Hypergraph Representation Learning for Efficient Recommender Systems

Towards Better Performance at Lower Computational Cost

Tendai Mukande

Supervisors:

Prof. Noel O'Connor (**DCU**) and Dr. Ruihai Dong (**UCD**)

A thesis presented for the degree of Doctor of Philosophy

School of Computing

Dublin City University

December 2025

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of **Doctor of Philosophy** is entirely my own work, and that I have exercised reasonable care to ensure that the work is original and have conformed to the regulations on the use and declaration of Generative AI, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signature: **Tendai Mukande** Student Number: **A20213350**

Date: **05/12/2025**

Declaration on Gen AI tool use

I hereby certify that no Generative Artificial Intelligence (Gen AI) tools have been used in the creation of the thesis.

Signature: **Tendai Mukande** Student Number: **A20213350**

Date: **05/12/2025**

Acknowledgements

I sincerely thank my supervisors **Dr. Annalina Caputo (RIP), Dr. Esraa Ali, Dr. Ruihai Dong and Prof. Noel O'Connor** for their invaluable support and guidance. I am also grateful to **Dr. Kevin McGuinness** (RIP) for introducing Graph Neural Networks (GNNs) and Transformers in the EE613 module, and to independent panel member **Dr. Paul Clarke**, for his regular support. Special thanks to my mentors at the RecSys'22 Doctoral Symposium, **Prof. Julian McAuley (University of California, San Diego (USA) and Dr. Hongning Wang (University of Virginia (USA)** for their insightful discussions. I greatly appreciate the administrative support of **Angela, Deirdre and Vicky**, as well as the opportunity to complete my industry placement at **Kantan AI, Keywords Studios Group**, which was coordinated by **Jonathan Costello (University College Dublin)**. My heartfelt gratitude goes to **ML-LABS, the School of Computing, the Faculty of Engineering and Computing, Dublin City University**, friends and family for their moral support and collaboration during the Ph.D. journey. I also appreciate the **Irish Centre for High-End Computing (ICHEC)** for providing the compute infrastructure used to execute experiments for this research project. This research was supported by **Taighde Éireann – Research Ireland** under Grant number 18/CRT/6183.

Dedication

Joseph, James, Jemimah and Jonathan

Thank you, Annalina.

Contents

List of Figures

List of Tables

List of Publications

1. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. "**A Flash Attention Transformer for Multi-Behaviour Recommendation**". In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management,* 2023, pp. 4210–4214.

2. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"A Sparse Sinkhorn Transformer for Multi-Behaviour Recommendation."** In *Proceedings of the 19th IEEE International Conference on e-Business Engineering (ICEBE 2023).* IEEE. 2023, pp. 90–96.

3. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"MMCRec: Towards Multi-modal Generative AI in Conversational Recommendation."** In *Proceedings of the 46th European Conference on Information Retrieval.* Springer. 2024, pp. 316–325.

4. **Tendai Mukande**. **"Heterogeneous Graph Representation Learning for Multi- Target Cross-Domain Recommendation."** In *Proceedings of the 16th ACM Conference on Recommender Systems.* 2022, pp 730 - 734.

5. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"Towards Efficient Hypergraph Representation Learning for Multi-Behaviour Recommender Systems."** *Companion Proceedings of the ACM on Web Conference 2025*, pp 1204-1208.

6. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"A Hypergraph-MoA Framework for Multi-Behaviour Recommendation."** Planned submission: *20th ACM Conference on Recommender Systems.*

7. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"A Survey of Hypergraph Representation Learning Methods for Recommender Systems."** *Planned Submission: ACM Computing Surveys.*

8. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **"HyperRec: Hyperbolic Geometry for Scalable Recommender Systems."** Planned Submission: *20th ACM Conference on Recommender Systems.*

9. **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **"Towards Lightweight Hypergraph Transformers Framework For Recommender Systems"**. Planned Submission: *ACM Transactions in Recommender Systems.*

10. **Tendai Mukande**, Dimiter Dinkov, Riccardo Superbo and Noel E. O'Connor. **"Accelerating Workflows in Video Game Localisation: A Recommender System for Review and Post-Edit Assignments."** *ACM Transactions on Recommender Systems: Special Issue on Recommender Systems in Industry.* 2025.

List of Abbreviations

BERT Bi-directional Encoder Representations from Transformers. 16

CDR Cross-Domain Recommendation. 53

DCT Discrete Cosine Transform. 125

DNN Deep Neural Network. 17

GAN Generative Adversarial Network. 16

GAT Graph Attention Network. 32

GCN Graph Convolutional Network. 31

GDPR General Data Protection Regulation. 53, 163

GNNs Graph Neural Networks. 2, 17

GPU Graphics Processing Unit. 103

GT Graph Transformer. 61

HBM High Bandwidth Memory. 104

HR Hit Ratio. 21

LLM Large Language Model. 54

MLP Multi-Layer Perceptron. 71

MPGNN Message Passing Graph Neural Network. 57

MPHGNN Message Passing Hypergraph Neural Network. 69

MRR Mean Reciprocal Rank. 21

NCF Neural Collaborative Filtering. 17

NDCG Normalised Discounted Cumulative Gain. 21

OOM Out-of-Memory. 138

POI Point of Interest. 45

RL Reinforcement Learning. 16

RPE Relative Positional Embedding. 109

RSGD Riemannian Stochastic Gradient Descent. 37

RSs Recommender Systems. 1

SRAM Static Random Access Memory. 104

TPU Tensor Processing Unit. 103

VAE Variational Autoencoder. 16

Hypergraph Representation Learning for Efficient Recommender Systems: Towards Better Performance at Lower Computational Cost

Tendai Mukande

Abstract

Recommender systems (RSs) facilitate decision making by providing customised suggestions that are tailored to user preferences. Most existing RS methods focus on single-type behaviour modelling, such as ratings, whereas real-world user interactions are diverse, including activities such as *view*, *add-to-cart* and *purchase*. The main hypothesis in this thesis is that incorporating multi-type behavioural indicators enables the development of more comprehensive models that better reflect user preferences, enhancing recommendation performance. To capture heterogeneous user-item interactions more effectively, graph transformer (GT)-based RS methods have been adopted in the literature. These GT algorithms offer a hybrid framework that improves recommendation performance. However, most existing GNN-based approaches struggle with scalability in large-scale settings and have limited capability in modelling higher-order dependencies beyond pairwise user-item interactions. To address these limitations, hypergraph neural networks (HGNNs) have been proposed, which allow the representation of higher-order relationships involving multiple nodes. Despite their advantages in capturing intricate dependencies, HGNN-based methods remain computationally expensive, posing challenges in computational resource-constrained environments. In addition, applying model compression techniques to address this issue often leads to performance degradation. Motivated by these challenges, this research explores the question: *Is it possible to reduce the computational cost of HGNN-based RS methods without compromising accuracy in large-scale settings?* To address the limitations of existing methods, four novel approaches are proposed, based on: 1) lightweight hypergraph transformers, 2) hardware-algorithm co-design, 3) *hyperbolic representation learning* and 4) HGNN-enhanced Mixture-of-Agents (MoA) frameworks. The methods introduced in this research yield effective models that achieve improved performance while reducing memory usage and runtime, providing a foundation for more efficient and scalable RSs.

Keywords: *Hypergraph Neural Networks*, *Transformers*, *Mixture-of-Agents*, *Hyperbolic Representation Learning*, *Recommender Systems*, *Computational Efficiency*, *Hyperbolic Neural Networks*.

Chapter 1

Introduction

1.1 Background

Recommender Systems (RSs) deliver relevant content to users by retrieving information from multiple sources to generate personalised suggestions. In doing so, they minimise the information overload that users often encounter on online platforms [98]. The provision of personalised suggestions enables products and services to be more closely aligned with user preferences. Personalisation has become an essential part of the digital age, shaping the decisions people make and the content they interact with every day. For example: Netflix[1], Amazon[2], Spotify[3] and YouTube[4] platforms deliver personalised content based on user behaviour traces (*such as views, purchases ratings and likes*) [109]. Due to the use of RS on many platforms, improving both the efficiency and the quality of recommendation in real-world scenarios has been of great interest in academia and industry [242].

