:2001.00846*, 2019.

[51] Shuang Geng, Xiaofu He, Gemin Liang, Ben Niu, Sen Liu, and Yuqin He. Accuracy-diversity optimization in personalized recommender system via trajectory reinforcement based bacterial colony optimization. *Information Processing & Management*, 60(2): 103205, 2023.

[52] Dusan Stamenkovic, Alexandros Karatzoglou, Ioannis Arapakis, Xin Xin, and Kleomenis Katevas. Choosing the best of both worlds: Diverse and novel recommendations through multi-objective reinforcement learning. In *Proc. of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 957–965, 2022.

[53] Fei Zhou, Biao Luo, Zhengke Wu, and Tingwen Huang. Smonac: Supervised multiobjective negative actor–critic for sequential recommendation. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[54] Omar Moling, Linas Baltrunas, and Francesco Ricci. Optimal radio channel recommendations with explicit and implicit feedback. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 75–82, 2012.

[55] Pengfei Wang, Yu Fan, Long Xia, Wayne Xin Zhao, ShaoZhang Niu, and Jimmy Huang. Kerl: A knowledge-guided reinforcement learning model for sequential recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 209–218, 2020.

[56] Daocheng Hong, Yang Li, and Qiwen Dong. Nonintrusive-sensing and reinforcement-learning based adaptive personalized music recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1721–1724, 2020.

[57] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Self-supervised reinforcement learning for recommender systems. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 931–940, 2020.

[58] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Supervised advantage actor-critic for recommender systems. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1186–1196, 2022.

[59] Zhaochun Ren, Na Huang, Yidan Wang, Pengjie Ren, Jun Ma, Jiahuan Lei, Xinlei Shi, Hengliang Luo, Joemon Jose, and Xin Xin. Contrastive state augmentations for reinforcement learning-based recommender systems. pages 922–931, 2023.

[60] Xin Xin, Tiago Pimentel, Alexandros Karatzoglou, Pengjie Ren, Konstantina Christakopoulou, and Zhaochun Ren. Rethinking reinforcement learning for recommendation: A prompt perspective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1347–1357, 2022.

[61] Jacob Devlin et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186, 2019. URL `https://aclanthology.org/N19-1423/`.

[62] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.

[63] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[64] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[65] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 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*, pages 1930–1939, 2018.

[66] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM conference on recommender systems*, pages 43–51, 2019.

[67] Hongyan Tang, Junning Liu, Ming Zhao, and Xudong Gong. Progressive layered extraction (ple): a novel multi-task learning (mtl) model for personalized recommendations. In *Fourteenth ACM conference on recommender systems*, pages 269–278, 2020.

[68] Fajie Yuan, Xiangnan He, Alexandros Karatzoglou, and Liguang Zhang. Parameter-efficient transfer from sequential behaviors for user modeling and recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in information retrieval*, pages 1469–1478, 2020.

[69] Fajie Yuan, Guoxiao Zhang, Alexandros Karatzoglou, Joemon Jose, Beibei Kong, and Yudong Li. 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*, pages 696–705, 2021.

[70] Jian Liu, Pengpeng Zhao, Fuzhen Zhuang, Yanchi Liu, Victor S Sheng, Jiajie Xu, Xiaofang Zhou, and Hui Xiong. Exploiting aesthetic preference in deep cross networks for cross-domain recommendation. In *Proceedings of The Web Conference 2020*, pages 2768–2774, 2020.

[71] Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Jun Ma, and Maarten de Rijke. 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*, pages 685–694, 2019.

[72] Chuhan Wu, Fangzhao Wu, Tao Qi, Jianxun Lian, Yongfeng Huang, and Xing Xie. Ptum: pre-training user model from unlabeled user behaviors via self-supervision. *arXiv preprint arXiv:2010.01494*, 2020.

[73] Feng Zhu, Yan Wang, Chaochao Chen, Guanfeng Liu, Mehmet A Orgun, and Jia Wu. A deep framework for cross-domain and cross-system recommendations. *arXiv preprint arXiv:2009.06215*, 2020.

[74] Kyuyong Shin, Hanock Kwak, Kyung-Min Kim, Minkyu Kim, Young-Jin Park, Jisu Jeong, and Seungjae Jung. One4all user representation for recommender systems in e-commerce. *arXiv preprint arXiv:2106.00573*, 2021.

