Single-behaviour and Multi-behaviour Streaming Recommender Systems

by

Yan Zhao

Master of Computer Science, Blaise Pascal University (Aubière, France), 2016

Bachelor of Computer Science, Harbin Institute of Technology (Harbin, China), 2014

A thesis submitted in fulfilment of

the requirements for the award of the degree

Doctor of Philosophy

from

Department of Computing

Faculty of Science and Engineering

MACQUARIE UNIVERSITY

Supervisor: Prof. Yan Wang

Associate Supervisor: Prof. Michael Sheng

Adjunct Supervisor: Dr. Shoujin Wang

May 2021

Statement of Candidate

I certify that this thesis entitled **"Single-behaviour and Multi-behaviour Streaming Recommender Systems"** is being submitted to Macquarie University and Harbin Institute of Technology in accordance with the Cotutelle agreement dated 20 September 2018.

I also certify that this thesis is an original piece of research and it has been written by me. Any help and assistance that I have received in my research work and the preparation of this thesis itself have been appropriately acknowledged.

In addition, I certify that all information sources and literature used are indicated in this thesis.

[illegible]

Yan Zhao

14 May 2021

To my father, my sister, and my wife,

who make me understand the true meaning of love.

In memory of my mother,

Fengyan Zhao

Abstract

In this information age, Recommender Systems (RSs) have played an increasingly important role in providing users with tailored suggestions that match their preferences. However, conventional offline RSs cannot well deal with the ubiquitous data streams of user-item interactions. This is because offline RSs are periodically trained with large-volume historical interaction data, and thus cannot well capture the latest preferences of users embedded in their recent interactions. To address this issue, Streaming Recommender Systems (SRSs) have emerged in recent years, which commonly train recommendation models with newly coming interaction data to capture the latest user preferences for streaming recommendations. For further improving the accuracies of streaming recommendations, this thesis proposes the following three approaches.

Firstly, training recommendation models with newly coming data only benefits overcoming the *preference drift* problem, but overlooks the *long-term user preferences* embedded in the historical data. Moreover, the common *heterogeneity* of users and items makes it more challenging to deliver accurate streaming recommendations, as different types of users (or items) have different preferences (or characteristics). To address these two issues, we propose a Variational and Reservoir-enhanced Sampling based Double-Wing Mixture-of-Experts framework, called VRS-DWMoE. Specifically, in VRS-DWMoE, we first devise variational and reservoir-enhanced sampling to wisely complement newly coming data with historical data for capturing long-term user preferences while addressing the issue of preference drift. Then, we propose a double-wing mixture-of-experts model to effectively learn the heterogeneous user preferences and item characteristics with two mixture of experts, respectively, where each individual expert model specialises in one type of users or items.

Secondly, the commonly existing *underload* (or *overload*) scenarios, where the data receiving speed is lower (or higher) than the data processing speed, should be well dealt with for accurate streaming recommendations. Therefore, we propose a Stratified and Time-aware Sampling based Adaptive Ensemble Learning framework, called STS-AEL. Specifically, in STS-AEL, we first devise stratified and time-aware sampling to extract training data from both newly coming data and historical data. This practice not only benefits utilising the idle resources in underload scenarios more effectively, but also helps capture long-term user preferences while addressing the preference drift issue. After that, we propose adaptive ensemble learning to first leverage multiple individual recommendation models for *concurrently* learning from the prepared training data, and then fuse the results of these individual models with a sequential adaptive mechanism for accurate streaming recommendations.

Thirdly, all the existing SRSs have been devised for dealing with data streams of user-item interactions w.r.t. a single behaviour type (e.g., purchases) and commonly suffer from the data sparsity issue caused by the limited number of such single-behaviour interactions. Therefore, we propose the *first* Multi-behaviour Streaming Recommender System, called MbSRS, to exploit more sufficient *multi-behaviour* interactions (e.g., purhcases, add-to-carts and views) for further improving the accuracies of streaming recommendations. In MbSRS, confronting data streams of multi-behaviour interactions, we first propose the Multi-behaviour Learning Module (MbLM) to accurately learn the short-term user preferences and stable item characteristics. Then, we propose the Attentive Memory Network (AMN) to effectively maintain the long-term user preferences. After that, these learnt short-term user preferences and long-term user preferences are merged by the elaborately devised User Preference Merging Module (UPMM). Note that MbLM, AMN and UPMM all effectively leverage the multi-behaviour interactions to further improve the accuracies of streaming recommendations.

The superiorities and the effectiveness of all the above three approaches proposed in this thesis have been validated by both the theoretical analysis and extensive experiments that are conducted on real-world datasets.

Acknowledgments

First of all, I would like to express my sincere gratitude to my supervisors, Prof. Yan Wang (principal supervisor), Prof. Michael Sheng (associate supervisor) and Dr. Shoujin Wang (adjunct supervisor), for their continuous help and patient supervision during my time in Macquarie University. Meanwhile, I would also like to thank my supervisor at my home university, Prof. Hongwei Liu, for his complete support throughout the years. Without their selfless devotions and practical suggestions, this thesis would not have been completed. Over the past years, I have learnt how to do high-quality research from them and established a rigorous research attitude. This experience will be of great value to my future academic career.

Second, I wish to express my thanks to my colleagues and the staff in the Department of Computing for their support. They have provided me great encouragement during my tough times and established a comfortable working environment.

Most importantly, I would like to thank my father and sister, Xianggui Zhao and Yujiao Zhao, for their unconditional love throughout my life. Sincerest thanks to my wife, Xiaoyu Zhang, who gives me the confidence and courage I needed to pursue my dream. Their love is my fundamental motivation to complete this thesis.

Publications

This thesis is based on the research work I completed with the help of my supervisors and other colleagues during my Cotutelle PhD at the Department of Computing, Macquarie University, between 2019 and 2021. Some parts of my research are contained in the following papers:

[1] **Yan Zhao**, Shoujin Wang, Yan Wang, Hongwei Liu, Weizhe Zhang: Double-Wing Mixture of Experts for Streaming Recommendations, 21st International Conference on Web Information Systems Engineering (WISE 2020), pages 269-284. (**regular paper, acceptance rate 19.5%, CORE2020**[1] **Rank A**)

[2] **Yan Zhao**, Shoujin Wang, Yan Wang, Hongwei Liu: Stratified and time-aware sampling based adaptive ensemble learning for streaming recommendations, Applied Intelligence (APIN), online published in November 2020. (**CORE2020 Rank B**)

[3] **Yan Zhao**, Shoujin Wang, Yan Wang, Hongwei Liu: MbSRS: a Multi-behaviour Streaming Recommender System. Submitted to IEEE Transactions on Knowledge and Data Engineering (TKDE, **CORE2020 Rank A***)

[1]CORE stands for Computing Research and Education Association of Australasia (`http://www.core.edu.au`).

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Significance

Recommender Systems (RSs) [70] are a type of information filtering systems, which assist users to select items more effectively and efficiently in the current big data era. With the ever-growing data volume, RSs have played an increasingly important role in both academia and industry to confront the information explosion. As data streams of user-item interactions (e.g., continuous purchase records generated by online shopping websites) are pervasive in this information age [6], delivering accurate streaming recommendations is an essential task we need to accomplish. However, this task remains a challenge [127] despite the fast development and wide applications of RSs. This is because that conventional offline RSs periodically train recommendation models with large-volume historical interaction data, which impedes capturing the latest preferences of users embedded in their recent interactions.

To deliver accurate streaming recommendations, recent years have seen an emerging trend where recommendation models are trained with newly coming interaction data instantly. The RSs following this trend are commonly referred to as *online* [38] or *Streaming Recommender Systems* (SRSs) [6]. Compared with the conventional offline RSs, SRSs have the following three main advantages.

1) SRSs can capture the latest user preferences. As user preferences commonly change over time, capturing the latest user preferences significantly contributes to delivering accurate recommendations based on users' current intentions for

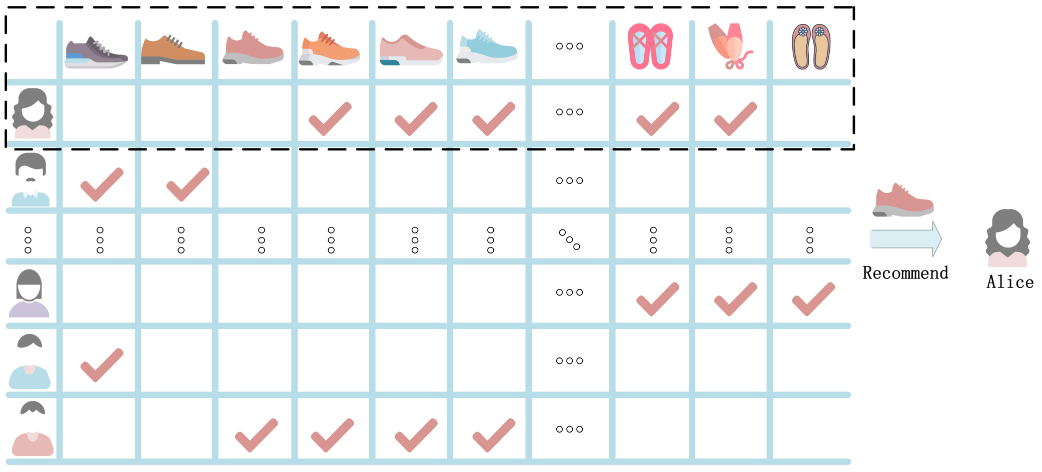
interacting with items. However, offline RSs treat all the available interactions equally, and thus can only learn the overall user preferences from all available interaction data. In contrast, SRSs are commonly trained with newly coming interaction data, and thus can well capture the latest user preferences embedded in the recent interaction data for more accurate streaming recommendations.

2) SRSs can better deal with the cold start problem. By training recommendation models with newly coming interaction data in a timely manner, SRSs can quickly learn the preferences of new users and characteristics of new items, and thus make accurate recommendations in terms of new users and new items.

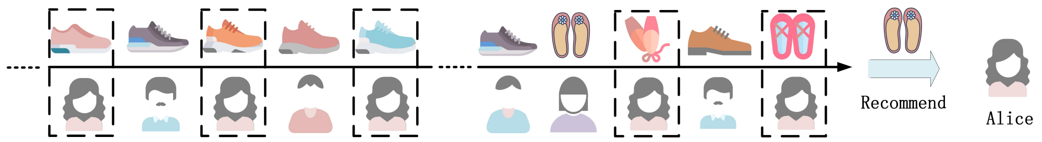
3) SRSs do not need to store large-volume interaction data. Offline RSs are periodically trained with large-volume historical data, and thus they require large-space storage for performing the offline recommendations. In contrast, most SRSs do not need to store any historical data, which benefits protecting user privacy and also helps reduce the extra expenses for the storage.

The superiorities of SRSs over offline RSs are also illustrated by the following motivating example.

Motivating Example 1. We present a motivating example in Fig. 1.1, which depicts an online shopping scenario. Specifically, after Alice has purchased three pairs of sports shoes last year and two pairs of dance shoes this year, she is recommended a pair of sports shoes and a pair of dance shoes by the offline RS and SRS, respectively. This is because the offline RS considers all Alice's purchased items equally, and infers that Alice prefers the sports shoes as she has purchased more sports shoes than the dance shoes. In contrast, the SRS successfully captures Alice's latest preferences towards dance shoes embedded in her recent purchases, and thus recommends her a pair of dance shoes. In the end, Alice purchases dance shoes that better match her latest preference. This motivating example clearly illustrates the fact that SRSs are able to deliver more accurate recommendations than offline RSs do by effectively capturing users' latest preferences.



(a) Offline Recommender System

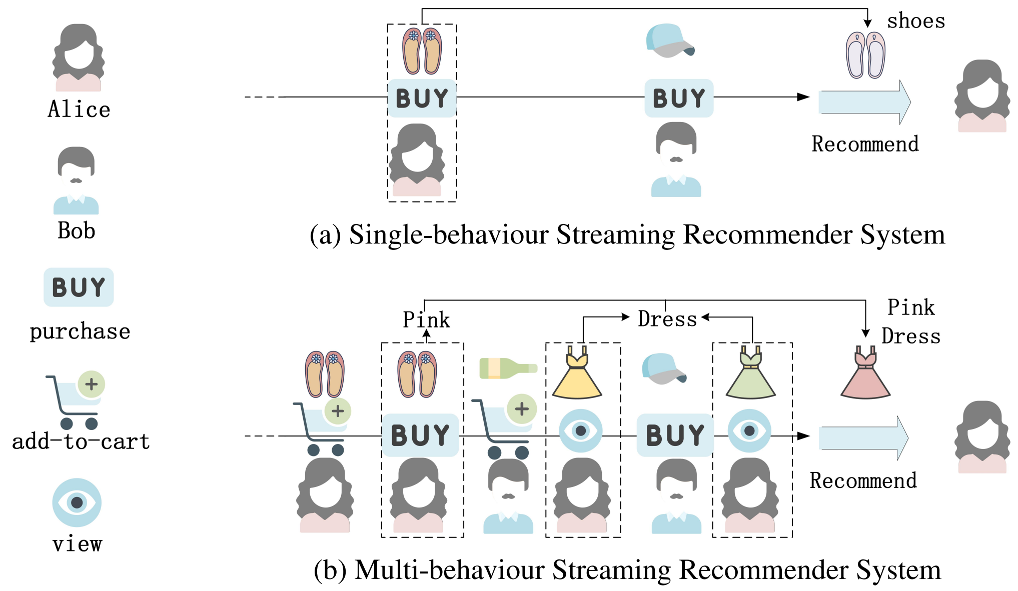


(b) Streaming Recommender System

Figure 1.1: Comparison between offline RS and streaming RS

The existing SRSs are mainly developed in two stages: 1) *adaption-based SRSs* and 2) *stream-oriented SRSs*. As an earlier attempt, adaption-based SRSs aim to adapt the conventional offline RSs to streaming scenarios by incrementally training their recommendation models with newly coming data, such as Incremental Collaborative Filtering (ICF) [86] and Element-wise Alternating Least Squares (eALS) [38]. Recently, stream-oriented SRSs have been proposed, which are specifically devised for the streaming scenarios, such as Stream-centered Probabilistic Matrix Factorisation (SPMF) [127] and Neural Memory Recommender Networks (NMRN) [121].

These SRSs can all be referred to as *single-behaviour* SRSs, as they make recommendations with data streams of interactions w.r.t. a single type of behaviour (e.g., purchases). Intuitively, in the cases where interactions w.r.t. multiple behaviour types (e.g., purchases, add-to-carts and views) are available, *multi-behaviour* SRSs should be proposed to incorporate such multi-behaviour interactions for accurate streaming recommendations. This is because incorporating the more sufficient multi-behaviour



Alice
Bob
BUY
purchase
add-to-cart
view
shoes
BUY
BUY
Recommend
(a) Single-behaviour Streaming Recommender System
Pink
Dress
Pink
Dress
BUY
BUY
Recommend
(b) Multi-behaviour Streaming Recommender System

Figure 1.2: Comparison between the single-behaviour SRS and multi-behaviour SRS

interactions benefits addressing the long-standing data sparsity issue, and thus contributes to more accurate streaming recommendations when confronting data streams of multi-behaviour interactions. This is also illustrated in the following motivating example.

Motivating Example 2. In the motivating example presented in Fig. 1.2, after Alice has purchased a pair of pink shoes and viewed two dresses with different colours, the single-behaviour SRS in Fig. 1.2a recommends her another pair of shoes based on her single-behaviour (*purchase*) interactions while the multi-behaviour SRS in Fig. 1.2b recommends her a pink dress based on her multi-behaviour (*purchase* and *view*) interactions. As Alice has already purchased a pair of similar shoes recently, she prefers the pink dress that is recommended based on her multi-behaviour interactions. This motivating example clearly illustrates the fact that exploiting multi-behaviour interactions significantly contributes to more accurate streaming recommendations.

This thesis aims to improve the accuracies of both single-behaviour recommendations (see Chapters 3 and 4) and the multi-behaviour recommendations (see Chapter 5). The following subsection introduces the challenges we need to address for achieving this goal.

1.2 Challenges of Streaming Recommendations

In this subsection, we introduce the four main challenges that are targeted in this thesis. Specifically, Challenges 1, 2 and 3 (**CH1**, **CH2** and **CH3**) are from single-behaviour streaming recommendations, while Challenge 4 (**CH4**) is from multi-behaviour streaming recommendations.

1.2.1 Preference Drift and Long-term User Preferences

The first challenge **CH1** in this thesis is 'how to well deal with the *preference drift* (i.e., user preferences towards items change over time) while capturing *long-term user preferences*'.

The evolutional user preferences over time cause preference drift (also known as concept drift) in streaming recommendations [127]. For example, the preferences of Alice towards the styles of clothes change as she ages. Another problem is the loss of long-term user preferences. For example, Bob likes to read history books; however, the online book store recommends him math books only after he recently bought some math books just for an examination. It is a challenging task for SRSs to simultaneously deal with the above two problems.

Targeting this challenge, SPMF [127] and NMRN [121] have proposed reservoir-based and neural memory network based approaches, respectively. However, SPMF has difficulties in dealing with preference drift as it does not effectively utilise newly coming data to capture the latest user preferences, while NMRN has a limited capability to capture long-term user preferences as the memory recording preferences might be updated frequently over the continuous data stream.

1.2.2 Heterogeneity of Users and Items

The second challenge **CH2** in this thesis is 'how to well deal with the *heterogeneity* of both users and items (i.e., different types of users and items have different preferences and characteristics, respectively)'.

Different preferences (or characteristics) of different types of users (or items) commonly exist in the real world. For example, ballet dancers tend to purchase ballet shoes while tennis players are more likely to purchase rackets. This heterogeneity makes accurate streaming recommendations for all types of users and items a challenging task. However, this issue has not been discussed in the literature of SRSs. Existing SRSs commonly utilise a unified model to learn both user preferences and item characteristics for all the users and items [32, 38]. Nevertheless, with a single unified model, SRSs cannot well deal with the intrinsic differences between user preferences and item characteristics or accurately learn the preferences (or characteristics) of different types of users (or items).

1.2.3 Underload Problem and Overload Problem

The third challenge **CH3** in this thesis is 'how to tackle the *underload* problem (i.e., the scenarios where the data coming speed is *lower* than the data processing speed) and the *overload* problem (i.e., the scenarios where the data coming speed is *higher* than the data processing speed)'. This challenge can be further decomposed into the following two sub-challenges.

CH3.1 *how to tackle the underload problem?* The low resource utilisation ratio caused by the underload problem commonly exists in real-world applications [62, 51], as the systems are devised to be scalable and prepared for the peak demand. Although resource management approaches [35, 75] have been studied to address the underload problem for general stream processing, to the best of our knowledge, no studies have been reported in the literature to particularly address the underload problem in streaming recommendations. However, it is an important issue, since the wasted computational resources in the widely-existing underload scenarios should be effectively utilised for improving the recommendation accuracies. Therefore, we argue that SRSs that can effectively deal with the underload problem should be proposed.

CH3.2 *how to tackle the overload problem?* The overload problem has attracted more research interests in the stream processing areas [93, 49], including the streaming recommendations [6, 127, 32]. As the generation velocity of data streams keeps increasing, SRSs should be well prepared to handle the intensive workload beyond their capacities. To address the overload problem in streaming recommendations, sampling methods have been proposed in the literature [32, 127] to reduce the training workload of monolithic SRSs (i.e., SRSs that employ single recommendation models for recommendations). However, monolithic SRSs cannot completely tackle the overload problem due to their limited computational capabilities.

1.2.4 Multi-behaviour Interactions for Streaming Recommendations

The fourth challenge **CH4** in this thesis is 'how to wisely leverage multi-behaviour interactions for improving the accuracies of streaming recommendations'. This challenge can be further decomposed into the following three sub-challenges.

CH4.1 *how to wisely exploit multi-behaviour interactions to accurately learn the short-term user preferences and stable item characteristics in the streaming scenarios?* Compared with the single-behaviour interactions, the multi-behaviour ones are much more difficult to deal with. This is because multi-behaviour interactions comprise the following two types of user preferences (or item characteristics): 1) the shared user preferences (or item characteristics) across multiple behaviour types, e.g., a user tends to *purchase* and *view* the same brand of mobile phones; and 2) the behaviour-specific user preferences (or item characteristics), e.g., a user *views* expensive mobile phones for curiosity but tends to *purchase* cheap ones for the financial issue. Therefore, these two types of user preferences (or item characteristics) need to be both accurately learnt from data streams of multi-behaviour interactions. Moreover, different from the relative stable item characteristics, user preferences vary over time. Thus, short-term preferences of users embedded in their recent interactions need to be well captured for accurate streaming recommendations.

Although some offline multi-behaviour RSs [28] attempt to utilise multi-behaviour interactions for offline recommendations, they fail to explicitly learn the shared user preferences (or item characteristics) and behaviour-specific user preferences (or item characteristics) for further improving recommendation accuracies. Furthermore, they are not devised to deal with data streams, and thus are not able to be employed for streaming recommendations.

CH4.2 *how to wisely utilise the multi-behaviour interactions for effectively maintaining the long-term user preferences in the streaming scenarios?* Compared with that in offline recommendation scenarios, maintaining long-term user preferences in streaming scenarios is more challenging. This is because recommendation models for streaming scenarios are commonly trained with newly coming data only, and thus their learnt long-term user preferences embedded in historical interactions are easily overwritten by the short-term ones. For example, Bob prefers reading history books, however, he is only recommended textbooks after he has purchased some textbooks for exams. Moreover, incorporating multi-behaviour interactions makes this problem more challenging. This is because interactions w.r.t. different behaviour types might comprise different long-term user preferences, and are difficult to be exploited simultaneously for maintaining such preferences w.r.t. the primary behaviour.

Although some single-behaviour SRSs [127, 32] have been proposed to maintain the long-term user preferences, they are not devised to exploit the multi-behaviour interactions for more accurately learning such long-term preferences.

CH4.3 *how to effectively merge the short-term and long-term user preferences when confronting data streams of multi-behaviour interactions?* Merging short-term user preferences and long-term user preferences benefits reflecting users' overall intention for interacting with items, and thus leads to more accurate recommendations. However, this merging process is challenging since the contributions of short-term preferences (or long-term preferences) might be different for different users, and even different for the same user towards different items. In addition, this problem becomes more challenging when incorporating multiple behaviour types, as we need to exploit the multi-behaviour interactions to obtain the merged user preferences w.r.t. the primary behaviour. For example, Alice *watched* romantic movies for months while *searching* several thriller movies this week, it is challenging to decide which type of movies to be recommended for her to *watch*.

Although some SRSs [150] have taken both short-term preferences and long-term preferences into consideration, they learn these two preferences simultaneously by a single module, and thus the merging process is not devised. However, we argue that short-term preferences and long-term preferences cannot be well learnt by a single module simultaneously, as these two types of preferences are inconsistent and might affect the learning processes of each other in some cases.

1.3 Thesis Contributions

Targeting the four challenges of streaming recommendations mentioned above, this thesis has three major contributions.

1. Targeting **CH1** and **CH2**, the first contribution is proposing a novel mixture of

expert based streaming recommendation framework along with an elaborately devised sampling method. The characteristics and contributions of this work are summarised as follows:

- In this work, we propose a novel VRS-DWMoE framework, which consists of Variational and Reservoir-enhanced Sampling (VRS) and Double-Wing Mixture of Experts (DWMoE), for accurate streaming recommendations.
- To address **CH1**, we propose VRS to wisely complement newly coming data with the sampled historical data while guaranteeing the proportion of newly coming data. In this way, VRS not only benefits addressing the preference drift issue by highlighting the importance of newly coming data but also helps learn the long-term user preferences from the sampled historical data.
- To address **CH2**, we propose DWMoE to effectively learn heterogeneous user preferences and item characteristics. Specifically, DWMoE not only learns user preferences and item characteristics with two elaborately devised MoEs, respectively, to deal with their intrinsic differences, but also allows each expert to specialise in one underlying type of users (or items) to more effectively learn the heterogeneous user preferences (or item characteristics).

2. Targeting **CH1** and **CH3**, the second contribution is proposing a novel ensemble learning based framework that can well deal with the underload problem and overload problem for delivering accurate streaming recommendations. The relevant characteristics and contributions of this framework are summarised as follows:

   - In this work, we propose a novel STS-AEL framework for more accurate streaming recommendations, which contains two main components: STS and AEL.

- To address **CH1** and **CH3.1**, we propose STS to extract representative data from both newly coming data and historical data while guaranteeing the proportion of newly coming data. In this way, through elaborately incorporating both newly coming data and historical data, STS can not only capture both short-term and long-term user preferences, but also effectively utilise idle resources in underload scenarios for delivering accurate recommendations in the streaming scenarios.
- To address **CH3.1** and **CH3.2**, we propose AEL to increase recommendation accuracies in both underload scenarios and overload scenarios. AEL first conducts concurrent training to address the excessive data in both underload scenarios (complemented by the sampled historical data) and overload scenarios via multiple individual models, and then fuses the prediction results of these models to achieve higher recommendation accuracies. Moreover, for the fusion process, we propose a sequential adaptive fusion method specifically for streaming scenarios to further improve the accuracies of streaming recommendations.

Extensive experiments demonstrate that the proposed STS-AEL framework significantly outperforms the state-of-the-art SRSs in terms of recommendation accuracies on all three widely-used datasets in both underload scenarios and overload scenarios.

3. Targeting **CH4**, the third contribution is proposing a novel multi-behaviour streaming recommender system, which effectively exploits data streams of interactions w.r.t. multiple behaviour types for improving the accuracies of streaming recommendations. To the best of our knowledge, this is the *first* SRS in the literature that exploits the multi-behaviour interactions for streaming recommendations. The characteristics and contributions of this work are summarised as follows:

   - In this work, we propose the *first* Multi-behaviour Streaming Recommender

System, called MbSRS. Specifically, MbSRS contains three main components: Multi-behaviour Learning Module (MbLM), Attentive Memory Network (AMN) and User Preference Merging Module (UPMM).

- To address **CH4.1**, we propose MbLM to accurately learn both the short-term user preferences and the stable item characteristics in streaming scenarios by elaborately learning shared embeddings and behaviour-specific embeddings for both users and items.
- To address **CH4.2**, we propose AMN to well maintain the long-term user preferences by first memorising the items that interacted with the target user in terms of all the behaviours, and then wisely utilising these items to represent the long-term preferences of the target user w.r.t. the target behaviour type via an attentive method.
- To address **CH4.3**, we propose UPMM to wisely merge the short-term user preferences and long-term user preferences with a gate mechanism. Through this way, the short-term user preferences and long-term user preferences w.r.t. the primary behaviour can be well merged for collaboratively contributing to the accurate streaming recommendations.

Extensive experiments on three real-world datasets demonstrate that our proposed MbSRS significantly outperforms both the state-of-the-art SRSs and the state-of-the-art offline multi-behaviour RSs that can be modified to adapt to the streaming scenarios.

1.4 Thesis Structure

The remainder of this thesis is structured as follows.

Chapter 2 reviews the literature on RSs related to the work in this thesis, including the single-behaviour offline RSs, single-behaviour streaming RSs and multi-behaviour offline RSs.

Chapter 3 introduces our proposed VRS-DWMoE framework, which not only addresses the preference drift issue while maintaining the long-term user preferences, but also well handles the heterogeneity of users and items for improving the accuracies of streaming recommendations. This chapter covers our work that has been published in WISE'2020 [150].

Chapter 4 introduces our proposed STS-AEL framework, which can well deal with both underload scenarios and overload scenarios for accurate streaming recommendations. It covers our work that has been published in APIN'2020 [149].

Chapter 5 introduces our proposed MbSRS, which effectively exploits the multi-behaviour interactions for further improving the accuracies of streaming recommendations. This chapter covers our work that is prepared to be submitted to TKDE.

Chapter 6 concludes this thesis and provides an outlook for future research directions.

Chapter 2

Literature Review

In this chapter, the existing literature related to the work in this thesis are introduced. Specifically, these related literature are reviewed based on the categories w.r.t. recommendation scenarios of their proposed RSs: (1) Single-behaviour Offline Recommender Systems (SbORSs), (2) Single-behaviour Streaming Recommender Systems (SbSRSs), and (3) Multi-behaviour Offline Recommender Systems (MbORSs). To the best of our knowledge, no multi-behaviour SRSs have been reported in the literature, thus they are not introduced in this chapter. In addition, as a detailed introduction of the widely-explored SbORSs is beyond the scope of this thesis, this chapter only reviews the SbORSs that have inspired our work, including representative SbORSs, memory network enhanced SbORSs and ensemble learning based SbORSs.

The remainder of this chapter is organized as follows:

- Section 2.1 reviews the related SbORSs, including representative SbORSs, memory network enhanced SbORSs and ensemble learning based SbORSs.
- Section 2.2 reviews the existing SbSRSs, including adaption-based SbSRSs and stream-oriented SbSRSs.
- Section 2.3 reviews the existing MbORSs, including 2-behaviour ORSs and n-behaviour ORSs.

2.1 Single-behaviour Offline Recommender Systems

Single-behaviour offline RSs is a well-explored research area, which has been studied from a wide range of perspectives. So far, many surveys have been proposed to systematically introduce the SbORSs devised for different recommendation scenarios, including group recommendations [13], conversational recommendations [46], explainable recommendations [148], session-based recommendations [123, 124], review-based recommendations [9] and social recommendations [135]; or with different recommendation mechanisms, including deep learning based recommendations [147], knowledge graph based recommendations [33], graph neural network based recommendations [125] and adversarial machine learning based recommendations [14]. According to the focus of this thesis, this section only reviews the representative SbORSs (see Subsection 2.1.1) and those are highly relevant to our work, including memory network enhanced SbORSs (see Subsection 2.1.2) and ensemble learning based SbORSs (see Subsection 2.1.3). Readers can refer to the corresponding surveys mentioned above for the detailed introduction to other types of SbORSs.

2.1.1 Representative SbORSs

Based on recommendation mechanisms, SbORSs can be roughly categorised into three types: 1) content-based SbORSs [69, 88, 4], 2) collaborative filtering based SbORSs [94, 37, 138] and 3) hybrid SbORSs [10, 106, 50] which combine the first two types. Content-based SbORSs make recommendations by analysing the profiles of users and properties of items. Differently, collaborative filtering based SbORSs recommend items to target users by analysing the interactions from both target users and other users who have similar preferences. During the past decade, the collaborative filtering based SbORSs attract the most research interests, and have achieved substantial progress towards delivering accurate recommendations. The most prevailing collaborative filtering based SbORSs are based on the matrix factorisation models or deep learning models. Generally, these SbORSs first learn the latent factors of users (a.k.a.

Table 2.1: The comparison of representative matrix factorisation based SbORSs

Category	**Representative approach**	**Technique adopted or basic idea**
Without side information	FunkSVD – Funk [99]	Singular value decomposition
	PMF – Mnih et al. [94]	Probability theory
	Koren et al. [56]	Personalized bias
	WRMF – Hu et al. [41]	Weighted factorisation
	WMF – Pan et al. [84]	Weighted factorisation
	BPR – Rendle et al. [91]	Bayesian optimisation & personalised ranking
With side information	SR – Cheng et al. [72]	Social constraints
	sRui – Ma [71]	Social constraints
	BMFSI – Porteous et al. [89]	Bayesian optimisation

user embeddings) and latent factors of items (a.k.a. item embeddings), and then leverage these latent factors to make recommendations. To be more specific, matrix factorisation based SbORSs rely on the linear operation (i.e., dot product) for learning users' preferences towards items from user-item interactions, while the deep learning based SbORSs employ multiple neural network layers to capture complex and non-linear user-item relations between users and items. In the following, we introduce representative Matrix Factorisation based SbORSs (MF-SbORSs) in Subsection 2.1.1.1 with a brief summary in Table 2.1, and introduce representative Deep Learning based SbORSs (DL-SbORSs) in Subsection 2.1.1.2 with a brief summary in Table 2.2.

2.1.1.1 Matrix Factorisation based SbORSs

In this subsection, we introduce MF-SbORSs with two parts, i.e., 1) MF-SbORSs without side information and 2) MF-SbORSs with side information.

MF-SbORSs without side information

Singular Value Decomposition (SVD) [104] is a well known mathematical approach for factorising a matrix. Inspired by SVD, Simon Funk proposed FunkSVD [99] to perform recommendations. Specifically, FunkSVD first factorises a rating matrix into two low-rank matrixes, which incorporate the latent factors of users and latent factors of items, respectively. Compared with the original SVD approach, FunkSVD does not only have a lower time complexity but can also apply to the matrixes with missing values, which are common for user-item interaction matrixes. Later on, many recommendation models have been proposed based on the matrix factorisation methods.

For example, Probabilistic Matrix Factorisation (PMF) [94] was proposed to improve recommendation accuracies with the probability theory. Specifically, PMF first initialises the latent factors of users and the latent factors of items with the Gaussian distribution, then learns user preferences and item characteristics based on these well initialised latent factors, and finally makes recommendations with these learnt user preferences and item characteristics. Moreover, the work in [94] extended PMF to constrained PMF, which assumes that users with similar rated items possibly have similar preferences. Their experimental results show that this constraint PMF is more effective and performs well on the Netflix dataset.

In addition, Koren et al. [56] added biases to the original matrix factorisation model. In this way, the personalised features of the target users and target items are also considered for recommendations. First, they calculated the overall average rating of the dataset, and the biases of the ratings for the target users and target items. Then, they summed up these three values to serve as regularisation terms for the matrix factorisation process. In this way, the predicted ratings are tuned for target users and the target items, and thus are possibly more accurate.

Later on, Hu et al. [41] and Pan et al. [84] improved the matrix factorisation models for performing recommendations w.r.t. implicit behaviours, such as purchases, views and clicks. Compared with the explicit behaviours such as ratings, implicit behaviours are generally more common and easier to collect. Thus, recommendations w.r.t. implicit behaviours should be well studied. However, this is a challenging task because implicit behaviours do not explicitly reflect user preferences. For example, users might find that they are not interested in a video after viewing it. To address this issue, Hu et al. [41] and Pan et al. [84] both proposed the weighted matrix factorisation models, which assign each interaction with a weight indicating the preferences of the target user on the target item. Moreover, the work in [84] also employs a negative sampling method to leverage the user-item pairs without observed interactions for effectively training the matrix factorisation models. This negative sampling method benefits improving the accuracies of recommendations w.r.t. implicit behaviours, and has been widely employed in recently proposed SbORSs [37, 38].

The above mentioned SbORSs are all element-wise SbORSs, which trains recommendation models by minimising the differences between real labels (or values) and the corresponding predicted ones. Different from these element-wise SbORSs, Rendle et al. [91] proposed Bayesian Personalised Ranking (BPR) to learn user preferences via a pair-wise method for accurate recommendations. Specifically, they assumed that users have more preferences for their interacted items than those of their missed items. Based on this assumption, they first sampled tuples of <target user, interacted item, missed item>, and trained the matrix factorisation model with these tuples via the Bayesian optimisation method [101]. The training objective is that the learnt preferences of users for their interacted items are more than those for their missed items. BPR has been proved to be effective for training recommendation models, but it introduces more time complexities into the training process.