In this thesis, a **real-world recommendation scenario** is defined as a practical context that involves large-scale datasets, complex user-item interactions and multiple objectives. These objectives could include factors such as maximising user satisfaction, increasing user engagement, ensuring diversity in recommendations, op-

[1]https://www.netflix.com/
[2]https://www.amazon.com/
[3]https://open.spotify.com/
[4]https://www.youtube.com/

timising long-term user retention, and balancing between relevance and novelty. For example, an RS on an e-commerce platform might recommend products that match user preferences (maximising satisfaction) while maintaining sufficient diversity to encourage exploration (promoting novelty). At the same time, it can also account for business objectives such as fostering user loyalty and increasing revenue.

The modelling paradigm in RSs has evolved from neighbourhood methods [277, 274] to *representation learning*-based frameworks [48, 75, 200]. Neighbourhood methods, such as *collaborative filtering* [107, 256], generate recommendations based on historical user-item interactions. These approaches have been widely used in traditional RSs due to their simplicity [337]. However, when applied to heterogeneous datasets, they face challenges such as sparsity and limited scalability [329]. In this context, "heterogeneous" refers to datasets that consist of different categories of users, items and interactions (e.g., ratings, views and purchases). This limits the applicability of RSs in complex real-world scenarios where user-item interactions are often incomplete or highly diverse [21].

Most existing research has focused on modelling user behaviour of a single type, such as user ratings, while real-world scenarios involve multi-type behaviour [111]. Figure 1.1 illustrates a real-life scenario of user behaviour in the e-commerce domain. This example shows that users can explore and interact with products from multiple categories in a single session. For example, a user might search for electronics, add a fashion item to their cart, and then proceed to purchase a book. The main challenge with the single-user behaviour modelling approach is that it does not capture the dynamics of various user–item interactions, thus limiting the ability of these approaches to provide novel suggestions to the user [19]. In addition, the RS should be able to capture the relationships between different product categories to provide complementary recommendations. For example, if a user buys a camera, the system should be able to recommend related accessories such as photography books.

Graph Neural Networks (GNNs) have been applied to model multi-type user

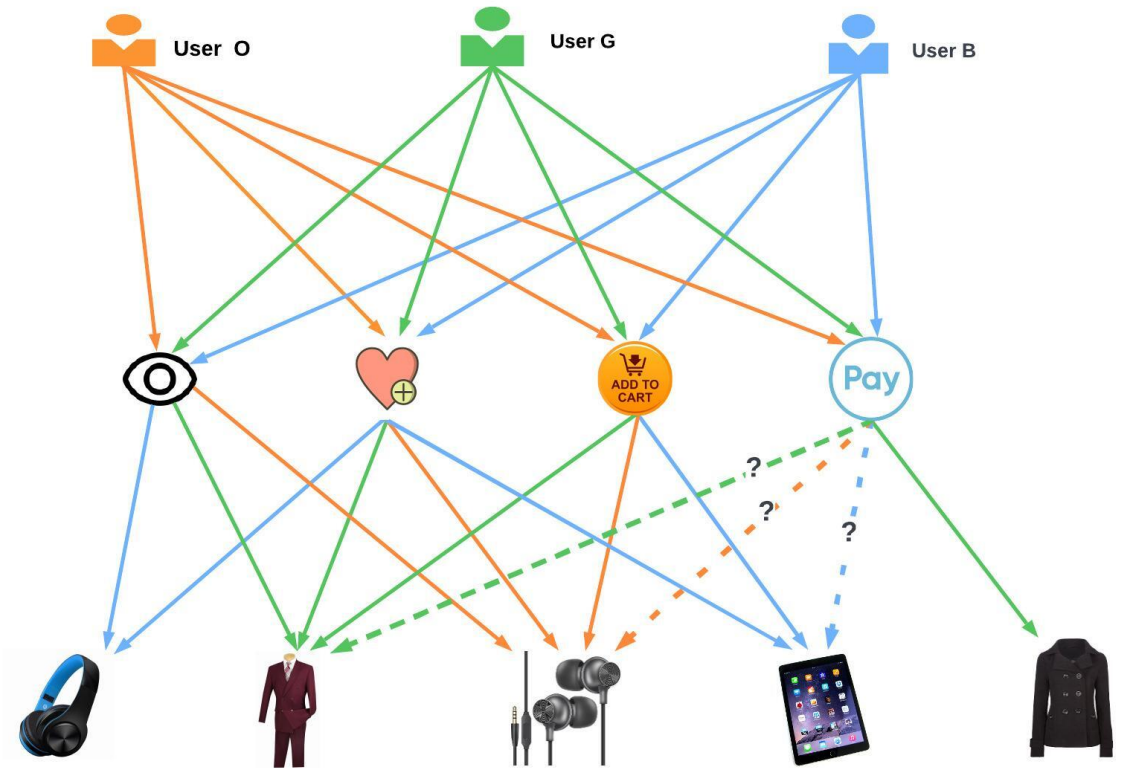

Figure 1.1: A real-world e-commerce recommendation scenario with multiple user behaviour types, where complex multi-way interactions among users and items are best modelled using a hypergraph.

behaviour in order to provide better recommendations [19, 111, 321]. In these methods, collaborative user behaviour such as *"view"*, *"add-to-cart"* and *"add-to-favourites"* is incorporated with the goal of improving the accuracy of predicting a target behaviour such as *purchase*. However, most existing GNN-based RS methods are limited to pairwise interactions and cannot model higher-order interactions such as those that involve three or more nodes simultaneously [138, 161]. To improve *graph representation learning* performance, some research works combine GNN with the transformer-based *self-attention* mechanism[5], resulting in hybrid graph transformer (GT) models [276, 350]. This approach takes advantage of the strengths of both architectures, offering a framework that captures complex patterns of interaction between users and items, improving recommendation performance [19]. Transformers have been applied to model heterogeneous data, such as recommen-

[5]The transformer-based neural network architecture introduced by Vaswani et al. [237] has become a foundational model for various tasks in natural language processing (NLP), computer vision [169] and life sciences [318].

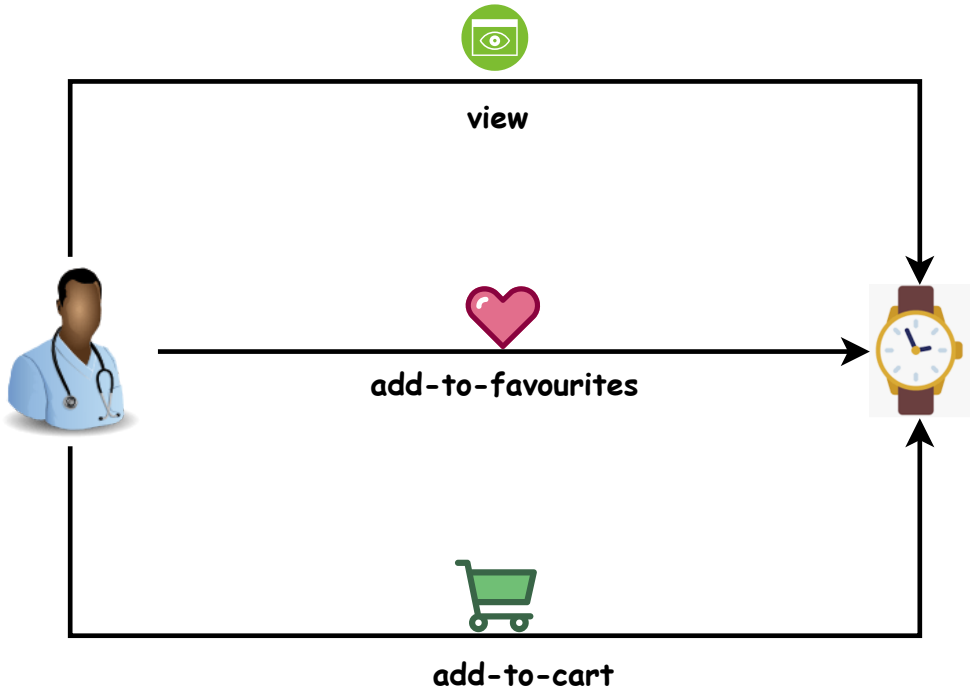

Figure 1.2: Example of a real-world higher-order user-item interaction involving triadic behaviour: *"view"*, *"add-to-cart"* and *"add-to-favourites"*. GNN methods are limited to pairwise interactions and do not effectively model higher-order relations.

dation scenarios in which users exhibit diverse behaviours in multiple dimensions [183]. The *self-attention* mechanism of the transformer allows the capture of global patterns and the aggregation of information at distant graph nodes [209]. In this way, GT-based models can incorporate contextual information on user behaviour (such as time of day and location), leading to better recommendations [244].