[75] Kyuyong Shin, Hanock Kwak, Kyung-Min Kim, Su Young Kim, and Max Nihlen Ramstrom. Scaling law for recommendation models: Towards general-purpose user representations. *arXiv preprint arXiv:2111.11294*, 2021.

[76] Chuhan Wu, Fangzhao Wu, Yang Yu, Tao Qi, Yongfeng Huang, and Xing Xie. Userbert: contrastive user model pre-training. *arXiv preprint arXiv:2109.01274*, 2021.

[77] Hao Ding, Yifei Ma, Anoop Deoras, Yuyang Wang, and Hao Wang. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318*, 2021.

[78] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[79] Ruining He and Julian McAuley. Vbpr: visual bayesian personalized ranking from implicit feedback. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[80] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

[81] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[82] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[83] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

[84] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, et al. How can recommender systems benefit from large language models: A survey. *arXiv preprint arXiv:2306.05817*, 2023.

[85] Wenqi Fan, Zihuai Zhao, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Jiliang Tang, and Qing Li. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*, 2023.

[86] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. *arXiv preprint arXiv:2305.00447*, 2023.

[87] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.

[88] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Xiangnan He, Zhijin Wang, Bo Hu, and Zang Li. Transrec: Learning transferable recommendation from mixture-of-modality feedback. *arXiv preprint arXiv:2206.06190*, 2022.

[89] Jianmo Ni, Gustavo Hernández Abrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. Sentence-T5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021. URL `https://arxiv.org/abs/2108.08877`.

[90] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. Recommender systems with generative retrieval. *arXiv preprint arXiv:2305.05065*, 2023.

[91] Minae Kwon, Sang Michael Xie, Kalesha Bullard, and Dorsa Sadigh. Reward design with language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=10uNUgI5Kl`.

[92] Wentao Shi, Xiangnan He, Yang Zhang, Chongming Gao, Xinyue Li, Jizhi Zhang, Qifan Wang, and Fuli Feng. Large language models are learnable planners for long-term recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1893–1903, 2024.

[93] Markus Schedl. The LFM-1b dataset for music retrieval and recommendation. In *Proceedings of the 2016 ACM International Conference on Multimedia Retrieval (ICMR '16)*, pages 103–110. ACM, 2016. doi: 10.1145/2911996.2912004. URL `https://doi.org/10.1145/2911996.2912004`.

[94] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. Recommendations as treatments: Debiasing learning and evaluation. In *international conference on machine learning*, pages 1670–1679. PMLR, 2016.

[95] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. *Stabilizing off-policy Q-learning via bootstrapping error reduction*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[96] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20(4):422–446, 2002. doi: 10.1145/582415.582418.

[97] Ataur Rahman and Max L Wilson. Exploring opportunities to facilitate serendipity in search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 939–942, 2015.

[98] Marius Kaminskas and Derek Bridge. Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. *ACM Transactions on Interactive Intelligent Systems*, 7(1):1–42, 2016.

[99] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.

[100] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[101] Xiang-Rong Sheng, Liqin Zhao, Guorui Zhou, Xinyao Ding, Binding Dai, Qiang Luo, Siran Yang, Jingshan Lv, Chi Zhang, Hongbo Deng, et al. One model to serve all: Star topology adaptive recommender for multi-domain ctr prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 4104–4113, 2021.

[102] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

[103] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. Mm-rec: Multimodal news recommendation. *arXiv preprint arXiv:2104.07407*, 2021.

[104] Muyang Li, Xiangyu Zhao, Chuan Lyu, Minghao Zhao, Runze Wu, and Ruocheng Guo. Mlp4rec: A pure mlp architecture for sequential recommendations. *arXiv preprint arXiv:2204.11510*, 2022.

[105] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 378–387, 2021.

[106] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[107] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[108] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

[109] Fajie Yuan, Guibing Guo, Joemon M Jose, Long Chen, Haitao Yu, and Weinan Zhang. Lambdafm: learning optimal ranking with factorization machines using lambda surrogates. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 227–236, 2016.

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

[111] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[112] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[113] Naihan Li, Shujie Liu, Yanqing Liu, Sheng Zhao, and Ming Liu. Neural speech synthesis with transformer network. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6706–6713, 2019.