MF-SbORSs with side information

Although the matrix factorisation model has been enhanced by the above work, the inherent data sparsity issue in recommendations has not been well solved. Recently, some researchers have proposed to employ side information to address this data sparsity issue. For example, Hao et al. [72] proposed a RS with social constraints, which leverages the social network information among users to improve the recommendation accuracies. Specifically, they assumed that friends in a social network with similar historical ratings would have similar preferences for items. Based on this assumption, the work in [72] utilises the L_2 norm of the differences between the latent factors of users and their friends as regularisation terms to more effectively train matrix factorisation models. Moreover, Hao et al. [72] multiplied these regularisation terms with the similarities of target users and their corresponding friends. In this way, the friends with more similar preferences to target users will contribute more to the regularisation terms.

Further to the work in [72], along with leveraging the latent factors of friends with similar preferences as regularisation terms, Ma [71] also leveraged the latent factors of friends with dissimilar preferences and the item social relations to further improve the recommendation accuracies. Specifically, Ma assumed that the learnt latent factors for target users and latent factors for dissimilar friends should be diversified, and thus the differences between them should be maximised. Moreover, Ma leveraged the similarities of ratings among items to represent the implicit social relations among items. Then, similar to the user social relations, such implicit item social relations were also served as regularisation terms to more effectively train the matrix factorisation models.

Apart from the social constraints, other side information such as user profiles and item properties have also been incorporated by some MF-SbORSs. For example, Porteous et al. [89] proposed a Bayesian optimisation based matrix factorisation model to well leverage such side information. Specifically, they first represented the given side information with vectors, and then incorporated these vectors into the priors of latent factors of corresponding users or items. After that, they employed the Gibbs sampler to conduct the inference process for final predictions based on these side information enhanced latent factors. Later on, this method has been further improved by Park et al. [87] with a hierarchical structure, which can more sufficiently exploit the side information to better assist the recommendations.

Despite continued efforts, the MF-SbORSs are commonly restricted by their linear operations and shallow structure, and thus have difficulty accurately learning nonlinear and complex user-item relations to further improve the recommendation accuracies.

Table 2.2: The comparison of representative deep learning based SbORSs

Category	**Representative approach**	**Technique adopted or basic idea**
Without side information	AutoRec – Sedhain et al. [97]	Autoencoder
	Cheng et al. [11]	Wide & deep learning
	DMF – Xue et al. [41]	Double-wing MLP
	NeuMF – He et al. [37]	Wide & deep learning
With side information	ConvMF – Kim et al. [52]	Convolutional Neural Network
	DeepCoNN – Zheng et al. [151]	Convolutional Neural Network
	SR-RNN – Hidasi et al. [40]	Recurrent Neural Network
	SR-GNN – Wu et al. [131]	Graph Neural Network

2.1.1.2 Deep Learning based SbORSs

Compared with the MF-SbORSs, DL-SbORSs are generally more effective. This effectiveness partially comes from their non-linear operations, deep structure, and high flexibilities to adapt to different recommendation scenarios. In this subsection, we introduce DL-SbORSs with two parts: 1) DL-SbORSs without side information and 2) DL-SbORSs with side information.

DL-SbORSs without side information

The AUTOencoder based RECommender system (AutoRec) was proposed by Sedhain et al. [97] to make recommendations with an unsupervised autoencoder. Specifically, given a rating matrix, AutoRec takes rows (or columns) as the input of the user-based (or item-based) autoencoder to learn the hidden states, and then reconstructs these rows (or columns). In this way, the missing ratings are filled by the reconstructed rows (or columns). Moreover, [97] proved that the deep structure benefits AutoRec to achieve higher recommendation accuracies. Further, AutoRec has been enhanced in [132] and [34] to be more robust and effective.

Later on, the Multiple-Layer Perceptron (MLP) based SbORSs have been proposed and achieve satisfactory recommendation accuracies. For example, Cheng et al. [11] proposed a wide and deep learning based RS that employs MLP to learn the complex and non-linear features of users and items. Moreover, besides this deep MLP component, a wide and shallow component has also been employed to memorise the raw input features and transformed input features. Then, the deep component and the wide component are fused to reinforce each other for accurate recommendations.

Besides, Deep Matrix Factorisation (DMF) was proposed by Xue et al. [138] to employ a double-wing MLP structure for accurate recommendations. First, they employed two MLPs to accurately learn user preferences and item characteristics, respectively. Then, these learnt user preferences and item characteristics were fused with the function of cosine similarity for the predictions. In this way, with the more accurately learnt user preferences and item characteristics, DMF is expected to achieve higher recommendation accuracies.

Moreover, Neural Matrix Factorisation (NeuMF) was proposed by He et al. [37] to further improve the recommendation accuracies. Specifically, they first enhance the original matrix factorisation model with non-linearities by replacing the linear dot production with one fully connected neural network layer. Further, the output vector of this enhanced matrix factorisation model is fused with that of an MLP model to complement each other for better modelling the complex user-item relations, and thus deliver accurate recommendations.

DL-SbORSs with side information

Similar to MF-SbORSs, the side information has also been employed by DL-SbORSs to improve the recomemndation accuracies. Moreover, DL-SbORSs are flexibly constructed with multiple types of neural network structures, such as Covolutional Neural Network (CNN) [48, 59], Recurrent Neural Network (RNN) [81, 144] and Graph Neural Network (GNN) [95, 152], for effectively exploiting different types of side information.

For example, in the recommendation scenarios where user reviews for items are available, ConvMF [52] addresses the data sparsity issue by leveraging such review information via CNN, and thus improves the recommendation accuracies. Specifically, ConvMF first embeds user reviews for items into latent factors with CNN, then leverages these latent factors of reviews to well initialise the latent factors of the corresponding items. Finally, the PMF [94] model is employed to make recommendations based on these well-initialised item latent factors and randomly initialised user latent factors.

Further to ConvMF [52], which leverages the review information to initialise the latent factors of items only, DeepCoNN [151] also employs the review information to initialise the latent factors of the corresponding users as well. Then, these initialised latent factors for users and items are first concatenated, and then fed to a fully connected layer for predictions. Compared with ConvMF, DeepCoNN better exploits the review information to learn user preferences, and thus can achieve higher recommendation accuracies. Recently, some other CNN based SbORSs have been proposed to well leverage the review information for delivering more accurate recommendations, such as ReGS [134], HSACN [145] and WCN-MF [120].

Moreover, RNN has been widely leveraged by SbORSs to perform accurate sequential recommendations. For example, Hidasi et al. [40] proposed a session-based RS, which employs RNN to capture the sequential dependencies among the items in a session. RNN was originally proposed for learning the features embedded in sequential data, and thus suitable for learning the dependencies from the item sequences in a session for accurate session-based recommendations. Later, many other RNN enhanced RSs for session-based recommendations have been proposed, such as DREAM [141], iGRU4Rec [39], NARM [65], HRNN [90] and MCPRN [126].

Furthermore, GNN has been employed to more effectively exploit the transitions among items to further improve the accuracies of session-based recommendations [131, 95, 152, 119]. For example, SR-GNN [131] models each session as a directed subgraph, and all these subgraphs composite the complete session graph that incorporates the information of all the sessions. Then, the transitions and dependencies among items are well captured by the gated graph neural networks for accurate session-based recommendations.

2.1.2 Memory network enhanced SbORSs

The memory network [130] is an effective machine learning model, which contains a readable and writable storage. It was originally proposed in [130] and is further enhanced to an end-to-end model in [108]. The effectiveness of the memory network has been widely proved in various areas where objects or information need to be stored for further usage, such as natural language processing [58, 111], image processing [110, 136] and pattern recognization [82, 57]. Recently, researchers have also exploited the potential of the memory network for achieving accurate recommendations [153, 42, 74, 73].

For example, Huang et al. [42] devised a double-wing memory network along with a hierarchical attention mechanism to perform mention[1] recommendations in Twitter-like applications. First, they employed two memories for storing the Twitter histories of authors and candidate users, respectively. Then, they utilised a word-level encoder and a sentence-level encoder, both of which rely on the attention mechanism, to collaboratively embed the interests of authors and candidate users from their corresponding Twitter histories, respectively. Finally, they leveraged the learnt interests for mention recommendations by judging which candidate users match the authors. Later, the memory-based mention recommendations have been further improved by the work in [74]. Besides, the hashtag recommendations [73] for Twitter-like applications have also benefited from the memory network for improving recommendation accuracies.

Recently, the memory network has also been leveraged by Dong et al. [22] to effectively address the long-standing cold-start problem in recommendations. Specifically, they employed a feature-specific memory and a task-specific memory to assist initialise the latent factors of users in a personalised manner and guide more effective predictions, respectively. By leveraging the feature-specific memory, which memorises the profiles and latent factors of existing users, the latent factors of new users can be initialised based on their profiles while referring to the latent factors of existing

[1]The action of $@username$ is called a mention in Twitter-like applications.

Table 2.3: The comparison of representative ensemble learning based SbORSs

Category	**Representative approach**	**Technique adopted or basic idea**
Ensemble learning based SbORSs	Ke et al. [47]	Bagging
	Su et al. [60]	Mixture of Experts (MoE)
	Heater – Zhu et al. [154]	MoE
	MMoE_SAC – Li et al. [64]	Multi-gate MoE

users, and thus can more accurately reflect the preferences of these new users. As for the task-specific memory that memorises the fast gradients of users, it is employed for well initialising the parameters for predictions by learning from the interactions from all users, and thus perform more accurate predictions for new users.

In addition, the memory network has also been widely employed by sequence-based SbORSs [67, 103, 29, 43] to exploit the long-term item dependencies embedded in the historical interactions. For example, Gligorijevic et al. [29] proposed to first memorise the historical sessions in a memory, and then leverage an end-to-end memory network structure to exploit users' preferences towards items from these historical sessions for accurate recommendations w.r.t. the ongoing session.

2.1.3 Ensemble Learning based SbORSs

Ensemble learning [23] combines multiple individual machine learning models for achiving better learning performance.

Based on the ensemble strategies, ensemble learning can be categorised into four main groups: Bagging [2, 3], Boosting [26, 25], Stacking [24, 114] and Mixture of Experts (MoE) [78, 142]. As the work in this thesis is highly related to Bagging and MoE, we introduce these two types of ensemble learning methods and the SbORSs devised based on them in this subsection. In addition, for a clear comparison, these ensemble learning based SbORSs and their features are summarised in Table 2.3.

2.1.3.1 Bagging based SbORSs

Bagging [2, 3] aims to achieve better learning performance by averaging the results of multiple individual models, which are trained independently with the bootstrapped samples from the complete training data. In this way, the ensembled model generated by Bagging could have a lower bias than each individual model, and thus improve the learning performance.

The effectiveness of Bagging for recommendations has been verified in [47], where Ji et al. proposed a Bagging-based matrix factorisation framework to increase the recommendation accuracies. Specifically, they first resampled multiple subsets of training data from the dataset, then leveraged these subsets to train multiple matrix factorisation models, respectively, and finally averaged the predicted values of these models for recommendations.

2.1.3.2 MoE based SbORSs

Besides, MoE [142, 78] is another effective ensemble learning method that wisely fuses multiple expert models to achieve better learning performance. Generally, MoE contains 1) multiple experts where each expert is an atomic model specialising in learning from a particular type of input data, 2) a gating network to calculate the gating weights that reflect the expertise of each expert regarding the input and 3) a fusion module to fuse the outputs of multiple experts with the gating weights.

Because of its effectiveness, MoE has been employed by SbORSs for achieving higher recommendation accuracies. Among the first attempts, Su et al. [60] proposed a sequential strategy and a joint strategy for leveraging MoE to ensemble different types of collaborative filtering methods. Specifically, in the sequential strategy, the predictions of the prior collaborative filtering method are fed to the posterior one for more accurate predictions. Differently, in the joint strategy, all the collaborative filtering methods take the original interaction matrix as input and then obtain the final predictions by voting. As an early attempt, Su et al. [60] did not effectively leveraged

MoE for accurate recommendations, as they only employed different types of collaborative filtering methods as expert models but did not allow them to specialise in certain types of interactions.

More recently, some RSs [154, 64] have better exploited the potential of MoEs to further improve the recommendation accuracies. Specifically, they employ the gating network to dynamically assign different weights for different expert models when confronting different types of interactions. In this way, each expert model is trained to specialise in a certain type of interactions, and then the specialised experts are assigned higher weights for predictions w.r.t. the corresponding types of interactions. In this way, through recommending items to users by specialised expert models, the recommendation accuracies are expected to be improved.

2.1.4 Single-behaviour Offline Recommender Systems: A Summary

The research area of SbORSs is well explored from various perspectives for delivering recommendations w.r.t. a single behaviour type in the offline scenarios. Among the existing SbORSs, the most representative ones are mainly based on the matrix factorisation models or deep learning models. The MF-SbORSs tend to learn the linear relations between users and items from interactions, and later leverage these learnt relations for recommendations. Compared with the MF-SbORSs, DL-SbORSs are able to capture the complex and non-linear relations, and thus are commonly more expressive and benefit delivering more accurate recommendations. Moreover, some advanced machine learning techniques, such as ensemble learning and memory network, have been employed by some existing SbORSs to further improve the recommendation accuracies.

However, all these SbORSs are devised for single-behaviour recommendations in the offline scenarios, and thus have following two main limitations.

- SbORSs can only deal with interactions w.r.t. the primary behaviour (e.g., purchases) but ignore the widely-existing and informative auxiliary behaviours (e.g., views and add-to-carts), and thus have difficulty well dealing with the data sparsity problem.

- SbORSs are all devised for delivering recommendations in the offline scenarios only, and thus cannot deal with the pervasive data stream of user-item interactions for accurate streaming recommendations.

2.2 Single-behaviour Streaming Recommender Systems

As discussed above, conventional SbORSs are all devised for offline scenarios and cannot well deal with the continuous data stream. Therefore, SbSRSs are proposed to deliver accurate recommendations in streaming scenarios. Based on their development stages, SbSRSs can be roughly categorised into two groups: 1) adaption-based SbSRSs and 2)stream-oriented SbSRSs. Specifically, adaptation-based SbSRSs aim to adapt the existing offline RSs to streaming scenarios by enhancing recommendation models with incremental update mechanisms, while stream-oriented SbSRSs are specifically devised for delivering accurate recommendations in the streaming scenarios. For a more detailed introduction, the adaptation-based SbSRSs are reviewed in Subsection 2.2.1 with a brief summary in Table 2.4, while the stream-oriented SbSRSs are reviewed in Subsection 2.2.2 with a brief summary in Table 2.5.

2.2.1 Adaptation-based SbSRSs

In the early stage, researchers tended to adapt the conventional offline RSs to streaming scenarios for streaming recommendations. These adaptation-based SRSs can be categorised into two groups: 1) adaptation of neighbourhood-based collaborative filtering models and 2) adaptation of matrix factorisation models.

Table 2.4: The comparison of adaptation-based SbSRSs

Category	Representative approach	Technique adopted or basic idea
Adaptation of neighbourhood-based collaborative filtering models	ICF – Papagelis et al. [86]	Incremental updates of user-to-user similarities
	VBMF – Silva et al. [98]	Bayesian optimisation & approximation algorithm
	STREAMREC – Subbian et al. [107]	Approximation algorithm
	OCF-DR – Li et al. [66]	Dynamic regularisation
	StreamRec – Chandramouli et al. [5]	Stream processing system
	TecnetRec – Huang et al. [44]	Stream processing system
Adaptation of matrix factorisation models	RKMF – Rendle et al. [92]	Regularised kernel
	RMFK – Diaz-Aviles et al. [16]	Reservoir technique
	ISGD – Vinagre et al. [116]	Stochastic gradient descent
	Devooght et al. [15]	Randomised block coordinate descent
	eAls – He et al. [38]	Alternative least square & popularity weight

2.2.1.1 Adaptation of neighbourhood-based Collaborative Filtering Models

Among the first attempts in the streaming recommendations, Papagelis et al. [86] proposed an Incremental Collaborative Filtering (ICF) method to conduct efficient streaming recommendations based on neighbourhood-based collaborative filtering. Specifically, ICF updates the user-to-user matrix with an incremental strategy, which avoids repetitive computations in the training process, and thus achieves efficient recommendations w.r.t. continuous data stream of user-item interactions. Moreover, this incremental update process does not rely on any approximation method, and thus does not lead to any degradation of recommendation accuracies.

Later on, the approximation method has been employed by Silva et al. [98] to increase the efficiency of streaming recommendations. Specifically, Silva et al. first leveraged a statistical model to sample a subset of informative user-item interactions as the training data. Then, they integrated the Bayesian inference method into the matrix factorisation model to increase the recommendation accuracies in the streaming scenarios. Finally, they trained this Bayesian inference based matrix factorisation model with the earlier sampled data via a fast and near-optimal method. Due to the approximation strategy, this SRS is computationally efficient, but this efficiency comes at a price of degradation of recommendation accuracy.

Further to [98], the approximation strategy has been further leveraged by Subbian et al. [107] for improving the recommendation efficiency of adapted neighbourhood-based collaborative filtering in the streaming scenarios. Specifically, Subbian et al. dynamically calculated the similarities between items with an approximate algorithm, and thus the computational complexity can be significantly reduced. Moreover, they deduced the theoretical bounds of the approximation error, which indicates the degradation of recommendation accuracy is controllable and acceptable.

Besides, neighbourhood-based collaborative filtering has also been enhanced by Li et al. [66] with dynamic regularisation for streaming recommendations. Specifically, they dynamically calculated the statistics about the users and items (e.g., the average rating of each user) and then utilised these statistics to regularise the predictions for more personalised recommendations. For example, the predicted rating will be adjusted to be lower for a user with a lower average rating for consistency with his/her usual practice.

For higher computational efficiency, Chandramouli et al. proposed StreamRec [5] to conduct neighbourhood-based collaborative filtering with a stream processing system [105] (e.g., Storm [115]) and thus delivering more effective streaming recommendations with efficient stream processing. Specifically, StreamRec first decomposes collaborative filtering into several native incremental streaming operators, and then deploys these operators in the stream processing system for delivering efficient recommendations in the streaming scenarios.

Further to [5], Huang et al. [44] proposed a general framework, named TecentRec, built on the stream processing system. To be more specific, TecentRec aims to conveniently integrate the offline RSs for streaming recommendations with high computational efficiency. Some representative recommendation models have already been deployed in TecentRec (e.g., neighbourhood-based collaborative filtering and demographic-based models). Moreover, this framework has been deployed in real-world applications and well served 10 billion user requests per day. Stream processing systems are highly computationally efficient for dealing with data stream, and thus are promising to be leveraged for streaming recommendations.

2.2.1.2 Adaptation of Matrix Factorisation Models

Apart from these neighbourhood-based collaborative filtering approaches, the matrix factorisation models, which are commonly more effective, have also been adapted to streaming scenarios for streaming recommendations. For example, Regularised Kernel Matrix Factorisation (RKMF) was proposed by Rendle et al. [92] to endow the matrix factorisation model the ability of dynamical updates in the streaming scenarios. Specifically, in RKMF, a regularised kernel is devised, which can not only dynamically update the matrix factorisation model without retraining this model from scratch, but also benefits capturing the non-linear relations between users and items with its non-linear structure.

In order to further improve the recommendation accuracies in streaming scenarios, Streaming Ranking Matrix Factorisation [16] was proposed to train the matrix factorisation model with newly receiving data along with the sampled historical data from the reservoir. Here, the reservoir is a set of representative samples of the historical data, which is partially used for training the matrix factorisation model. Note that this reservoir technique is useful and inspiring, and is later employed and enhanced by some more advanced SRSs [127, 32].

Later on, Stochastic Gradient Descent (SGD) [1] was employed by Vinagre et al. [116] to train the matrix factorisation model in the streaming scenarios. Through stochastic gradient descent, the matrix factorisation model can be updated once a user-item interaction is received, and thus can make accurate real-time recommendations. Note that, SGD is a generalised training technique for machine learning methods, and is leveraged by some recently proposed SRSs [140].

In addition, missing interactions (i.e., user-item pairs without any observed interaction) have been employed by Devooght et al. [15] to train the matrix factorisation model in the streaming scenarios. Specifically, they first assigned missing interactions a fixed and small value (e.g. 0), and then employed Randomised Block Coordinate (RCD) to effectively train the matrix factorisation model with these missing interactions and observed interactions for streaming recommendations. Actually, the practise of employing missing interactions for training streaming recommendation models significantly benefits improving the recommendation accuracies, and is widely employed in the SbSRSs [38, 32, 140] proposed recently.

Different from the work in [15], where missing interactions are all assigned to the same weight for training recommendation models, He et al. [38] weighted these missing interactions based on the popularities of their incorporated items to more effectively train the recommendation models. For example, a popular item that is purchased by many users might not match Alice's special preferences if Alice has not purchased this popular item, and thus this unobserved interaction should be assigned larger weights to more accurately learn the special preferences of Alice. Then, He et al. [38] proposed an element-wise Alternating Least Squares (eAls) algorithm to efficiently and effectively train recommendation models while accounting for these weighted missing interactions.

2.2.2 Stream-oriented SbSRSs

Recently, Stream-oriented SbSRSs (So-SbSRSs) have been specifically proposed for addressing the challenges of streaming recommendations. Based on the recommendation scenarios, So-SbSRSs can be roughly categorised into three groups: 1) interactio-based So-SbSRSs, 2) session-based So-SbSRSs and 3) other So-SbSRSs.

Table 2.5: The comparison of stream-oriented SbSRSs

Category	**Representative approach**	**Technique adopted or basic idea**
interaction-based	sRec – Chang et al. [6]	Random process & Brownian motion
	streamGBMF – Chen et al. [8]	Forgetting mechanism
	SPMF – et al. [127]	Ranking-based sampling & reservoir technique
	NMRN – Wang et al. [121]	Key-value memory network
	OCFIF – Yin et al. [140]	Ensembling multiple matrix factorisation models
	Vinagre et al. [117]	Online bagging
Session-based	SSRM – Guo et al. [32]	Attention mechanism & reservoir technique
	SANSR – Sun et al. [109]	Self-attention network
	MAN – Mi et al. [80]	Memory network
Other	TRM – Yin et al. [139]	POI recommendation & topic model
	SIMF – Jakomin et al. [45]	Side information & relational learning

2.2.2.1 Interaction-based So-SbSRSs

Among various types of So-SbSRSs, the interaction-based ones are the most common and relatively well explored. These So-SbSRSs make recommendations based on data streams of user-item interactions only. For example, Targeting the preference drift issue, Change et al. [6] proposed a random process to track the evolution of user preferences. Specifically, they first assumed that preferences of new users followed the general preferences of the existing users, and then utilised the Brownian motion to track the evolution of such preferences. It should be noted that using a random process to capture user preferences and address the preference drift issue might be a promising research direction, as it is explainable and computationally efficient.

Then, this preference drift issue has also been addressed by Chen et al. [8] with two forgetting mechanisms. Specifically, the first forgetting mechanism discards the historical user-item interactions that do not match the users' latest preferences embedded in the recent interactions. However, this consistency is measured by the difference between the predicted rating and the real rating, which might not be accurate as such differences might also come from the low prediction accuracy of the recommendation model. As for the second forgetting mechanism, it is devised based on the time-decay confidence, which assumes that more recent interactions should be assigned larger weights for training the recommendation models. This time-decay method is reasonable, as more recent interactions commonly better reflect users' latest preferences, and thus contribute more to accurate streaming recommendations.

Later on, the reservoir technique leveraged in [16] was employed by Stream-centered Probabilistic Matrix Factorisation (SPMF) [127] to capture long-term user preferences embedded in the historical interactions. Moreover, a ranking-based sampling method was proposed to deal with overload scenarios, where the data receiving speed is higher than the data processing speed. Furthermore, SPMF also deals with the preference drift problem with a probabilistic Gaussian classification model. However, the proposed ranking-based sampling method needs to first make predictions for all the historical interactions, and then sample the ones with the largest prediction errors. Although this sampling method can provide informative interactions to more effectively train the recommendation models, it is highly computationally expensive, and thus might not be suitable for the streaming scenarios. Moreover, the proposed SPMF relies on a linear operation (i.e., the dot product) to learn from the interactions, and thus might have difficulty well capturing the non-linear user preferences and item characteristics.

Additionally, the memory network has also been employed to learn both short-term user preferences and long-term user preferences. Specifically, Wang et al. [121] devised a key-value memory network, where the key memory slot memorises the user-specific weights of the latent factors memorised in the value memory. The weights remain unchanged for each user, while the memorised latent factors evolve to track both short-term user preferences and long-term user preferences. This memory-based streaming recommendation model has low computational complexity; however, the short-term user preferences and long-term user preferences can be hardly well learnt simultaneously within this single memory network structure. This is because, in some cases, these two types of preferences are inconsistent and might affect the learning processes of each other.

Moreover, ensemble learning has also been employed by the stream-oriented Sb-SRSs to further improve the recommendation accuracies in the streaming scenarios. For example, Yin et al. [140] proposed to leverage ensemble learning to address the limitations of a single SRS. Specifically, they first trained multiple matrix factorisation models of different parameters with the same training data, and then selected one model only for the final recommendations. Ensemble learning is an effective machine learning method and should be well studied for streaming recommendations. However, this approach in [140] feeds the same input to all the individual models while selects only one recommendation model for the final prediction. Therefore, it might not sufficiently exploit the potential of ensemble learning to effectively leverage all recommendation models for accurate streaming recommendations.

Further to [140], Vinagre et al. [117] proposed to leverage online bagging to more effectively ensemble multiple individual factorisation models for accurate streaming recommendations. Specifically, they first trained each individual factorisation model incrementally with the same input, and then averaged their predictions for the final prediction. In this way, all the individual models contribute to the final prediction, and thus can reduce the bias of each individual model for improving recommendation accuracies. However, as this model feeds all the individual models with the same input, it does not fully exploit the potential of ensemble learning to confront the high-speed and continuous data stream.

2.2.2.2 Session-based So-SbSRSs

Recently, session-based So-SbSRSs have been proposed. Compared with the interaction-based So-SbSRSs, these session-based ones additionally take the session information to improve the recommendation accuracies. For example, Guo et al. [32] proposed an attention-based matrix factorisation model along with the reservoir technique to deliver accurate session-based recommendations in the streaming scenarios. First, they employed the reservoir technique to keep a set of representative historical sessions, which are later sampled with an elaborately devised sampler for training the recommendation model along with the newly receiving sessions. Then, they enhanced the matrix factorisation model with an attention technique to capture both long-term and short-term user preferences for more accurate session-based streaming recommendations.

Later on, Sun et al. [109] employed the self-attention network for more accurate session-based recommendations in the streaming scenarios. Specifically, this proposed self-attention network predicts the next item by first calculating the similarities between the target item and the existing items in the ongoing session, then embedding the ongoing session into a unified vector with these similarities, and finally comparing this unified vector with the embedding of the target item for the prediction. Although the importance of each item within the ongoing session is measured by the calculated similarities, the sequential dependencies among items have not been well considered for higher session-based recommendations in the streaming scenarios.

Further to the work in [32, 109], Mi et al. [80] explored the potential of memory networks in session-based streaming recommendations. Specifically, in [80], they proposed a Memory Augmented Neural model (MAN) for accurate session-based recommendations in the streaming scenarios. The effectiveness of MAN mainly lies in its memory network component, which can effectively and efficiently capture long-term user preferences embedded in historical sessions stored in this memory, and thus benefits delivering accurate session-based recommendations in streaming scenarios.

2.2.2.3 Other So-SbSRSs

Besides, Point-of-Interest (POI) recommendations in streaming scenarios are studied by Yin et al. [139]. Specifically, they first proposed a topic-region model based on the probabilistic theory to learn from the users' check-in activities, and then leveraged the learnt knowledge for POI recommendations. Moreover, this topic-region model was further enhanced with the online learning method to track the evolution of user preferences and accelerate the training process. The POI recommendations in streaming scenarios remain largely unexplored, and might be a potential research topic.

Moreover, targeting the inherent data sparsity issue in streaming recommendations, Jakomin et al. [45] proposed to leverage side information, such as user profiles and item properties, via relational learning to better learn user preferences and item characteristics. Specifically, in addition to data streams of user-item interactions, they also analysed the user profiles and item properties, and thus could better learn user preferences and item characteristics with such side information. This practice of considering side information to improve the recommendation accuracies in streaming scenarios is promising and less explored, and thus should attract more research attention.

2.2.3 Single-behaviour Streaming Recommender Systems: A Summary

As discussed above, the conventional offline RSs cannot deal with the ubiquitous data stream, and SbSRSs have been proposed to address this issue by training recommendation models incrementally with continuous user-item interactions w.r.t. a single behaviour type. Based on their development stages, the SbSRSs can be roughly categorised into two groups: adaptation-based SbSRSs and stream-oriented SbSRSs.

Adaptation-based SbSRSs aim to adapt existing SbORSs to streaming scenarios for streaming recommendations. Specifically, they commonly enhance SbORSs with the incremental update mechanisms for well dealing with the continuous data stream. However, these adaptation-based SbSRSs mainly focus on dealing with the continuous user-item interactions for streaming recommendations, but pay less attention to improving the accuracies of streaming recommendations.

By contrast, stream-oriented SbSRSs have been proposed specifically for delivering accurate streaming recommendations. Compared with the adaptation-based SbSRSs, stream-oriented SbSRSs focus more on addressing the challenges of streaming recommendations to improve recommendation accuracies in the streaming scenarios.

Although efforts have been made, the challenge of well learning both short-term and long-term user preferences still needs to be well addressed. Moreover, the existing SbSRSs are all devised for dealing with data streams of single-behaviour interactions, and thus might have difficulty well dealing with the data sparsity issue.

2.3 Multi-behaviour Offline Recommender Systems

The above reviewed SbORSs and SbSRSs are both devised to deal with interactions w.r.t. a single behaviour type, and suffer from the data sparsity issue caused by the limited number of such single-behaviour interactions. Therefore, MbORSs [18, 77] have been proposed recently to improve recommendation accuracies by leveraging multiple types of user behaviours (e.g., purchases, add-to-carts and views) in the offline scenarios. Based on the maximal number of behaviours could be incorporated, the MbORSs can be categorised into two groups: 2-behaviour ORSs [85, 18, 20] and n-behaviour ($n > 2$) ORSs [21, 113, 7]. Specifically, 2-behaviour ORSs are devised to deal with the interactions w.r.t. two behaviour types, i.e., the primary behaviour and one type of auxiliary behaviour, while n-behaviour ORSs are more generalised and able to incorporate an arbitrary number types of auxiliary behaviours. For a clear introduction, the 2-behaviour ORSs are reviewed in Subsection 2.3.1 with a brief summary in Table 2.6, while the n-behaviour ORSs are reviewed in Subsection 2.3.2 with a brief summary in Table 2.7.

Table 2.6: The comparison of 2-behaviour ORSs

Category	**Representative approach**	**Technique adopted or basic idea**
Pair-wise	Ding et al. [17]	Bayesian optimisation & personalised ranking
	BPR+view – Ding et al. [18]	Bayesian optimisation & personalised ranking
	ABPR – Pan et al. [85]	Bayesian optimisation & personalised ranking
	VALS – Ding et al. [20]	Alternative least square & max-margin learning
Element-wise	SVD++ – Koren et al. [54]	Regularised matrix factorisation
	TimeSVD++ – Koren et al. [55]	Regularised matrix factorisation & temporal pattern capture
	MGNN-SPred – Wang et al. [128]	Gated graph neural network

2.3.1 2-behaviour Offline Recommender Systems

2-behaviour ORSs [85, 17, 18, 20, 128] focus on improving the recommendations w.r.t. the primary behaviour with the additional interactions w.r.t. one type of auxiliary behaviour. In this subsection, the 2-behaviour ORSs are introduced with two parts: 1) pair-wise 2-behaviour ORSs and 2) element-wise 2-behaviour ORSs.

2.3.1.1 Pair-wise 2-behaviour ORSs

Most 2-behaviour ORSs [17, 18, 85, 20] have adopted the pair-wise ranking strategy to endow the primary behaviour higher priority over the auxiliary one. Here, the priority of a certain behaviour type indicates how much it reflects the user preferences for items (e.g., the primary behaviour usually has the highest priorities). Note that, the unobserved behaviour, which serves as a virtual behaviour between a user and a missed item by this user, has also been considered with lowest priority by most pair-wise 2-behaviour ORSs [85, 17, 21, 68] to help more effectively train the recommendation models.

For example, to leverage the view behaviour to assist purchase recommendations, Ding et al. [17] proposed an improved sampler for BPR by leveraging the interactions w.r.t. both the purchase behaviour and the view behaviour. Specifically, this view-enhanced sampler samples three types of tuples: <target user, purchased item, viewed item>, <target user, purchased item, missed item> and <target user, viewed item, missed item>. After that, each of these three types of tuples is trained with the BPR method [91] independently for learning the relations between users and items. However, this work might not well learn the overall semantics of the view behaviour, as it trains the tuples of <target user, purchased item, viewed item> and <target user, viewed item, missed item> independently.

Later on, Ding et al. extends the work in [17] to view-enhanced BPR [18] for addressing the limitation mentioned above. Specifically, this view-enhanced BPR adopts a joint training method to better learn from both purchase behaviours and view behaviours. This joint training method incorporates the partial priority order among behaviours, i.e., purchase $>$ view $>$ the unobserved, with a unified loss function, and thus benefits effectively learn user preferences from both the purchase behaviour and view behaviour. Although efforts have been made, this approach relies on linear operations to model the relations between users and items, and thus might not well capture the complex and non-linear user-item relations from the interactions.

Different from the above two studies, Adaptive Bayesian Personalized Ranking (ABPR) [85] focuses more on the uncertainty of the auxiliary behaviour. That is, compared with the primary behaviour, the auxiliary behaviour does not explicitly reflect the preferences of a user for an item, and thus is more difficult to be interpreted. To address this issue, ABPR adopts a margin-based method to adaptively calculate the confidence of auxiliary behaviours for reflecting the user preferences. Then, this confidence is integrated into the loss function for more effectively training the recommendation model along with the BPR method. Although this confidence benefits more accurately interpreting the auxiliary behaviours, it might not be precise in the cases when prediction accuracy is low. This is because the confidence is simply calculated based on the difference between the predicted probability of an interaction and its real label.

In addition to the above mentioned BPR-based methods, the pair-wise strategy was adopted along with the ALS method [41]. For example, View-enhanced Alternative Least Square (VALS) [20] improves eALS [38] to conduct recommendations w.r.t. both the primary behaviour and the auxiliary behaviour. Specifically, VALS first devises a view-enhanced objective function, which leverages a margin-based formulation to incorporate all the interactions w.r.t. the purchase behaviour, the view behaviour, and the unobserved behaviour. Different from the BPR-based 2-behaviour ORSs, which sample limited unobserved behaviours for training the recommendation models, VALS assumes that all the user-item pairs without interactions are with the unobserved behaviours. Although this practice utilises all possible user-item pairs for learning the relations between users and items, it introduces substantial computational cost. To address this issue, VALS adopts a fast learning algorithm, which accelerates its training process by avoiding repetitive calculations when updating the parameters of the recommendation model. However, in essence, VALS is a regression method, and thus might not be well suited to delivering purchase recommendations, which need to accurately identify the users' purchase behaviour from view behaviour and the unobserved one. Moreover, similar to the limitations of BPR-based MbORSs, VALS also relies on linear operations to learn from the multi-behaviour interactions, and thus has limited abilities to model complex and non-linear user-item relations.