To address limitations of GNNs in modelling higher-order interactions, exploring RS methods based on hypergraph representation learning has become an active research area [279]. HGNNs can model relationships that involve multiple nodes at the same time, providing more expressive frameworks to capture complex interactions [142, 310]. In this context, **expressivity** refers to the ability of a model to capture complex structural dependencies and distinguish between different graph topologies. Unlike GNNs, a hyperedge can link three or more nodes simultaneously [255], allowing the representation of higher-order relationships that frequently occur in real-world scenarios, as illustrated in Figure 1.2. In contrast, standard GNNs represent these interactions as multiple pairwise edges, linking a user to each prod-

uct individually, which can result in the loss of important information about the joint nature of the interaction [315]. This limits the model's ability to fully capture the contextual and co-occurrence patterns that drive user preferences in complex recommendation scenarios [142].

1.2 Motivation

Given the potential of HGNN-based RS methods [219, 255], this research focuses on challenges that arise when deploying these models in real-world scenarios. One major concern is efficiency, raising the following question: *Can we design HGNN-based RS methods that are efficient in large-scale environments, while maintaining high performance?* Moreover, optimising the efficiency of these algorithms opens up new possibilities, such as implementing RSs on smartphones for personalised health monitoring. The central question driving this thesis is:

> **Can we reduce the memory and time cost of HGNN-based RS methods without compromising the quality of recommendation? What novel techniques can be applied to improve the performance-efficiency trade-off of these methods in real-world settings?**

In real-world scenarios, graphs can be large and complex, with diverse user interactions across various item types, leading to millions of nodes and edges. This results in the need for large computational resources. This high computational resource demand comes with an increase in energy consumption for model training and inference, which contributes to a larger carbon footprint [128]. Developing more efficient RS models is one way to contribute to the United Nations Sustainable Development Goal 13 on combating climate change [202, 222]. In existing research, practical concerns such as efficiency are often overlooked or addressed with solutions that seem feasible but, in practice, offer limited real-world benefits [182]. Effective solutions that produce measurable improvements are challenging, mainly due to the

difficulty in balancing modelling requirements with algorithmic and hardware constraints [302]. This research aims to tackle these challenges by focusing on four key areas: lightweight graph transformers, hardware-software co-design, non-message-passing algorithms and hypergraph-enhanced Mixture-of-Agents (MoA) frameworks.

1.2.1 Graph Representation Learning Bottlenecks

Existing GNN-based multi-type behaviour RSs have shown improved performance compared to single-type behaviour models [268]. However, they fall short in large-scale real-world scenarios, where large amounts of heterogeneous data are frequently generated [63, 211, 313]. One limitation of GNN methods is their inability to effectively represent complex relationships that consist of more than two entities [56, 142]. For example, some user-item interactions involve higher-order relations, such as the triadic behaviour of *"view", "add-to-cart"* and "add-to-favourites" in the same user-item set, as illustrated in Figure 1.2. Furthermore, GNNs operate with a localised receptive field, where each node aggregates information from its immediate neighbours through message-passing layers [105]. Information loss from aggregation limits the model's ability to capture long-range dependencies and global structural information [245], which is challenging in large-scale graphs where meaningful relationships extend multiple hops beyond the effective neighbourhood radius of the GNN [72]. In addition, GNN-based methods are prone to oversmoothing[6] and over-squashing[7] problems in large-scale recommendation scenarios.

1.2.2 High Computational Cost

State-of-the-art RS methods based on GNN and transformer architectures have limited scalability due to high computational complexity [233]. In particular, GNN-based RSs become computationally expensive as the size of the graph increases

[6]Oversmoothing refers to a phenomenon in GNN where node representations become indistinguishable as layers increase, leading to the loss of useful information.

[7]Oversquashing refers to the phenomenon in GNNs where long-range dependencies are compressed into low-dimensional representations, limiting the model's ability to propagate information effectively. This is discussed in detail in Chapter 2.

[352]. Similarly, transformers have quadratic memory and time complexity relative to the number of nodes in the graph [233], which limits their scalability. Algorithms that have quadratic complexity are inefficient for large datasets because the time or memory required to process them grows proportionally to the square of the input size [232]. This makes practical deployment challenging for large-scale HGNN-based RSs, especially in scenarios with limited computational resources. In light of the highlighted challenges, this thesis presents architectural innovations designed to efficiently capture higher-order, complex user-item interactions, aiming to enhance recommendation performance while reducing the computational cost of HGNN-based RS methods in real-world scenarios.

1.3 Problem Statement

This research aims to improve the efficiency of HGNN-based recommender systems by introducing novel approaches that reduce computational cost **without compromising but ideally improving performance** in large-scale scenarios. This thesis introduces five innovative methods that address the aforementioned challenges. In doing so, the reported research contributes to advancing the practical applicability of HGNN-based RS models. The results of this research represent a significant step toward the development of resource-efficient and high-performance RSs.

1.3.1 Research Questions

This work addresses three research questions aimed at enhancing recommendation performance while minimising computational costs in large-scale real-world scenarios. To address these research questions, five HGNN-based RS frameworks are proposed, designed to optimise both performance and efficiency while enhancing scalability in practical settings.

> **RQ1: Model compression in GNN / transformers, and their hybrid models often results in accuracy loss. Can lightweight**

Graph Transformer approaches address this issue without degrading recommendation performance?

This research question investigates whether the incorporation of hypergraph representation combined with efficient lightweight attention can improve recommendation performance. The focus is on developing faster and more scalable RSs that maintain high performance, even in complex large-scale settings. Model compression in GNN and transformer-based models often leads to accuracy degradation [159]. RQ1 explores whether lightweight graph transformer-based approaches can mitigate high computational cost while maintaining strong recommendation performance.

RQ2: The message-passing scheme in HGNN models is prone to issues such as oversmoothing and oversquashing in large-scale settings. Does the use of HGNN-based RS methods without message-passing improve model performance and efficiency?

HGNN-based message-passing often suffers from limitations such as oversmoothing, where node representations become indistinguishable, and oversquashing, where information from distant nodes is compressed into fixed-size representations as the graph size increases, leading to degraded performance in large-scale settings. The hypothesis is that addressing these limitations requires moving beyond the message-passing paradigm. This question explores how to design and optimise large-scale RS methods that do not apply message-passing to address performance, efficiency, and scalability challenges present in real-world RSs without compromising recommendation performance. This question leads to two closely related questions.

RQ 2.1: *Does hardware-algorithm co-design (aligning the recommendation algorithm's computational patterns with the structure of the underlying hardware accelerator platform, i.e. GPU), without applying HGNN-message-passing, improve performance and efficiency in real-world RS scenarios?*

The irregular structure of hypergraph data introduces memory bottlenecks and redundancies, which hinder efficient computation and memory access, which are essential to achieve high performance on hardware platforms. Hardware accelerator platforms such as GPUs and TPUs, which are optimised primarily for workloads with regular computational and memory access patterns, struggle to efficiently accelerate these methods [27].