[114] Valerii Likhosherstov, Anurag Arnab, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, and Mostafa Dehghani. Polyvit: Co-training vision transformers on images, videos and audio. *arXiv preprint arXiv:2111.12993*, 2021.

[115] Weiwei Deng, Peihu Zhu, Han Chen, Tao Yuan, and Ji Wu. Knowledge-aware sequence modelling with deep learning for online course recommendation. *Information Processing & Management*, 60(4):103377, 2023.

[116] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 582–590, 2019.

[117] Eli Pariser. *The filter bubble: What the Internet is hiding from you*. Penguin UK, 2011.

[118] Yair Censor. Pareto optimality in multiobjective problems. *Applied Mathematics and Optimization*, 4(1):41–59, 1977.

[119] F Waltz. An engineering approach: hierarchical optimization criteria. *IEEE Transactions on Automatic Control*, 12(2):179–180, 1967.

[120] Tommaso Di Noia, Jessica Rosati, Paolo Tomeo, and Eugenio Di Sciascio. Adaptive multi-attribute diversity for recommender systems. *Information Sciences*, 382:234–253, 2017.

[121] Shanfeng Wang, Maoguo Gong, Haoliang Li, and Junwei Yang. Multi-objective optimization for long tail recommendation. *Knowledge-Based Systems*, 104:145–155, 2016.

[122] Ankush Jain, Pramod Kumar Singh, and Joydip Dhar. Multi-objective item evaluation for diverse as well as novel item recommendations. *Expert Systems with Applications*, 139: 112857, 2020.

[123] Feng Zou, Debao Chen, Qingzheng Xu, Ziqi Jiang, and Jiahui Kang. A two-stage personalized recommendation based on multi-objective teaching–learning-based optimization with decomposition. *Neurocomputing*, 452:716–727, 2021.

[124] Po-Lung Yu. A class of solutions for group decision problems. *Management science*, 19 (8):936–946, 1973.

[125] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23, 2010.

[126] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[127] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 368–377, 2018.

[128] Mengqi Zhang, Shu Wu, Meng Gao, Xin Jiang, Ke Xu, and Liang Wang. Personalized graph neural networks with attention mechanism for session-aware recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3946–3957, 2020.

[129] Yitong Pang, Lingfei Wu, Qi Shen, Yiming Zhang, Zhihua Wei, Fangli Xu, Ethan Chang, Bo Long, and Jian Pei. Heterogeneous global graph neural networks for personalized session-based recommendation. In *Proceedings of the fifteenth ACM international conference on web search and data mining*, pages 775–783, 2022.

[130] Zhizhuo Yin, Kai Han, Pengzi Wang, and Xi Zhu. H3gnn: Hybrid hierarchical hypergraph neural network for personalized session-based recommendation. *ACM Transactions on Information Systems*, 42(3):1–30, 2023.

[131] Yu Du, Sylvie Ranwez, Nicolas Sutton-Charani, and Vincent Ranwez. Is diversity optimization always suitable? toward a better understanding of diversity within recommendation approaches. *Information processing & management*, 58(6):102721, 2021.

[132] Kun Zhou, Hui Yu, Wayne Xin Zhao, and Ji-Rong Wen. Filter-enhanced mlp is all you need for sequential recommendation. In *Proceedings of the ACM web conference 2022*, pages 2388–2399, 2022.

[133] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[134] Dongyang Zhao, Liang Zhang, Bo Zhang, Lizhou Zheng, Yongjun Bao, and Weipeng Yan. Mahrl: Multi-goals abstraction based deep hierarchical reinforcement learning for recommendations. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 871–880, 2020.

[135] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017. URL `https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html`.

[136] Zhe Fu, Xi Niu, and Mary Lou Maher. Deep learning models for serendipity recommendations: A survey and new perspectives. *ACM Computing Surveys*, 2023.

[137] Zhe Fu, Xi Niu, and Li Yu. Wisdom of crowds and fine-grained learning for serendipity recommendations. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 739–748, 2023.

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

[139] Gangyi Zhang. User-centric conversational recommendation: Adapting the need of user with large language models. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1349–1354, 2023.

[140] Yingpeng Du, Di Luo, Rui Yan, Hongzhi Liu, Yang Song, Hengshu Zhu, and Jie Zhang. Enhancing job recommendation through llm-based generative adversarial networks. *arXiv preprint arXiv:2307.10747*, 2023.