2.3.1.2 Element-wise 2-behaviour ORSs

Apart from these pair-wise approaches, element-wise 2-behaviour ORSs have also been proposed. For example, Koren et al. [54] incorporated both explicit behaviour and implicit behaviour for recommendations via an element-wise manner. Specifically, they improved the traditional matrix factorisation model by leveraging the interactions w.r.t. implicit behaviour as a regularisation term to assist the prediction of the explicit rating. Later, this method was further extended in [55], where the temporal patterns of user preferences were considered and captured for more accurate recommendations. Although these two recommendation models are computationally efficient, they might not well explore the user preferences and item characteristics embedded in the implicit behaviour, as these two models both treat the implicit behaviour as a regularisation term only.

Besides, GNN has also been employed by MGNN-SPred [128] to well learn the user-item relations from 2-behaviour interactions. Specifically, in this graph neural network, two types of edges are devised for representing the primary behaviour and one type of auxiliary behaviour, respectively. Then, the sequence representation w.r.t. a certain behaviour type is achieved by sum-pooling over all the items within this sequence. After that, a gating mechanism is devised to merge the sequence representations of the primary behaviour and the auxiliary behaviour for reflecting user preferences. Although the item-item relations have been learnt by MGNN-SPred, the importance of the last item for sequence representation might not be sufficiently highlighted with the sum-pooling operation, which might affect the recommendation accuracies.

2.3.2 n-behaviour Offline Recommender Systems

Different from the 2-behaviour ORSs, n-behaviour ($n > 2$) ORSs have been devised to deal with an arbitrary number of types of auxiliary behaviours in the offline scenarios, and thus are more flexible. In this subsection, we introduce n-behaviour ORSs with two parts, i.e., 1) pair-wise n-behaviour ORSs and 2) element-wise n-behaviour ORSs.

2.3.2.1 Pair-wise n-behaviour ORSs

Some techniques in 2-behaviour ORSs have been enhanced by n-behaviour ORSs to well deal with more types of auxiliary behaviours. For example, the BPR method was enhanced by MFPR [68] to incorporate one type of explicit behaviour (primary behaviour), such as ratings, and multiple types of implicit behaviours (auxiliary behaviours), such as purchases and clicks. Specifically, similar to the work in [85, 18],

Table 2.7: The comparison of n-behaviour ORSs

Category		**Representative approach**	**Technique adopted or basic idea**
Pair-wise		MFPR – Liu et al. [68]	Bayesian optimisation & personalised ranking
		AALS – Ding et al. [21]	Alternative least square & joint training
Element-wise	Non-sequential recommendation	Liang et al. [112]	Ensemble learning
		CMF – Singh et al. [100]	Collective matrix factorisaion
		DCMF – Mariappan et al. [77]	Collective matrix factorisaion & deep learning
		MATN – Xia et al. [133]	Memory network & transformer
		M-MLP – Wen et al. [129]	Fusing MLP and matrix factorisation
		NMTR – Gao et al. [28]	Multi-task learning cascaded prediction
		EHCF – Chen et al. [7]	Transfer-learning
	Sequential recommendation	ASLI – Mehrab et al. [113]	Ensemble learning
		DPIN – Guo et al. [31]	Bi-directional RNN & multi-task learning & edge computing

a generation algorithm was first proposed to generate tuples of $<$target user, item$_1$, item$_2$ $>$ with an accurate priority order between item$_1$ and item$_2$. Then, a BPR-based approach was proposed to learn user preferences from these generated tuples. Nevertheless, similar to the BPR-based 2-behaviour MbORSs [85, 18], MFPR relies on linear operations to learn from the multi-behaviour interactions, and might have limited ability to capture the complex and non-linear user-item relations.

Apart from BPR-based approaches, the 2-behaviour ORS VALS [20] has also been improved into AALS [21] for conducting recommendations w.r.t. an arbitrary number of types of auxiliary behaviours. The main improvement of AALS over the VALS lies in its loss function, which is able to incorporate more behaviour types. Specifically, the multiple behaviour types are first sorted by their priorities, where the purchase is usually with the highest and the unobserved is with the lowest. Then, the differences between the predicted values of all adjacent behaviour types are summed up to composite the loss function for learning the user-item relations from multi-behaviour interactions. However, this loss function can only reflect the local partial priorities of the adjacent behaviour types only, which might not be able to well learn the global partial priorities of all behaviour types.

2.3.2.2 Element-wise n-behaviour ORSs

Besides the above pair-wise n-behaviour ORSs, element-wise n-behaviour ORSs have also been proposed. In this subsection, we introduce n-behaviour ORSs with two parts: 1) non-sequential recommendations and 2) sequential recommendations.

Non-sequential recommendations

Liang et al. [112] provided an empirical study where three training strategies, including Model Combination (MC), Prior Combination (PC) and Constrained Regression (CR), are devised to simultaneously exploit interactions w.r.t. multi-behaviour interactions. Specifically, 1) MC trains multiple individual recommendation models independently with each model focusing on a specific behaviour type, and makes final predictions based on a weighted sum of the predictions of all these individual models; 2) PC trains multiple individual recommendation models sequentially where the parameters of a prior model are utilised to regularise the parameters of the posterior one, and the recommendations are made based on the last recommendation model; and 3) the CR utilises only one recommendation model, where the training loss considers the interactions w.r.t. all behaviour types. Although three strategies have been proposed, they are still in the early stage of multi-behaviour recommendations, which rely on the simple combination of individual models or naive joint training loss to exploit multi-behaviour interactions, and could be further improved with more advanced techniques.

Besides, Collective Matrix Factorisation (CMF) originally proposed in [100] attempts to factorise multiple user-item interaction matrixes simultaneously, e.g., the interaction matrix w.r.t. the primary behaviour and interaction matrixes w.r.t. auxiliary behaviours. Thus, CMF is able to incorporate multiple behaviour types for recommendations. Moreover, CMF allows the user embeddings (or item embeddings) to be different w.r.t. different behaviour types, and thus achieve more flexibilities when learning from the multi-behaviour interactions. However, it has difficulties in learning the complex and non-linear relations between users and items, as it relies on the shadow structure to learn from the multi-behaviour interactions..

Later on, Deep Collective Matrix Factorisation (DCMF) [77] enhances CMF [100] with a deep neural network structure, and thus have stronger abilities to capture the complex and non-linear user-item relations. Moreover, it employs the Bayesian optimisation method to address the optimisation challenges caused by the dependencies of different behaviour types. However, this work mainly leverages the shared user embeddings and shared item embeddings to complement the modelling of the behaviours with one another, but might not sufficiently highlight the distinctions of user preferences and item characteristics w.r.t. different behaviour types.

To better leverage multi-behaviour interactions, Memory-Augmented Transformer Network (MATN) [133] models the dependencies among behaviours with a transformer-based multi-behaviour relation encoder. Moreover, it also learns the contextual information (e.g., view behaviour appear more frequently than purchase behaviour does) for each behaviour type. Furthermore, a cross-behaviour aggregation component is devised to collaborate these behaviours for assisting recommendations. Although MATN models dependencies among different behaviour types well and considers the contextual information, it might pay insufficient attention to explicitly learning user preferences and item characteristics w.r.t. each behaviour, which might affect the recommendation accuracies.

Inspired by NeuMF [37], Wen et al. [129] proposed Multi-branch Multi-Layer Perceptron (M-MLP) to capture the non-linearity from the multi-behaviour interactions, and then made recommendations by collaborating M-MLP with a matrix factorisation module. Similar to NeuMF [37], with both M-MLP and the matrix factorisation, this recommendation model can capture both complex non-linear relations and shallow linear relations, respectively, for accurate recommendations. However, its fusion process for M-MLP is not personalised for the target user and the target item, which might affect the fusion effectiveness.

Moreover, the multi-task learning method has also been employed by Neural Multi-Task Recommendation (NMTR) [28] to exploit multi-behaviour interactions. Specifically, NMTR first leverages multiple individual recommendation models to learn from the interactions from interactions w.r.t. multiple behaviours, respectively. Then, cascaded predictions are made across these individual recommendation models, where the prediction w.r.t. the precedent behaviour is used to help make predictions w.r.t. the posterior behaviour. In this way, NMTR can capture the dependencies among different behaviour types. However, for making these cascaded predictions, the training data for NMTR must be interactions w.r.t. behaviours with strict priority order (e.g., <view, click, add-to-cart, purchase>). Nevertheless, the generating process of interactions w.r.t. such ordered behaviour types is not clearly stated in this paper, and the quality of the generated interactions might greatly affect the recommendation accuracies.

Further, Chen et al. [7] proposed a transfer-learning based approach to effectively leverage multi-behaviour interactions. Specifically, they devised a transfer-based prediction layer, where the parameters w.r.t. a behaviour type are regularised by those of other behaviour types. Transfer-learning based techniques are promising in utilising the knowledge learnt for one behaviour type to help the recommendations w.r.t. another behaviour type. Thus, researchers should conduct more research on transfer-learning based multi-behaviour RSs.

Sequential recommendations

Recently, the multi-behaviour interactions have also been employed for next-item recommendations by Attentive Sequential model of Latent Intent (ASLI) [113]. Specifically, it first employs a self-attention layer to learn the item similarities from users' interaction histories. Then, a temporal convolutional layer is leveraged to learn the behaviour representation on a particular category. Finally, the above self-attention layer and the temporal convolutional layer are fused by a fully connected layer to recommend an item to a user w.r.t. a specific behaviour type. However, ASLI learns from behaviour sequences and item sequences independently, and thus might not well model the overall relations between behaviours and items.

Besides, Deep Intent Prediction Network (DIPN) [31] incorporates both browse-interactive behaviours (e.g., purchases and views) and touch-interactive behaviours (e.g., swipes and taps) to deliver more accurate session-based recommendations. Specifically, it first learns the dependencies among items by bi-directional RNN layers from sequences w.r.t. different behaviour types, respectively. Then, a hierarchical attention layer fuses the hidden outputs of different sequences w.r.t. all behaviours. Finally, this recommendation model is trained on the server side with multi-task learning, and is then deployed on the client side for recommendations. DIPN is elaborately devised; however, it can achieve the best performance only with an edge computing platform where the RS is trained on the server side and deployed on the client side. This feature restricts its generalisation. Moreover, as this model utilises touch-interactive behaviours, it might not be suitable for performing recommendations when only browse-interactive behaviours are available (e.g., online shopping with browsers on PC).

2.3.3 Multi-behaviour Offline Recommender Systems: A Summary

Targeting the data sparsity issue caused by the limited interactions w.r.t. the single behaviour type, MbORSs are proposed to incorporate multiple behaviour types for improving the recommendation accuracies in offline scenarios. Based on the maximal number of behaviours that could be incorporated, MbORSs can be categorised into two groups: 2-behaviour ORSs that are devised for leveraging only one type of auxiliary behaviour, and n-behaviour ORSs that could incorporate an arbitrary number of types of auxiliary behaviours.

Most 2-behaviour ORSs adopt the pair-wise strategy to distinguish the priority of the primary behaviour and that of the auxiliary behaviour. Specifically, they commonly first generate tuples containing the target user and two items with an accurate priority order w.r.t. behaviour types, and then employ BPR based or ALS based approaches to train these generated tuples.

In contrast, n-behaviour ORSs can incorporate an arbitrary number of behaviour types, and thus are more flexible and promising. To achieve this goal, n-behaviour ORSs employ advanced machine learning methods, such as multi-task learning, transfer learning, deep learning, temporal convolutional network, and memory network.

Although efforts have been made, it is still a challenging task to accurately learn both the shared user preferences (or item characteristics) and behaviour-specific user preferences (or item characteristics) to further improve the recommendation accuracies. More importantly, the existing MbSRSs are all devised for the offline scenarios, and thus have difficulty dealing with the widely-existing data stream of interactions w.r.t. multiple behaviour types.

2.4 Chapter Summary

In this chapter, we review the related literature on SbORSs, SbSRSs and MbORSs. The SbORSs are well studied and aim to deliver recommendations w.r.t. a single behaviour type in the offline scenarios. However, they cannot either well deal with the pervasive data stream of user-item interactions for streaming recommendations or well leverage the multiple behaviour types to address data sparsity issue for improving the recommendation accuracies. Later on, SbSRSs and MbORSs are proposed to well deal with the widely-existing data stream and exploit interactions w.r.t. multiple behaviour types, respectively.

In regard to existing SbSRSs, their main categories, characteristics, advantags and disadvantages, are summarised as follows:

- **Categories and characteristics.**
  - **Adaptation-based SbSRSs.** Adaptation-based SbSRSs adapt the existing offline RSs to streaming scenarios for streaming recommendations. However, they mainly focus on how to perform recommendations in streaming scenarios but pay less attention to addressing the challenges of streaming recommendations for higher recommendation accuracies.
  - **Stream-oriented SbSRSs.** Stream-oriented SbSRSs target the challenges of streaming recommendations to further improve the recommendation accuracies in streaming scenarios. However, they are devised to deal with a single behaviour type, and thus suffer from the data sparsity issue.
- **Advantages.** These SbSRSs are able to well deal with the pervasive data stream of single-behaviour interactions by updating recommendation models in a timely manner. Thus, they are able to make recommendations in streaming scenarios based on users' latest preferences.
- **Disadvantages.** These SbSRSs cannot well exploit the widely-existing multi-behaviour interactions to address the data sparsity issue in the streaming scenarios, and thus have difficulty delivering accurate streaming recommendations.

Besides, in regard to existing MbORSs, their main categories, characteristics, advantages and disadvantages, are summarised as follows:

- **Categories and characteristics.**
  - **2-behaviour ORSs.** 2-behaviour ORSs aim to incorporate one type of auxiliary behaviour to assist the recommendations w.r.t. the primary behaviour. However, their inabilities of dealing with more types of auxiliary behaviours restrict their generalisations and performance.
  - **n-behaviour ORSs.** n-beahvior RSs are able to incorporate an arbitrary number of types of auxiliary behaviours, and thus are more flexible. In

addition, existing n-behaviour ORSs are all devised for delivering recommendations in offline scenarios only.

- **Advantages.** These MbORSs are able to address the long-standing data sparsity issue by well incorporating interactions w.r.t. auxiliary behaviours.

- **Disadvantages.** These MbORSs are all devised for the recommendations in the offline scenarios, and thus cannot deal with the pervasive data stream of user-item interactions for streaming recommendations.

Chapter 3

Double-Wing Mixture of Experts for Streaming Recommendations

To well deal with data streams of user-item interactions for delivering accurate streaming recommendations, recent years have seen an emerging trend where the newly coming interaction data are leveraged to train recommendation models in a timely manner. The RSs falling in this new trend are commonly referred to as Streaming Recommender Systems (SRSs). These SRSs train recommendation models with newly coming interaction data, and thus can capture the latest preferences of users embedded in their recent interactions for delivering accurate streaming recommendations. Note that the SRSs in this chapter all refer to the Single-behaviour SRSs, as multi-behaviour SRSs have not been reported in the literature and are either not considered by the work reported in this chapter.

Although various SRSs have been proposed, the two challenges introduced in Subsection 1.2.1 and Subsection 1.2.2, respectively, still need to be well dealt with for delivering accurate streaming recommendations. Specifically, these two challenges are expressed by **CH1**: 'how to address *preference drift* while capturing *long-term user preferences*', and **CH2**: 'how to handle the *heterogeneity* of both users and items', respectively.

To address the above two challenges, this chapter proposes a novel Variational and Reservoir-enhanced Sampling based Double-Wing Mixture of Experts framework, called VRS-DWMoE, for improving the accuracies of streaming recommendations.

Specifically, VRS-DWMoE contains two key components, i.e, 1) Variational and Reservoir-enhanced Sampling (VRS), which wisely samples historical data from the reservoir (i.e., a set of representative historical data) with an adjustable sampling size to complement newly coming data, and 2) Double-Wing Mixture of Experts (DWMoE), which first learns heterogeneous user preferences and item characteristics with the training data prepared by VRS, and then utilises these learnt preferences and characteristics to perform streaming recommendations. Specifically, DWMoE contains two elaborately devised Mixture of Experts (MoEs) for learning user preferences and item characteristics, respectively. Note that MoE is an effective ensemble learning model that wisely fuses the results of multiple experts (i.e., atomic models specialising in different types of input) for better learning performance [79]. Moreover, the multiple experts in each of the aforementioned two MoEs learn the preferences (or characteristics) of different types of users (or items) where each expert specialises in one underlying type.

The remainder of this chapter is organized as follows. We first formulate our research problem of streaming recommendations. Then, we propose our VRS-DWMoE framework. After that, we present the results of the experiments that are conducted to verify the effectiveness of VRS-DWMoE. Finally, we summarise this chapter.

3.1 Problem Statement

In this section, we formulate the research problem of streaming recommendations with implicit interactions, such as the users' purchase records of items. Given the user set $\mathbf{U}$ and item set $\mathbf{V}$, we use $y_{u,v}$ to denote an interaction between user $u \in \mathbf{U}$ and item $v \in \mathbf{V}$. Then, the list of currently received interactions is denoted by $\mathbf{Y} = \{y^1_{u^1,v^1}, y^2_{u^2,v^2}, \dots, y^k_{u^k,v^k}, \dots\}$. Note that the interactions in $\mathbf{Y}$ are sorted based on their coming time, for example, $y^k_{u^k,v^k}$ indicates the k^{th} received interaction. In addition, as in the real-world data stream, adjacent interactions (e.g., $y^k_{u^k,v^k}$ and $y^{k+1}_{u^{k+1},v^{k+1}}$) may involve different users (e.g., user u^k is different from user u^{k+1}). With the above notations, given the target user u' and target item v', the task of SRSs can be formulated as $\hat{y}_{u',v'} = P(y_{u',v'}|\mathbf{Y})$; that is predicting the probability of an interaction between the target user and target item conditionally on the currently received interactions. Compared with conventional offline recommendations, streaming recommendations take continuous and infinite data streams (e.g., streaming clicks) as input, thus they are more challenging.

3.2 Our Proposed VRS-DWMoE Framework

In this chapter, we first formulate our research problem. After that, we propose Variational and Reservoir-enhanced Sampling based Double-Wing Mixture of Experts (VRS-DWMoE), and then introduce its two key components: VRS and DWMoE.

3.2.1 Overall Structure

To simultaneously address preference drift while capturing long-term user preferences and handle the heterogeneity of users and items, we propose VRS-DWMoE, which contains two key components: 1) Variational and Reservoir-enhanced Sampling (VRS) and 2) Double-Wing Mixture of Experts (DWMoE). Specifically, as shown in Fig. 3.1, VRS first complements the newly coming data with sampled historical data while guaranteeing the proportion of newly coming data in preparation for training. Then, with the training data prepared by VRS, DWMoE better learns heterogeneous user preferences and item characteristics with two elaborately devised Mixture of Experts (MoEs), i.e., Mixture of User Experts (MoUE) and Mixture of Item Experts (MoIE), respectively. After that, the recommendations are made by learning the similarities between the learnt user preferences and item characteristics.

3.2.2 Variational and Reservoir-enhanced Sampling

The continuous and infinite data stream makes it impractical for SRSs to train recommendation models with all the interaction data. Therefore, we propose VRS to prepare

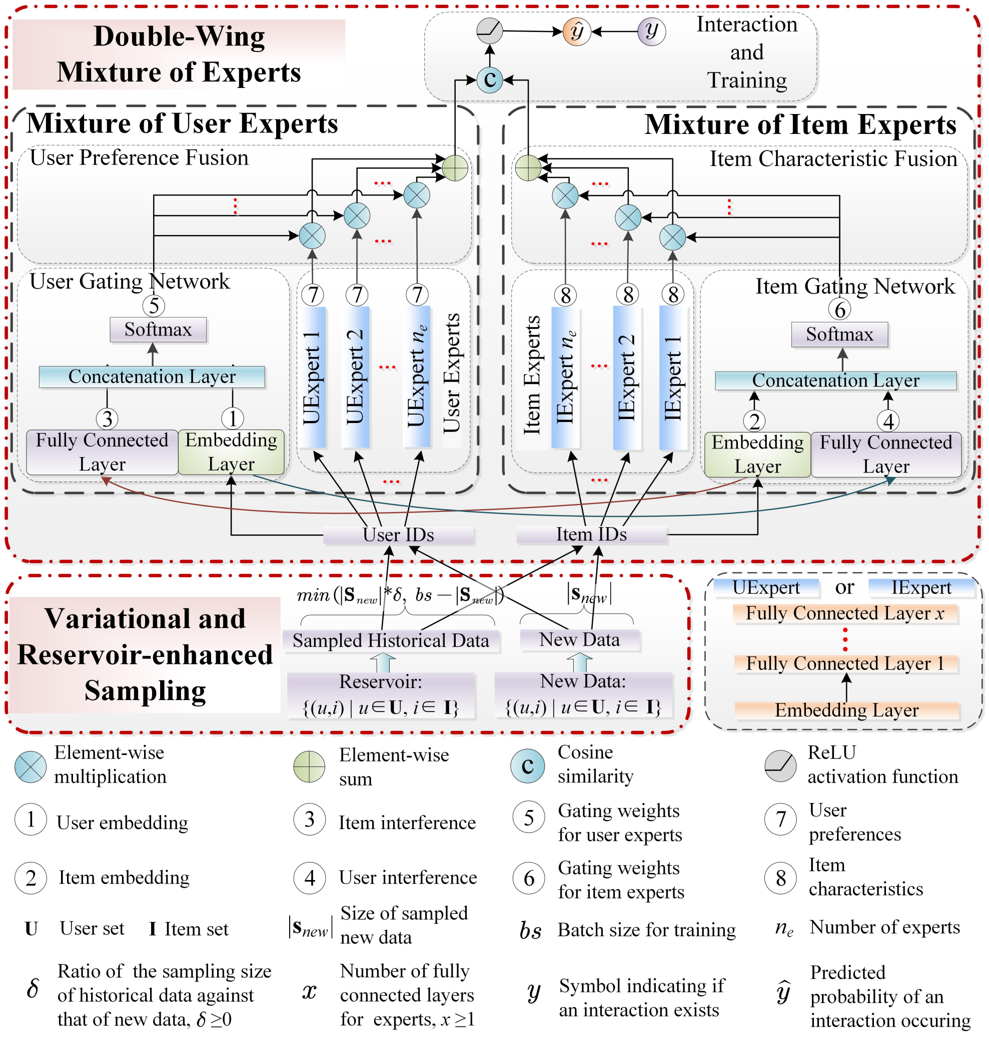

Double-Wing Mixture of Experts
Interaction and Training
Mixture of User Experts
User Preference Fusion
User Gating Network
Softmax
Concatenation Layer
Fully Connected Layer
Embedding Layer
UExpert 1
UExpert 2
UExpert n_e
User Experts
Mixture of Item Experts
Item Characteristic Fusion
Item Gating Network
Item Experts
IExpert n_e
IExpert 2
IExpert 1
User IDs
Item IDs
Variational and Reservoir-enhanced Sampling
$min(|\mathbf{S}_{new}|*\delta,\ bs-|\mathbf{S}_{new}|)$
$|\mathbf{S}_{new}|$
Sampled Historical Data
New Data
Reservoir: $\{(u,i) \mid u \in \mathbf{U}, i \in \mathbf{I}\}$
New Data: $\{(u,i) \mid u \in \mathbf{U}, i \in \mathbf{I}\}$
UExpert or IExpert
Fully Connected Layer x
Fully Connected Layer 1
Embedding Layer
Element-wise multiplication
Element-wise sum
Cosine similarity
ReLU activation function
1 User embedding
3 Item interference
5 Gating weights for user experts
7 User preferences
2 Item embedding
4 User interference
6 Gating weights for item experts
8 Item characteristics
U User set
I Item set
$|\mathbf{S}_{new}|$ Size of sampled new data
bs Batch size for training
n_e Number of experts
δ Ratio of the sampling size of historical data against that of new data, $\delta \geq 0$
x Number of fully connected layers for experts, $x \geq 1$
y Symbol indicating if an interaction exists
$\hat{y}$ Predicted probability of an interaction occuring

Figure 3.1: The structure of the VRS-DWMoE Framework

the training data by wisely complementing newly coming data with sampled historical data while guaranteeing the proportion of newly coming data.

Specifically, following [63, 127], we first maintain a reservoir to store a set of representative historical data. As newer data commonly reflect more recent user preferences, we put newly coming data into the reservoir and discard the oldest data when the reservoir runs out of space. With this reservoir and newly coming data, VRS generates the training data with two different strategies in two typical scenarios for streaming recommendations: 1) the *underload* scenarios where the data receiving speed is lower than the data processing speed, and 2) the *overload* scenarios where the data receiving speed is higher than the data processing speed. Note that the contribution of VRS mainly lies in underload scenarios, which are more common in the real world [51]. More details are presented below.

Underload Scenario. In underload scenarios, VRS generates training data by first sampling representative historical data $\mathbf{S}_{his}$ from the reservoir with a variational sampling size, and then merging $\mathbf{S}_{his}$ with all the newly coming data $\mathbf{N}$. Specifically, the sampling size $|\mathbf{S}_{his}|$ of the reservoir can be calculated as below,

$$|\mathbf{S}_{his}| = min(s_{new} * \delta, bs - s_{new}). \tag{3.1}$$

where bs denotes the training batch size and δ ($\delta \geq 0$) is a predetermined parameter measuring the ratio of the sampling size $|\mathbf{S}_{his}|$ of the reservoir against the size s_{new} of newly coming data. In this way, the sampling size of the reservoir can be adjusted by δ based on the characteristics of data streams. For example, δ should be set to a small value (e.g., 0.1) for data stream where user preferences frequently change to focus more on newly coming data. Then, $\mathbf{S}_{his}$ is sampled from the reservoir based on their coming time for assigning higher sampling probabilities to more recently received interactions. Specifically, taking the sampling probability p_k of the k^{th} received interaction as an example,

$$p_k = p_{k-1} * \lambda_{res}, \tag{3.2}$$

where λ_{res} $(\lambda_{res} > 1)$ denotes the decay ratio for assigning higher sampling probabilities to newer data. Assuming that the sampling probability of the earliest received interaction in the reservoir is p_1, we can calculate p_k by iteratively performing Eq. (3.2),

$$p_k = p_1 * (\lambda_{res})^{k-1}. \tag{3.3}$$

Then, with Eq. (3.3), we can obtain the normalised sampling probabilities, taking the probability of the k^{th} received interaction as an example,

$$P(k|\lambda_{res}, s_{res}) = \frac{p_k}{\sum_{i=1}^{s_{res}} p_i} = \frac{(\lambda_{res})^{k-1} * (1 - \lambda_{res})}{1 - (\lambda_{res})^{s_{res}}}, \tag{3.4}$$

where s_{res} denotes the size of the reservoir.

With the normalised sampling probabilities and the sampling size calculated in Eq. (3.1), VRS samples representative historical data $\mathbf{S}_{his}$ from the reservoir. Finally, the training data **T** is obtained by merging $\mathbf{S}_{his}$ with the newly coming data **N**.

Overload Scenario. In overload scenarios, VRS only samples $\mathbf{S}_{new}$ from newly coming data to form the training data for effectively capturing the latest user preferences. The sampling probability of the newly coming data can be calculated in a similar way to that described by Eqs. (3.2) to (3.4),

$$P(k|\lambda_{new}, s_{new}) = \frac{p_k}{\sum_{i=1}^{s_{new}} p_i} = \frac{(\lambda_{new})^{k-1} * (1 - \lambda_{new})}{1 - (\lambda_{new})^{s_{new}}}, \tag{3.5}$$

where λ_{new} and s_{new} denote the decay ratio and size, respectively, of newly coming data. With this sampling probability and setting the sampling size to the batch size bs, VRS samples $\mathbf{S}_{new}$ from the newly coming data as the training data **T**. Note that in the cases where the data receiving speed exactly equals to the data processing speed, VRS utilises the entire newly coming data to form the training data **T**.

3.2.3 Double-Wing Mixture of Experts

With the training data **T** prepared by VRS, DWMoE first utilises MoUE and MoIE to learn user preferences and item characteristics, respectively, for dealing with their intrinsic differences [122]. Moreover, each expert in MoUE (or MoIE) specialises in one underlying type of users (or items) to more effectively learn heterogeneous user preferences (or item characteristics). Then, DWMoE makes recommendations with the learnt user preferences and item characteristics.

Specifically, MoUE and MoIE share the same structure with different parameters. This structure has three key parts: 1) multiple experts, 2) a gating network and 3) a fusion module. Taking MoUE as an example, multiple experts first learn user preferences in parallel. Then, the gating network calculates the gate weights to measure the expertise of each expert regarding each input user. After that, the fusion module calculates the unified preferences for each user by fusing the preferences learnt by all experts with the gating weights. Note that we set the numbers (n_e) of experts in MoUE and MoIE the same in this work, and will study the effect of different numbers of experts in MoUE and MoIE in the future work. More details are presented below.

Expert. The experts in MoUE and MoIE learn the user preferences and item characteristics, respectively. Taking MoUE as an example, each expert first utilises an embedding layer to learn the user embedding $\mathbf{p}_u^i$, where i denotes the index of the expert. Then, the user preferences are learnt by the experts with x ($x \geq 1$) fully connected layers, taking the user preferences $\mathbf{P}_u^i$ for user u learnt by the i^{th} expert model as an example,

$$\mathbf{P}_u^i = a_{i,x}^{MoUE}(\cdots a_{i,2}^{MoUE}(\mathbf{W}_{i,2}^{MoUE} a_{i,1}^{MoUE}(\mathbf{W}_{i,1}^{MoUE}\mathbf{p}_u^i + \mathbf{b}_{i,1}^{MoUE}) + \mathbf{b}_{i,2}^{MoUE}) \cdots + b_{i,x}^{MoUE}), \tag{3.6}$$

where a_*^*, $\mathbf{W}_*^*$, and $\mathbf{b}_*^*$ denote the activation function, weight matrix, and bias vector, respectively. For example, $a_{i,x}^{MoUE}$, $\mathbf{W}_{i,x}^{MoUE}$, and $\mathbf{b}_{i,x}^{MoUE}$ denote the activation function, weight matrix and bias vector, respectively, in the x^{th} layer for the i^{th} expert in MoUE. Note that the symbols a_*^*, $\mathbf{W}_*^*$ and $\mathbf{b}_*^*$ are used in the rest of this chapter with different superscripts and subscripts to introduce the fully connected layers. In a similar way to that described by Eq. (3.6), item characteristics $\mathbf{Q}_v^j$ can be learnt by the j^{th} expert in MoIE with the item embedding $\mathbf{q}_v^j$ as input,

$$\mathbf{Q}_v^j = a_{j,x}^{MoIE}(\cdots a_{j,2}^{MoIE}(\mathbf{W}_{j,2}^{MoIE} a_{j,1}^{MoIE}(\mathbf{W}_{j,1}^{MoIE}\mathbf{q}_v^j + \mathbf{b}_{j,1}^{MoIE}) + \mathbf{b}_{j,2}^{MoIE}) \cdots + b_{j,x}^{MoIE}). \tag{3.7}$$

Gating Network. The gating networks in MoUE and MoIE calculate the gating weights measuring the expertise scales of each expert in learning the preferences of input users and characteristics of input items, respectively. To achieve this goal, taking the gating network in MoUE as an example, we first employ an embedding layer and a fully connected layer to calculate user embedding $\mathbf{p}_u^{MoUE}$ and item interference $\mathbf{I}^{MoUE}$, respectively. Note that the item interference is employed to more accurately calculate the gating weights in MoUE by considering the items interacted with the corresponding users. Specifically, the item interference can be calculated with the item embedding $\mathbf{q}_v^{MoIE}$ in MoIE,

$$\mathbf{I}^{MoUE} = a_{gate}^{MoUE}(\mathbf{W}_{inter}^{MoUE}\mathbf{q}_v^{MoIE} + \mathbf{b}_{inter}^{MoUE}). \tag{3.8}$$

Then, the user embedding and item interference are concatenated as follows,

$$\mathbf{c}^{MoUE} = \left[\mathbf{p}_u^{MoUE}; \mathbf{I}^{MoUE}\right]. \tag{3.9}$$

After that, $\mathbf{c}^{MoUE}$ is fed into a softmax layer to get the gating weights $\mathbf{g}^{MoUE}$ by the following equation,

$$\mathbf{g}^{MoUE} = softmax(\mathbf{W}_{soft}^{MoUE}\mathbf{c}^{MoUE} + \mathbf{b}_{soft}^{MoUE}). \tag{3.10}$$

Likewise, the gating weights $\mathbf{g}^{MoIE}$ for the experts in MoIE can be calculated in a similar way to that described by Eqs. (3.8) to (3.10).

Fusion Module. The user preferences (or item characteristics) learnt by multiple experts in MoUE (or MoIE) are fused to the unified ones to fully utilise the expertise of all the experts. Specifically, with the above calculated user preferences, item characteristics and their corresponding gating weights, the unified user preferences $\mathbf{P}_u^{uni}$ and unified item characteristics $\mathbf{Q}_v^{uni}$ can be calculated with the dot production, respectively,

$$\mathbf{P}_u^{uni} = [\mathbf{P}_u^1; \cdots; \mathbf{P}_u^{n_e}]^T\mathbf{g}^{MoUE}, \tag{3.11}$$

$$\mathbf{Q}_v^{uni} = [\mathbf{Q}_v^1; \cdots; \mathbf{Q}_v^{n_e}]^T\mathbf{g}^{MoIE}, \tag{3.12}$$

where n_e denotes the number of experts.

Interaction Module. To make recommendations based on unified user preferences and unified item characteristics, we first utilise cosine similarity for measuring to what degree the unified preferences match the corresponding unified characteristics,

$$cos\_sim_{<\mathbf{P}_u^{uni},\mathbf{Q}_v^{uni}>} = cosine(\mathbf{P}_u^{uni}, \mathbf{Q}_v^{uni}) = \frac{(\mathbf{P}_u^{uni})^T \mathbf{Q}_v^{uni}}{\|\mathbf{P}_u^{uni}\|_2 \|\mathbf{Q}_v^{uni}\|_2}, \tag{3.13}$$

and then we obtain the predicted probability $\hat{y}_{u,v}$ of the interaction between user u and item v by performing a nonlinear transformation of this cosine similarity,

$$\hat{y}_{u,v} = a_{out}^{predict}(\mathbf{W}_{out}^{predict} cos\_sim_{<\mathbf{P}_u^{uni},\mathbf{Q}_v^{uni}>} + \mathbf{b}_{out}^{predict}). \tag{3.14}$$

Optimisation. To learn the parameters of our proposed DWMoE, we train the model by minimising the following loss ℓoss^{total} with stochastic gradient descent,

$$\ell oss^{total} = \ell oss^{acc} + \gamma(\ell oss^{gate}), \tag{3.15}$$

where ℓoss^{acc} is the loss for the recommendation accuracies, ℓoss^{gate} is used as the regularisation term for gating weights to avoid local optimisations, and γ is the coefficient to adjust the importance of ℓoss^{gate}.