> **RQ 2.2** explores the following: *HGNN message-passing is prone to over-smoothing and oversquashing due to repeated averaging in hierarchical real-world data (e.g., Clothing → Jacket → Black Biker Jacket), as well as high computational cost. To address these issues, the following investigation is conducted: Is hyperbolic hypergraph representation learning, without relying on extensive message-passing, effective in improving performance and efficiency in real-world RS settings?*

Traditional HGNNs operating in Euclidean space often struggle to represent hierarchical and scale-free structures efficiently. Hyperbolic geometry has been applied to embed hierarchical data with lower distortion and fewer dimensions and has been shown to allow better separation of nodes in large and complex graphs [192]. Thus, the hypothesis is that incorporating hyperbolic geometry can improve HGNN expressivity by enabling richer and more discriminative representations, thus improving the recommendation performance.

> **RQ3: Most real-world recommendation scenarios exhibit complex and evolving user-item interaction dynamics, which makes static hypergraph representation learning models insufficient to capture temporal patterns effectively. This raises the question: Does hypergraph-enhanced MoA modelling effectively address these challenges?**

In practice, user preferences are dynamic and continue to evolve. However, most existing GNN-based RS methods are static, making them inadequate for capturing

these dynamics in practical settings [53]. As a result, these methods often show suboptimal performance when applied to dynamic scenarios [161]. Moreover, recent methods such as LLMs [97, 140], which attempt to address this limitation, are often computationally intensive [35]. Thus, temporal *representation learning* is required to capture these evolution patterns [22]. In RQ3, investigation is carried out on whether HGNN-enhanced learning combined with a Mixture-of-Agents (MoA) architecture can achieve strong performance while minimising computational cost.

1.4 Thesis Contributions

In real-world scenarios, HGNN-based RS methods can be large and complex, often scaling to millions of nodes and edges. In such cases, architectural innovations are necessary to allow models to scale efficiently without compromising performance. Methods introduced in this thesis cover four areas: lightweight graph transformers, hardware-algorithm co-design without message-passing, hyperbolic *hypergraph representation learning* and MoA frameworks. These **contributions** are summarised below.

Literature Survey on HGNNs. In **Chapter 2**, an overview of RSs, GNNs, hyperbolic geometry and transformers is presented, followed by a review of the state-of-the-art methods and discussion of open challenges in HGNN-based recommendation. The literature survey, which covers **Chapter 2** content in greater detail, has been submitted to ACM.

> **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **A Survey of Hypergraph Representation Learning for Recommender Systems.** Under review: *ACM Computing Surveys.*

GTRec: Towards Lightweight Hypergraph Transformers for Multi-Behaviour Recommendation. Graph transformers (GTs) have emerged as a promising architecture for *graph representation learning* tasks [94, 133, 259]. However,

despite their potential, they face scalability challenges [215]. Model compression techniques, such as pruning or quantisation, that are commonly applied to improve efficiency in GT-based methods have shown degraded model performance [198]. To address these issues, **GTRec** is introduced in **Chapter 3**, a lightweight graph transformer framework that effectively captures heterogeneous user-item interactions. GTRec combines innovative modules to achieve superior recommendation performance compared to state-of-the-art methods, with significantly lower computational costs. This work will be submitted to the *ACM Transactions on Recommender Systems (TORS)* journal.

> **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **"Towards a Lightweight Hypergraph Transformer Framework for Multi-Behaviour Recommendation"**. Planned Submission: *ACM TORS* journal.

STMBR: A Sparse Sinkhorn Attention Model for Multi-Behaviour Recommendation. To address the high computational cost of GT-based methods without compromising performance, STMBR, another lightweight GT-based method, is introduced in Chapter 3. This is achieved by partitioning input user-item interaction sequences into blocks and the use of a meta-sorting network that learns to rearrange the block sequences based on token relevance. The work reported in this section has been published.

> **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **"A Sparse Sinkhorn Transformer for Multi-Behaviour Recommendation"**. In: *"Proceedings of the 19th IEEE International Conference on e-Business Engineering.* ICEBE. 2023, pp. 90–96.

FATH: A Hardware-Algorithm Co-design Framework for Recommender Systems. The irregularity of graph-structured data poses significant challenges for efficient computation and memory access, both of which are essential to enhancing

model performance and efficiency. Hardware accelerator platforms, such as GPU and TPU, are mainly optimised for workloads that involve regular and predictable patterns of computation [27]. In **Chapter 4**, the **FATH** framework is introduced with the aim of improving the efficiency of HGNN-based RSs. FATH addresses the challenges of adapting algorithms to hardware constraints, enabling the scaling of HGNN models for deployment on devices with limited computational and memory resources. The work reported in this chapter has been published in *CIKM'23* and *WWW'25* conference proceedings.

> **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **"A Flash Attention Transformer for Multi-Behaviour Recommendation".** In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management.* 2023, pp. 4210–4214.
>
> **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"Towards Efficient Hypergraph Representation Learning for Multi-Behaviour Recommender Systems".** *Companion Proceedings of the ACM on Web Conference 2025*, pp 1204-1208.

HyperRec: Hyperbolic Representation Learning for Scalable Recommender Systems. Most Graph Transformer (GT) models have high computational cost and therefore limited scalability in large graphs [215]. Hyperbolic geometry has shown potential in modelling heterogeneous data that exhibits hierarchical structures and scaling to millions of nodes [192]. To address the limitations of GNN message-passing methods that are discussed in the previous paragraph, *HyperRec*, a novel hyperbolic-based model, is introduced in **Chapter 5**. HypeRrec uses a hyperbolic *self-attention* mechanism to capture complex relationships between multiple user behaviours in a unified framework for scalable recommendation. The work reported in this chapter has been submitted to the *RecSys'26* conference.

> **Tendai Mukande**, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel O'Connor. **"HyperRec: Towards Hyperbolic Representation Learn-**

ing for Scalable Recommender Systems." Planned Submission: *20th ACM Conference on Recommender Systems (RecSys'26).*

HGLMRec: An Efficient Hypergraph-MoA Framework for Recommender Systems. User preferences and item characteristics change over time, making it important for recommender systems to adapt to these dynamics [312]. Most existing HGNN-based recommendation methods are limited in capturing these dynamic scenarios [53]. To address these issues, **HGLMRec**, a model that incorporates a hypergraph encoder combined with a Mixture-of-Agents framework is introduced, which enhances performance and reduces computational overhead. The work reported in this chapter will be submitted to the *RecSys'26* conference.

Tendai Mukande, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"An Efficient Hypergraph-MoA framework for Multi-Behaviour Recommendation."** Planned submission:*20th ACM Conference on Recommender Systems (RecSys'26).*

In addition, there are other publications related to this research, listed below, but they fall outside the scope of this thesis.

Tendai Mukande. **"Heterogeneous Graph Representation Learning for Multi- Target Cross-Domain Recommendation."** In *Proceedings of the 16th ACM Conference on Recommender Systems.* 2022, pp 730 - 734.

Tendai Mukande, Esraa Ali, Annalina Caputo, Ruihai Dong and Noel E. O'Connor. **"MMCRec: Towards Multi-modal Generative AI in Conversational Recommendation."** In *Proceedings of the 46th European Conference on Information Retrieval.* Springer. 2024, pp. 316–325.