[141] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[142] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*, 2021.

[143] Wes Gurnee and Max Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=jE8xbmvFin`.

[144] Kenneth Li, Aspen K Hopkins, David Bau, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=DeG07_TcZvT`.

[145] Shwai He, Liang Ding, Daize Dong, Miao Zhang, and Dacheng Tao. Sparseadapter: An easy approach for improving the parameter-efficiency of adapters. *arXiv preprint arXiv:2210.04284*, 2022.

[146] Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*, 2023.

[147] Lei Li, Yongfeng Zhang, and Li Chen. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1348–1357, 2023.

[148] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. How to index item ids for recommendation foundation models. *arXiv preprint arXiv:2305.06569*, 2023.

[149] Amir massoud Farahmand, Rémi Munos, and Csaba Szepesvári. *Error propagation for Approximate Policy and Value Iteration*, page 568–576. NIPS'10. Curran Associates Inc., Red Hook, NY, USA, 2010.

[150] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[151] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[152] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[153] Jie Wang, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. Reinforcement learning-based recommender systems with large language models for state reward and action modeling. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 375–385, 2024.

[154] Yuanqing Yu, Chongming Gao, Jiawei Chen, Heng Tang, Yuefeng Sun, Qian Chen, Weizhi Ma, and Min Zhang. Easyrl4rec: A user-friendly code library for reinforcement learning based recommender systems. *arXiv preprint arXiv:2402.15164*, 2024.

[155] Shuchang Liu, Qingpeng Cai, Bowen Sun, Yuhao Wang, Ji Jiang, Dong Zheng, Peng Jiang, Kun Gai, Xiangyu Zhao, and Yongfeng Zhang. Exploration and regularization of the latent action space in recommendation. In *Proceedings of the ACM Web Conference 2023*, pages 833–844, 2023.

[156] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[157] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Annual Conference on Robot Learning*, 2021.

[158] Shideh Rezaeifar, Robert Dadashi, Nino Vieillard, Léonard Hussenot, Olivier Bachem, Olivier Pietquin, and Matthieu Geist. Offline reinforcement learning as anti-exploration. In *AAAI Conference on Artificial Intelligence*, 2022.

[159] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.

[160] Younggyo Seo, Lili Chen, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. State entropy maximization with random encoders for efficient exploration. In *International Conference on Machine Learning*, pages 9443–9454. PMLR, 2021.

[161] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[162] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *arXiv preprint arXiv:1402.6028*, 2014.

[163] Dalin Guo, Sofia Ira Ktena, Pranay Kumar Myana, Ferenc Huszar, Wenzhe Shi, Alykhan Tejani, Michael Kneier, and Sourav Das. Deep bayesian bandits: Exploring in online personalized recommendations. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 456–461, 2020.

[164] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695, 2022.

[165] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022. doi: 10.48550/arXiv.2204.06125. URL https://arxiv.org/abs/2204.06125.

[166] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. Generative recommendation: Towards next-generation recommender paradigm. *arXiv preprint arXiv:2304.03516*, 2023. URL https://arxiv.org/abs/2304.03516.

[167] Yashar Deldjoo, Zhankui He, Julian McAuley, Anton Korikov, Scott Sanner, Arnau Ramisa, Rene Vidal, Maheswaran Sathiamoorthy, Atoosa Kasrizadeh, Silvia Milano, et al. Recommendation with generative models. *arXiv preprint arXiv:2409.15173*, 2024. URL https://arxiv.org/abs/2409.15173.

[168] Wuchao Li, Rui Huang, Haijun Zhao, Chi Liu, Kai Zheng, Qi Liu, Na Mou, Guorui Zhou, Defu Lian, Yang Song, Wentian Bao, Enyun Yu, and Wenwu Ou. DimeRec: A unified framework for enhanced sequential recommendation via generative diffusion models. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 726–734. ACM, 2025. doi: 10.1145/3701551.3703555. URL https://doi.org/10.1145/3701551.3703555.

[169] Zhao Liu, Yichen Zhu, Yiqing Yang, Guoping Tang, Rui Huang, Qiang Luo, Xiao Lv, Ruiming Tang, Kun Gai, and Guorui Zhou. Diffgrm: Diffusion-based generative recommendation model. *arXiv preprint arXiv:2510.21805*, 2025. URL https://arxiv.org/abs/2510.21805.