Specifically, to measure the difference between the ground truth and the prediction, we employ the cross-entropy loss as below,

$$\ell oss^{acc}(y_{u,v}, \hat{y}_{u,v}) = -(y_{u,v} log(\hat{y}_{u,v}) + (1 - y_{u,v}) log(1 - \hat{y}_{u,v})), \tag{3.16}$$

where $y_{u,v}$ is the label of the interaction between user u and item v; that is, it is 1 if this interaction exists and 0 otherwise, and $\hat{y}_{u,v}$ denotes the predicted probability for this interaction. Moreover, we introduce ℓoss^{gate} to avoid the local optimisation caused by the imbalanced utilisation of experts; for example, some experts receive large gating weights for most interactions while others always receive small gating weights. Specifically, we employ the standard derivations of gating weights in both

MoUE and MoIE to form ℓoss^{gate} as follows,

$$\begin{aligned} \ell oss^{gate} &= \ell oss^{gate}_{MoUE} + \ell oss^{gate}_{MoIE} \\ &= \sqrt{\frac{1}{n_e}\sum\nolimits_{i=1}^{n_e}(\mathbf{g}_i^{MoUE} - \bar{\mathbf{g}}^{MoUE})^2} + \sqrt{\frac{1}{n_e}\sum\nolimits_{j=1}^{n_e}(\mathbf{g}_j^{MoIE} - \bar{\mathbf{g}}^{MoIE})^2}, \end{aligned} \tag{3.17}$$

where $\mathbf{g}_i^{MoUE}$ and $\mathbf{g}_j^{MoIE}$ denote the gating weight for the i^{th} expert in MoUE and the gating weight for the j^{th} expert in MoIE, respectively, and $\bar{\mathbf{g}}^{MoUE}$ and $\bar{\mathbf{g}}^{MoIE}$ denote the average of gating weights for MoUE and MoIE, respectively. Through minimising ℓoss^{gate}, DWMoE encourages MoUE and MoIE to more effectively utilise all their experts to learn the heterogeneous user preferences and item characteristics, respectively, and thus to increase the accuracies of streaming recommendations.

3.3 Experiments

In this section, we present the results of the extensive experiments that are conducted to answer the following four Research Questions (RQs).

RQ1: How does our proposed VRS-DWMoE perform when compared with the state-of-the-art approaches?

RQ2: How does our proposed DWMoE perform when compared with the existing recommendation models?

RQ3: How does our proposed VRS perform when compared with the existing sampling methods?

RQ4: How does the number of experts in VRS-DWMoE affect the recommendation accuracies?

Table 3.1: Statistics of the tunned datasets for VRS-DWMoE

Dataset	**#User**	**#Item**	**#Interaction**	**Sparsity**
MovieLens	6,400	3,703	994,169	95.81%
Netflix	5,000	16,073	1,010,588	98.74%
Yelp	25,677	25,815	731,671	99.89%

The symbol # in this table denotes the number (#User represents the number of users).

3.3.1 Experimental Settings

Datasets. In the experiments, we employ three widely-used real-world datasets [127, 38] in the literature of streaming recommendations, including MovieLens (1M)[1], Netflix[2] and Yelp[3], to verify the effectiveness of our proposed VRS-DWMoE. Note that these three datasets all contain timestamps and thus can be utilized to simulate data stream for evaluating our proposed approach and baselines. We extract the interactions of 5000 randomly selected users from the Netflix dataset for the experiments, as processing the original Netflix dataset, which contains more than 100 million interactions, is beyond our computational capacity. In addition, following the common practice [38, 91], for each dataset, we retain the interactions from users who have more than 10 interactions to reduce data sparsity. The statistics of the tuned datasets are summarised in Table 3.1. Moreover, following [140, 121], we transform the explicit ratings in all three datasets into the implicit ones, where it is 1 if an explicit rating exists and 0 otherwise, as this work focuses on the recommendations with implicit interactions.

Evaluation Policy. Following [38], we first sort the data in each of the three datasets by their coming time, and then divide them into two parts: 1) a training set to simulate the historical data and 2) a test set to simulate the upcoming data in the streaming scenarios. Specifically, the data in the training set are used to incrementally train recommendation models while the data in the test set are first used for the test and then used to incrementally train the recommendation models. Then, we select the first 85%, 90% and 95% of interactions from each dataset as training sets, while the remainder

[1]https://grouplens.org/datasets/movielens/1m
[2]https://www.kaggle.com/netflix-inc/netflix-prize-data
[3]https://www.yelp.com/dataset

serves as the corresponding test sets. Note that the k-fold cross validation is infeasible in streaming scenarios, as only the latest interactions (i.e., the interactions with the largest timestamps) can be utilised for the test to be consistent with the data coming order. In addition, following [38], we report the results in the cases where the proportion of the training set is 90$\%$ while the results in other two cases are similar to the reported ones. Moreover, to verify the effectiveness of our proposed VRS-DWMoE in both underload scenarios and overload scenarios, we train the recommendation model with a fixed number n_p ($n_p = 256$ in this chapter) of interactions each time and adjust the number n_r of interactions received in this training period to indicate different workload intensities. For the sake of simplicity, we use n_p and n_r to simulate the data processing speed s_p and data receiving speed s_r, respectively. In this way, underload scenarios and overload scenarios can be simulated by the cases where $s_p > s_r$ and $s_p < s_r$, respectively, via adjusting the amount of data received during the training process in the last iteration. Note that this work is among the first attempt in the literature to evaluate streaming recommender systems in both the underload and overload scenarios, and the evaluation policies will be improved in the future work.

Evaluation Metrics. Following the common practice [127, 38], we adopt the ranking-based evaluation strategy. Specifically, for each interaction between a target user and a target item, we first randomly sample 99 items that are not interacted with by the target user as negative items, and then rank the target item among these 100 items (i.e., the target one plus the 99 negative ones). Finally, the recommendation accuracies are measured by two widely-used metrics: Hit Ratio (**HR**) and Normalised Discounted Cumulative Gain (**NDCG**) [38, 127]. To be more specific, HR@k judges if the target item is among the top-k recommended items, while NDCG@k also considers the specific ranking position of the target item in the top-k recommendation list. Normally, larger HR and NDCG indicates higher recommendation accuracies. In this Chapter, we utilize HR@10 and NDCG@10 as the evaluation metrics.

Baselines. The following eight baselines are used for comparisons, including iBPR, iGMF, iMLP, iNeuMF, RCD, eAls, SPMF and OCFIF.

- *Bayesian Personalised Ranking* (BPR) [91] is a representative personalised ranking method to optimise the matrix factorisation. We adapt BPR to the streaming scenarios, named as **iBPR**, by training it with newly coming data continuously via stochastic gradient descent.

- *Neural Matrix Factorisation* (NeuMF) [37] is an advanced matrix factorisation model, which combines two other recommendation models: *Generalized Matrix Factorisation* (GMF) and *Multi-Layer Perceptron* (MLP), to achieve higher recommendation accuracies. We adapt NeuMF, GMF and MLP, to streaming scenarios by training recommendation models with newly coming data continuously via stochastic gradient descent. The adapted streaming versions of these three recommendation models are named as **iNeuMF**, **iGMF** and **iMLP**, respectively.

- *Randomised block Coordinate Descent* (**RCD**) [15] and *Element-wise Alternating Least Squares* (**eAls**) [38] are two representative approaches for optimising the matrix factorisation models in streaming scenarios. We enhance RCD and eAls with abilities of batch processing to increase their throughput for fair comparisons.

- *Stream-centered Probabilistic Matrix Factorisation* (**SPMF**) [127] is a state-of-the-art SRS. SPMF was originally performed along with a time-consuming sampling method and does not perform well w.r.t. our evaluation policy where sampling needs to be frequently performed. For a fair comparison, we employ our proposed VRS to prepare training data for SPMF.

- *Online Collaborative Filtering with Implicit Feedback* (**OCFIF**) [140] is the only SRS reported in the literature that employs multiple models (i.e., matrix factorisation) to avoid the limitations of a single model for higher recommendation accuracies.

In addition, we equip our proposed **VRS-DWMoE** with different numbers of experts (i.e., 2, 4, 6 and 8) for comparisons. For example, VRS-DWMoE_8 indicates the VRS-DWMoE that is equipped with 8 experts for both MoUE and MoIE.

Parameter Setting. For a fair comparison, we initialise the baselines with parameters reported in their papers and optimise them for our experimental settings. For our VRS-DWMoE, we empirically set the learning rate to 0.001, the batch size bs to 256, the loss coefficient γ to 0.01, and the volume of the reservoir to 10,000 interactions. Besides, we employ the widely-used negative sampling technique [140, 126, 137], where the reservoir is used to check if an interaction exists, to improve the learning performance with the negative sampling size set to four. We also adopt L_2 regularisation and Adam optimiser to avoid overfitting and for the optimisation purpose, respectively. Other parameters including δ, λ_{res} and λ_{new} are adjusted via cross validation to achieve the best performance in different cases.

3.3.2 Performance Comparison and Analysis

Experiment 1: Comparison with Baselines (for RQ1 and RQ2)

Setting. To answer **RQ1** and **RQ2**, we compare our proposed VRS-DWMoE (the number n_e of experts is set to eight) with all eight baselines with a fixed data processing speed $s_p = 256$ and different data receiving speeds, where $s_r = 128$ and $s_r = 512$ indicate underload scenarios and overload scenarios, respectively.

Result 1 (for RQ1). Table 3.2 shows the results of our proposed approach and eight baselines on all three datasets. In all the cases, VRS-DWMoE_8 delivers the highest recommendation accuracies (marked with **bold** font), and the improvement percentages of VRS-DWMoE_8 over the best-performing baselines (with the results marked by <u>underline</u>) are introduced in the last row for each dataset, ranging from 2.0% to 37.4% with an average of 8.5% in terms of HR@10, and ranging from 1.9% to 42.9% with an average of 10.4% in terms of NDCG@10.

The superiority of VRS-DWMoE can be explained in two aspects: 1) VRS ad-

[4]Improvement percentages over the best-performing baseline(s)

Table 3.2: Performance comparison for VRS-DWMoE

Datasets	Metrics		HR@10			NDCG@10		
	Data Receiving Speeds (s_r)		128	256	512	128	256	512
MovieLens	Baselines	eAls	0.231	0.231	0.234	0.106	0.106	0.108
		RCD	0.287	0.297	0.278	0.139	0.145	0.138
		iBPR	0.303	0.303	0.279	0.147	0.147	0.134
		SPMF	0.460	0.453	0.440	0.251	0.246	0.238
		iGMF	0.525	0.529	0.477	0.295	0.297	0.265
		iMLP	0.538	0.539	0.488	0.304	0.303	0.272
		iNeuMF	0.551	0.546	0.496	0.311	0.307	0.275
		OCFIF	0.532	0.508	0.467	0.291	0.279	0.256
	Ours	VRS-DWMoE_8	**0.563**	**0.558**	**0.535**	**0.317**	**0.313**	**0.299**
	Improvement percentages[4]		2.20%	2.20%	7.90%	1.90%	2.00%	8.70%
Netflix	Baselines	eAls	0.395	0.389	0.362	0.211	0.207	0.192
		RCD	0.447	0.436	0.435	0.226	0.226	0.219
		iBPR	0.685	0.686	0.627	0.396	0.395	0.360
		SPMF	0.701	0.669	0.640	0.425	0.397	0.378
		iGMF	0.747	0.748	0.577	0.482	0.482	0.352
		iMLP	0.787	0.782	0.624	0.519	0.510	0.369
		iNeuMF	0.801	0.798	0.711	0.531	0.529	0.430
		OCFIF	0.745	0.734	0.606	0.457	0.453	0.357
	Ours	VRS-DWMoE_8	**0.821**	**0.814**	**0.790**	**0.553**	**0.548**	**0.515**
	Improvement percentages[5]		2.50%	2.00%	11.1%	4.10%	3.60%	19.8%
Yelp	Baselines	eAls	0.287	0.289	0.290	0.167	0.167	0.169
		RCD	0.454	0.452	0.447	0.260	0.257	0.259
		iBPR	0.307	0.295	0.188	0.180	0.172	0.108
		SPMF	0.197	0.192	0.184	0.104	0.100	0.097
		iGMF	0.499	0.470	0.396	0.294	0.276	0.228
		iMLP	0.573	0.574	0.438	0.338	0.338	0.246
		iNeuMF	0.566	0.570	0.435	0.331	0.334	0.247
		OCFIF	0.260	0.249	0.203	0.135	0.129	0.107
	Ours	VRS-DWMoE_8	**0.608**	**0.603**	**0.602**	**0.354**	**0.358**	**0.353**
	Improvement percentages[5]		6.10%	5.10%	37.4%	4.70%	5.90%	42.9%

dresses preference drift while capturing long-term user preferences by wisely complementing newly coming data with sampled historical data, and 2) DWMoE better learns the heterogeneous user preferences and item characteristics with two MoEs, where each expert specialises in one underlying type of users or items.

Result 2 (for RQ2). The superiority of our DWMoE is verified by the cases where $s_r = s_p$ (i.e., $s_r = 256$) in Table 3.2. In these cases, both our approach and baselines utilise all the newly coming data to train recommendation models, thus their recommendation accuracies only depend on their recommendation models. Therefore, the superiority of DWMoE is confirmed by the highest recommendation accuracies delivered by VRS-DWMoE in these cases. The reason for this superiority is that DWMoE not only learns heterogeneous user preferences and item characteristics with two dedicated MoEs, respectively, but also allows each of their experts to specialise in one

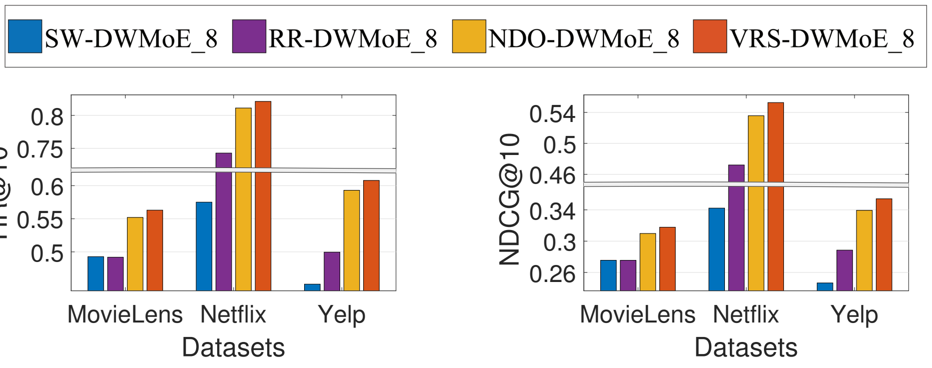


Figure 3.2: Performance comparison for VRS

underlying type of users or items.

Experiment 2: Performance of VRS (for RQ3)

Setting. To answer **RQ3**, we replace our proposed VRS with existing sampling methods, including Newly coming Data Only (NDO) [140], Reservoir-enhanced Random sampling (RR) [16] and Sliding Window (SW) [102], for comparisons. In this experiment, we report the results in the underload scenarios only to save space while the results in the overload scenarios are similar to the reported ones.

Result 3 (for RQ3). As Fig. 3.2 illustrates, our proposed VRS outperforms all the other sampling methods. The improvements of VRS over the best-performing baseline — NDO, range from 1.2% (on Netflix) to 2.0% (on Yelp) with an average of 1.9% in terms of HR@10, and range from 3.2% (on Netflix) to 4.3% (on Yelp) with an average of 3.4% in terms of NCDG@10. The effectiveness of VRS comes from wisely complementing newly coming data with sampled historical data while guaranteeing the proportion of newly coming data, and thus addressing preference drift while capturing long-term user preferences.

Experiment 3: Effect of Number of Experts (for RQ4)

Setting. To answer **RQ4**, we compare the performance of VRS-DWMoE when equipped with different numbers (i.e., 2, 4, 6 and 8) of experts. In this experiment, we report the results in the overload scenarios only to save space while the results in the underload are similar to the reported ones.

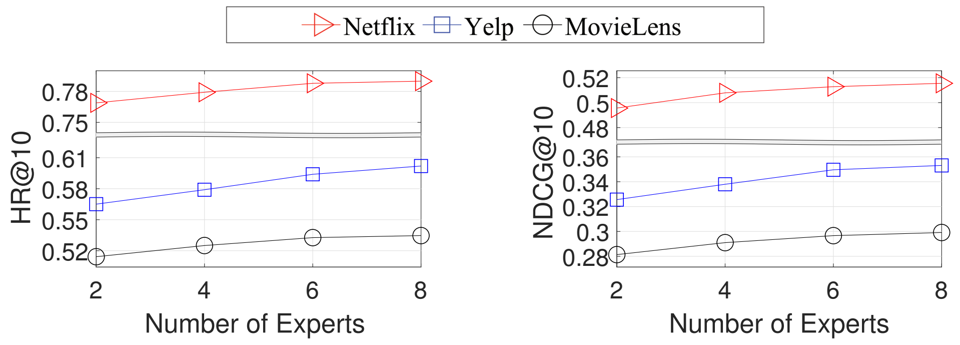


Figure 3.3: Effect of the number (n_e) of experts for VRS-DWMoE

Result 4 (for RQ4). As Fig. 3.3 illustrates, our proposed VRS-DWMoE delivers higher recommendation accuracies when equipped with more experts. The improvements of VRS-DWMoE equipped with eight experts over that equipped with two experts range from 2.7% (on Netflix) to 6.5% (on Yelp) with an average of 4.4% in terms of HR@10, and range from 4.0% (on Netflix) to 8.4% (on Yelp) with an average of 6.3% in terms of NCDG@10. The reason for the superiority of more experts is that more experts better complement one another with their expertise to more effectively learn the user preferences and item characteristics. Specifically, each expert model is specified in learning the preferences of one underlying type of users or the characteristics of one underlying type of items. Therefore, more experts improve the learning processes as each of them can focus more on learning the preferences of fewer users or the characteristics of fewer items, and thus contribute to increasing the recommendation accuracies.

3.4 Chapter Summary

In this chapter, we have proposed a Variational and Reservoir-enhanced Sampling based Double-Wing Mixture of Experts framework (VRS-DWMoE) for delivering accurate streaming recommendations. We first propose VRS to wisely complement newly coming data with sampled historical data to address preference drift while capturing long-term user preferences. After that, with these sampled data, DWMoE learns heterogeneous user preferences and item characteristics with two MoEs: MoUE and MoIE, respectively, and then makes recommendations with learnt preferences and characteristics. The superiority of VRS-DWMoE has been verified by extensive experiments.

Chapter 4

Stratified and Time-aware Sampling based Adaptive Ensemble Learning for Streaming Recommendations

Various SRSs[1] have been proposed to deliver recommendations w.r.t. data streams of user-item interactions. These SRSs commonly assume that their data processing speed equals the data receiving speed from the applications. However, in practice, the data receiving speed varies over time and might not equal the data processing speed. This leads to a new challenge that has been introduced in Subsection 1.2.3 and expressed by **CH3**: 'how to well deal with the *underload* scenarios where the data receiving speed is *lower* than the data processing speed, and the *overload* scenarios where the data receiving speed is *higher* than the data processing speed'.

Targeting **CH3** mentioned above and **CH1** introduced in Subsection 1.2.1, this chapter proposes a novel **S**tratified and Time-aware Sampling based Adaptive Ensemble Learning framework, called STS-AEL, for higher accuracies of streaming recommendations. STS-AEL contains two main components: 1) Stratified and Time-aware Sampling (STS) and 2) Adaptive Ensemble Learning (AEL). Specifically, STS samples training data from both newly coming data and historical data both by assigning higher sampling probabilities to newer data in the sample space. In addition, the sam-

[1]The SRSs in this chapter all refer to the Single-behaviour SRSs as multi-behaviour SRSs have not been reported in the literature and are either not considered in the work reported in this chapter.

ple sizes of the newly coming data and reservoir are both determined based on the characteristics and receiving speed of the streaming data. As for AEL, it first trains multiple individual models in parallel with the sampled data and then fuses the results of these trained models with a novel sequential adaptive mechanism. Note that AEL is specifically devised for the streaming scenarios, where training processes and test processes are iteratively conducted over each data stream. To be more specific, AEL dynamically calculates the fusion weights with a sequential adaptive mechanism based on the test accuracy achieved by each individual model in the last iteration to conduct more effective fusions in the current iteration.

Although the target problem of this chapter is similar to the one of Chapter 3, the works in these two chapters aim to address different challenges. Specifically, the work in Chapter 3 focuses more on effectively learning the heterogeneous user preferences and item characteristics by mixture-of-expert models, especially in the underload scenarios; while the work in this chapter focuses more on effectively dealing with the overload scenarios concurrently with multiple individual recommendation models via an ensemble learning method.

The remainder of this chapter is organized as follows. We first formulate our research problem of streaming recommendations. Then, we introduce our proposed STS-AEL framework and its two components (i.e., STS and AEL). After that, we present the results of the experiments that are conducted to verify the effectiveness of STS-AEL. Finally, we summarise this chapter.

4.1 Problem Statement

In this chapter, we focus on streaming recommendations with implicit user-item interactions, e.g., users' clicks on items. As the problem studied in this chapter is similar to the one in Chapter 3, we briefly introduce this problem in this section, and readers can refer to Section 3.1 for more details. Specifically, with the interaction set $\mathbf{Y}$, user set $\mathbf{U}$ and item set $\mathbf{V}$, let $\mathcal{Y} = \{y^1_{u^1,v^1}, y^2_{u^2,v^2}, \ldots, y^k_{u^k,v^k}, \ldots\}$ be the list of currently received

Algorithm 1: STS-AEL Framework

```
   Input  : New Data N, Reservoir R, Set of Individual Models IM
   Output : Recommendations
 1 if N is for training then
       /* Stratified and Time-aware Sampling */
 2     Sample |IM| sets of training data from N and R by Eqs. (4.1) to (4.4)
       /* Concurrent Training */
 3     do in parallel for each model im in IM
 4         Update im with sampled training data by optimising the loss
           in Eq. (4.12)
 5 else
       /* Sequential Adaptive Fusing */
 6     do in parallel for each model im in IM
 7         Get the prediction results of im, taking NeuMF as an example,
           by Eqs. (4.5) to (4.11)
 8     Get the fusion weights fw for the models in IM by Eqs. (4.13) to (4.18)
 9     Get the final predictions $\hat{\mathbf{y}}^{final}$ by fusing the prediction results by Eq. (4.19)
10     Store the prediction accuracies
11     return $\hat{\mathbf{y}}^{final}$
```

interactions, the task of the SRS is to predict the probability of a future interaction between the given user u' and item v' based on the currently received interactions $\mathcal{Y}$, i.e., $\hat{y} = P(y_{u',v'}|\mathcal{Y})$.

4.2 Our Proposed STS-AEL Framework

In this section, we first propose a novel Stratified and Time-aware Sampling based Adaptive Ensemble Learning framework, called STS-AEL, and then introduce its two key components: Stratified and Time-aware Sampling (STS) and Adaptive Ensemble Learning (AEL).

4.2.1 Overall Structure

To perform accurate streaming recommendations, we propose Stratified and Time-aware Sampling based Adaptive Ensemble Learning framework, called STS-AEL. As

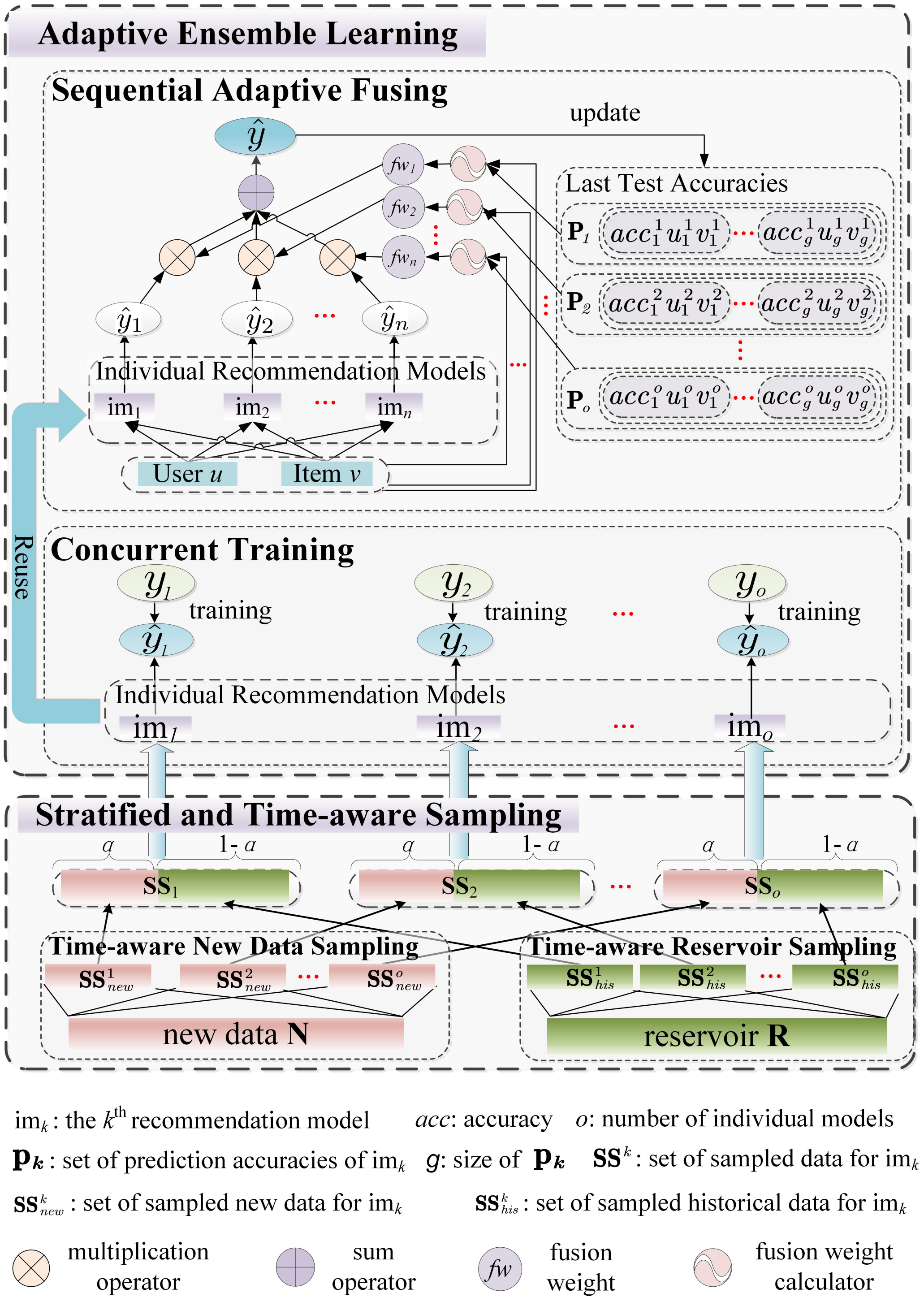

Adaptive Ensemble Learning
Sequential Adaptive Fusing
update
Last Test Accuracies
P_1 $acc_1^1 u_1^1 v_1^1$ ⋯ $acc_g^1 u_g^1 v_g^1$
P_2 $acc_1^2 u_1^2 v_1^2$ ⋯ $acc_g^2 u_g^2 v_g^2$
P_o $acc_1^o u_1^o v_1^o$ ⋯ $acc_g^o u_g^o v_g^o$
fw_1 fw_2 fw_n
$\hat{y}$ $\hat{y}_1$ $\hat{y}_2$ $\hat{y}_n$
Individual Recommendation Models
im_1 im_2 im_n
User u
Item v
Reuse
Concurrent Training
y_1 y_2 y_o
training
$\hat{y}_1$ $\hat{y}_2$ $\hat{y}_o$
Individual Recommendation Models
im_1 im_2 im_o
Stratified and Time-aware Sampling
α $1-\alpha$
SS_1 SS_2 SS_o
Time-aware New Data Sampling
SS_{new}^1 SS_{new}^2 SS_{new}^o
new data **N**
Time-aware Reservoir Sampling
SS_{his}^1 SS_{his}^2 SS_{his}^o
reservoir **R**
im_k: the k^{th} recommendation model
acc: accuracy
o: number of individual models
$\mathbf{p}_k$: set of prediction accuracies of im_k
g: size of $\mathbf{p}_k$
SS^k: set of sampled data for im_k
SS_{new}^k: set of sampled new data for im_k
SS_{his}^k: set of sampled historical data for im_k
multiplication operator
sum operator
fusion weight
fusion weight calculator

Figure 4.1: The structure of the STS-AEL Framework

Fig. 4.1 shows, the proposed STS-AEL mainly contains two components, i.e., STS and AEL, which are introduced in detail in Section 3.3 and Section 3.4, respectively.

Specifically, STS first samples representative data from both newly coming data and the reservoir. After that, with the sampled data, AEL efficiently performs the concurrent training process for all individual recommendation models, and then effectively fuses the results of all these models with a sequential adaptive fusion method to obtain the final recommendation results.

To better introduce the workflow of STS-AEL, we have presented its high-level procedure in Algorithm 1. Specifically, as Algorithm 1 illustrates, the training data are first prepared by STS (line 2), with which the multiple individual models are trained in parallel (lines 3 and 4). Then, when conducting the predictions, the trained individual models generate prediction results in parallel (lines 6 and 7). After that, these prediction results from multiple individual models are fused into the final one with the sequential adaptive fusion method, which elaborately calculates the fusion weights based on the recommendation accuracies of the last batch of received interactions (lines 8 and 9). Finally, the prediction accuracies are stored for calculating the fusion weights regarding the next batch of received interactions (line 10). More details about STS-AEL are presented in the following.

4.2.2 Stratified and Time-aware Sampling

Training the recommendation model with the entire dataset is impractical for SRSs, as streaming data is continuous and infinite. Therefore, we propose STS to sample representative data to reduce the training workload effectively.

To capture both short-term and long-term user preferences, STS elaborately incorporates both newly coming data and historical data while guaranteeing the proportion of newly coming data. Specifically, STS contains five key steps: 1) maintain a reservoir containing representative historical data, which is a widely-used technology [63, 127] in the streaming processing area; 2) calculate the sample sizes of both this reservoir and newly coming data; 3) calculate the probabilities to sample user-item interactions from both the reservoir and newly coming data; 4) obtain sample sets SS_{his} and SS_{new}

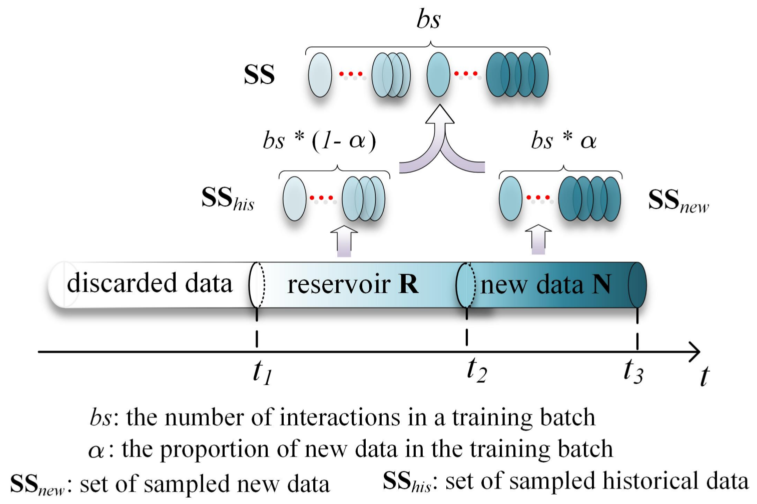

bs
SS
bs * (1- α)
bs * α
SS_his
SS_new
discarded data
reservoir R
new data N
t1
t2
t3
t
bs: the number of interactions in a training batch
α: the proportion of new data in the training batch
SS_new: set of sampled new data
SS_his: set of sampled historical data

Figure 4.2: Stratified and Time-aware Sampling (STS) approach

from the reservoir **R** and newly coming data **N**, respectively, with the sampling sizes and sampling probabilities calculated in the preceded steps; and 5) merge $\mathbf{SS}_{his}$ and $\mathbf{SS}_{new}$ to form the final sample set **SS** as the input of the subsequent concurrent training process.

To better illustrate our proposed STS approach, we present its sample process in Fig. 4.2, where the darker color indicates the newer data (i.e., the data received more recently). As shown in Fig. 4.2, the whole data can be partitioned into three parts: the newly coming data, reservoir, and discarded data, based on their coming time. During reservoir maintenance, STS incorporates newly coming data and discards the oldest data, as newer data contains more timely user preferences towards items.

To guarantee the proportion of the sampled newly coming data in the entire training data, STS adopts a stratified sampling strategy and utilises parameter α to adjust the proportion of the sampled newly coming data. With this proportion α and the training batch size bs, STS first calculates the sample size of newly coming data: $|\mathbf{SS}_{new}| = bs{*}\alpha$ and the sample size of the historical data from the reservoir: $|\mathbf{SS}_{his}| = bs{*}(1{-}\alpha)$. And then, $\mathbf{SS}_{new}$ and $\mathbf{SS}_{his}$ are sampled from newly coming data and reservoir, respectively, both in a time-aware manner. Specifically, to assign newer data in the sample space higher sampling probabilities, we employ decay ratios λ_{new} ($\lambda_{new} \geq 1$) and λ_{res}

($\lambda_{res} \geq 1$) for the newly coming data **N** and the reservoir **R**, respectively. Next, we present the sampling process for $\mathbf{SS}_{new}$ in detail, while the sampling process for $\mathbf{SS}_{his}$ is similar. Given the sampling probability p_{k-1} of the $(k-1)^{th}$ user-item interaction, the sampling probability p_k of the k^{th} user-item interaction can be calculated as below,

$$p_k = p_{k-1} * \lambda_{new}. \tag{4.1}$$

In this way, we can adjust the ratio (i.e., λ_{new} for new interactions and λ_{res} for interactions in the reservoir) of the sampling probability of the k^{th} received interaction against that of the $(k-1)^{th}$ received interaction, and thus adjust the emphasis of our approach on newer interactions with more flexibilities. After that, by iteratively performing Eq. (4.1) and assuming the sampling probability of the earliest user-item interaction is p_1, we can get p_k as follows,

$$p_k = p_1 * \overbrace{\lambda_{new} * \lambda_{new} * \cdots * \lambda_{new}}^{k-1} = p_1 * (\lambda_{new})^{k-1}. \tag{4.2}$$

Based on Eq. (4.2), with the size $|\mathbf{N}|$ of newly coming data, we can infer the normalised sampling probability of the k^{th} user-item interaction by the following equation,

$$P(k|\lambda_{new}, |\mathbf{N}|) = \frac{p_k}{\sum_{i=1}^{|\mathbf{N}|} p_i} = \frac{\lambda_{new}^{k-1} * (1-\lambda_{new})}{1-\lambda_{new}^{|\mathbf{N}|}}. \tag{4.3}$$

Then, with $P(k|\lambda_{new}, |\mathbf{N}|)$, STS samples $\mathbf{SS}_{new}$ from the newly coming data. Note that the sampling process is with replacement among the individual models, which means that one user-item interaction can be possibly sampled by multiple individual models. Using the similar method, STS samples $\mathbf{SS}_{his}$ from the reservoir. Then, the final sampled training data set SS can be obtained by merging $\mathbf{SS}_{new}$ and $\mathbf{SS}_{his}$ as follows,

$$\mathbf{SS} = \mathbf{SS}_{new} \cup \mathbf{SS}_{his}. \tag{4.4}$$

The aforementioned parameters α, λ_{new} and λ_{res} provide STS with flexibilities to effectively handle data streams with various characteristics and receiving speeds. For example, in the scenarios where the preference drift happens frequently, we can increase the values of α, λ_{new} and λ_{res} to increase the proportion of sampled newly coming data and the sampling probabilities of newer data in the sample space. In this way, we can emphasize more on the newly coming data, and thus more accurately capture the short-term user preferences for better handling the preference drift. Furthermore, STS has a strong generalisation capability and can be easily derived to the existing sampling approaches. For example, STS can be derived to the sliding window based sampling [143] by setting α to $\frac{|\mathbf{N}|}{bs}$ and setting both λ_{new} and λ_{res} to large values, e.g., 1.5, and it has a similar effect to random sampling when randomly selecting α from $[0, 1]$ and setting both λ_{new} and λ_{res} to 1.