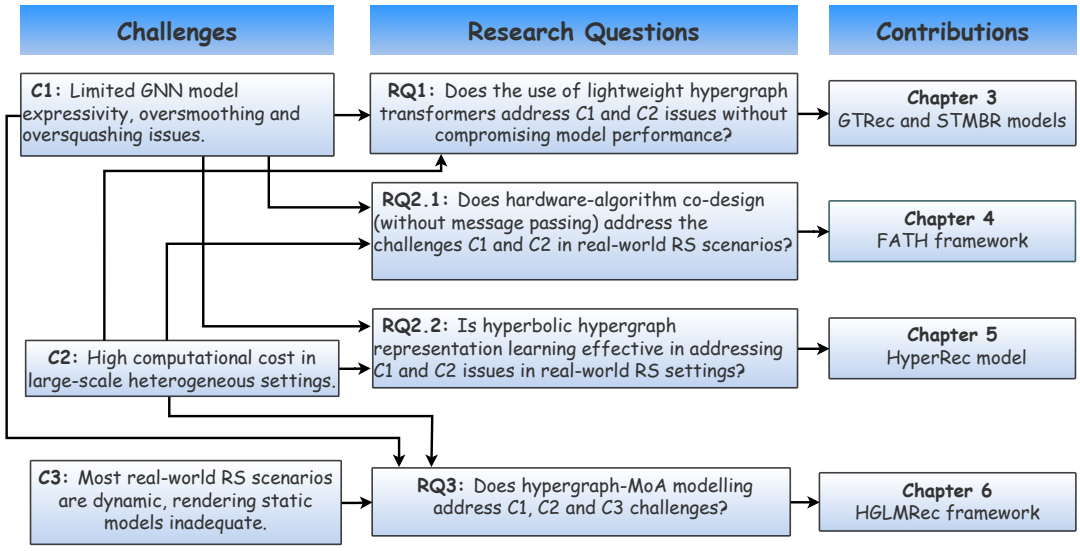

Figure 1.3: Thesis content structure: Overview of the challenges explored, research questions addressed and contributions proposed in this thesis.

1.5 Thesis Outline

In **Chapter 2**, an overview of RSs, GNNs, hyperbolic neural networks and transformers is provided, followed by a literature review on state-of-the-art methods and open challenges in hypergraph-based recommendation. In **Chapter 3**, the GTRec and STMBR methods are presented. **Chapters 4 and 5** introduce FATH and HyperRec, respectively. In **Chapter 6**, HGLMRec, a Hypergraph-MoA recommendation model is introduced. Finally, **Chapter 7** summarises the entire thesis and presents opportunities for future research. A visualisation of the thesis contributions is illustrated in Figure 1.3.

Chapter 2

Background and Literature Survey

In this chapter, a useful background on RSs, GNNs and transformers is provided. This is followed by a review of state-of-the-art recommendation methods based on HGNNs. Lastly, open challenges and future directions towards the development of hypergraph-based real-world RSs are discussed.

2.1 Recommender Systems

RSs have evolved significantly over the years, from traditional methods such as matrix factorisation, content-based and collaborative filtering to advanced deep learning-based approaches that model complex relationships [329].

2.1.1 Overview of Recommender Systems

Traditional methods. Traditional RS methods include content-based filtering, collaborative filtering and matrix factorisation. Content-based filtering recommends items similar to those in which a user has previously shown interest, based on common features of the items. For example, a movie recommender might use genres, directors and actors to find similar movies [235]. User-based collaborative filtering recommends items to a user based on the preferences of similar users, which are measured using methods such as cosine similarity or Pearson correlation [36]. In item-based collaborative filtering, items similar to those in which the user has previ-

ously shown interest are recommended. The similarity between items is calculated on the basis of user ratings. Matrix factorisation uses techniques such as Singular Value Decomposition to decompose the user-item interaction matrix to identify latent factors that represent the characteristics of the user and the item [293]. Hybrid methods combine methods such as collaborative and content-based filtering to mitigate their individual limitations and improve the quality of recommendations.

Deep Learning Methods. Modern recommendation methods use neural networks to capture complex patterns and interactions between users and items. Some advanced RS methods incorporate contextual information, such as time and location, into the recommendation process [18]. In one class of deep learning methods, contrastive learning aims to learn representations for items and users such that similar items are close in the embedding space [60]. This approach helps to learn representations that capture the similarity and dissimilarity between items effectively. Reinforcement Learning (RL) models interactions as a sequence of actions in which the objective is to maximise cumulative rewards [43]. For recommendations, the system learns policies that optimise long-term user engagement. RL models adapt over time based on user interactions, making them well suited for environments where user preferences evolve [3]. Other research works apply methods such as Generative Adversarial Network (GAN) and Variational Autoencoder (VAE) which generate new content (user profiles or items) based on learned patterns [42]. GANs have been applied to create high-quality synthetic data that is used to train RSs with the goal of improving performance by increasing the diversity and richness of training data [264]. VAE-based models generate predictions by learning a robust latent space that captures the underlying structure of user-item interactions. VAE-based methods have been shown to be effective in capturing latent factors of user preferences and item characteristics [311]. Another class of RS methods such as Bidirectional Encoder Representations from Transformers (BERT) [224] and GPT4Rec [136] leverage the transformer self-attention mechanism [237] to capture complex dependencies and contextual information in user-item interactions. The self-attention

mechanism allows for dynamic weighting of various parts of the input data, which is useful for handling sequential and context-sensitive information in recommendations [25].

State-of-the-art RS Methods. To address the limitations of traditional approaches, representation learning-based methods have been introduced and have shown better recommendation performance [217, 349]. For example, deep learning-based recommendation models have attracted great interest in academic research and industrial applications due to their ability to incorporate heterogeneous data such as contextual and visual information [212]. These approaches have been shown to capture complex patterns present in user-item interactions. Examples include Neural Collaborative Filtering (NCF) [86], deep matrix factorisation [293] and autoencoder-based techniques that map user-item interactions into dense, low-dimensional latent representations [96, 305, 311]. Despite their performance improvement over traditional methods, Deep Neural Network (DNN) approaches face several limitations. One issue is that they struggle to explicitly model relationships beyond simple user-item pairs and often fail to capture relational data such as user-user or item-item correlations [267, 285]. Moreover, most DNN-based models are less able to adapt to dynamic user-item interactions [269]. Recent work has explored the use of GNNs to represent the relationships between users and items in real-world scenarios [244]. Using GNNs, user-item interactions are represented in lower-dimensional vector spaces where multiple relations (edges) connect user/items (nodes) [234]. GNN-based methods have been shown to improve *representation learning* through feature aggregation [213], facilitating the incorporation of structured external information, such as social relationships between users, and providing a unified perspective to model abundant heterogeneous data [63, 268].

In general, transformers and GNNs are currently considered among the most advanced methods due to their ability to handle complex data relationships and capture contextual information effectively [350]. *Graph representation learning* provides a unified approach to model heterogeneous data in RS by encoding the topological

structure that aims to improve user and item representations [133]. For example, the user-item interaction data is represented by a graph where users and items are nodes, and edges represent the interactions between the corresponding users and items [268]. Compared with the other DNN methods, GNNs are more flexible and convenient to model multi-hop connectivity from user-item interactions [63]. Transformer-based methods have also attracted significant research interest, with their applications extending to generative methods such as large-language models (LLMs) [250].

2.1.2 Evaluation of Recommender Systems

The evaluation of recommender systems determines their effectiveness and reliability in providing relevant recommendations to users. In this thesis, we adopt the 80/10/10 data split, allocating 80% for training, 10% for validation, and 10% for testing, a strategy commonly used in RS evaluations [74]. This partitioning ensures that the model is trained using the largest portion of the data, facilitating the learning of robust user-item interaction patterns [89]. The validation set allows for tuning hyperparameters and preventing overfitting, while the test set provides an unbiased evaluation of the model's performance on unseen data. Compared to alternatives such as the 60/20/20 split, the 80/10/10 ratio offers a balanced approach, providing ample data for training while preserving sufficient data for reliable evaluation, as shown in Table 2.1. The 60/20/20 setting is often used in small-scale or highly imbalanced datasets to provide a larger validation/test pool, but it reduces the amount of training data, which can degrade the accuracy of recommendations in large and sparse datasets [37]. Furthermore, recent studies [13] adopt the 80/10/10 split as standard practice in RS performance benchmarking, highlighting its effectiveness for large-scale experiments.[1]

[1]The 80/10/10 split is widely used in RS evaluation frameworks such as RecBole [336].

Split Ratio	Training Set Size	Validation/Test Sets	Typical Use Cases
80/10/10	Large training pool, enabling better learning of user–item interaction patterns	Moderate validation/test size, sufficient for reliable evaluation in large-scale datasets	Widely adopted in large-scale RS benchmarks
60/20/20	Reduced training pool, which may weaken generalisation in sparse data settings	larger validation/test pools, improves robustness of evaluation in small datasets	Small-scale or imbalanced datasets where more test samples are needed

Table 2.1: Comparison of dataset splitting strategies in RS evaluation.