[170] Zihe Wang and Bo Jin. Diffsbr: A diffusion model for session-based recommendation. *Information Processing & Management*, 63(1):104284, 2026. URL https://doi.org/10.1016/j.ipm.2025.104284.

[171] Zhu Zhang, Bo Yang, and Yimeng Lu. A local context enhanced consistency-aware mamba-based sequential recommendation model. *Information Processing & Management*, 62(3):104076, 2025. URL https://doi.org/10.1016/j.ipm.2025.104076.

[172] Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to sequence text generation with diffusion models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=jQj-_rLVXsj.

[173] Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35:4328–4343, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

[174] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. URL https://ieeexplore.ieee.org/abstract/document/10377858/.

[175] Pu Cao, Feng Zhou, Qing Song, and Lu Yang. Controllable generation with text-to-image diffusion models: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. URL https://ieeexplore.ieee.org/abstract/document/11304732.

[176] Dawen Liang, Rahul G. Krishnan, Matthew D. Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*, pages 689–698. ACM, 2018. doi: 10.1145/3178876.3186150. URL https://doi.org/10.1145/3178876.3186150.

[177] Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *Advances in Neural Information Processing Systems*, volume 36, pages 24247–24261, 2023. URL https://dl.acm.org/doi/abs/10.5555/3666122.3667176.

[178] Tao Meng, Sidi Lu, Nanyun Peng, and Kai-Wei Chang. Controllable text generation with neurally-decomposed oracle. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2205.14219.

[179] Yashar Deldjoo, Alejandro Bellogin, and Tommaso Di Noia. Explaining recommender systems fairness and accuracy through the lens of data characteristics. *Information Pro-*

cessing & Management, 58(5):102662, 2021. URL https://doi.org/10.1016/j.ipm.2021.102662.

[180] Ludovico Boratto, Gianni Fenu, Mirko Marras, and Giacomo Medda. Practical perspectives of consumer fairness in recommendation. *Information Processing & Management*, 60(2):103208, 2023. URL https://doi.org/10.1016/j.ipm.2022.103208.

[181] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015. URL http://proceedings.mlr.press/v37/sohl-dickstein15.html.

[182] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

[183] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015. URL https://link.springer.com/chapter/10.1007/978-3-319-24574-4_28.

[184] Xin Ma, Yiming Wang, Guangyiting Jia, Xue Chen, Zhiqiang Liu, Ya-Feng Li, Chen Chen, and Yu Qiao. Latte: Latent diffusion transformer for video generation. *arXiv preprint arXiv:2401.03048*, 2024. URL https://arxiv.org/abs/2401.03048.

[185] Zeyi Li, Jiahao Zhang, Qing Lin, Junjie Xiong, Yu Long, Xian Deng, Yike Zhang, Xuan Liu, Minlie Huang, Zhen Xiao, et al. Hunyuan-DiT: A powerful multi-resolution diffusion transformer with fine-grained chinese understanding. *arXiv preprint arXiv:2405.08748*, 2024. URL https://arxiv.org/abs/2405.08748.

[186] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8162–8171. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/nichol21a.html.

[187] Abhimanyu Dubey, Akhil Jauhri, Abhijeet Pandey, Abhinav Kadian, Ammar Al-Dahle, Albert Letman, Anjali Mathur, Andre Schelten, Aishwarya Yang, Angela Fan, et al. The LLaMA 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. URL https://arxiv.org/abs/2407.21783.

[188] Jonathan Ho et al. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[189] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, 2004. URL https://aclanthology.org/W04-1013.pdf.

[190] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019. URL https://arxiv.org/abs/1904.09675.

[191] Zhiyuan Liu, Yujia Wang, and Percy Liang. Controllable text generation with attribute-aware diffusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, pages 13342–13356, 2023.

[192] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[193] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models, February 2023. URL http://arxiv.org/abs/2302.13971. arXiv:2302.13971.

[194] Dan Hendrycks and Kevin Gimpel. Bridging nonlinearities and stochastic regularizers with gaussian error linear units, 2016. URL https://arxiv.org/abs/1606.08415.

[195] William Peebles and Saining Xie. Adaptive normalization for vision transformers. *CVPR*, 2023.

[196] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016. URL https://arxiv.org/abs/1607.06450.