With the representative data sampled by STS from both newly coming data and reservoir, the individual recommendation models can be trained by AEL, which will be introduced in the following subsection.

Time Cost Analysis. The time cost of STS is acceptable, as the time complexities of Eqs. (4.1) to (4.4) are $O(bs)$ for sampling a training batch of user-item interactions.

4.2.3 Adaptive Ensemble Learning

With the data sampled by STS, our proposed AEL first concurrently trains multiple individual models, and then fuses the results of these trained models via an effective sequential adaptive fusion method to obtain the final recommendation results with higher accuracies. More details about AEL are presented below.

Concurrent Training. AEL trains multiple individual recommendation models concurrently for higher computational efficiency. This concurrent training process is possible, as the individual models are independent from one another during the training process. Moreover, this feature of concurrency contributes to more effectively handling the streaming data, especially when confronting the excessive amount of data in overload scenarios. Furthermore, AEL performs negative sampling, a technique that has been widely used in the literature [76, 38], to overcome the natural absence of negative feedback in recommendations with implicit user-item interactions. In addition, AEL utilises the aforementioned reservoir to check if a sampled interaction exists for guaranteeing the effectiveness of negative sampling.

For individual recommendation models, AEL can employ existing monolithic SRSs directly or adapt offline RSs to streaming scenarios by incrementally updating recommendation models with an online update mechanism; for example, training recommendation models with stochastic gradient descent [1]. In this work, AEL delivers the best performance by adapting Neural Matrix Factorisation (NeuMF) proposed in [37] to the streaming scenarios. Specifically, NeuMF is a neural network based RS, which combines other two basic RSs, i.e., Generalized Matrix Factorisation (GMF) and Multiple Layer Perceptron (MLP), to achieve more accurate recommendations. As our proposed STS-AEL achieves high recommendation accuracies when ensembling these three monolithic models, i.e., GMF, MLP and NeuMF, we briefly introduce them in the following.

As its name indicates, GMF is a generalised matrix factorisation model that enhances the original matrix factorisation model with non-linear transformation to achieve stronger modelling capabilities. Specifically, GMF improves matrix factorisation with a nonlinear activation function a_{out}^{GMF} as follows,

$$\phi^{GMF} = \mathbf{p}_u^{GMF} \otimes \mathbf{q}_v^{GMF}, \tag{4.5}$$

$$\hat{y}_{u,v}^{GMF} = a_{out}^{GMF}(\mathbf{W}_{GMF}^{T}\phi^{GMF} + \mathbf{b}_{GMF}), \tag{4.6}$$

where $\mathbf{p}_u$ and $\mathbf{q}_v$ represent the embedding of user u and the embedding of item v, respectively, $\otimes$ denotes the element-wise multiplication, $\mathbf{W}_{GMF}$ denotes the weight matrix, $\mathbf{b}_{GMF}$ denotes the bias vector, and $\hat{y}_{u,v}^{GMF}$ indicates the predicted probability for an interaction between user u and item v. In such a way, through enhancing the conventional matrix factorisation with nonlinearity, GMF can achieve stronger fitting ability, and thus make more accurate predictions.

Different from GMF that learns user preferences from the interactions based on a fixed dot product between the user embedding and the item embedding, MLP aims to improve modelling flexibilities with a multiple-layer-perception structure, and thus achieve higher recommendation accuracies. Specifically, MLP first concatenates the user embedding and item embedding as follows,

$$\phi_0^{MLP} = \left[\mathbf{p}_u^{MLP}; \mathbf{q}_v^{MLP}\right], \tag{4.7}$$

and then feeds the concatenated embedding to a L-layer perceptron for training with interactions between users and items with the following equations,

$$\begin{aligned} \phi_1^{MLP} &= a_1^{MLP}(\mathbf{W}_1^T \phi_0^{MLP} + \mathbf{b}_1), && (4.8)\\ &\vdots \\ \phi_k^{MLP} &= a_k^{MLP}(\mathbf{W}_k^T \phi_{k-1}^{MLP} + \mathbf{b}_k), && (4.9)\\ &\vdots \\ \hat{y}_{u,v}^{MLP} &= a_L^{MLP}(\mathbf{W}_L^T \phi_{L-1}^{MLP} + \mathbf{b}_L), && (4.10) \end{aligned}$$

where $\mathbf{W}_k$, $\mathbf{b}_k$ and a_k denote the weight matrix, bias vector and activation function for the k^{th} $(1 \leq k \leq L)$ layer, respectively. MLP obtains much flexibility from the concatenated embedding and nonlinear perceptrons, and thus can capture the user preferences more effectively.

To further improve recommendation accuracies, NeuMF fuses these two recommendation models (i.e, GMF and MLP) for complementing each other to better learn the user preferences towards items. Specifically, NeuMF first concatenates the features learned by GMF and MLP, and then transforms the concatenated feature by a nonlinear function a_{out}^{NeuMF} as below,

$$\hat{y}_{u,v}^{NeuMF} = a_{out}^{NeuMF}(\mathbf{W}_{NeuMF}^{T}\left[\phi^{GMF};\phi_{L-1}^{MLP}\right] + \mathbf{b}_{NeuMF}), \tag{4.11}$$

where $\mathbf{W}_{NeuMF}$ and $\mathbf{b}_{NeuMF}$ denote the weight matrix and the bias vector, respectively. Through such a process, NeuMF is expected to combine the advantages of both GMF and MLP, and thus delivers more accurate recommendations.

For the training process, following the work in [37, 32, 126], we employ the binary cross-entropy loss as the loss function,

$$\ell oss_{<u,v>} = -(y_{<u,v>}\text{log}\hat{y}_{<u,v>} + (1 - y_{<u,v>})\text{log}(1 - \hat{y}_{<u,v>})), \tag{4.12}$$

where $y_{u,v}$ indicates if an interaction between user u and item v exists, and $\hat{y}_{u,v}$ is the predicted probability for this interaction. Specifically, this loss function encourages larger $\hat{y}_{u,v}$ if the interaction between user u and item v exists (i.e., $y = 1$), and encourages smaller $\hat{y}_{u,v}$ otherwise. With this binary cross-entropy loss function defined in Eq. (4.12), the individual recommendation models can be trained via stochastic gradient descent.

Sequential Adaptive Fusing. The proposed sequential adaptive fusion approach improves fusion performance by assigning elaborately calculated weights to multiple individual models in the streaming scenarios, where training processes and test processes are iteratively conducted with data streams. Specifically, AEL contains four key steps: 1) calculate and store the prediction accuracy for each interaction and the corresponding user-item pair (i.e., the user and item related to this interaction) for each individual model in the current iteration, 2) with the prediction accuracies and the corresponding user-item pairs stored in the last iteration, estimate the confidence of each individual model to predict the interaction for the target user-item pair, 3) based on the calculated confidence, use an AdaBoost-like method [102, 118] to calculate the fusion weights for all the individual models, and 4) fuse the predictions made by these models with the fusion weights to obtain the ensembled prediction. More details are presented below.

AEL maintains a set $\mathbf{P}$ containing tuples of prediction accuracies and the corresponding user-item pairs in the last iteration for each individual model, taking the k^{th} individual model as an example,

$$\mathbf{P}_k = \{\langle acc_1^k, {u_1}^k, {v_1}^k \rangle, \cdots, \langle acc_j^k, {u_j}^k, {v_j}^k \rangle, \cdots, \langle acc_g^k, {u_g}^k, {v_g}^k \rangle\}, \tag{4.13}$$

where acc_j^k $(1 \leq j \leq g)$ represents the accuracy of the k^{th} individual model regarding user-item pair $\langle u_j^k, v_j^k \rangle$ and g is the size of $\mathbf{P}_k$. To predict the interaction between user u and item v by the k^{th} individual model, AEL first calculates the similarity (we employ cosine similarity in this work to achieve the best performance) between the target user-item pair $\langle u, v \rangle$ and each of the user-item pairs $\langle u', v' \rangle$ in $\mathbf{P}_k$ based on their embeddings,

$$\text{cos\_sim}^k_{\langle u,v \rangle, \langle u',v' \rangle} = \frac{([\mathbf{p}_u^k; \mathbf{q}_v^k])^T \cdot [\mathbf{p}_{u'}^k; \mathbf{q}_{v'}^k]}{\|[\mathbf{p}_u^k; \mathbf{q}_v^k]\|_2 \|[\mathbf{p}_{u'}^k; \mathbf{q}_{v'}^k]\|_2}, \tag{4.14}$$

where $[\mathbf{p}_u; \mathbf{q}_v]$ indicates the concatenation of embeddings $\mathbf{p}_u$ and $\mathbf{q}_v$, and $\| * \|_2$ represents the L_2 norm. Then, based on these similarities, AEL creates subset $\mathbf{S}^k \subseteq \mathbf{P}_k$ (taking the k^{th} individual model as an example) for each individual model by extracting the top e (a predetermined parameter representing the size of $\mathbf{S}^k$) tuples that have the most similar user-item pairs to the target $\langle u, v \rangle$ from $\mathbf{P}_k$. Then, we can estimate the confidence of each model for the prediction of the interaction for $\langle u, v \rangle$. Specifically, with $\mathbf{S}_k$, AEL calculates the confidence $c_{u,v}^k$ of the k^{th} individual model to predict the interaction for the target $\langle u, v \rangle$ as follows,

$$c_{u,v}^k = \frac{1}{|\mathbf{S}_{u,v}^k|} \sum_{\langle \mathbf{p}_{u'}^k, \mathbf{q}_{i'}^k, a_{u',i'}^k \rangle \in \mathbf{S}_{u,v}^k} a_{u',v'}^k. \tag{4.15}$$

For the sake of simplicity, we use $\mathbf{c}_{u,v}$ to represent the vector containing the confidence of all the individual models, i.e.,

$$\mathbf{c}_{u,v} = [c^1_{u,v}, \cdots, c^o_{u,v}]^T, \tag{4.16}$$

where o is the number of individual models. With this estimated confidence $\mathbf{c}_{u,v}$, the fusion weights for the prediction of the interaction between user u and item v can be calculated and normalised with an AdaBoost-like [12] method as below,

$$\mathbf{fw}'_{u,v} = \frac{\mathbf{c}_{u,v}}{1 - \mathbf{c}_{u,v}}, \tag{4.17}$$

$$\mathbf{fw}_{u,v} = \frac{\mathbf{fw}'_{u,v}}{||\mathbf{fw}'_{u,v}||_1}, \tag{4.18}$$

where $|| * ||_1$ represents the L_1 norm and $\mathbf{fw}_{u,v}$ represents the fusion weights of the individual models for $\langle u, v \rangle$. As shown in Eqs. (4.17) and (4.18), AEL assigns higher fusion weights to the models those with more confidence (i.e., larger $\mathbf{c}_{u,v}$), for better fusion effectiveness. Finally, AEL fuses the predictions $\hat{\mathbf{y}}$ from the multiple individual recommendation models with $\mathbf{fw}_{u,v}$ to get the final prediction as follows,

$$\hat{y}^{final}_{u,v} = \mathbf{fw}^T_{u,v} \hat{\mathbf{y}}_{u,v}. \tag{4.19}$$

Time Cost Analysis. The time cost of AEL mainly contains two parts: 1) the time cost of the concurrent training process and 2) the time cost of the sequential adaptive fusion method. As multiple individual models are trained in parallel in the concurrent training process, the time complexity for the concurrent training is roughly equal to that of the corresponding monolithic recommendation model. As for the sequential adaptive fusion, the prediction processes of the individual models can also be parallel. Thus, compared with the monolithic recommendation models, the extra time cost introduced by AEL mainly comes from the calculation of fusion weights, which is described by Eqs. (4.14) to (4.18). The time complexities of Eqs. (4.14) to (4.18) can be easily calculated; they are $O(|\mathbf{p_u}| * |\mathbf{q_v}| * |\mathbf{S}_k|)$, $O(|\mathbf{S}_k|)$, $O(1)$, $O(o)$ and $O(o)$, respectively. Obviously, the extra time cost mainly depends on Eq. (4.14), which is $O(|\mathbf{p_u}| * |\mathbf{q_v}| * |\mathbf{S}_k|)$, since it is the highest one. This time complexity is acceptable for the following two reasons: 1) it remains constant once the the sizes of latent factors $|\mathbf{p_u}|$ and $|\mathbf{q_v}|$) and the size of S_k (i.e., a set of the interactions and corresponding recommendation accuracies from the k^{th} individual model), are determined, and 2) it is not affected by the number of users or the number of items. Although AEL has a higher time complexity than the classic Bagging [2] based ensemble learning, which commonly averages the results of multiple individual models for the fusion purpose with a time complexity of $O(1)$, our proposed AEL greatly improves the fusion performance by fusing the results of multiple individual models with elaborately calculated weights. As for the Boosting [27] based ensemble learning methods, they typically train the individual models one by one, and thus need much more training time compared with our proposed AEL that trains the individual models in parallel.

4.3 Experiments

In this section, we present the results of the extensive experiments that are conducted to answer the following four Research Questions (RQs).

RQ1: How does our proposed STS-AEL perform when compared with the state-of-the-art approaches?

RQ2: How does the number of individual models ensembled by STS-AEL affect the recommendation accuracies?

RQ3: How does our proposed STS perform when compared with the existing sampling methods?

RQ4: How does our proposed AEL perform when compared with the existing ensemble methods?

4.3.1 Experimental Settings

Before presenting the results and analysis of the experiments, we first introduce the experimental settings, including the datasets, evaluation policy, evaluation metrics and comparison approaches, for better readability.

Datasets. In the experiments, we employ three real-world datasets, including MovieLens (1M)[2], Netflix[3] and Yelp[4], all of which are widely used in the literature [127, 38]. Note that these three datasets all contain timestamps and thus can be utilized to simulate data stream for evaluating our proposed approach and baselines. Since the original Netflix dataset contains over 100 million interactions, which is beyond our computation capacity, we extract the interactions of 5000 randomly selected users for the experiments. In addition, we follow the common practice [38, 91] to retain the users who have more than ten interactions on all three datasets to reduce the data sparsity. The statistics of the tuned datasets are summarised in Table 3.1. Since this work focuses on the recommendations with implicit user-item interactions, following the common practice [140, 121, 38], we transform the explicit data (i.e., users' ratings on items) in all three datasets into the implicit ones, where it is 1 if an explicit interaction exists and 0 otherwise.

Evaluation Policy. Similar to [38, 127], we first sort the data by their coming time, and then divide them into a training set (where the data are used for incrementally training the recommendation models) and a test set (where the data are first used for testing and then used for incrementally training the recommendation models) to simulate the historical data and upcoming data, respectively, in the streaming scenarios. After that, we select the first 85%, 90% and 95% of interactions from each dataset as training sets, while the remainder serves as the corresponding test sets. Note that the k-fold cross validation is infeasible in streaming scenarios, as only the latest interactions (i.e., the interactions with the largest timestamps) can be utilised for the test to

[2]https://grouplens.org/datasets/movielens/1m
[3]https://www.kaggle.com/netflix-inc/netflix-prize-data
[4]https://www.yelp.com/dataset

be consistent with the data coming order. In addition, following [38], we report the results in the cases where the proportion of the training set is 90$\%$ while the results in other two cases are similar to the reported ones. Moreover, to observe the performance of our proposed STS-AEL and that of the baselines w.r.t. different workload intensities, we train all the models with a fixed number n_p (we set $n_p = 256$ in this chapter) of user-item interactions in each iteration and adjust the number n_r of user-item interactions received in this training period to simulate the cases with different workload intensities. For the sake of simplicity, we use n_p and n_r to simulate the data processing speed sp_p and data receiving speed sp_r, respectively, where $sp_p > sp_r$ and $sp_p < sp_r$ indicate the underload scenarios and overload scenarios, respectively, via adjusting the amount of data received during the training process in the last iteration. Note that this work is among the first attempt in the literature to evaluate streaming recommender systems in both the underload and overload scenarios, and the evaluation policies will be improved in the future work.

Evaluation Metrics. We adopt the ranking-based evaluation strategy, which is widely used for the evaluation of streaming recommendations with implicit data [127, 38]. Specifically, for each given interaction between a target user and a target item, we randomly sample 99 items that are not interacted with this user as negative items, and rank the target item among the 100 items (i.e., the target one plus the 99 negative ones). Then, the recommendation accuracies are evaluated by two widely-used metrics: Hit Ratio (**HR**) and Normalised Discounted Cumulative Gain (**NDCG**) [36, 38, 127]. In this Chapter, we utilize HR@10 and NDCG@10 to evaluate the recommendation accuracies. More details about HR and NDCG can be found in Subsection 3.3.1

Baselines. We compare the performance of our proposed STS-AEL framework with that of nine baseline models, including one ensemble model (i.e., OCFIF) and eight monolithic models (i.e., iBPR, iGMF, iMLP, iNeuMF, iTPMF-CF, RCD, eAls and SPMF). The brief introduction of these baselines are as follows.

- *Bayesian Personalised Ranking* (BPR) is a representative pair-wise ranking approach proposed by Rendle et al. [91] to optimise the matrix factorisation. We adapt this work to the streaming scenarios, named as **iBPR**, by incrementally training the recommendation model with newly coming data.

- *Neural Matrix Factorisation* (NeuMF) [37] is an advanced matrix factorisation model, which combines two recommendation models: *Generalized Matrix Factorisation* (GMF) and *Multi-Layer Perceptron* (MLP) for higher recommendation accuracies. We adapt these three offline recommendation models (i.e., NeuMF, GMF and MLP) to the streaming scenario, named as **iNeuMF**, **iGMF** and **iMLP**, respectively, by feeding them with newly coming data continuously.

- *Time-window based probabilistic Matrix Factorisation for Collaborative Filtering* (TPMF-CF) [146] is a representative probabilistic matrix factorisation based approach that adopts the time window technique to construct a 3D user-item-time model. As the TPMF-CF was originally devised for the offline scenarios, we adapt TPMF-CF to the streaming scenarios, named as **iTPMF-CF**, by incrementally training the recommendation model with newly coming data. Furthermore, to keep the features of the TPMF-CF, we also employ a sliding window based sampling approach to improve its recommendation accuracies.

- *Randomised block Coordinate Descent* (**RCD**) and *Element-wise Alternating Least Squares* (**eAls**) are two representative approaches employed by Devooght et al. [15] and He et al. [38], respectively, to optimise the streaming matrix factorisation. To be consistent with the proposed STS-AEL and other baselines, we enhance eAls and RCD with abilities of batch processing to increase their throughput.

- *Stream-centered Probabilistic Matrix Factorisation* (**SPMF**) [127] is a state-of-the-art monolithic SRS based on the probabilistic matrix factorisation, which improves the work in [116]. SPMF was originally performed along with a ranking-based sampling method. However, this ranking-based sampling method has excessive computation complexity since it evaluates all the user-item interactions in the reservoir and rank them by the test accuracies for each sampling process, and thus is not suitable for our evaluation policy where sampling needs to be performed frequently. Therefore, we sample data for SPMF with our proposed STS for fair comparisons.

- *Online Collaborative Filtering with Implicit Feedback* (**OCFIF**) [140] is the only ensemble SRS, which combines multiple matrix factorisation models to deliver more accurate streaming recommendations. It is specifically devised to ensemble matrix factorisation models, and thus cannot ensemble other models for the evaluation.

- *Stratified and Time-aware Sampling based Adaptive Ensemble Learning* (**STS-AEL**) is our proposed SRS. For the evaluation, we take each of the top three best-performing monolithic baselines — iNeuMF, iMLP and iGMF in Experiment 1 as its individual model and set different numbers (i.e., 2, 4, 6 and 8) of individual models to compose different forms of STS-AELs. For example, STS-AEL_8-iNeuMF indicates an STS-AEL framework ensembling 8 iNeuMF models.

Parameter Setting. For baselines, we initialise them with the parameters reported in their papers and tune them based on our experimental scenarios to achieve the best performance for fair comparisons. For our proposed STS-AEL, we empirically set the learning rate to 0.001, and initialise the parameters in embedding layers with the Gaussian distribution $X \sim N(0, 0.25)$, inner layers with Glorot initialisation [30] and the output layer with LeCun initialisation [61]. Besides, we adopt L_2 regularisation to avoid overfitting and Adaptive Moment Estimation (Adam) [53] for the regularisation. In addition, we manually adjust the parameters α, λ_{new} and λ_{res} based on the data receiving speed and the characteristics of the dataset to achieve the best performance. Without loss of generality, following the common practice in [127, 32], all the baselines and the proposed STS-AEL process data stream in batch to increase the throughput.

Table 4.1: Performance comparison for STS-AEL (results on MovieLens)

Dataset			MovieLens					
Metric			HR@10			NDCG@10		
Data Receiving Speed			128	256	512	128	256	512
Baseline	Monolithic	eAls	0.231	0.231	0.234	0.106	0.106	0.108
		RCD	0.287	0.297	0.278	0.139	0.145	0.138
		iBPR	0.303	0.303	0.279	0.147	0.147	0.134
		iTPMF-CF	0.408	0.401	0.426	0.222	0.216	0.229
		SPMF	0.472	0.466	0.434	0.262	0.259	0.234
		iGMF	0.525	0.529	0.477	0.295	0.297	0.265
		iMLP	0.538	0.539	0.488	0.304	0.303	0.272
		iNeuMF	<u>0.551</u>	<u>0.546</u>	<u>0.496</u>	<u>0.311</u>	<u>0.307</u>	<u>0.275</u>
	Ensemble	OCFIF	0.532	0.508	0.467	0.291	0.279	0.256
Our STS-AEL framework	STS-AEL_8-iGMF		0.592	0.584	0.590	0.344	0.340	0.344
	STS-AEL_8-iMLP		0.591	0.586	0.583	0.341	0.340	0.338
	STS-AEL_8-iNeuMF		**0.608**	**0.607**	**0.598**	**0.351**	**0.353**	**0.346**
Improvement percentage over the best-performing baseline			10.3%	11.2%	20.6%	12.9%	15.0%	25.8%

4.3.2 Performance Comparison and Analysis

Experiment 1: Performance Comparison with Baselines (for RQ1)

Setting. In this experiment, we take each of iGMF, iMLP and iNeuMF as the individual model of STS-AEL and set the number of individual models to eight for the evaluation. In addition, the proposed STS-AEL and nine baselines are evaluated on all three datasets with a fixed data processing speed, i.e., $sp_p = 256$, and three different data receiving speed, i.e., $sp_r = 128$ (simulating the underload scenario), $sp_r = 256$ (simulating the ideal case where the data processing speed is equal to the data receiving speed) and $sp_r = 512$ (simulating the overload scenario) to make comparisons in different cases.

Result. As Tables 4.1 to 4.3 show, in all the cases, STS-AEL_8-iNeuMF delivers the highest recommendation accuracies (marked with **bold** font), and the improvement percentages of STS-AEL_8-iNeuMF over the best-performing baseline (with the results marked by <u>underline</u>) in each case are introduced in the last rows, ranging from 4.0% (compared with iNeuMF on Netflix w.r.t. $sp_r = 256$) to 51.8% (compared with iMLP on Yelp w.r.t. $sp_r = 512$) with an average of 17.1% in terms of HR@10, and

Table 4.2: Performance comparison for STS-AEL (results on Netflix)

Dataset			Netflix					
Metric			HR@10			NDCG@10		
Data Receiving Speed			128	256	512	128	256	512
Baseline	Monolithic	eAls	0.395	0.389	0.362	0.211	0.207	0.192
		RCD	0.447	0.436	0.435	0.226	0.226	0.219
		iBPR	0.685	0.686	0.627	0.396	0.395	0.360
		iTPMF-CF	0.514	0.5231	0.540	0.290	0.298	0.309
		SPMF	0.699	0.676	0.636	0.425	0.410	0.374
		iGMF	0.747	0.748	0.577	0.482	0.482	0.352
		iMLP	0.787	0.782	0.624	0.519	0.510	0.369
		iNeuMF	0.801	0.798	0.711	0.531	0.529	0.443
	Ensemble	OCFIF	0.745	0.734	0.606	0.457	0.453	0.357
Our STS-AEL framework	STS-AEL_8-iGMF		0.789	0.783	0.785	0.524	0.517	0.518
	STS-AEL_8-iMLP		0.813	0.805	0.798	0.546	0.535	0.525
	STS-AEL_8-iNeuMF		**0.840**	**0.830**	**0.821**	**0.576**	**0.562**	**0.552**
Improvement percentage over the best-performing baseline			4.80%	4.00%	15.5%	8.40%	6.20%	24.6%

ranging from 6.2% (compared with iNeuMF on Netflix w.r.t. $sp_r = 256$) to 64.0% (compared with iMLP on Yelp w.r.t. $sp_r = 512$) with an average of 22.9% in terms of NDCG@10.

Analysis. The superiority of our proposed STS-AEL can be explained in the following three aspects: 1) the proposed STS addresses preference drift while capturing long-term user preferences by wisely incorporating both newly coming data and historical data through a stratified and time-aware strategy, 2) ensembling multiple individual models can not only avoid the limitations of monolithic models by complementing one another, but also contributes to mining user preferences more effectively by the concurrent training process, especially in overload scenarios, and 3) the proposed AEL improves fusion effectiveness by assigning elaborately calculated fusion weights to multiple individual models for effective fusion.

As Tables 4.1 to 4.3 show, OCFIF, which is the only existing ensemble SRS, delivers lower recommendation accuracies even than some monolithic baseline models, e.g., iGMF, iMLP and iNeuMF. This can be explained by the following three reasons: 1) OCFIF trains multiple individual models with the same data. This practice not

Table 4.3: Performance comparison for STS-AEL (results on Yelp)

Dataset			Yelp					
Metric			HR@10			NDCG@10		
Data Receiving Speed			128	256	512	128	256	512
Baseline	Monolithic	eAls	0.287	0.289	0.290	0.167	0.167	0.169
		RCD	0.454	0.452	0.447	0.260	0.257	0.259
		iBPR	0.307	0.295	0.188	0.180	0.172	0.108
		iTPMF-CF	0.172	0.174	0.186	0.089	0.090	0.098
		SPMF	0.203	0.203	0.177	0.107	0.106	0.091
		iGMF	0.499	0.470	0.396	0.294	0.276	0.228
		iMLP	0.573	0.574	0.438	0.338	0.338	0.246
		iNeuMF	0.566	0.570	0.435	0.331	0.334	0.247
	Ensemble	OCFIF	0.260	0.249	0.203	0.135	0.129	0.107
Our STS-AEL framework	STS-AEL_8-iGMF		0.647	0.614	0.600	0.399	0.371	0.362
	STS-AEL_8-iMLP		0.676	0.671	0.639	0.414	0.402	0.378
	STS-AEL_8-iNeuMF		**0.717**	**0.677**	**0.665**	**0.456**	**0.415**	**0.405**
Improvement percentage over the best-performing baseline			25.1%	17.9%	51.8%	34.9%	22.8%	64.0%

only reduces the data processing speed, which affects the sufficient training of models, but also harms the diversities of individual models, which is essential for effective ensemble learning, 2) OCFIF selects only one individual model for the final prediction, which does not fully utilise all the individual models to obtain more accurate recommendations and 3) OCFIF is specifically devised to ensemble matrix factorisation models, which only capture the linear relations with the dot product between the user embedding and the item embedding, while the aforementioned three models (i.e., iGMF, iMLP and iNeuMF) all capture the more complex nonlinear relations with nonlinear operations (e.g., sigmoid function), and thus they can more accurately learn the user preferences towards items.

Another observation from Tables 4.1 to 4.3 is that higher data receiving speed commonly leads to the degradation of the recommendation accuracies. Taking STS-AEL_8-iNeuMF on MovieLens as an example, the recommendation accuracies decrease 1.6% w.r.t. HR@10 and decrease 1.4% w.r.t. NDCG@10 when the data receiving speed increases from 128 to 512. The reason is that high data receiving speed impedes the recommendation models from sufficiently learning the user preferences towards the items from the data stream. Especially, the long-term user preferences cannot be well learned from the historical data, as the new data arrives at a high speed. Thus, this insufficient training process leads to the degradation of recommendation accuracies when the data receiving speed is high.

Summary. Our proposed STS-AEL significantly outperforms all the baseline models in all the cases, including the underload scenarios ($sp_r = 128$) and overload scenarios ($sp_r = 512$).

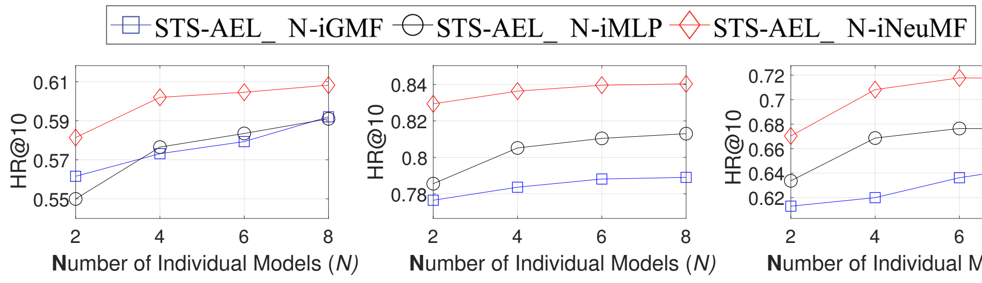
Experiment 2: Effect of the Number of Individual Models (for RQ2)

Setting. To answer **RQ2**, we allow our proposed STS-AEL ensembling different numbers (i.e., 2, 4, 6 and 8) of individual models for comparison. In this experiment, we report the results on all three datasets in both underload scenarios (i.e., the cases where $sp_r = 128$) and overload scenarios (i.e., the cases where $sp_r = 512$).

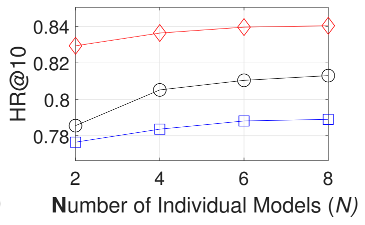
Result. As shown in Fig. 4.3, on all three datasets in both underload scenarios and overload scenarios, our proposed STS-AEL delivers higher recommendation accuracies when ensembling more individual models w.r.t. all three types of individual models, i.e., iNeuMF, iMLP and iGMF.

Analysis. The improvements when ensembling more individual models can be explained in two aspects: 1) more individual models can better complement one another through the fusion process when making recommendations, and 2) with the concurrent training process, the streaming data can be better exploited with more individual models, both when complemented by the historical data in the underload scenarios and when confronting the excessive amount of data in overload scenarios. In addition, It can be observed that the improvement ratios generally decrease as the number of individual models increases. This is because after having enough individual models to effectively complement one another and to sufficiently mine streaming data in parallel, getting more individual models will no longer increase the recommendation accuracies much.

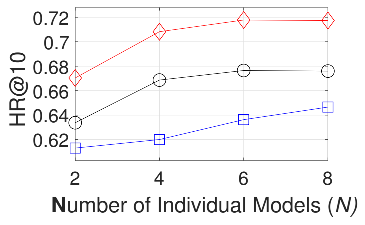
Since STS-AEL performs the best when ensembling eight individual models, as shown in Fig. 4.3, we set the number of individual models to eight in the following



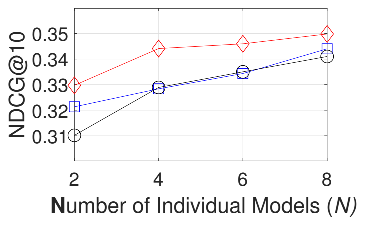
(a) HR@10 on MovieLens w.r.t. $sp_r = 128$



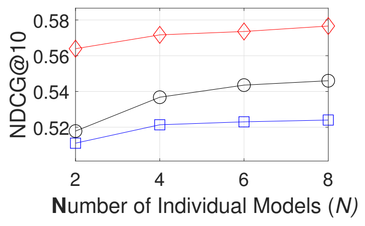
(b) HR@10 on Netflix w.r.t. $sp_r = 128$



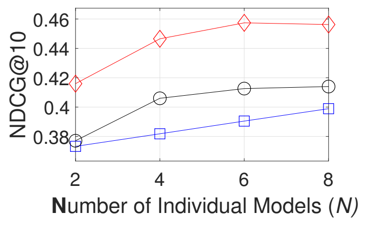
(c) HR@10 on Yelp w.r.t. $sp_r = 128$



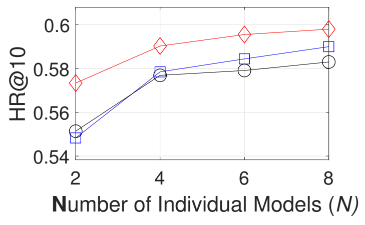
(d) NDCG@10 on MovieLens w.r.t. $sp_r = 128$



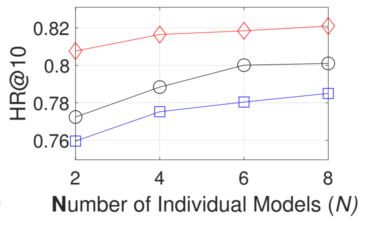
(e) NDCG@10 on Netflix w.r.t. $sp_r = 128$



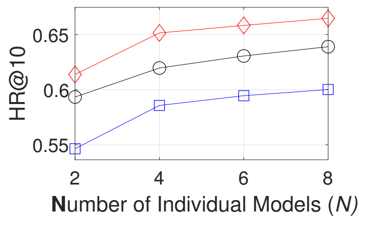
(f) NDCG@10 on Yelp w.r.t. $sp_r = 128$



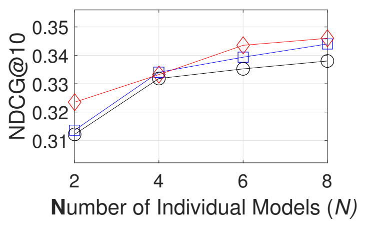
(g) HR@10 on MovieLens w.r.t. $sp_r = 512$



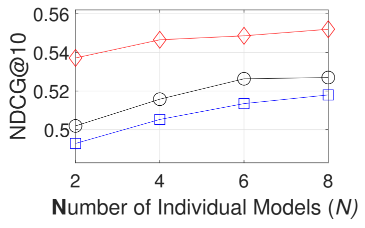
(h) HR@10 on Netflix w.r.t. $sp_r = 512$



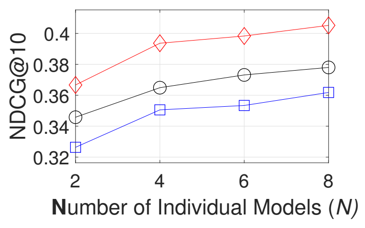
(i) HR@10 on Yelp w.r.t. $sp_r = 512$



(j) NDCG@10 on MovieLens w.r.t. $sp_r = 512$

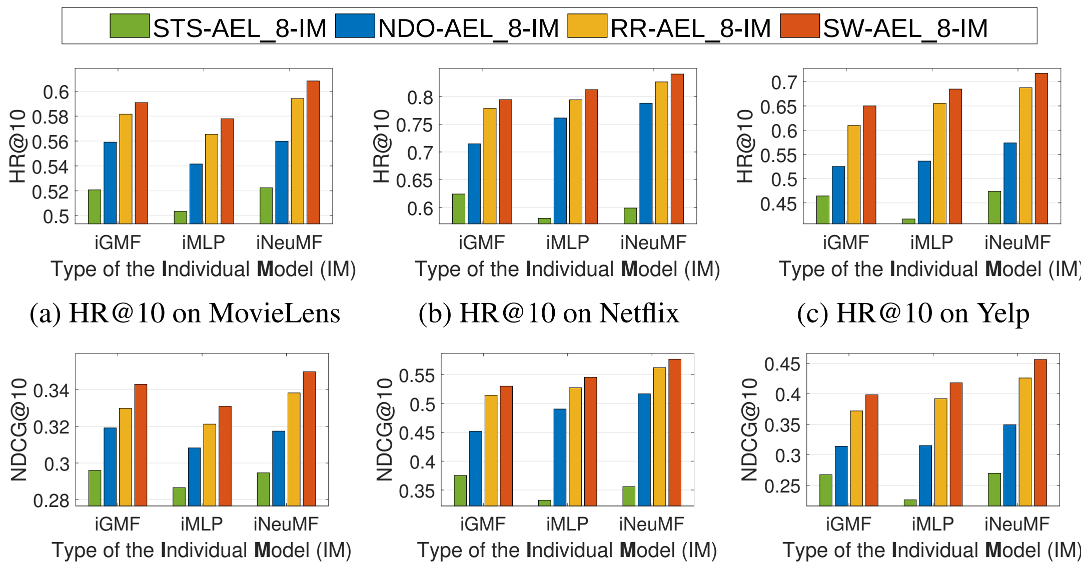


(k) NDCG@10 on Netflix w.r.t. $sp_r = 512$



(l) NDCG@10 on Yelp w.r.t. $sp_r = 512$

Figure 4.3: Effect of the number of individual models.