Evaluation Protocols

Recommender systems are typically evaluated using two widely adopted protocols: *leave-one-out* evaluation and *k-fold cross-validation*.

- ***Leave-one-out* evaluation:** In this method, at least one interaction per user is left out for testing, while the remaining interactions are used for training and validation. This protocol is effective for datasets with temporal dynamics, as it uses the most recent interactions as the test set. It also mirrors real-world recommendation scenarios, where the RS predicts the next interaction of a user [37].

- ***K-fold cross-validation*:** In this method, the dataset is partitioned into k equal subsets, and the model is trained and evaluated k times, each time using a different fold as the test set and the remaining folds for training and validation. This approach offers a more reliable estimate of the generalisation of the model across different data splits and is commonly used when the datasets is relatively small or when temporal ordering is not a concern [89].

In this thesis, we use the *leave-one-out* protocol for evaluation, which is closely aligned with real-world recommendation tasks where the objective is to predict the next user-item interaction. For each user, we use the most recent interaction as the test sample and the second most recent as the validation sample, and all earlier interactions constitute the training set. This chronological split ensures that

the model is tested on later interactions, leading to a more realistic performance in practical recommendation settings [74]. Moreover, the *leave-one-out* evaluation method is computationally more efficient for large-scale datasets compared to more resource-intensive methods such as k-fold cross-validation [89].

In evaluating our methods, we adopt Cloze task settings, used in benchmarks such as [224, 307]. In the Cloze task, our objective is to predict the next item which a user will interact with, given their historical sequence of interactions [39]. Given a user u and their past interactions $S_u = [i_1, i_2, ..., i_{n-1}]$, the task is to predict the next item i_n. We consider the temporal order of interactions, reflecting how user preferences evolve over time, aligning with the goal of recommending the next item that is likely to be purchased.

RS Model Perfomance Evaluation

RS objectives can be understood from two primary perspectives. From the user's perspective, an RS aims to filter through vast amounts of information to help users discover the most relevant and interesting items. From the perspective of service providers, the goal is to improve user engagement, increase traffic and ultimately increase profits [99]. To achieve these objectives, several evaluation metrics such as accuracy, novelty and diversity are considered during the evaluation of RSs [156].

Novelty and Diversity. **Novelty** refers to the ability of the RS to present users with items they have not previously encountered [195]. Novelty ensures that users are exposed to items they have not seen before, increasing engagement and satisfaction [113]. This factor is vital in maintaining user interest and encouraging the exploration of new items [109]. **Diversity** ensures that the recommendation list includes a wide variety of items, and enhancing user satisfaction [51]. Diversity reduces redundancy in the recommendation list, providing a wider range of content and better catering to diverse user interests [201]. A system that prioritises accuracy recommends similar items; therefore, diversity is essential to provide users with more choices, thus increasing the likelihood of interaction with different items [113].

Despite the importance of novelty and diversity in recommendations, research in this area remains limited [354]. Both aspects are important in top-k recommendation tasks, where the goal is not only to maximise accuracy, but also to provide a rich and engaging user experience [113]. Commonly used metrics include *coverage*, which evaluates how much of the item space is represented; and *unexpectedness*, which quantifies how surprising or novel the recommendations are for the user [195].

Accuracy: Accuracy is a fundamental metric for evaluating the performance of an RS, as it measures how well the recommended items align with the preferences of users [51]. In this thesis, the proposed methods are primarily evaluated in terms of accuracy using commonly adopted top-k metrics: **Hit Ratio (HR), Mean Reciprocal Rank (MRR)**, and **Normalised Discounted Cumulative Gain (NDCG)** [74]. These metrics are widely used in sequential and top$k$ recommendation tasks because they directly quantify the model's ability to rank relevant items higher in the recommendation list. In addition, accuracy-orientated evaluation provides a standardised and comparable measure of performance across different models and datasets, making it suitable for benchmarking the proposed approaches against existing baselines [89].

- **Hit Ratio (HR):** HR@N evaluates the ability of a RS to place the ground-truth item within the top-N recommended items for each user. A hit is recorded if the ground-truth item appears in the top-N list. The Hit Ratio at N, commonly used in leave-one-out evaluation protocols, is defined as

$$\text{HR@N} = \frac{1}{|\mathcal{U}|} \sum_{u \in \mathcal{U}} \mathbb{I}\left(r_u \in \text{Top-}N(u)\right) \tag{2.1}$$

  where $\mathcal{U}$ denotes the set of all users in the test set, r_u represents the ground-truth items for user u, Top-$N(u)$ represents the set of top-N items recommended to user u, and $\mathbb{I}(\cdot)$ is a function that returns 1 if the ground-truth item is present in the top-N list and 0 otherwise. HR@N measures the proportion of users for whom the system successfully includes ground-truth items

in the top-N recommendations.

- **Reciprocal Rank (RR)**: RR is defined based on the position (rank) of the first ground-truth item in a user's recommendation list:

$$\text{RR} = \frac{1}{\text{rank}} \tag{2.2}$$

where *rank* refers to the position of the first ground-truth item in the ranked list of recommendations for a given user. The Mean Reciprocal Rank (MRR) is the average of these reciprocal ranks across all users:

$$\text{MRR} = \frac{1}{|\mathcal{U}|} \sum_{u=1}^{|\mathcal{U}|} \frac{1}{\text{rank}_u} \tag{2.3}$$

where $\mathcal{U}$ denotes the set of users in the evaluation set, and rank_u is the position of the first ground-truth item for user u. MRR measures how early, on average, ground-truth items appear in the recommendation list [157].

- **Normalised Discounted Cumulative Gain (NDCG)**: NDCG is a ranking-based evaluation metric that assesses the quality of the recommendation list considering both the position and the relevance of the recommended items, and gives higher scores when relevant items appear higher in the ranking [254]. NDCG at N is defined as:

$$\text{NDCG@N} = \frac{\text{DCG@N}}{\text{IDCG@N}}, \quad \text{where} \quad \text{DCG@N} = \sum_{i=1}^{N} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)} \tag{2.4}$$

where rel_i denotes the relevance score of the item at position i in the ranked list, DCG@N is the Discounted Cumulative Gain at position N, and IDCG@N is the Ideal DCG, i.e., the maximum possible DCG@N achievable by a perfect ranking where all relevant items are ranked at the top. NDCG@N ranges from 0 to 1, with higher values indicating better ranking quality.

Efficiency

The efficiency of the methods in this thesis is evaluated using metrics that reflect different dimensions of computational and economic performance, particularly those relevant to large-scale deployment. These metrics offer a comprehensive view of efficiency in terms of hardware utilisation, computational overhead, execution speed, and economic feasibility, factors that are essential considerations for deployment in real-world scenarios [38]. They go beyond accuracy-based evaluation by providing a multi-dimensional perspective on the trade-offs between computational cost, resource usage, responsiveness, and scalability, key determinants for the practical adoption of large-scale real world RSs.

- *Memory Usage:* We measure peak GPU memory usage during training and inference, as this metric indicates whether a model can be executed within the limits of target hardware. Models with high memory demand may require specialised accelerators, limiting their portability and raising deployment costs. In contrast, lower memory usage allows models to run on commodity hardware, such as GPUs, cloud instances with smaller budgets, or even edge devices with restricted resources [40]. Memory efficiency also enables the handling of longer user-item interaction sequences, which is particularly relevant in domains such as e-commerce or personalised health monitoring, where histories can be extensive. From a practical point of view, memory efficient models help reduce infrastructure costs by avoiding frequent hardware upgrades and allow broader adoption between institutions with heterogeneous hardware capabilities [353].

- *Model Runtime:* We measure the model training time (in seconds per epoch) and the effective inference latency (time per query). Runtime is a key indicator of practical performance: shorter training cycles accelerate model development, hyperparameter tuning, and continuous retraining, while lower inference latency improves user experience [233]. In production environments such as streaming platforms, social networks, and e-commerce systems, high latency

can lead to noticeable delays that degrade user experience and reduce engagement [197]. In addition, runtime efficiency determines system throughput, the number of queries served per second. Hence, optimising runtime not only improves research and deployment efficiency but also translates into tangible business and user-centric benefits [352].

- *TFLOPs:* We also evaluate model computational cost in terms of *Tera Floating Point Operations per second (TFLOPs)*:

$$\text{TFLOPs} = \frac{\text{Operations per forward/backward pass} \times \text{Batch Size}}{\text{Execution Time (seconds)} \times 10^{12}} \tag{2.5}$$

This metric provides a hardware-agnostic measure of computational efficiency [247]. Unlike runtime, which depends on software and hardware configuration, TFLOPs indicate the algorithmic efficiency of a model. Models with lower TFLOPs are more sustainable because they consume less energy, leading to a reduced carbon footprint [262]. This becomes increasingly important as AI systems scale, given the increase in the economic and environmental costs associated with training large models [24].

- *API Cost:* For methods that depend on API calls to LLMs, we quantify operational expenditure (OpEx) in USD, calculated as:

$$\text{API Cost} = \text{Cost per Token} \times \text{Number of Tokens Processed} \tag{2.6}$$

This metric determines the financial feasibility of deploying models at scale. In high-traffic scenarios such as livestreaming platforms [317], even modest per-request costs can accumulate into substantial operational expenses. Minimising API cost is therefore a measure of commercial viability, especially in resource-constrained environments [251]. Furthermore, API usage introduces additional latency from network communication and external process-

ing, which can create bottlenecks in practical deployment settings [151]. Thus, API cost captures not only direct monetary implications but also indirect performance trade-offs, making it an essential metric for evaluating the efficiency of RSs in practical deployments [247].

2.1.3 Datasets.

We evaluate the methods introduced in this thesis using four heterogeneous real-world datasets: **Retailrocket, Taobao, IJCAI** and **Tianchi**. These datasets provide rich contextual information that enables a comprehensive evaluation of model performance in practical recommendation settings. Using these publicly available datasets also ensures that our experiments address the complexity and diversity characteristics of real-world scenarios. The variation in the scale of the datasets further makes them valuable benchmarks for assessing models that incorporate heterogeneity and dynamics features in realistic recommendation environments. To improve the reliability of learning signals, we filter out users and items with fewer than five interactions, reducing sparsity. A summary of the statistics of the filtered datasets is shown in Table 2.2.

Retailrocket. This dataset was collected over a period of 4.5 months from *Retailrocket* [2], an e-commerce company that provides personalised and recommendation services for online retailers. The dataset consists of three main components: (i) an events log containing anonymised user–item interactions with event type and timestamp, (ii) item properties describing metadata such as categorical attributes, and (iii) a category tree defining the hierarchical taxonomy of products. The event log records three types of user behaviour: *view, add-to-cart*, and *purchase*, all reflecting the entire decision funnel from browsing to buying. Events are chronologically ordered, supporting sequential and session-based modelling, while the additional item metadata and category tree enable side-information-aware and cold-start recommen-

[2] `https://retailrocket.net/`

dation. The Retailrocket dataset is publicly available at the following link[3].

Taobao. This dataset was collected from Taobao[4], one of the largest e-commerce marketplaces in China. It contains more than 100 million user interaction logs that span a 4-month period (from November 2017 to March 2018), covering a diverse population of consumers. Each log entry includes an anonymised user identifier, an item identifier, the type of interaction, and a timestamp. Four different types of user-item interactions are recorded in the dataset: *view*, *add-to-cart*, *add-to-favourites*, and *purchase*, which together reflect the entire pipeline from initial exploration to final "purchase" transaction. These heterogeneous users-item interactions make the Taobao dataset a realistic benchmark for assessing the scalability of modern recommendation models. For example, *view* shows exploratory intent, *add-to-favourites* indicate preference retention, and *add-to-cart* reflects a consideration to purchase an item. The Taobao dataset is publicly available at the following link[5].

IJCAI. The IJCAI dataset was released for the *24th International Joint Conference on Artificial Intelligence (IJCAI'15)* Repeat Buyers Prediction Competition[6]. IJCAI contains user–item interaction logs covering four types of user-item interaction: *view, add-to-favourite, add-to-cart, purchase*. Each record includes anonymised user and item IDs, timestamps, and contextual metadata such as category and shop identifiers, enabling detailed modelling of temporal usage patterns. The goal of the *IJCAI'15* competition was to apply advanced machine learning techniques to predict which shoppers would become repeat buyers after sales promotion, a task that is of practical importance for e-commerce platforms that aim to retain high user transaction volumes. The IJCAI dataset is widely used in recommendation research to evaluate large-scale RS models [223], and is publicly available at the following link[7].

[3] `https://www.kaggle.com/datasets/retailrocket/ecommerce-dataset`

[4] `https://www.taobao.com/`

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

[6] `https://ijcai-15.org/repeat-buyers-prediction-competition/`

[7] `https://drive.google.com/file/d/1OFT_5Xp_az-GSHIl7QEPB9zhulbooLzE/view?usp=sharing`

Tianchi. The Tianchi dataset, released by Alibaba[8], a global e-commerce platform that connects businesses with suppliers and manufacturers. Tianchi contains user transaction logs that record four types of interaction: *view, add-to-favourite, add-to-cart*, and *purchase*. Each interaction is linked to anonymised user and item IDs, enabling the construction of behaviour sequences for individual users. Tianchi includes metadata related to items and users, such as category and shop identifiers, providing additional context to better predict purchase behaviour, a key business metric in e-commerce RSs [70]. The dataset differentiates between various types of user feedback, reflecting distinct stages of the decision-making process: *views* indicate exploration, favourites indicate long-term interest, *add-to-cart* actions signal an intent to purchase. Jointly modelling these interactions provides a richer representation of user intent than relying sorely on purchase data [163]. The Tianchi dataset is publicly available[9].

Dataset Properties

The properties of a dataset, such as density and average sequence length, and several other factors determine the complexity of RS datasets, and help to evaluate the robustness of the performance of recommendation models. For example, high-density data is less prone to cold-start problems [260]. In contrast, low-density datasets make it difficult for algorithms to find reliable similarities between users or items, often leading to less effective recommendations [271]. *Average sequence length* reflects the depth of user interactions over time; short sequences may lack sufficient context for accurate preference inference, while very long sequences can introduce modelling complexity and dilution of relevant signals [191, 301].

Average Sequence Length. We define the *average sequence length* as the number of interactions per user, representing the amount of behavioural context available to model user preferences:

[8] `https://www.alibaba.com/`
[9] `https://tianchi.aliyun.com/competition/entrance/231576/information`

$$\text{Average Sequence Length} = \frac{\#\text{Interactions}}{\#\text{Users}} \tag{2.7}$$

Long sequences generally provide a rich context, allowing for more expressive user representations [191]. However, excessively long sequences can lead to *oversmoothing* and *oversquashing* in graph-based architectures [44, 330], while very short sequences may restrict the model's ability to capture evolving user preferences [45].

Dataset Density. We define the *density* of a dataset as the proportion of observed user-item interactions relative to all possible user-item pairs:

$$\text{Density} = \frac{\#\text{Interactions}}{\#\text{Users} \times \#\text{Items}} \tag{2.8}$$

Sparsity. We define *sparsity* as the proportion of unobserved user-item interactions:

$$\text{Sparsity} = 1 - \frac{\#\text{Interactions}}{\#\text{Users} \times \#\text{Items}} \tag{2.9}$$

High density indicates that users interact with a large proportion of available items, providing richer behavioural data for training. Conversely, low density implies higher sparsity, a common property in large-scale RSs where most user-item interactions remain unobserved. Evaluating models across datasets with varying densities reveals how effectively they handle data scarcity and incomplete feedback. As shown in Table 2.2, datasets that have a density below 2×10^{-4} (*Retailrocket* and *IJCAI*) are classified as **sparse**, while those above 3×10^{-4} (*Tianchi* and *Taobao*) are **dense**. User behaviour is categorised as **short**, **medium**, or **long** sequences, reflecting the context available for modelling preferences.