(a) HR@10 on MovieLens (b) HR@10 on Netflix (c) HR@10 on Yelp

(d) NDCG@10 on MovieLens (e) NDCG@10 on Netflix (f) NDCG@10 on Yelp

Figure 4.4: The superiority of STS

experiments to answer **RQ3** and **RQ4**.

Experiment 3: Superiority of STS (for RQ3)

Setting. To answer **RQ3**, we replace the proposed STS with three representative sampling methods for comparisons: Newly coming Data Only (NDO) [140] which only uses the newly coming data for the training purpose, Reservoir-enhanced Random sampling (RR) [16] which conducts random sampling with a reservoir and Sliding Window (SW) [102] which uses the latest k (a predetermined parameter) user-item interactions received for the training purpose. Similar to the naming scheme in the preceding experiments, we use *-AEL (e.g., NDO-AEL) to indicate which fusion method (e.g., NDO) is employed for the sampling purpose. In this experiment, we report the results on all three datasets in the underload scenarios (i.e., $sp_r = 128$) to show the superiority of STS while the results in the overload scenarios (i.e., $sp_r = 512$) are similar to the reported ones.

Result. As Fig. 4.4 indicates, our proposed STS consistently outperforms all the other sampling methods on all three datasets. Taking the individual model of iNeuMF as an example, with which our proposed approach achieves the best overall performance, the improvements of our proposed STS over the best-performing baseline — NDO, range from 1.7% (on Netflix) to 4.3% (on Yelp) with an average of 2.8% in terms of HR@10, and range from 2.6% (on Netflix) to 7.1% (on Yelp) with an average of 4.3% in terms of NDCG@10.

Analysis. The superiority of STS can be explained by elaborately incorporating both newly coming data and historical data in a time-aware manner while guaranteeing the proportion of newly coming data. Thus, STS can well address preference drift while capturing long-term user preferences. Besides, it can be observed that NDO, which takes newly coming data only to train the recommendation models, outperforms two other baselines: RR and SW, which both take the historical data into consideration. This indicates that improperly incorporating historical data may reduce the recommendation accuracies with our experimental settings. It is possibly caused by an insufficient emphasis on newly coming data, which hinders effectively addressing preference drift, and ineffectual sampling for historical data, which impedes effectively capturing long-term user preferences.

Experiment 4: Superiority of AEL (for RQ4)

Setting. To answer **RQ4**, we replace the sequential adaptive fusion in AEL with three representative fusion methods for comparisons; they are 1) Attentive Weighting (AttW) [96] which adopts an attention mechanism for the fusion, 2) AVeraGing (AVG) [83] which simply averages the results of the individual models and 3) AdaBoost-like Weighting (AdaW) [12] which considers the previous recommendation accuracies of individual models when conducting the fusion process. Similar to the naming scheme in the preceding experiments, we use STS-* (e.g., STS-AVG) to indicate which sampling method (e.g., AVG) is employed for the fusion purpose. In this experiment, we report the results on all three datasets in the overload scenarios (i.e. $sp_r = 512$) while the results in the underload scenarios (i.e. $sp_r = 128$) are similar to the reported ones.

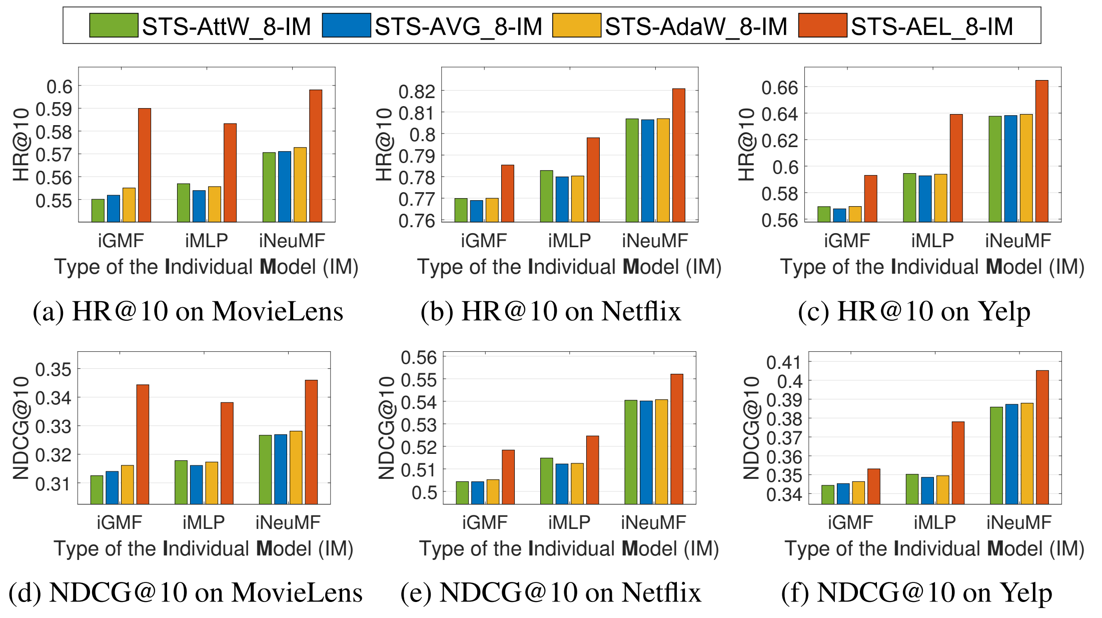


Figure 4.5: The superiority of AEL

Result. As Fig. 4.5 indicates, our proposed AEL significantly outperforms all the baselines on all three datasets. Taking the individual model of iNeuMF as an example, with which the ensembling approach achieves the best overall performance, the improvements over the best-performing baseline — AdaW, range from 1.7% (on Netflix) to 6.6% (on Yelp) with an average of 4.2% in terms of HR@10, and range from 2.2% (on Netflix) to 6.6% (on Yelp) with an average of 4.7% in terms of NDCG@10.

Analysis. The superiority of AEL mainly comes from the elaborately calculated fusion weights, which leads to effective fusion for higher recommendation accuracies. Specifically, when calculating the fusion weights, AEL not only considers the previous recommendation accuracies of individual models but also takes the specific characteristic of the target user-item pair into account. Besides, it can be observed that AttW does not deliver good performance in this experiment, and the possible reason is that AttW was originally devised for the mixture of expert model [96] and is not suitable for our proposed framework.

4.4 Chapter Summary

In this chapter, we have proposed a Stratified and Time-aware Sampling based Adaptive Ensemble Learning framework, called STS-AEL, for delivering accurate streaming recommendations. Our proposed STS-AEL addresses preference drift while capturing long-term user preferences through wisely sampling the newly coming data and historical data through a stratified and time-aware manner. Moreover, the incorporation of the sampled historical data also benefits addressing the underload problem. Furthermore, STS-AEL addresses the overload problem by first training the multiple individual models concurrently and then fusing the results of these trained models with a sequential adaptive fusion method. The extensive experiments show that the proposed STS-AEL significantly outperforms the state-of-the-art approaches. In addition, the effectiveness of the two main components: Stratified and Time-aware Sampling (STS) and Adaptive Ensemble Learning (AEL), have also been explicitly verified by the experiments.

Chapter 5

MbSRS: a Multi-behaviour Streaming Recommender System

Various SRSs have been proposed to deliver recommendations in the streaming scenarios. However, all the existing SRSs are devised to deal with a single behaviour type (e.g., purchases). These single-behaviour SRSs are restricted by the limited number of such single-behaviour interactions, and thus commonly suffer from the data sparsity problem. Therefore, the more sufficient multi-behaviour interactions (e.g., purchases, add-to-carts and views) might be utilised for more accurate streaming recommendations. However, delivering streaming recommendations with multi-behaviour interactions leads to another new challenge without any solution reported in the literature. This has been introduced in Subsection 1.2.4 and expressed by **CH4**: 'how to wisely leverage multi-behaviour interactions for improving the accuracies of streaming recommendations'.

Targeting **CH4**, this chapter proposes the *first* Multi-behaviour Streaming Recommender System, called MbSRS, for delivering more accurate streaming recommendations by utilising data streams of multi-behaviour interactions. Specifically, MbSRS contains the following three key components.

1) Multi-behaviour Learning Module (MbLM), which not only leverages shared embeddings to accurately learn shared user preferences and shared item characteristics across multiple behaviour types, but also employs behaviour-specific embeddings to learn behaviour-specific user preferences and behaviour-specific

item characteristics. Moreover, this module is trained with the newly coming interaction data, and thus is able to capture the latest user preferences.

2) Attentive Memory Network (AMN), which effectively maintains the long-term user preferences w.r.t. the primary behaviour. Specifically, AMN first memorises the items interacted (w.r.t. all behaviour types) by users in a memory. Then, AMN elaborately represents the long-term user preferences with these memorised items using an attentive method to emphasize those items more relevant to the target item. Specifically, this attentive method first learns the weights of memorised items to indicate their relevance scales with the target item. After that, with these calculated weights, the memorised items are elaborately combined to represent the long-term preferences of the target user for the target item

3) User Preference Merging Module (UPMM), which wisely merges the short-term user preferences and long-term user preferences with a gate mechanism. Specifically, our proposed UPMM employs two gates to adaptively calculate the gate weights of short-term user preferences and long-term user preferences, respectively, based on the embeddings of the target user and target item. Then, the short-term user preferences and long-term user preferences are merged based on these gate weights. Through this way, UPMM not only considers the different contributions of short-term and long-term user preferences via these gate weights, but also is more specialised in the target user and the target item during the preference merging process.

The remainder of this chapter is organized as follows. We first formulate the research problem studied in this chapter. Then, we introduce our proposed MbSRS and its three key components (i.e., MbLM, AMN and UPMM). After that, we present the results of the experiments that are conducted to verify the effectiveness of MbSRS. Finally, we summarise this chapter.

5.1 Problem Statement

In this section, we formulate our research problem of streaming recommendations w.r.t. multi-behaviour interactions as follows. First, we utilise $U = \{u_1, u_2, ..., u_{|U|}\}$, $V = \{v_1, v_2, ..., v_{|V|}\}$, and $B = \{b_1, b_2, ..., b_n\}$ ($n = |B|$) to represent the user set, the item set and the set of behaviour types, respectively. Note that, for $b_i \in B$, we use the subscript 1 to $(n-1)$ for denoting the first type of auxiliary behaviour to the $(n-1)^{th}$ type of the auxiliary behaviour, respectively, and use the subscript n for denoting the primary behaviour. Moreover, these subscripts also indicate the priority levels of the corresponding behaviour types among the total behaviour types in terms of to what degree these behaviour types reflect user preferences towards to an item. Taking the behaviour types in online shopping as an example, purchase behaviour can best reflect user preferences and thus is always assigned to the largest subscript (i.e., *n*). Similarly, the view behaviour is usually with the smallest subscript (i.e., 1) and the add-to-cart behaviour is usually with a subscript between 1 and n.

With the notations presented above, an interaction between user u and item v w.r.t. behaviour type b is denoted as $t = \{u, v, b\}$, where $u \in U$, $v \in V$ and $b \in B$. Consequently, a data stream of multi-behaviour interactions is denoted as $s = \{t^1, t^2, ..., t^k, ...\}$, where $t^k = \{u^k, v^k, b^k\}$. Note that the order of these interactions in each data stream is determined by their receiving timestamps and marked by their superscripts (e.g., t^k indicates the k^{th} received interaction). Moreover, similar to the real-world cases, adjacent interactions are not necessarily from the same user (e.g., user u^k is possibly different from user u^{k+1}). In this work, without loss of generality, we consider the implicit behaviours (e.g., purchases and views) only, which are more common and more difficult to deal with than the explicit ones (e.g., ratings).

Then, given the currently received data stream s' of multi-behaviour interactions, the task of a multi-behaviour SRS is to predict the probability $\hat{y}$ of a future interaction t' w.r.t. the primary behaviour; that is, $\hat{y} = P(t'|s')$. Similar to the single-behaviour SRSs, multi-behaviour SRSs should learn both short-term user preferences and long-term user preferences to deliver accurate streaming recommendations. Moreover, the multi-behaviour SRSs are also expected to well exploit the interactions w.r.t. auxiliary behaviours while eliminating their negative interference for further increasing the accuracies of streaming recommendations. Thus, our research problem of streaming recommendations w.r.t. data streams of multi-behaviour interactions is challenging.

5.2 Our Proposed Multi-behaviour Recommender System

In this section, we propose the *first* Multi-behaviour Streaming Recommender System, called MbSRS. Specifically, we first present its overall structure. Then, we introduce its three key components: Multi-behaviour Learning Module (MbLM), Attentive Memory Network (AMN) and User Preference Merging Module (UPMM). Finally, we introduce its prediction process and training process.

5.2.1 Overall Structure

Aiming at delivering accurate streaming recommendations w.r.t. data streams of multi-behaviour interactions, we propose the first Multi-behaviour Streaming Recommender System, called MbSRS. As Fig. 5.1 illustrates, our proposed MbSRS contains three key components: MbLM, AMN and UPMM, along with the prediction process and the training process. The details about these three components are presented in Subsections 5.2.2 to 5.2.4, respectively, while the prediction process and the training process are introduced in Subsection 5.2.5. Here, we provide a brief introduction for each of them as follows.

Firstly, MbLM accurately learns short-term user preferences and stable item characteristics via both the shared embeddings across multiple behaviour types and behaviour-specific embeddings of users and items, respectively. Secondly, AMN effectively maintains the long-term user preferences w.r.t. the primary behaviour by first mem-

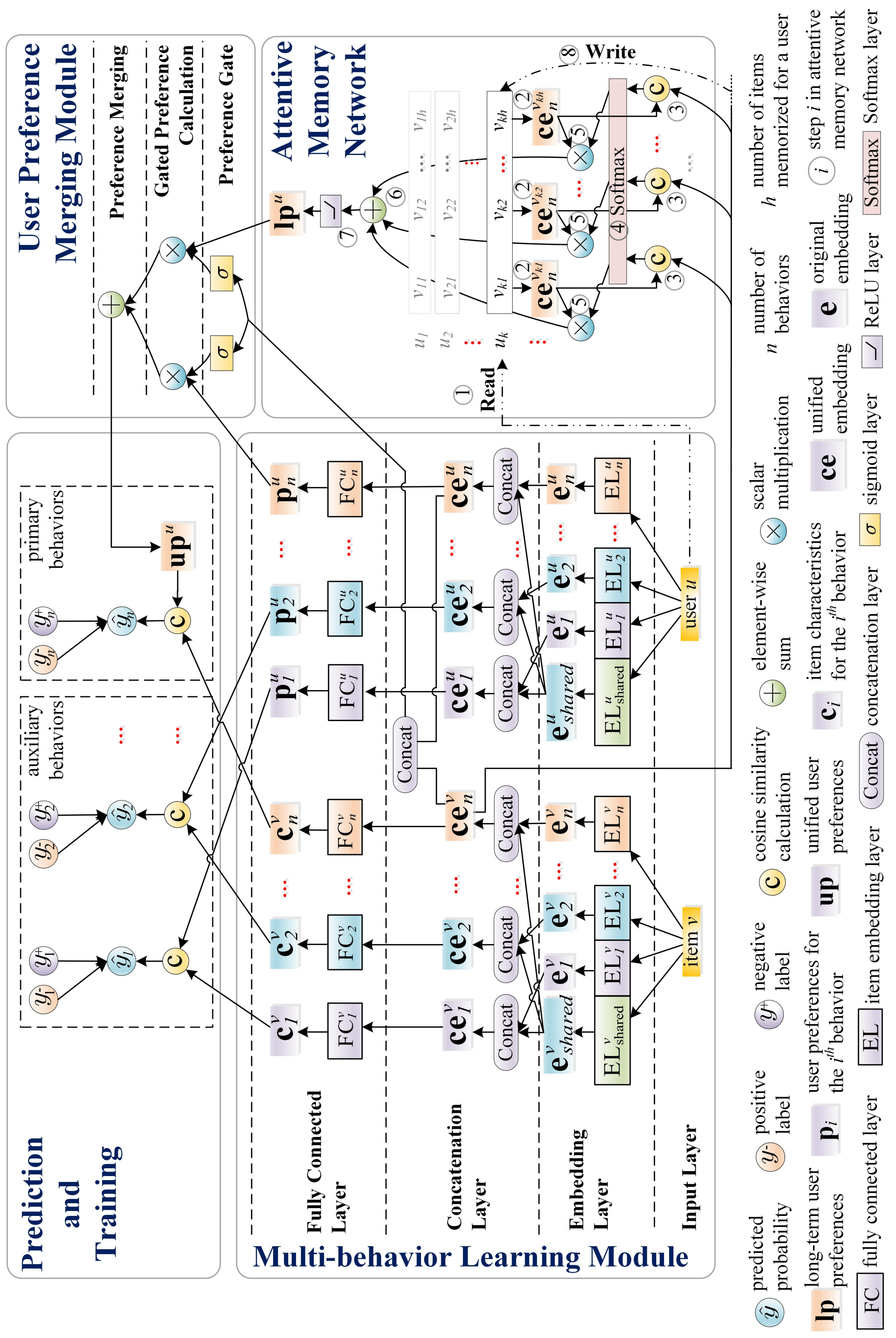


Figure 5.1: The structure of MbSRS.

orising the items that interacted by users in the memory, and then representing the long-term preferences of the target user with these memorised items via an attentive method. After that, UPMM elaborately merges the short-term user preferences learnt by MbLM and the long-term user preferences learnt by AMN into the unified user preferences w.r.t. the primary behaviour. Finally, the predictions are conducted based on the user preferences and item characteristics learnt by the above three components. As for the training process, Stochastic Gradient Descent (SGD) is employed to train MbSRS with newly coming interaction data in a timely manner.

5.2.2 Multi-behaviour Learning Module

We propose MbLM to accurately learn the short-term user preferences and stable item characteristics. To achieve this purpose, MbLM first learns the shared embeddings across multiple behaviour types and behaviour-specific embeddings for both users and items. After that, these shared embeddings and behaviour-specific embeddings for users (or items) are concatenated and then transformed with fully connected layers to obtain both the shared user preferences (or item characteristics) and the behaviour-specific user preferences (or item characteristics). Note that, in MbLM, we not only devise the shared layers to enhance the learning process for user preferences (or item characteristics) w.r.t. the primary behaviour by that w.r.t the auxiliary behaviours, but also devise the behaviour-specific layers to highlight the unique user preferences (or item characteristics) w.r.t. each behaviour type.

Input Layer. MbLM takes data streams of multi-behaviour interactions as the input. For each interaction, the target user and target item serve as the direct input to learn user preferences and item characteristics for streaming recommendations, while the target behaviour type is utilised to activate the corresponding behaviour-specific layers for learning such preferences and characteristics w.r.t. this target behaviour type. For better readability, we utilise the subscripts in the names to indicate which behaviour-specific layers are activated by the target behaviour type. Taking an embedding layer with the name of EL_i as an example, it will be activated when an interaction w.r.t. the i^{th} behaviour type is received.

Embedding Layer. We not only devise shared user (or item) embedding layers to learn the shared user (or item) embeddings for reinforcing the learning process of user prferences (or item characteristics) among multiple behaviour types, but also deivse behaviour-specific user (or item) embedding layers to learn the behaviour-specific user (or item) embeddings to reflect the unique latent features of users (or items) w.r.t. each behaviour type. Specifically, taking the user embeddings w.r.t. the i^{th} behaviour type as an example, shared user embedding $\mathbf{e}^u_{shared}$ and behaviour-specific user embedding $\mathbf{e}^u_i$ for user u are learnt by the shared user embedding layer EL^u_{shared} and the behaviour-specific user embedding layer EL^u_i, respectively. Similarly, the shared item embedding $\mathbf{e}^v_{shared}$ and the behaviour-specific item embedding $\mathbf{e}^v_j$ for item v w.r.t. the j^{th} behaviour type are learnt by the shared item embedding layer EL^v_{shared} and the behaviour-specific item embedding layer EL^v_i, respectively.

Concatenation Layer. This layer concatenates the shared user embedding (or item embedding) and the corresponding behaviour-specific user embedding (or item embedding) to incorporate both shared latent features and behaviour-specific latent features of users (or items). For example, the concatenated user embedding $\mathbf{ce}^u_i$ of user u w.r.t. the i^{th} behaviour type is obtained as follows,

$$\mathbf{ce}^u_i = \left[\mathbf{e}^u_{shared}; \mathbf{e}^u_i\right]. \tag{5.1}$$

With a similar calculation process to the one presented in Eq. (5.1), the concatenated item embedding $\mathbf{ce}^v_j$ of item v w.r.t. the j^{th} behaviour type can be obtained as follows,

$$\mathbf{ce}^v_j = \left[\mathbf{e}^v_{shared}; \mathbf{e}^v_j\right]. \tag{5.2}$$

Fully Connected Layer. In this layer, the concatenated user embeddings $\mathbf{ce}^u$ (or item embeddings $\mathbf{ce}^i$) are taken as the input to learn both the shared preferences (or characteristics) and the behaviour-specific preferences (or characteristics) via a fully connected layer. Taking the user preference $\mathbf{p}_i^u$ of user u w.r.t. the i^{th} behaviour type as an example, this fully connected layer can be experssed as follows,

$$\mathbf{p}_i^u = a_i^{pref}(\mathbf{W}_i^{pref}\mathbf{ce}_i^u + \mathbf{b}_i^{pref}), \tag{5.3}$$

where a_i^{pref}, $\mathbf{W}_i^{pref}$ and $\mathbf{b}_i^{pref}$ denote the activation function, weight matrix and the bias vector, respectively. With similar notations, the item characteristic $\mathbf{q}_j^v$ w.r.t. the j^{th} behaviour type is calculated as follows,

$$\mathbf{q}_j^v = a_i^{char}(\mathbf{W}_i^{char}\mathbf{ce}_j^v + \mathbf{b}_i^{char}). \tag{5.4}$$

Note that we have tried different numbers of fully connected layers for learning these user preferences and item characteristics in this step, but found that more layers can hardly improve the overall recommendation accuracies and introduce more computational complexities. Therefore, we employ a single fully connected layer here to perform this transformation process.

In addition, notations a, $\mathbf{W}$ and $\mathbf{b}$ are also utilised in the remainder of this chapter for the similar meaning with different subscripts and superscripts to represent fully connected layers. Besides, MbLM is trained by SGD with the newly coming interaction data to ensure that the cpatured user preferences are the latest. As for the item characteristics, they are relatively stable in essential, and thus can be well learend from the newly coming interaction data.

5.2.3 Attentive Memory Network

Besides the short-term user preferences captured by MbLM, the long-term user preferences w.r.t. the primary behaviour should also be effectively maintained for delivering accurate streaming recommendations. Therefore, we propose AMN to first memorise the items that are interacted by users in a key-value memory, where the users serve as the keys and the corresponding items serve as the values of the corresponding users.

After that, the long-term preferences w.r.t. the primary behaviour of the target users are represented by their interacted items memorised in the memory via an attentive method. Specifically, AMN utilises the following eight steps to maintain the long-term user preferences.

Step 1. The AMN module is activated when an interaction w.r.t. the primary behaviour is received. Specifically, the memorised items are read with the key of the target user u in this received interaction. Note that, in this step, the memorised items interacted by the target user w.r.t. all the behaviour types are read to more effectively represent the long-term user preferences.

Step 2. Then, these read items are transformed into their concatenated embeddings (which contain both the shared embeddings and behaviour-specific embeddings, see Subsection 5.2.2 for more details) w.r.t. the primary behaviour to serve as the input for the following steps.

Step 3. After that, we assign higher weights to the read items that are more similar to the target item for emphasizing more on these similar items. Specifically, we measure such similarities with the cosine similarities between the embeddings (the embeddings all refer to the concatenated embeddings w.r.t. the primary behaviour in this step) of these read items and the embedding of the target item. Taking the similarity s^i between the embedding $\mathbf{ce}^i$ of the i^{th} read item and the embedding $\mathbf{ce}'$ of the target item as an example,

$$s^i = cosine\_similarity(\mathbf{ce}^i, \mathbf{ce}') = \frac{(\mathbf{ce}^i)^T \mathbf{ce}'}{||\mathbf{ce}^i||_2 ||\mathbf{ce}'||_2}, \tag{5.5}$$

where $(*)^T$ and $|| * ||_2$ denote the transposition and L_2 norm, respectively, of a vector.

Step 4. These calculatd similarities are then fed to a softmax layer to obtain normalized similarities, taking the normalised similarity ns^i in terms of the i^{th} memorised item as an example,

$$ns^i = \frac{e^{(s^i)}}{\sum_{j=1}^{ms} e^{(s^j)}} \tag{5.6}$$

where ms denotes the memory size and e is the natural logarithm.

Step 5. After that, these normalised similarities are leveraged for calculating the weighted item embeddings. For example, the weighted item embedding $\mathbf{we}^i$ of the i^{th} read item is calculated as follows,

$$\mathbf{we}^i = ns^i \otimes \mathbf{ce}^i, \tag{5.7}$$

where $\otimes$ denotes the scalar multiplication.

Step 6. These weighted item embeddings are then summed up to obtain the unified item embedding **ue** as follows,

$$\mathbf{ue} = \sum_{i=1}^{h} \mathbf{we}^i. \tag{5.8}$$

Through this way, this unified embedding contains the overall latent features of the memorised items for the target user.

Step 7. Finally, this unified item embedding **ue** is passed to a fully connected layer for obtaining the long-term user preference $\mathbf{lp}^u$ of the target user u w.r.t. the primary behaviour,

$$\mathbf{lp}^u = a_{long}(\mathbf{W}_{long}\mathbf{ue} + \mathbf{b}_{long}). \tag{5.9}$$

Step 8. The last step of AMN is to write the target item to the memory with the target user as the key for future usage. The memory size — the maximal number of items w.r.t. each user can be memorised — is dynamically set based on the features of data streams for achieving the best performance. For maintaining this memory, AMN adopts a First-In-First-Out (FIFO) strategy where the earliest memorised item will be discarded when the number of the memorised items exceeds the memory size. In the future, we will study more advanced strategies for maintaining this memory.

5.2.4 User Preference Merging Module

In this section, we propose UPMM to elaborately merge the short-term user preferences learnt from MbLM and the long-term user preferences learnt from AMN both w.r.t. the primary behaviour. Although the short-term user preferences and long-term user preferences are both essential for delivering accurate recommendations, it is not an easy task to merge these two types of preferences. This is because the contributions of short-term preferences (or long-term preferences) might be different for different users, and even different for the same user towards different items. To address this issue, UPMM conducts this merging process via a gate mechanism. Specifically, UPMM first employs embeddings of both target users and target items to dynamically calculate the gate weights, and then leverages the calculated gate weights for effectively merging these two types of preferences. More details are presented in the below.

Preference Gate. First, the concatenated embedding $\mathbf{ce}_n^u$ of the target user u and the concatenated embedding $\mathbf{ce}_n^v$ ($\mathbf{ce}_n^u$ and $\mathbf{ce}_n^v$ are both learnt in MbLM, see Subsection 5.2.2 for more details) of the target item v w.r.t. the primary behaviour are further concatenated as follows,

$$\mathbf{ce}'_{<u,v>} = \left[\mathbf{ce}_n^u ; \mathbf{ce}_n^v\right] . \tag{5.10}$$

Then, inspired by the cell structure in LSTM, UPMM calculates the gate weights gw_{short}^u and gw_{long}^u for the short-term user preferences and long-term user preferences to reflect their contributions to the overall user preferences. Specifically, gw_{short}^u and gw_{long}^u are calculated by the following Eq. (5.11) and Eq. (5.12), respectively,

$$gw_{short}^u = sigmoid(\mathbf{W}_{short}^{gate}\mathbf{ce}'_{<u,v>} + \mathbf{b}_{short}), \tag{5.11}$$

$$gw_{long}^u = sigmoid(\mathbf{W}_{long}^{gate}\mathbf{ce}'_{<u,v>} + \mathbf{b}_{long}), \tag{5.12}$$

where the sigmoid function serves as the activation function.

Gated Preference Calculation. With these gate weights, the gated short-term preference $\mathbf{gp}_{short}^u$ and the gated long-term preference $\mathbf{gp}_{long}^u$ of user u are calculated by the following Eq. (5.13) and Eq. (5.14), respectively,

$$\mathbf{gp}_{short}^u = gw_{short}^u \otimes \mathbf{p}_n^u, \tag{5.13}$$

$$\mathbf{gp}_{long}^u = gw_{long}^u \otimes \mathbf{lp}^u, \tag{5.14}$$

where the short-term preference $\mathbf{p}_n^u$ and the long-term preference $\mathbf{lp}^u$ are learnt by Eq. (5.3) in MbLM and Eq. (5.9) in UPMM, respectively.

Preference Merging. Finally, the unified user preference $\mathbf{up}^u$ of user u is obtained by summing $\mathbf{gp}_{short}^u$ and $\mathbf{gp}_{long}^u$ up as follows,

$$\mathbf{up}^u = \mathbf{gp}_{short}^u \oplus \mathbf{gp}_{long}^u, \tag{5.15}$$

where $\oplus$ denotes the element-wise addition. This unified user preferences are later be utilised to recommend items to the users w.r.t. the primary behaviour in the following Subsection 5.2.5.

5.2.5 Prediction Process and Training Process

In this subsection, we introduce the prediction process and training process of MbSRS. Our proposed MbSRS deliver streaming recommendations w.r.t. the primary behaviour by predicting the probabilities of interactions w.r.t. the primary behaviour. In addition, we also allow MbSRS to predict the probabilities of interactions w.r.t. auxiliary behaviours as well as train MbSRS using these interactions. This practice benefits improving the accuracies of streaming recommendations w.r.t. the primary behaviour, as exploiting the more sufficient additional interactions w.r.t. auxiliary behaviours help more accurately learn the shared user preferences and shared item characteristics.

5.2.5.1 Prediction Process

The prediction is conducted by learning the relevance scales between the preferences of the target users and the characteristics of the target items. In this chapter, we utilise the cosine similarity to measure this relevance scale for calculating the predicted probabilities of interactions. Note that user preferences w.r.t. the primary behaviour and the auxiliary behaviours are calculated in two different ways. Specifically, the user preference of user u w.r.t. the primary behaviour is represented by $\mathbf{up}^u$ calculated through Eq. (5.3) in UPMM, while the preference of user u w.r.t. the k^{th} $(1 \leq k \leq n-1)$

type auxiliary behaviour is represented by $\mathbf{p}_k^u$ calculated through Eq. (5.15) in MbLM. Differently, the item characteristic $\mathbf{q}_k^v$ of item v w.r.t. the k^{th} ($k = n$ for the primary behaviour) behaviour type is calculated by Eq. (5.4) in MbLM for both the primary behaviour and auxiliary behaviours.

With these well learnt user preferences and item characteristics, the predicted probability $\hat{y}_n^{<u,v>}$ of the interaction between user u and item v w.r.t. the primary behaviour is calculated by the following equation,

$$\hat{y}_n^{<u,v>} = \frac{(\mathbf{up}_u)^T \mathbf{q}_v^n}{\|\mathbf{up}_u\|_2 \|\mathbf{q}_v^n\|_2}. \tag{5.16}$$

In a similar way, the predicted probability $\hat{y}_k{}^{<u,v>}$ of the interaction between user u and item v w.r.t. the k^{th} ($1 \leq k \leq n-1$) type of auxiliary behaviour is calculated as follows,

$$\hat{y}_k^{<u,v>} = \frac{(\mathbf{p}_u^k)^T \mathbf{q}_v^k}{\|\mathbf{p}_u^k\|_2 \|\mathbf{q}_v^k\|_2}. \tag{5.17}$$

Then, the recommendations are performed based on these predicted probabilities. For example, item v might be recommended to user u w.r.t. the primary behaviour b if an interaction of $< u, v, b >$ gets a high predicted probability.

5.2.5.2 Training Process

MbSRS is trained with the interactions related to each behaviour type independently to learn from the continuous data stream effectively. To be more specific, 1) when an interaction w.r.t. the primary behaviour is received, AMN, UPMM and the neural network layers related to the primary behaviour in MbLM will be trained; and 2) when an interaction w.r.t. the i^{th} type of auxiliary behaviour is received, only the neural network layers related to the i^{th} type of auxiliary behaviour in MbLM will be trained. Through this independent training process, compared with the joint training process employed in some literature [28, 77], the parameters in MbSRS can be updated once an interaction is received and thus performs more accurate streaming recommendations.

We employ the cross-entropy loss to measure the differences between the predicted probabilities and the corresponding real labels for interactions. For example, the cross-entropy loss $\ell oss_k^{<u,v>}$ w.r.t. the interaction of $< u, v, k >$ is calculated as follows,

$$\ell oss_k^{<u,v>} = -(y_k^{<u,v>} log(\hat{y_k}^{<u,v>}) + (1 - y_k^{<u,v>}) log(1 - \hat{y_k}^{<u,v>})), \tag{5.18}$$

where $\hat{y_k}^{<u,v>}$ denotes the predicted probability of the interaction between user u and item v w.r.t. the k^{th} $(1 \leq k \leq n)$ behaviour type, while $y_k^{<u,v>}$ represents the real label of this interaction; that is $y_k^{<u,v>}$ is 1 if this interaction exists and 0 otherwise.

Then, SGD is employed to minimise the above loss by training MbSRS once a new interaction is received. Through this way, our proposed MbSRS is able to capture the latest user preferences towards items, and thus make more accurate streaming recommendations.

In addition, inspired by the work in [37, 38, 32], we employ the negative sampling strategy to more effectively train our proposed MbSRS. Specifically, for each target interaction, the corresponding negative interactions are sampled from both the unobserved interactions and interactions w.r.t. behaviour types that are with lower priority levels than the target behaviour type (e.g., views $<$ add-to-carts $<$ purchases). Then, these negative interactions are utilised to more effectively train MbSRS along with the target one.

5.3 Experiments

In this section, we present the results of extensive experiments that are conducted to answer the following five Research Questions (RQs).

RQ1: How does our proposed MbSRS perform when compared with the state-of-the-art baselines?

RQ2: How does each component in our proposed MbSRS contribute to the performance improvement?

Table 5.1: Statistics of the tuned datasets for MbSRS

Dataset	**Beibei**	**Taobao**	**Tmall**
#Total	2,108,192	442,254	735,356
#Purchase	166,883	96,237	179,035
#Add-to-cart	n.a.	30,203	n.a.
#Favorite	n.a.	11,484	n.a.
#Collect	n.a.	n.a.	24,681
#View	1,941,309	304,330	531,640
#User	10,000	13,777	12,921
#Item	49,488	27,652	22,570

The symbol # in this table denotes the number (e.g., #item represents the number of items).

RQ3: How do the sizes of embeddings affect the recommendation accuracies?

RQ4: How does the memory size affect the recommendation accuracies?

RQ5: How does the number of incorporated behaviour types affect the recommendation accuracies?

5.3.1 Experimental Settings

Datasets. For the experiments, we employ three real-world datasets, including Beibei, Taobao and Tmall, all of which are widely utilised in the literature [19, 21, 7]. These three datasets are suitable for evaluating our proposed MbSRS and baselines. This is because these datasets all contain more than one behaviour types and provide the timestamps of the interactions, and thus can be utilised to simulate data streams of multi-behaviour interactions for the evaluation of streaming recommender systems. Note that real-world datasets that contain both multi-behaviour interactions and timestamps are not common, and these three datasets are the most suitable ones we have found for our experiments. More details about these three datasets are as follows.

1) Beibei[1] provided and tunned by Ding et al. [19]. It records interactions w.r.t.

[1]https://github.com/dingjingtao/NegativeSamplerBPR/tree/master/BPRplusView/data

the purchase and view from Beibei, which is a popular Chinese E-commerce platform for maternal and infant products.

2) Taobao[2] released on Tianchi by Alibaba. This dataset contains interactions w.r.t. the purchase, add-to-cart, favourite and view collected from Taobao, which is another popular Chinese E-commerce platform.

3) Tmall[3] tuned by Ding et al. [21] based on the original version that is released for the IJCAI-15 competition. This dataset records interactions w.r.t. the purchase, collect and view from Tmall, i.e., the Chinese version of Amazon.

As processing the original Taobao dataset, which contains more than 100 million interactions, is beyond our computational capacity, we tune this dataset with the method that is utilised in [19, 21] for tunning the Beibei dataset and Tmall dataset. Specifically, we first merge the repetitive purchases between the same user and the same item using the earliest timestamp, and then filter out the users' auxiliary behaviours involving their purchased items to avoid the information leak. Moreover, following the practice in [150, 38], we also filter out the users and items that are not involved in an interaction w.r.t. the primary behaviour (i.e., purchase), and then extract the interactions of users who conduct more than ten purchases and of items that are involved in more than one purchases. The statistics of the tunned datasets are presented in Table 5.1.

Evaluation Policy. To simulate data streams of multi-behaviour interactions for the evaluation of our proposed MbSRS, following the work in [150, 38], we first sort the interactions based on their timestamps in the ascending order for each of the aforementioned three datasets. After that, we divide each dataset into a training set (which simulates the historical interactions used to incrementally train recommendation models) and a test set (which simulates the upcoming interactions first used for the evaluation and then used to incrementally train recommendation models). Specifically, we select the first 85%, 90% and 95% of interactions from each dataset as training sets,

[2]https://tianchi.aliyun.com/dataset/dataDetail?dataId=649

[3]https://github.com/dingjingtao/Auxiliary_enhanced_ALS/tree/master/data/tmall

while the remainder serves as the corresponding test sets. Note that the k-fold cross validation is infeasible in streaming scenarios, as only the latest interactions (i.e., the interactions with the largest timestamps) can be utilised for the test to be consistent with the data coming order. In addition, following the work in [38, 150], we report the results in terms of 90% split only for saving space, while the results in terms of other cases are similar to the reported ones.

Evaluation Metrics. Following [37, 38], we employ the widely-used ranking-based evaluation strategy in our experiments. Specifically, given each interaction between a target user and a target item w.r.t. the primary behaviour, we first sample 99 items that are not involved in the interactions w.r.t. the primary behaviour conducted by the target user as negative items. Then, we rank the target item among these 100 items (i.e., the target item plus the 99 negative items). Finally, the recommendation accuracies are measured with Hit Ratio (HR) and Normalised Discounted Cumulative Gain (NDCG) In this chapter, we utilize HR@5, HR@10, NDCG@5, and NDCG@10 to observe the performance of our proposed MbSRS and that of baselines. More details about HR and NDCG can be found in Subsection 3.3.1

Comparison Approaches. We have employed eight baselines, including six single-behaviour SRSs (i.e., SPMF, iGMF, iMLP, iNeuMF, iDMF and DWMoE) and two multi-behaviour offline RSs (i.e., DCMF and NMTR), for the comparisons with our proposed MbSRS. Note that no multi-behaviour SRSs are available for the comparisons, as our proposed MbSRS is the first SRS to exploit multi-behaviour interactions for streaming recommendations. To evaluate the single-behaviour SRSs w.r.t. data streams of multi-behaviour interactions, we extract the interactions w.r.t. the primary behaviour only for both the training process and test process of these single-behaviour SRSs. As for the multi-behaviour offline SRSs, we adapt them to streaming scenarios while keeping their features described in their papers. The brief introduction of these eight baselines is as follows.

- *Stream-centered Probabilistic Matrix Factorisation* (**SPMF**) [127] is a representative SRS that is based on the probabilistic matrix factorisation model. The original SPMF is trained along with a ranking-based sampling strategy. However, this ranking-based sampling strategy causes long delays during the training process, and thus is not suitable for our experimental settings, where the recommendations need to be performed in a timely manner. Therefore, similar to the practice in [150], we train SPMF with newly coming interaction data in a timely manner.

- *Incremental Neural Matrix Factorisation* (**iNeuMF**), *Incremental Generalized Matrix Factorisation* (**iGMF**) and *Incremental Multiple Layer Perception* (**iMLP**) are three representative SRSs devised in [150] to perform streaming recommendations. Note that these three SRSs are adapted from NeuMF, GMF and MLP, respectively, all of which are originally proposed in [37]. Specifically, these transformed approaches are trained by SGD to learn from the newly coming interaction data incrementally in the streaming scenarios.

- *Incremental Deep Matrix Factorisation* (**iDMF**) is the streaming version of Deep Matrix Factorisation (DMF) [138], which employs a double-wing MLP structure to better learn the user-item relations from the user-item interactions. We train iDMF with newly coming data incrementally by SGD for streaming recommendations.

- *Double-wing Mixture of Experts* (**DWMoE**) [150] is a state-of-the-art SRS that employs the mixture of experts to accurately learn user preferences and item characteristics for more accurate recommendations. Note that, in its original paper, DWMoE is trained using different methods in different scenarios, while we train DWMoE with newly coming interaction data only for fair comparisons.

- *Deep Collective Matrix Factorisation* (**DCMF**) [77] is a state-of-the-art multi-behaviour offline RS that enhances CMF [100] with the deep learning technique, and thus can capture the complex and non-linear user-item relations from multi-behaviour interactions. The original DCMF is devised to be jointly trained with interactions w.r.t. all behaviour types, and thus is not suitable for our experimental settings where interactions are received one-by-one and in random order. Therefore, we adapt DCMF to streaming scenarios by decoupling its training process, so that it can be trained with the interactions w.r.t. each behaviour type independently.

- *Neural Multi-task Recommendation* (**NMTR**) [28] is a state-of-the-art multi-behaviour offline RS, which utilises a multi-task learning technique for the offline recommendations with multi-behaviour interactions. The original NMTR needs to be trained with interactions w.r.t. all behaviour types based on the priority level (e.g., a purchase behaviour must be preceded by a view behaviour). Thus, the original NMTR is not suitable for our experimental settings where the interactions of different behaviours are received in an arbitrary order. To adapt NMTR to our experimental settings, we tailor its cascaded prediction module, so that NMTR can be independently trained with the interactions w.r.t. each behaviour type. In addition, inspired by the practice in its original paper, we equip NMTR with NeuMF for delivering the best performance in our experimental settings.

Note that we have not considered some earlier SRSs, including eAls [38] and RCD [15], as baselines, since they have been significantly outperformed by DWMoE [150], which has been employed as our baseline. As for the ensemble learning based single-behaviour SRSs, such as OCFIF [140] and STS-AEL [149], they rely on ensembling multiple individual recommendation models to improve the recommendation accuracies, and thus are not suitable to serve as our baselines. In addition, some recently proposed multi-behaviour offline RSs, such as EHCF [7] and MATN [133], have either not been considered as baselines, as they are essentially devised for the offline scenarios, and cannot be well adapted to streaming scenarios while keeping their features described in their papers.

Simplified Versions of MbSRS. To clearly demonstrate the contributions of all three components of MbSRS, i.e., MbLM, AMN and UPMM, we have devised six simplified versions of MbSRS and conducted experiments to evaluate these simplified versions for the ablation analysis. Specifically, each of these six simplified versions simplifies or replaces one target component of MbSRS while keeping the other two components unchanged to verify the effectiveness of the target component. More details about these six simplified versions are introduced as follows.

- **MbSRS_MbLM_BsE** and **MbSRS_MbLM_SE** are both devised to verify the effectiveness of MbLM. The original MbLM employs shared embeddings and behaviour-specific embeddings for users (or items) to learn shared user preferences (or item characteristics) and behaviour-specific user preferences (or item characteristics), respectively. For comparisons, MbSRS_MbLM_BsE employs the behaviour-specific embeddings only and MbSRS_MbLM_SE employs the shared embeddings only for learning user preferences and item characteristics.

- **MbSRS_AMN_AVG** and **MbSRS_AMN_MAX** are both devised to verify the effectiveness of AMN, which represents the long-term user preferences by combining the embeddings of historical items with an attentive method. Specifically, MbSRS_AMN_AVG combines the embeddings of historical items simply with an average-pooling layer, while MbSRS_AMN_MAX combines these embeddings simply with a max-pooling layer.

- **MbSRS_UPMM_AVG** and **MbSRS_UPMM_MAX** are both devised to verify the effectiveness of UPMM, which merges the short-term user preferences and long-term user preferences with a gated merging process. Specifically, MbSRS_UPMM_AVG replaces UPMM with an average-pooling layer to merge the short-term and long-term user preferences, while MbSRS_UPMM_MAX replaces UPMM with a max-pooling layer to conduct this merging process.

Parameter Setting. For fair comparisons, the baseline RSs are first initialised with the parameters reported in their papers and then tuned on our datasets and evaluation

Table 5.2: Performance comparison for MbSRS (results on Beibei)

Dataset		Beibei			
Metric		HR@5	NDCG@5	HR@10	NDCG@10
Single-behaviour streaming baseline	SPMF	0.338	0.266	0.390	0.283
	iGMF	0.356	0.276	0.403	0.301
	iMLP	0.461	0.391	0.516	0.409
	iNeuMF	0.461	0.390	0.518	0.408
	iDMF	0.470	0.392	0.526	0.410
	DWMoE	0.475	0.402	0.534	0.418
Multi-behaviour offline baseline	DCMF	0.515	0.421	0.618	0.449
	NMTR	0.531*	0.432*	0.633*	0.465*
Simplified version of our MbSRS	MbSRS_MbLM_BsE	0.439	0.365	0.493	0.382
	MbSRS_MbLM_SE	0.552	0.448	0.656	0.481
	MbSRS_AMN_AVG	0.553	0.454	0.659	0.488
	MbSRS_AMN__MAX	0.535	0.434	0.639	0.467
	MbSRS_UPMM_AVG	0.534	0.436	0.638	0.470
	MbSRS_UPMM_MAX	0.537	0.430	0.647	0.466
Full version of our MbSRS		**0.559**	**0.461**	**0.665**	**0.495**
Improvement percentage over the best-performing baseline		5.27%	6.71%	5.06%	6.45%

policy to achieve the best performance. As for our proposed MbSRS, we empirically initialise its weight matrixes with the Glorot initialiser [30] and initialise the bias vectors with zeros for neural network layers. Moreover, we adopt the Adam optimiser [53] to more effectively train MbSRS with the learning rate of 0.001. Furthermore, to achieve the best performance, we set the sizes of shared embeddings to 64 for all three datasets; and set the sizes of behaviour-specific embeddings to 64 for the Taobao dataset and Tmall dataset, and to 32 for the Beibei dataset. As for the memory size, it is set to 100 for all three datasets. Note that, these embedding sizes and the memory size are set based on the results from Experiment 3 and Experiment 4, respectively. In addition to these general settings, specific settings for a certain experiment are introduced in the first part of the corresponding content for that experiment.

Table 5.3: Performance comparison for MbSRS (results on Taobao)

Dataset		Taobao			
Metric		HR@5	NDCG@5	HR@10	NDCG@10
Single-behaviour streaming baseline	SPMF	0.068	0.044	0.122	0.061
	iGMF	0.072	0.047	0.126	0.064
	iMLP	0.158	0.108	0.218	0.128
	iNeuMF	0.161	0.112	0.222	0.132
	iDMF	0.156	0.108	0.221	0.129
	DWMoE	0.160	0.115	0.226	0.134
Multi-behaviour offline baseline	DCMF	0.171	0.124	0.248	0.142
	NMTR	0.177*	0.126*	0.254*	0.151*
Simplified version of our MbSRS	MbSRS_MbLM_BsE	0.142	0.097	0.212	0.120
	MbSRS_MbLM_SE	0.185	0.129	0.263	0.155
	MbSRS_AMN_AVG	0.190	0.133	0.266	0.158
	MbSRS_AMN__MAX	0.194	0.136	0.27	0.161
	MbSRS_UPMM_AVG	0.189	0.133	0.262	0.157
	MbSRS_UPMM_MAX	0.181	0.128	0.255	0.152
Full version of our MbSRS		**0.198**	**0.141**	**0.276**	**0.167**
Improvement percentage over the best-performing baseline		11.86%	11.90%	8.66%	10.60%

5.3.2 Performance Comparison and Analysis

Experiment 1: Performance Comparison with Baselines (for RQ1)

Setting. To answer **RQ1**, in this experiment, we compare our proposed MbSRS with all eight baselines, including six single-behaviour streaming ones and two multi-behaviour offline ones on all three datasets.

Result. As Tables 5.2 to 5.4 demonstrate, on all three datasets, our proposed MbSRS delivers the highest recommendation accuracies (marked with the **bold** font). Specifically, the improvement percentages of MbSRS over the best-performing baseline (with the results marked by '*'), i.e., NMTR, on each dataset are introduced in the last rows, ranging from 5.27% (on Beibei) to 11.86% (on Taobao) with an average of 8.72% w.r.t. HR@5, ranging from 6.71% (on Beibei) to 12.43% (on Tmall) with an average of 10.35% w.r.t. NDCG@5, ranging from 5.06% (on Beibei) to 8.66% (on Tmall) with an average of 6.76% w.r.t. HR@10 and ranging from 6.45% (on Beibei) to 10.72% on (Taobao) with an average of 9.26% w.r.t. NDCG@10. Moreover, Tables 5.2 to 5.4

Table 5.4: Performance comparison for MbSRS (results on Tmall)

Dataset		Tmall			
Metric		HR@5	NDCG@5	HR@10	NDCG@10
Single-behaviour streaming baseline	SPMF	0.089	0.058	0.147	0.076
	iGMF	0.097	0.064	0.157	0.083
	iMLP	0.444	0.333	0.549	0.367
	iNeuMF	0.450	0.337	0.569	0.371
	iDMF	0.461	0.341	0.567	0.375
	DWMoE	0.466	0.345	0.572	0.382
Multi-behaviour offline baseline	DCMF	0.473	0.352	0.589	0.390
	NMTR	0.488*	0.362*	0.609*	0.401*
Simplified version of our MbSRS	MbSRS_MbLM_BsE	0.295	0.198	0.437	0.244
	MbSRS_MbLM_SE	0.502	0.377	0.627	0.418
	MbSRS_AMN_AVG	0.527	0.398	0.642	0.436
	MbSRS_AMN__MAX	0.522	0.394	0.638	0.433
	MbSRS_UPMM_AVG	0.512	0.384	0.637	0.425
	MbSRS_UPMM_MAX	0.513	0.377	0.639	0.418
Full version of our MbSRS		**0.532**	**0.407**	**0.649**	**0.444**
Improvement percentage over the best-performing baseline		9.02%	12.43%	6.57%	10.72%

show that multi-behaviour RSs consistently outperform the single-behaviour RSs in all the cases.

Analysis. The superiority of our proposed MbSRS can be mainly explained from the following three aspects, i.e., 1) our proposed MbLM accurately learns the short-term user preferences and stable item characteristics via both shared embeddings (for enhancing the modelling of the primary behaviour with the auxiliary ones) and the behaviour-specific embeddings (for highlighting the unique latent features w.r.t. the primary behaviour) in the streaming scenarios; 2) our proposed AMN elaborately represents the long-term preferences of users with their interacted (w.r.t. all behaviour types) items via an attentive method; and 3) our proposed UPMM effectively merges the short-term user preferences learnt from MbLM and the long-term user preferences learnt from AMN via a gated merging process. By learning the relevance scales of between these well-learnt user preferences and item characteristics, MbSRS achieves significantly higher recommendation accuracies than the baselines do.

Moreover, the advantages of multi-behaviour RSs over the single-behaviour RSs lie in that the multi-behaviour ones are able to incorporate more sufficient multi-behaviour interactions to enhance the recommendations w.r.t. the primary behaviour. This practice benefits addressing the data sparsity problem suffered by the single-behaviour RSs and caused by the limited number of single-behaviour interactions, and thus leads to higher accuracies of streaming recommendations.

Experiment 2: Ablation Analysis (for RQ2)

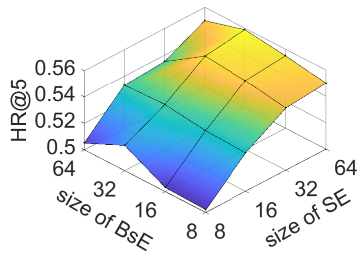
Setting. To answer **RQ2**, in this experiment, we compare all six simplified versions of our proposed MbSRS with the full version to verify the effectiveness of all three components: MbLM, AMN and UPMM in MbSRS, on all three datasets.

Results. As presented in Tables 5.2 to 5.4, the full version of MbSRS outperforms all the simplified versions in all the cases. Moreover, the simplified version MbSRS_MbLM_BsE performs even much worse than other simplified versions.

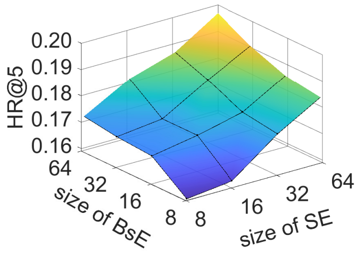
Analysis. These experimental results verify that MbLM, AMN and UPMM are all essential for delivering accurate streaming recommendations. Specifically, MbSRS outperforms MbSRS_MbLM_SE and MbSRS_MbLM_BsE, as MbLM accurately learns the short-term user preferences and stable item characteristics. Additionally, MbSRS delivers more accurate streaming recommendations than MbSRS_AMN_AVG and MbSRS_AMN_MAX do, since AMN effectively maintains the long-term user preferences. Besides, MbSRS achieves higher recommendation accuracies than MbSRS_UPMM_AVG and MbSRS_UPMM_MAX do, because UPMM wisely merges the short-term user preferences and long-term user preferences. In addition, the reason for the unsatisfactory recommendation accuracies achieved by MbSRS_MbLM_BsE is that it fails to leverage the shared embeddings to exploit the interactions w.r.t. auxiliary behaviours for improving the recommendation accuracies. This practice of exploiting the single-behaviour interactions only significantly reduces the accuracies of streaming recommendations.

Experiment 3: Effect of Embedding Sizes (for RQ3)

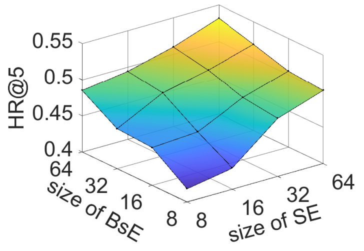
Setting. To answer **RQ3**, in this experiment, we evaluate our proposed MbSRS with



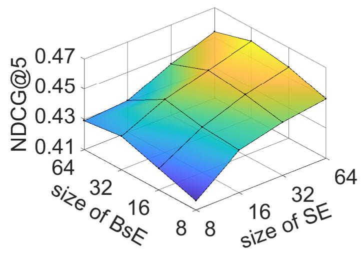
(a) HR@5 on Beibei



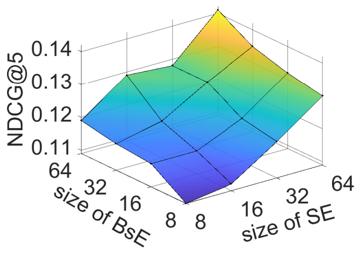
(e) HR@5 on Taobao



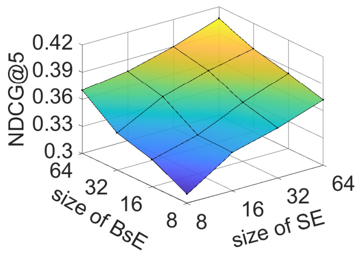
(i) HR@5 on Tmall



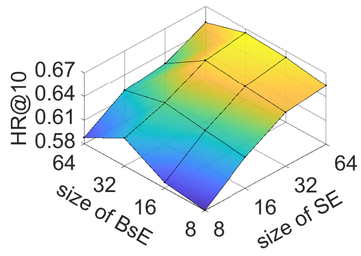
(b) NDCG@5 on Beibei



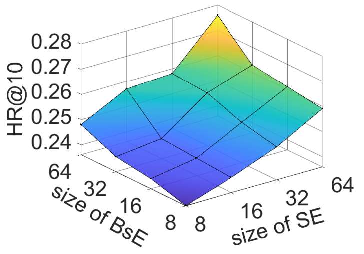
(f) NDCG@5 on Taobao



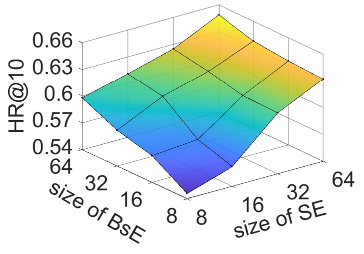
(j) NDCG@5 on Tmall



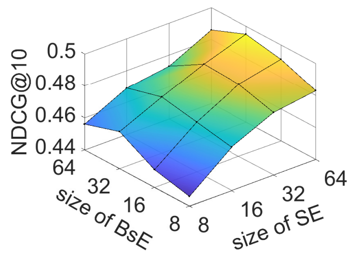
(c) HR@10 on Beibei



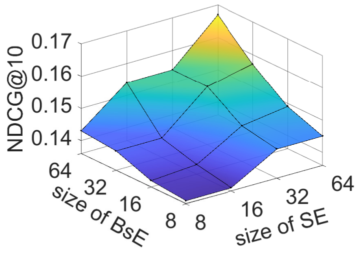
(g) HR@10 on Taobao



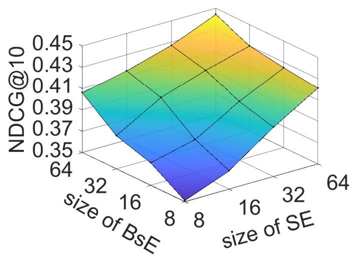
(k) HR@10 on Tmall



(d) NDCG@10 on Beibei



(h) NDCG@10 on Taobao



(l) NDCG@10 on Tmall

Figure 5.2: Effect of the size of behaviour-specific Embedding (BsE) and the size of Shared Embedding (SE).

different sizes (i.e., 8, 16, 32 and 64) of both the Shared Embeddings (SE) and the behaviour-specific Embeddings (BsE) to compare the overall recommendation accuracies delivered by MbSRS with various embedding sizes.

Result. Fig. 5.2 demonstrates that larger sizes of the shared embeddings benefit improving the recommendation accuracies delivered by MbSRS in all the cases. As for the behaviour-specific embeddings, the recommendation accuracies increase consistently with larger sizes of the behaviour-specific embeddings on the Taobao dataset (see Figs. 5.2e to 5.2h) and Tmall dataset (see Figs. 5.2i to 5.2l), while peak when the sizes of the behaviour-specific embeddings reach 32 on the Beibei dataset (see Figs. 5.2a to 5.2d).

Analysis. Larger sizes of the shared embeddings help better represent the shared user preferences and shared item characteristics, and thus benefit more effectively leveraging the interactions w.r.t. the auxiliary behaviours for improving the recommendation accuracies w.r.t. the primary behaviour. This explains why larger sizes of the shared embeddings lead to higher recommendation accuracies in all the cases. Moreover, this reason also partially explains the effect of the sizes of behaviour-specific embeddings. That is larger sizes of behaviour-specific embeddings generally benefit more accurately representing the behaviour-specific user preferences, and thus commonly lead to higher recommendation accuracies. However, too large sizes of embeddings might cause the overfitting problem in some cases, such as the case when the sizes of behaviour-specific embeddings reach 64 on the Beibei dataset, which reduces the recommendation accuracies.

Experiment 4: Effect of the Memory Size (for RQ4)

Setting. To answer **RQ4**, in this experiment, we allow the memory size to range from 40 to 140 with an interval of 20 to compare the overall recommendation accuracies delivered by MbSRS with different memory sizes.

Result. As Fig. 5.3 illustrates, the recommendation accuracies delivered by MbSRS first increase with larger memory sizes, and then keep stable after the memory size

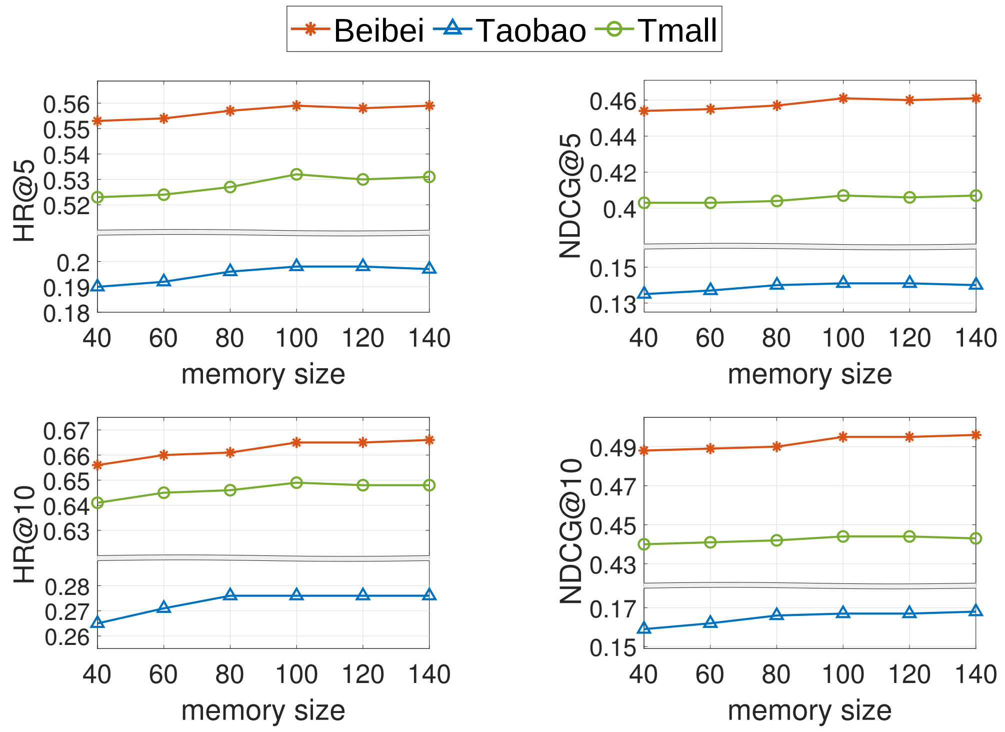


Figure 5.3: Effect of the memory size.

exceeds 100 on all three datasets w.r.t. all four evaluation metrics.

Analysis. Larger memory sizes benefit memorising more historical items, which comprise richer long-term user preferences. This explains why recommendation accuracies improve with increasing memory sizes when these sizes are small. However, too many historical items (e.g., more than 100 items in our cases) might exceed the modelling capability of AMN, and thus do not benefit further improving the recommendation accuracies.

Experiment 5: Effect of the Number of Behaviour Types (for RQ5)

Setting. To answer **RQ5**, in this experiment, we let MbSRS incorporate different numbers of behaviour types to compare the overall recommendation accuracies achieved by MbSRS with different numbers of behaviour types. Specifically, the number of behaviour types is set from 1 to the total number (i.e., two for Beibei, four for Taobao and three for Tmall) of behaviour types contained in the corresponding datasets. Moreover,

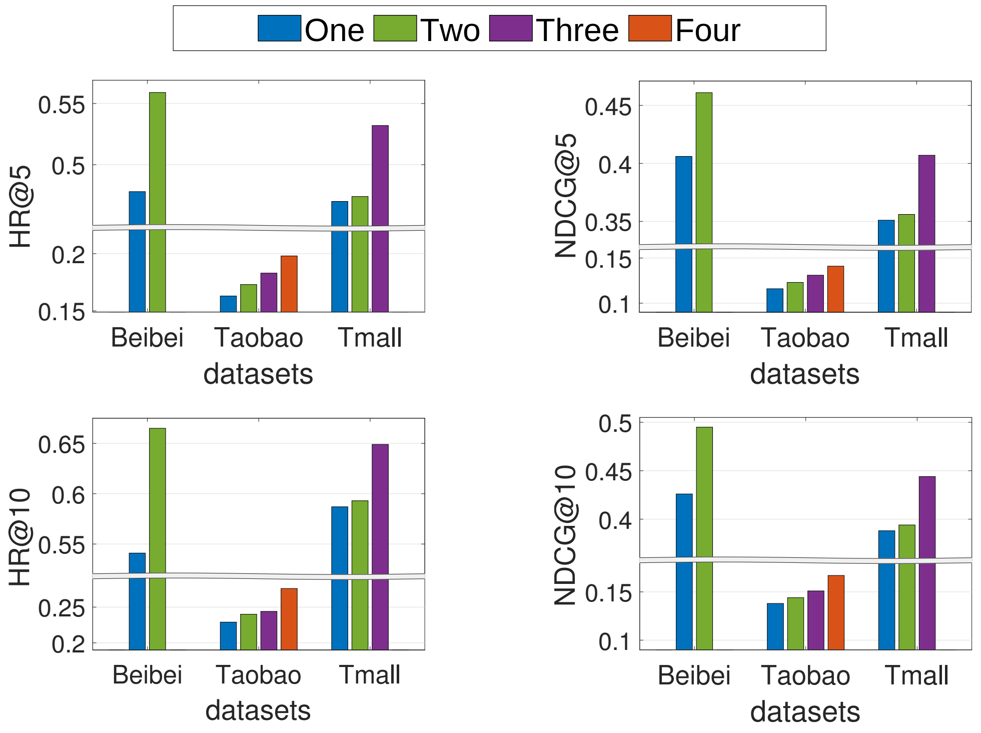


Figure 5.4: Effect of the number of behaviour types.

we add the behaviour types based on their priority levels; for example, the purchase behaviour is the first behaviour type we add while the view behaviour is the last one we add for all three datasets.

Result. Fig. 5.4 demonstrates that our proposed MbSRS achieves higher recommendation accuracies when incorporating more behaviour types in all the cases.

Analysis. Incorporating more behaviour types allows MbSRS to exploit more sufficient multi-behaviour interactions to better learn the user-item relations, and thus leads to higher accuracies of streaming recommendations.

5.4 Chapter Summary

In this chapter, we have proposed the *first* Multi-behaviour Streaming Recommender System, called MbSRS, for delivering accurate streaming recommendations by utilising data streams of multi-behaviour interactions. First, we propose a Multi-behaviour Learning Module (MbLM) to wisely exploit multi-behaviour interactions for learning the short-term user preferences and stable item characteristics. Then, we propose Attentive Memory Network (AMN) to effectively maintain the long-term user preferences w.r.t. the primary behaviour. After that, the short-term user preferences learnt by MbLM and the long-term user preferences learnt by AMN are carefully merged by our proposed User Preference Merging Module (UPMM). Finally, the recommendations are delivered by learning the relevance scales between the learnt user preferences and the learnt item characteristics. The superiority of our proposed MbSRS has been verified by extensive experiments on three real-world datasets.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

Recommender Systems (RSs) have been widely studied for improving the satisfaction of users and increasing the revenue of enterprises. However, the conventional offline RSs cannot well deal with the ubiquitous data stream for streaming recommendations. This is because the offline RSs periodically train recommendation models with large-volume historical interaction data, and thus cannot well capture the latest user preferences embedded in the recent interaction data. Therefore, single-behaviour SRSs have emerged to train recommendation models with newly coming data of single-behaviour interactions in a timely manner, and thus capture the latest user preferences for accurate streaming recommendations.

Although efforts have been made, the following three challenges still need to be well addressed to further improve the recommendation accuracies of single-behaviour SRSs; they are **CH1**: *'how to capture long-term user preferences while addressing the preference drift issue'*, **CH2**: *'how to well handle the heterogeneity of both users and items'* and **CH3**: *'how to well deal with the underload problem and the overload problem'*. Moreover, to the best of our knowledge, no multi-behaviour SRSs have been reported in the literature. Nevertheless, multi-behaviour interactions are pervasive in streaming scenarios and reflect the users' preferences for items. Thus, multi-behaviour interactions should be well leveraged for further improving the accuracies of streaming recommendations. This leads to the fourth challenge in this research area — **CH4**:

'how to wisely leverage multi-behaviour interactions for improving the accuracies of streaming recommendations'.

To address these four challenges, this thesis has proposed the following three approaches:

1) Targeting **CH1** and **CH2**, in Chapter 3, we have proposed a Variational and Reservoir-enhanced Sampling based Double-Wing Mixture-of-Experts framework, called VRS-DWMoE, to deliver accurate streaming recommendations w.r.t. data streams of single-behaviour interactions. Specifically, we first propose reservoir-enhanced sampling to wisely complement new data with sampled historical data to serve as the training data for capturing long-term user preferences while addressing the preference drift issue. Then, we propose a double-wing mixture-of-experts model to learn the heterogeneous user preferences and item characteristics with two mixture-of-experts models (i.e., mixture-of-user-experts model and mixture-of-item-experts model), respectively, from the prepared training data. After that, we make recommendations by merging these learned user preferences and item characteristics. The superiority of VRS-DWMoE has been verified by extensive experiments that are conducted on three real-world datasets.

2) Targeting **CH1** and **CH3**, in Chapter 4, we have proposed a Stratified and Time-aware Sampling based Adaptive Ensemble Learning framework, called STS-AEL, to improve the accuracies of streaming recommendations w.r.t. data streams of single-behaviour interactions. Specifically, our proposed STS-AEL captures long-term user preferences while addressing the preference drift issue through wisely sampling from both the historical data and new data with an elaborately devised stratified and time-aware sampler. Moreover, incorporating the historical data also benefits addressing the underload problem. Furthermore, STS-AEL addresses the overload problem by first training the multiple individual models concurrently, and then fusing the results of these trained models with a

sequential adaptive fusion approach. Extensive experiments conducted on three real-world datasets have demonstrated that our proposed STS-AEL significantly outperforms the state-of-the-art approaches.

3) Targeting **CH4**, in Chapter 5, we have proposed the *first* Multi-behaviour Streaming Recommender System, called MbSRS, to wisely leverage multi-behaviour interactions for further improving the recommendation accuracies in streaming scenarios. Specifically, by wisely exploiting data streams of multi-behaviour interactions, our proposed MbSRS first accurately captures the short-term user preferences and stable item characteristics, then effectively maintains the long-term user preferences via an attentive memory network, and afterwards elaborately merges the learned short-term user preferences and long-term preferences through a gated merging process. Extensive experiments conducted on three real-world datasets have demonstrated that our proposed MbSRS significantly outperforms the state-of-the-art approaches.

6.2 Future Work

This thesis has mainly focused on studying the single-behaviour and multi-behaviour SRSs to improve the recommendation accuracies in streaming scenarios. We have proposed three approaches to address the four challenges of streaming recommendations. Moreover, extensive experiments have been conducted to demonstrate the effectiveness and superiorities of our proposed three approaches. However, the following three issues still need to be well studied in the future work.

1) Chapter 3 proposes a double-wing mixture-of-experts framework for streaming recommendations. Currently, all the individual expert models are employed for dealing with every interaction, even if some of these individual expert models do not specialise in some types of interactions. In the future, researchers should propose approaches that allow individual expert models to deal with their spe-

cialised types of interactions. Through this way, we might not only further improve the recommendation accuracies by reducing the negative interference of unspecialised expert models, but also possibly more efficiently deal with data streams as specialised experts can concurrently deal with multiple types of interactions.

2) Chapter 4 proposes an ensemble learning based framework, which ensembles multiple individual recommendation models to learn the user-item relations from data streams in parallel. Currently, this framework ensembles the same types of individual recommendation models only, and its effectiveness might be restricted by such single types of individual models. In the future, approaches should be proposed to elaborately ensemble multiple complementary types of individual models, so that these individual models can better reinforce one another for delivering more accurate streaming recommendations.

3) Chapter 5 proposes a multi-behaviour SRS to address the long-standing data sparsity issue by incorporating multiple behaviour types. However, this data sparsity issue might not be completely solved by this proposed multi-behaviour SRS, as the interactions w.r.t. all available behaviour types might be still quite sparse. In the future, researchers could consider leveraging user profiles, item properties and even social relations to further address this data sparsity issue, and thus further improve the accuracies of streaming recommendations.

Appendix A

The Notations in the Thesis

Table A.1: The important notations in Chapter 3

Notation	Description
α	the proportion of $\mathbf{SS}_{new}$ in $\mathbf{SS}$
γ	the loss coefficient
δ	the ratio of the sampling size from $\mathbf{R}$
a	the activation function
bs	the batch size for training
$\mathbf{N} \subseteq \mathbf{Y}$	the newly received data from data stream
$\mathbf{p_u}$ and $\mathbf{q_v}$	the user embedding, and the item embedding
$\mathbf{P_u}$ and $\mathbf{Q_v}$	the vector of user preferences, and the vector of item characteristics
$\mathbf{R} \subseteq \mathbf{Y}$	reservoir, i.e., a set that contains representative historical data
sp_p and sp_r	data processing speed, and data receiving speed
$\mathbf{SS}_{his} \subseteq \mathbf{R}$	the set of sampled data from $\mathbf{R}$
$\mathbf{U} = \{u_1, u_2, ..., u_m\}$	the set of users
$\mathbf{V} = \{v_1, v_2, ..., v_n\}$	the set of items
$\mathbf{W}$ and $\mathbf{b}$	the weight matrix, and the bias vector
$y_{i,j} \in \mathbf{Y}$	the notation that indicates whether an interaction exists, i.e., it is 1 if an interaction exists between user u_i and item v_j, and 0 otherwise
$\mathbf{Y} \in R^{m*n}$	the matrix of interactions between $\mathbf{U}$ and $\mathbf{V}$
$\lvert * \rvert$	the size of a set
$\lvert\lvert * \rvert\rvert_1$	the L_1 norm of a vector
$\lvert\lvert * \rvert\rvert_2$	the L_2 norm of a vector
$[\mathbf{a}; \mathbf{b}]$	the concatenation of vector $\mathbf{a}$ and vector $\mathbf{b}$

Table A.2: The important notations in Chapter 4

Notation	**Description**
α	the proportion of $\mathbf{SS}_{new}$ in $\mathbf{SS}$
λ_{new} and λ_{res}	the decay ratio for $\mathbf{N}$, and the decay ratio for $\mathbf{R}$
a	the activation function
acc	the recommendation accuracy
bs	the size of training batch
$c^k_{u,v}$	the estimated confidence of im_k for the prediction of the interaction between user u and item v
fw_k	the fusion weights for im_k
im_k	the k^{th} individual model
IM = $\{\text{im}_1, \text{im}_2, ..., \text{im}_o\}$	the set of individual models
m	the number of users
n	the number of items
$\mathbf{N} \subseteq \mathbf{Y}$	the newly received data from data stream
o	the number of individual models
$\mathbf{p_u}$ and $\mathbf{q_v}$	the user embedding, and the item embedding
$\mathbf{P}_k = \{\langle acc^k_1, {u_1}^k, {v_1}^k\rangle, \cdots, \langle acc^k_g, {u_g}^k, {v_g}^k\rangle\}$	the set of recommendation accuracies from im_k and corresponding user-item pairs in last test iteration
$\mathbf{R} \subseteq \mathbf{Y}$	reservoir, i.e., a set that contains representative historical data
sp_p and sp_r	data processing speed, and data receiving speed
$\mathbf{S}_k \subseteq \mathbf{P}_k$	the set of top e tuples from $\mathbf{P}_k$ those have most similar user-item pairs to the target user-item pair
$\mathbf{SS}_{new} \subseteq \mathbf{N}$ and $\mathbf{SS}_{his} \subseteq \mathbf{R}$	the set of sampled data from $\mathbf{N}$, and the set of sampled data from $\mathbf{R}$
$\mathbf{U} = \{u_1, u_2, ..., u_m\}$	the set of users
$\mathbf{V} = \{v_1, v_2, ..., v_n\}$	the set of items
$\mathbf{W}$ and $\mathbf{b}$	the weight matrix, and the bias vector
$y_{i,j} \in \mathbf{Y}$	the notation that indicates whether an interaction exists, i.e., it is 1 if an interaction exists between user u_i and item v_j, and 0 otherwise
$\mathbf{Y} \in R^{m*n}$	the matrix of interactions between $\mathbf{U}$ and $\mathbf{V}$
$*^T$	the transpose of a vector
$\lvert * \rvert$	the size of a set
$\lvert\lvert * \rvert\rvert_1$	the L_1 norm of a vector
$\lvert\lvert * \rvert\rvert_2$	the L_2 norm of a vector
$[\mathbf{a}; \mathbf{b}]$	the concatenation of vector $\mathbf{a}$ and vector $\mathbf{b}$

Table A.3: The important notations in Chapter 5

Notation	Description
$\mathbf{B} = \{b_1, b_2, ..., b_{\vert B\vert}\}$	the set of behavior types
n	the number of behaviour types
$\mathbf{p_u}$ and $\mathbf{q_v}$	the user embedding, and the item embedding
$\mathbf{U} = \{u_1, u_2, ..., u_{\vert U\vert}\}$	the set of users
$\mathbf{V} = \{v_1, v_2, ..., v_{\vert V\vert}\}$	the set of items
$\mathbf{W}$ and $\mathbf{b}$	the weight matrix, and the bias vector
y_i	the notation that indicates whether an interaction w.r.t. the i^{th} behaviour exists, i.e., it is 1 if this interaction exists, and 0 otherwise
$\hat{y}_i$	the predicted value of y_i
$*^T$	the transpose of a vector
$\vert * \vert$	the size of a set
$\vert\vert * \vert\vert_2$	the L_2 norm of a vector
$[\mathbf{a}; \mathbf{b}]$	the concatenation of vector $\mathbf{a}$ and vector $\mathbf{b}$

Appendix B

The Acronyms in the Thesis

Table B.1: Main acronyms in all the chapters (part 1)

Acronyms	**Explanations**
AEL	adaptive ensemble learning
AMN	attentive memory network
ASLI	attentive sequential model of latent intent
AutoRec	autoencoder-based recommender system
AVG	averaging
BPR	Bayesian personalised ranking
CH1, CH2, CH3 and CH4	challenges 1, 2, 3 and 4
CMF	collective matrix factorisation
DCMF	deep collective matrix factorisation
DIPN	deep intent prediction network
DL-SBORS	deep learning-based single-behaviour offline recommender system
DMF	deep matrix factorisation
DWMoE	double-wing mixture of experts
eAls	element-wise alternating least squares
FW	fusion weights
GMF	generalised matrix factorisation
GNN	graph neural network
HR	hit ratio
iBPR	adapted Bayesian personalised ranking

Table B.2: Main acronyms in all the chapters (part 2)

Acronyms	**Explanations**
ALS	alternating least square
ICF	incremental collaborative filtering
iGMF	adapted generalised matrix factorisation
iMLP	adapted multiple layer perceptron
iNeuMF	adapted neural matrix factorisation
iTPMF-CF	adapted time window-based probabilistic matrix factorisation for collaborative filtering
MATN	memory-augmented transformer network
MbLM	multi-behaviour learning module
MbORS	multi-behaviour offline recommender system
MbSRS	multi-behaviour streaming recommender system
MF-SBORS	matrix factorisation-based single-behaviour offline recommender system
MLP	multiple layer perceptron-based
M-MLP	multi-branch multi-layer perceptron
MoE	mixture of experts
MoIE	mixture of item experts
MoUE	mixture of user experts
NDCG	normalised discounted cumulative gain
NDO	new data only
NeuMF	neural matrix factorisation
NMRN	neural memory recommender network

Table B.3: Main acronyms in all the chapters (part 3)

Acronyms	**Explanations**
OCFIF	online collaborative filtering with implicit feedback
ORS	offline recommender system
PMF	probabilistic matrix factorisation
RCD	randomised block coordinate descent
RNN	recurrent neural network
RQ	research question
RR	reservoir-enhanced random sampling
RS	recommender system
SBORS	single-behaviour offline recommender system
SBSRS	single-behaviour streaming recommender system
SGD	stochastic gradient descent
SPMF	stream-centred probabilistic matrix factorisation
SRS	streaming recommender system
STS	stratified and time-aware sampling
STS-AEL	stratified and time-aware sampling-based adaptive ensemble learning
SVD	singular value decomposition
SW	sliding window
TPMF-CF	time window-based probabilistic matrix factorisation for collaborative filtering
UPMM	user preference merging module
VALS	view-enhanced alternative least square
VRS	variational and reservoir-enhanced sampling
VRS-DWMoE	variational and reservoir-enhanced sampling-based double-wing mixture of experts

Bibliography

[1] S. Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(3):185–196, 1993.

[2] L. Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, 1996.

[3] P. Bühlmann, B. Yu, et al. Analyzing bagging. *The Annals of Statistics*, 30(4):927–961, 2002.

[4] I. Cantador, A. Bellogín, and D. Vallet. Content-based recommendation in social tagging systems. In *Proceedings of the 4th ACM Conference on Recommender Systems*, pages 237–240, 2010.

[5] B. Chandramouli, J. J. Levandoski, A. Eldawy, and M. F. Mokbel. Streamrec: a real-time recommender system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 1243–1246, 2011.

[6] S. Chang, Y. Zhang, J. Tang, D. Yin, Y. Chang, M. A. Hasegawa-Johnson, and T. S. Huang. Streaming recommender systems. In *Proceedings of the 26th International Conference on World Wide Web, Perth*, pages 381–389, 2017.

[7] C. Chen, M. Zhang, Y. Zhang, W. Ma, Y. Liu, and S. Ma. Efficient heterogeneous collaborative filtering without negative sampling for recommendation. In *Proceedings of the 34th AAAI Conference on Artificial Intelligence*, pages 19–26, 2020.

[8] J. Chen, H. Li, Q. Xie, L. Li, and Y. Liu. Streaming recommendation algorithm with user interest drift analysis. In *Proceedings of the 4th Asia-Pacific Web and Web-Age Information Management Joint International Conference on Web and Big Data*, volume 11642, pages 121–136, 2019.

[9] L. Chen, G. Chen, and F. Wang. Recommender systems based on user reviews: the state of the art. *User Model. User Adapt. Interact.*, 25(2):99–154, 2015.

[10] W. Chen, Z. Niu, X. Zhao, and Y. Li. A hybrid recommendation algorithm adapted in e-learning environments. *World Wide Web*, 17(2):271–284, 2014.

[11] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu, and H. Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pages 7–10, 2016.

[12] F. Chu and C. Zaniolo. Fast and light boosting for adaptive mining of data streams. In *Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, volume 3056, pages 282–292, 2004.

[13] S. Dara, C. R. Chowdary, and C. Kumar. A survey on group recommender systems. *J. Intell. Inf. Syst.*, 54(2):271–295, 2020.

[14] Y. Deldjoo, T. D. Noia, and F. A. Merra. Adversarial machine learning in recommender systems: State of the art and challenges. *CoRR*, abs/2005.10322, 2020.

[15] R. Devooght, N. Kourtellis, and A. Mantrach. Dynamic matrix factorization with priors on unknown values. In *Proceedings of the 21th ACM International Conference on Knowledge Discovery and Data Mining*, pages 189–198, 2015.

[16] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl. Real-time top-n recommendation in social streams. In *Proceedings of the 6th ACM Conference on Recommender Systems*, pages 59–66, 2012.

[17] J. Ding, F. Feng, X. He, G. Yu, Y. Li, and D. Jin. An improved sampler for bayesian personalized ranking by leveraging view data. In *Proceedings of the 27th International World Wide Web Conferences*, pages 13–14, 2018.

[18] J. Ding, G. Yu, X. He, F. Feng, Y. Li, and D. Jin. Sampler design for bayesian personalized ranking by leveraging view data. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[19] J. Ding, G. Yu, X. He, F. Feng, Y. Li, and D. Jin. Sampler design for bayesian personalized ranking by leveraging view data. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[20] J. Ding, G. Yu, X. He, Y. Quan, Y. Li, T. Chua, D. Jin, and J. Yu. Improving implicit recommender systems with view data. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 3343–3349, 2018.

[21] J. Ding, G. Yu, Y. Li, X. He, and D. Jin. Improving implicit recommender systems with auxiliary data. *ACM Trans. Inf. Syst.*, 38(1):11:1–11:27, 2020.

[22] M. Dong, F. Yuan, L. Yao, X. Xu, and L. Zhu. MAMO: memory-augmented meta-optimization for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event*, pages 688–697, 2020.

[23] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma. A survey on ensemble learning. *Frontiers Comput. Sci.*, 14(2):241–258, 2020.

[24] S. Dzeroski and B. Zenko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54(3):255–273, 2004.

[25] Y. Freund, R. Schapire, and N. Abe. A short introduction to boosting. *Journal-Japanese Society For Artificial Intelligence*, 14(771-780):1612, 1999.

[26] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156, 1996.

[27] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997.

[28] C. Gao, X. He, D. Gan, X. Chen, F. Feng, Y. Li, T. Chua, and D. Jin. Neural multi-task recommendation from multi-behavior data. In *Proceedings of the 35th IEEE International Conference on Data Engineering*, pages 1554–1557, 2019.

[29] D. Gligorijevic, J. Gligorijevic, A. Raghuveer, M. Grbovic, and Z. Obradovic. Modeling mobile user actions for purchase recommendation using deep memory networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1021–1024, 2018.

[30] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, volume 9, pages 249–256, 2010.

[31] L. Guo, L. Hua, R. Jia, B. Zhao, X. Wang, and B. Cui. Buying or browsing?: Predicting real-time purchasing intent using attention-based deep network with multiple behavior. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1984–1992.

[32] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Q. V. Hung. Streaming session-based recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1569–1577, 2019.

[33] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong, and Q. He. A survey on knowledge graph-based recommender systems. *CoRR*, abs/2003.00911, 2020.

[34] K. Gupta, M. Y. Raghuprasad, and P. Kumar. A hybrid variational autoencoder for collaborative filtering. *CoRR*, abs/1808.01006, 2018.

[35] G. Han, W. Que, G. Jia, and W. Zhang. Resource-utilization-aware energy efficient server consolidation algorithm for green computing in IIOT. *J. Netw. Comput. Appl.*, 103:205–214, 2018.

[36] X. He, T. Chen, M. Kan, and X. Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 1661–1670, 2015.

[37] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182, 2017.

[38] X. He, H. Zhang, M. Kan, and T. Chua. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the 39th International ACM conference on Research and Development in Information Retrieval*, pages 549–558, 2016.

[39] B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852, 2018.

[40] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of the 4th International Conference on Learning Representations*, 2016.

[41] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 263–272, 2008.

[42] H. Huang, Q. Zhang, and X. Huang. Mention recommendation for twitter with end-to-end memory network. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1872–1878, 2017.

[43] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 505–514, 2018.

[44] Y. Huang, B. Cui, W. Zhang, J. Jiang, and Y. Xu. Tencentrec: Real-time stream recommendation in practice. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Melbourne*, pages 227–238, 2015.

[45] M. Jakomin, Z. Bosnic, and T. Curk. Simultaneous incremental matrix factorization for streaming recommender systems. *Expert Syst. Appl.*, 160:113685, 2020.

[46] D. Jannach, A. Manzoor, W. Cai, and L. Chen. A survey on conversational recommender systems. *CoRR*, abs/2004.00646, 2020.

[47] K. Ji, Y. Yuan, R. Sun, K. Ma, Z. Chen, and J. Liu. A bagging-based ensemble method for recommendations under uncertain rating data. In *Proceedings of the 2018 International Conference on Security, Pattern Analysis, and Cybernetics*, pages 446–450, 2018.

[48] N. Kalchbrenner, E. Grefenstette, and P. Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, 2014.

[49] E. Kalyvianaki, M. Fiscato, T. Salonidis, and P. R. Pietzuch. THEMIS: fairness in federated stream processing under overload. In *Proceedings of the 42th International Conference on Management of Data*, pages 541–553, 2016.

[50] A. A. Kardan and M. Ebrahimi. A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups. *Inf. Sci.*, 219:93–110, 2013.

[51] B. Kepes. 30% of servers are sitting ”Comatose” according to research. *Forbes*, 2015.

[52] D. H. Kim, C. Park, J. Oh, S. Lee, and H. Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 233–240, 2016.

[53] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, pages 1–15, 2015.

[54] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

[55] Y. Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 447–456, 2009.

[56] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[57] D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. In *Proceedings of the 30th Conference on Neural Information Processing Systems*, pages 1172–1180, 2016.

[58] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher. Ask me anything: Dynamic memory networks for natural language processing. In *Proceedings of the 33nd International Conference on Machine Learning*, volume 48, pages 1378–1387, 2016.

[59] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Networks*, 8(1):98–113, 1997.

[60] B. T. Le and R. Dieng-Kuntz. A graph-based algorithm for alignment of OWL ontologies. In *Proceedings of the 2007 IEEE / WIC / ACM International Conference on Web Intelligence*, pages 466–469, 2007.

[61] Y. LeCun, L. Bottou, G. B. Orr, and K. Müller. Efficient backprop. In *Neural Networks: Tricks of the Trade - Second Edition*, volume 7700, pages 9–48. 2012.

[62] G. Lee. *Cloud networking: Understanding cloud-based data center networks*. 2014.

[63] L. Lefakis and F. Fleuret. Reservoir boosting : Between online and offline ensemble learning. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pages 1412–1420, 2013.

[64] D. Li, X. Li, J. Wang, and P. Li. Video recommendation with multi-gate mixture of experts soft actor critic. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1553–1556, 2020.

[65] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *Proceedings of the 26th ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[66] K. Li, X. Zhou, F. Lin, W. Zeng, B. Wang, and G. Alterovitz. Sparse online collaborative filtering with dynamic regularization. *Inf. Sci.*, 505:535–548, 2019.

[67] Y. Li, J. Song, X. Li, and W. Liu. Gated sequential recommendation with dynamic memory network. In *Proceedings of the International Joint Conference on Neural Networks*, pages 1–8, 2019.

[68] J. Liu, C. Shi, B. Hu, S. Liu, and P. S. Yu. Personalized ranking recommendation via integrating multiple feedbacks. In *Proceedings of the 21st Pacific-Asia*

Conference on Advances in Knowledge Discovery and Data Mining, volume 10235, pages 131–143, 2017.

[69] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender Systems Handbook*, pages 73–105. 2011.

[70] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang. Recommender system application developments: A survey. *Decis. Support Syst.*, 74:12–32, 2015.

[71] H. Ma. An experimental study on implicit social recommendation. In *Proceedings of the 36th International ACM SIGIR conference on research and development in Information Retrieval*, pages 73–82, 2013.

[72] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining*, pages 287–296, 2011.

[73] R. Ma, X. Qiu, Q. Zhang, X. Hu, Y.-G. Jiang, and X. Huang. Co-attention memory network for multimodal microblog's hashtag recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[74] R. Ma, Q. Zhang, J. Wang, L. Cui, and X. Huang. Mention recommendation for multimodal microblog with cross-attention memory network. In *Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 195–204, 2018.

[75] K. G. S. Madsen and Y. Zhou. Dynamic resource management in a massively parallel stream processing engine. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management*, pages 13–22, 2015.

[76] J. Manotumruksa, C. Macdonald, and I. Ounis. A deep recurrent collaborative filtering framework for venue recommendation. In *Proceedings of the 26th*

ACM Conference on Information and Knowledge Management, pages 1429–1438, 2017.

[77] R. Mariappan and V. Rajan. Deep collective matrix factorization for augmented multi-view learning. *Mach. Learn.*, 108(8-9):1395–1420, 2019.

[78] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artif. Intell. Rev.*, 42(2):275–293, 2014.

[79] S. Masoudnia and R. Ebrahimpour. Mixture of experts: a literature survey. *Artif. Intell. Rev.*, 42(2):275–293, 2014.

[80] F. Mi and B. Faltings. Memory augmented neural model for incremental session-based recommendation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 2169–2176, 2020.

[81] T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur. Extensions of recurrent neural network language model. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5528–5531, 2011.

[82] J. C. Núñez, R. Cabido, J. J. Pantrigo, A. S. Montemayor, and J. F. Vélez. Convolutional neural networks and long short-term memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognit.*, 76:80–94, 2018.

[83] N. C. Oza. Online bagging and boosting. In *Proceedings of the 2005 IEEE International Conference on Systems, Man and Cybernetics*, pages 2340–2345, 2005.

[84] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining*, pages 502–511, 2008.

[85] W. Pan, H. Zhong, C. Xu, and Z. Ming. Adaptive bayesian personalized ranking for heterogeneous implicit feedbacks. *Knowl. Based Syst.*, 73:173–180, 2015.

[86] M. Papagelis, I. Rousidis, D. Plexousakis, and E. Theoharopoulos. Incremental collaborative filtering for highly-scalable recommendation algorithms. In *Proceedings of the 15th International Symposium on Methodologies for Intelligent Systems*, volume 3488, pages 553–561, 2005.

[87] S. Park, Y. Kim, and S. Choi. Hierarchical bayesian matrix factorization with side information. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1593–1599, 2013.

[88] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The Adaptive Web, Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341, 2007.

[89] I. Porteous, A. U. Asuncion, and M. Welling. Bayesian matrix factorization with side information and dirichlet process mixtures. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.

[90] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of 11th ACM Conference on Recommender Systems*, pages 130–137, 2017.

[91] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, pages 452–461, 2009.

[92] S. Rendle and L. Schmidt-Thieme. Online-updating regularized kernel matrix factorization models for large-scale recommender systems. In *Proceedings of the 2nd ACM Conference on Recommender Systems*, pages 251–258, 2008.

[93] H. Röger and R. Mayer. A comprehensive survey on parallelization and elasticity in stream processing. *ACM Comput. Surv.*, 52(2):36:1–36:37, 2019.

[94] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the 21st Annual Conference on Neural Information Processing Systems*, pages 1257–1264, 2007.

[95] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Trans. Neural Networks*, 20(1):61–80, 2009.

[96] P. Schwab, D. Miladinovic, and W. Karlen. Granger-causal attentive mixtures of experts: Learning important features with neural networks. In *Proceedings of the 33th AAAI Conference on Artificial Intelligence*, pages 4846–4853, 2019.

[97] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie. Autorec: Autoencoders meet collaborative filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion*, pages 111–112, 2015.

[98] J. G. Silva and L. Carin. Active learning for online bayesian matrix factorization. In *Proceedings of the 18th ACM International Conference on Knowledge Discovery and Data Mining*, pages 325–333, 2012.

[99] F. Simon. Funksvd. In *https://sifter.org/simon/journal/20061211.html*, 2006 (accessed December 14, 2020).

[100] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 650–658, 2008.

[101] J. Snoek, H. Larochelle, and R. P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 26th Annual Conference on Neural Information Processing Systems*, pages 2960–2968, 2012.

[102] S. G. Soares and R. Araújo. An on-line weighted ensemble of regressor models to handle concept drifts. *Eng. Appl. Artif. Intell.*, 37:392–406, 2015.

[103] B. Song, Y. Cao, W. Zhang, and C. Xu. Session-based recommendation with hierarchical memory networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2181–2184, 2019.

[104] G. W. Stewart. On the early history of the singular value decomposition. *SIAM Rev.*, 35(4):551–566, 1993.

[105] M. Stonebraker, U. Çetintemel, and S. B. Zdonik. The 8 requirements of real-time stream processing. *SIGMOD Rec.*, 34(4):42–47, 2005.

[106] F. Strub, R. Gaudel, and J. Mary. Hybrid recommender system based on autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, DLRS@RecSys 2016, Boston, MA, USA, September 15, 2016*, pages 11–16, 2016.

[107] K. Subbian, C. C. Aggarwal, and K. Hegde. Recommendations for streaming data. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management*, pages 2185–2190, 2016.

[108] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus. End-to-end memory networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, pages 2440–2448, 2015.

[109] S. Sun, Y. Tang, Z. Dai, and F. Zhou. Self-attention network for session-based recommendation with streaming data input. *IEEE Access*, 7:110499–110509, 2019.

[110] Y. Tai, J. Yang, X. Liu, and C. Xu. Memnet: A persistent memory network for image restoration. In *Proceedings of the 16th IEEE International Conference on Computer Vision*, pages 4549–4557, 2017.

[111] D. Tang, B. Qin, and T. Liu. Aspect level sentiment classification with deep memory network. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 214–224, 2016.

[112] L. Tang, B. Long, B. Chen, and D. Agarwal. An empirical study on recommendation with multiple types of feedback. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 283–292, 2016.

[113] M. M. Tanjim, C. Su, E. Benjamin, D. Hu, L. Hong, and J. J. McAuley. Attentive sequential models of latent intent for next item recommendation. In *WWW '20: The Web Conference 2020, Taipei, Taiwan, April 20-24, 2020*, pages 2528–2534, 2020.

[114] K. M. Ting and I. H. Witten. Stacking bagged and dagged models. In *Proceedings of the 14th International Conference on Machine Learning*, pages 367–375, 1997.

[115] A. Toshniwal, S. Taneja, A. Shukla, K. Ramasamy, J. M. Patel, S. Kulkarni, J. Jackson, K. Gade, M. Fu, J. Donham, N. Bhagat, S. Mittal, and D. V. Ryaboy. Storm@twitter. In *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, pages 147–156, 2014.

[116] J. Vinagre, A. M. Jorge, and J. Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In *Proceedings of 22nd International Conference on User Modeling, Adaptation, and Personalization*, volume 8538, pages 459–470, 2014.

[117] J. Vinagre, A. M. Jorge, and J. Gama. Online bagging for recommender systems. *Expert Syst. J. Knowl. Eng.*, 35(4), 2018.

[118] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams

using ensemble classifiers. In *Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining*, pages 226–235, 2003.

[119] N. Wang, S. Wang, Y. Wang, Q. Z. Sheng, and M. A. Orgun. Modelling local and global dependencies for next-item recommendations. In *Proceedings of the 21st International Conference on Web Information Systems Engineering*, volume 12343 of *Lecture Notes in Computer Science*, pages 285–300, 2020.

[120] Q. Wang, S. Li, and G. Chen. Word-driven and context-aware review modeling for recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1859–1862, 2018.

[121] Q. Wang, H. Yin, Z. Hu, D. Lian, H. Wang, and Z. Huang. Neural memory streaming recommender networks with adversarial training. In *Proceedings of the 24th ACM International Conference on Knowledge Discovery & Data Mining*, pages 2467–2475, 2018.

[122] S. Wang and L. Cao. Inferring implicit rules by learning explicit and hidden item dependency. *IEEE Trans. Syst. Man Cybern. Syst.*, 50(3):935–946, 2020.

[123] S. Wang, L. Cao, and Y. Wang. A survey on session-based recommender systems. *CoRR*, abs/1902.04864, 2019.

[124] S. Wang, L. Hu, Y. Wang, L. Cao, Q. Z. Sheng, and M. Orgun. Sequential recommender systems: challenges, progress and prospects. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 6332–6338, 2019.

[125] S. Wang, L. Hu, Y. Wang, X. He, Q. Z. Sheng, M. Orgun, L. Cao, F. Ricci, and P. S. Yu. Graph learning based recommender systems: a review. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 1–9, 2021.

[126] S. Wang, L. Hu, Y. Wang, Q. Z. Sheng, M. A. Orgun, and L. Cao. Modeling multi-purpose sessions for next-item recommendations via mixture-channel purpose routing networks. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3771–3777, 2019.

[127] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen. Streaming ranking based recommender systems. In *Proceedings of the 41st International ACM Conference on Research & Development in Information Retrieval*, pages 525–534, 2018.

[128] W. Wang, W. Zhang, S. Liu, Q. Liu, B. Zhang, L. Lin, and H. Zha. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In *Proceedings of the 29th International World Wide Web Conferences*, pages 3056–3062, 2020.

[129] H. Wen, X. Liu, C. Yan, L. Jiang, Y. Sun, J. Zhang, and H. Yin. Leveraging multiple implicit feedback for personalized recommendation with neural network. In *Proceedings of the 1st International Conference on Artificial Intelligence and Advanced Manufacturing*, pages 6:1–6:6, 2019.

[130] J. Weston, S. Chopra, and A. Bordes. Memory networks. In *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[131] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan. Session-based recommendation with graph neural networks. In *Proceedings of the 33th AAAI Conference on Artificial Intelligence*, pages 346–353, 2019.

[132] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester. Collaborative denoising autoencoders for top-n recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*, pages 153–162, 2016.

[133] L. Xia, C. Huang, Y. Xu, P. Dai, B. Zhang, and L. Bo. Multiplex behavioral relation learning for recommendation via memory augmented transformer network.

In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 2397–2406, 2020.

[134] S. Xing, F. Liu, X. Zhao, and T. Li. Points-of-interest recommendation based on convolution matrix factorization. *Appl. Intell.*, 48(8):2458–2469, 2018.

[135] G. Xu, Z. Wu, Y. Zhang, and J. Cao. Social networking meets recommender systems: survey. *Int. J. Soc. Netw. Min.*, 2(1):64–100, 2015.

[136] W. Xu, G. Xu, Y. Wang, X. Sun, D. Lin, and Y. Wu. Deep memory connected neural network for optical remote sensing image restoration. *Remote. Sens.*, 10(12):1893, 2018.

[137] Y. Xu, Y. Zhu, Y. Shen, and J. Yu. Leveraging app usage contexts for app recommendation: a neural approach. *World Wide Web*, 22(6):2721–2745, 2019.

[138] H. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen. Deep matrix factorization models for recommender systems. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3203–3209, 2017.

[139] H. Yin, B. Cui, X. Zhou, W. Wang, Z. Huang, and S. W. Sadiq. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Trans. Inf. Syst.*, 35(2):11:1–11:44, 2016.

[140] J. Yin, C. Liu, J. Li, B. Dai, Y. Chen, M. Wu, and J. Sun. Online collaborative filtering with implicit feedback. In *Proceedings of the 24th International Conference on Database Systems for Advanced Applications*, volume 11447, pages 433–448, 2019.

[141] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan. A dynamic recurrent model for next basket recommendation. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 729–732, 2016.

[142] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE Trans. Neural Networks Learn. Syst.*, 23(8):1177–1193, 2012.

[143] U. Yun and G. Lee. Sliding window based weighted erasable stream pattern mining for stream data applications. *Future Gener. Comput. Syst.*, 59:1–20, 2016.

[144] W. Zaremba, I. Sutskever, and O. Vinyals. Recurrent neural network regularization. *CoRR*, abs/1409.2329, 2014.

[145] H. Zeng and Q. Ai. A hierarchical self-attentive convolution network for review modeling in recommendation systems. *CoRR*, abs/2011.13436, 2020.

[146] P. Zhang, Z. Zhang, T. Tian, and Y. Wang. Collaborative filtering recommendation algorithm integrating time windows and rating predictions. *Appl. Intell.*, 49(8):3146–3157, 2019.

[147] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv.*, 52(1):5:1–5:38, 2019.

[148] Y. Zhang and X. Chen. Explainable recommendation: A survey and new perspectives. *Found. Trends Inf. Retr.*, 14(1):1–101, 2020.

[149] Y. Zhao, S. Wang, Y. Wang, and H. Liu. Stratified and time-aware sampling based adaptive ensemble learning for streaming recommendations. *Appl. Intell.*, pages 1–21, 2020.

[150] Y. Zhao, S. Wang, Y. Wang, H. Liu, and W. Zhang. Double-wing mixture of experts for streaming recommendations. In *Proceedings of the 21st International Conference on Web Information Systems Engineering*, volume 12343 of *Lecture Notes in Computer Science*, pages 269–284, 2020.

[151] L. Zheng, V. Noroozi, and P. S. Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the 10th ACM International Conference on Web Search and Data Mining*, pages 425–434, 2017.

[152] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. Graph neural networks: A review of methods and applications. *CoRR*, abs/1812.08434, 2018.

[153] X. Zhou, C. Mascolo, and Z. Zhao. Topic-enhanced memory networks for personalised point-of-interest recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3018–3028, 2019.

[154] Z. Zhu, S. Sefati, P. Saadatpanah, and J. Caverlee. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 1121–1130, 2020.