In addition to sequence length and density, other factors contribute to the complexity of RS datasets. **Item Popularity curve (IPS)** refers to bias in the RS towards popular items, leading to a "rich-get-richer" effect, where items with more interactions continue to receive more, potentially neglecting niche items [17]. **Time-**

Drift (TD) refers to changes in user preferences and item relevance over time, which requires models that can adapt to these changes to maintain the accuracy of the recommendation [90]. **Side Information Availability (SIA)** refers to additional data, such as item attributes or user demographics, that enrich the recommendation process, especially in sparse datasets, by providing alternative sources of information [236].

As the size of the dataset increases, factors such as item IPS and TD become more pronounced, leading to degradation of model performance. For example, a study by Arabzadeh et al. [6] explores the impact of downsampling datasets on RSs. They found that reducing the size of the dataset can reduce computational costs. However, this can also lead to a loss in the quality of the recommendation, particularly for algorithms sensitive to the size of the data. Furthermore, Roy et al. [206] highlighted that scalability problems arise when the RS input data increases, which poses challenges in maintaining performance and efficiency. Therefore, we consider these factors in the evaluation of the models introduced in this thesis, as they are essential for the development of effective and sustainable RSs.

Statistics / Dataset	**Retailrocket**	**Taobao**	**IJCAI**	**Tianchi**
# Users	11,649	147,894	423,423	25,000
# Items	36,223	99,037	874,328	500,900
# Interactions	87,822	7,658,926	36,222,123	4,619,389
Avg. Sequence Length	7.54	51.78	85.54	184.78
Density	2×10^{-4}	5×10^{-4}	1×10^{-4}	4×10^{-4}
Behaviour Types	[view, cart, buy]	[view, fav, cart, buy]	[view, fav, cart, buy]	[view, fav, cart, buy]
Category	Sparse-Short	Dense-Medium	Sparse-Long	Dense-Long

Table 2.2: Statistical information of the datasets after filtering out users with fewer than five interactions, including qualitative categories reflecting sparsity level and typical sequence length.

Dataset Summary. Table 2.2 summarises the statistics of the four benchmark datasets. *IJCAI* is the largest, with more than 423K users and 36M interactions, while *Retailrocket* is the smallest, with approximately 12K users and fewer than 90K interactions. *Taobao* and *Tianchi* represent medium- to large-scale e-commerce scenarios, collectively providing a diverse range of sparsity levels, sequence length, and other factors such as TD and IPS. Variations in **density** and **average sequence**

length offer different levels of interaction richness, allowing a rigorous evaluation of the robustness of a model under various conditions. Higher density facilitates more reliable preference estimation, while longer sequences provide richer temporal information, but may also introduce noise or oversmoothing in graph-based encoders [93]. In general, these datasets provide a comprehensive and challenging benchmark suite that allows a robust evaluation of the models introduced in this thesis.

2.2 Graph Neural Networks

Recent studies have applied GNNs to model heterogeneous data in RSs [62, 267, 268]. *Graph representation learning* involves encoding each node into a low-dimensional vector that captures both its features and its structural context [5]. Message-passing allows nodes to exchange information by propagating and transforming features across edges, often incorporating edge types and semantics in heterogeneous graphs [68]. Neighbourhood aggregation then combines these messages, using functions such as mean, sum or attention to iteratively update node representations [252].

2.2.1 GNN Preliminaries

Definition: A graph is defined as $G = (V, E)$ where: V is the set of nodes (for example, users and items) and E is the set of edges that connect these nodes. Each node v_i has a feature vector $\mathbf{x}_i$, and each edge (v_i, v_j) has a feature vector $\mathbf{e}_{ij}$. The goal of a GNN is to aggregate information from neighbouring nodes to update each node's representation. For a node v_i, the representation $\mathbf{h}_i^{(k)}$ after the k-th layer of the GNN is given by:

$$\mathbf{h}_i^{(k)} = \text{AGGREGATE}^{(k)}\left(\{\mathbf{h}_j^{(k-1)} \mid \forall j \in \mathcal{N}(i)\}\right) \tag{2.10}$$

where: $\mathbf{h}_i^{(k)}$ is the representation of node i in the k -th layer, $\mathcal{N}(i)$ is the set of neighbours of node i, and AGGREGATE is an aggregation function such as mean, sum or maximum. After aggregation, the node representation is updated using a

neural network:

$$\mathbf{h}_i^{(k)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_i^{(k-1)}, \text{AGGREGATE}^{(k)}\left(\{\mathbf{h}_j^{(k-1)} \mid \forall j \in \mathcal{N}(i)\}\right)\right) \tag{2.11}$$

where: $\text{UPDATE}^{(k)}$ is a neural network that updates the node's representation.

Message passing. Most GNN architectures use the message-passing paradigm [68] during the aggregation process. In message-passing, each node generates a message based on its state and its neighbour's state. For example, at node v_i, the message m_{ij} sent to its neighbour v_j is:

$$m_{ij}^{(k)} = \phi\left(\mathbf{h}_i^{(k-1)}, \mathbf{h}_j^{(k-1)}, \mathbf{e}_{ij}\right) \tag{2.12}$$

where ϕ is a function that generates messages based on node representations and edge features.

Message Aggregation. Messages from all neighbours are then aggregated as follows:

$$\mathbf{m}_i^{(k)} = \text{AGGREGATE}^{(k)}\left(\{m_{ij}^{(k)} \mid \forall j \in \mathcal{N}(i)\}\right) \tag{2.13}$$

Node Update. The node representation is updated by combining its current state with the aggregated messages:

$$\mathbf{h}_i^{(k)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_i^{(k-1)}, \mathbf{m}_i^{(k)}\right) \tag{2.14}$$

Graph Neural Network Architectures. GNN methods use various message-passing architectures to learn meaningful graph-level and node representations. In this thesis, *graph convolution* and *graph attention* are commonly applied.

- **Graph Convolutional Network (GCN):** GCNs extend the concept of con-

volution from Euclidean data to graph structures. They use normalised adjacency matrices to aggregate and propagate information, ensuring smooth feature transitions across neighbourhoods. GCNs are used mainly in undirected graphs and are motivated by the principles of graph signal processing [263].

- **Graph Attention Network (GAT):** GATs introduce attention mechanisms to dynamically weigh the importance of neighbouring nodes during aggregation. Attention coefficients are calculated on the basis of node similarity, enabling GATs to adaptively focus on the most relevant neighbours [238].

2.2.2 Hypergraph Neural Networks

Hypergraph Representation. A hypergraph is defined as $H = (V, E)$ where: V is the set of nodes and E is the set of hyperedges, where each hyperedge is a subset of V that includes more than two nodes.

Hypergraph Neural Networks. HGNNs have been proposed to address the limitations of GNNs when modelling complex and heterogeneous relationships that go beyond pairwise [5]. In HGNNs, the nodes and hyperedges are represented in low-dimensional space to learn latent representations [138]. Unlike GNNs, where an edge links a user to a single item, a hyperedge simultaneously connects a user with multiple items. For example, a hyperedge may represent a user interacting with a group of items within a session, capturing collective preferences and co-consumption patterns. This enables the model to learn richer and more nuanced representations of user behaviour compared to standard bipartite user-item graphs [147]. HGNNs have been used to model interactions for several tasks, such as link prediction [181], computer vision [141], life sciences [124, 265, 346] and recommendation [142, 255]. A summary of the benefits of using HGNNs is given below.

- **Higher-order Relationships:** Hypergraphs model interactions among more than two nodes, capturing complex relationships more effectively [320].

- **Flexibility:** Hyperedges in hypergraphs allow for more flexible and expressive representations compared to traditional edges in graphs [5].
- **Rich Contextual Information:** Hypergraphs incorporate richer contextual information by connecting multiple nodes through hyperedges, enhancing the ability to capture intricate patterns [255].

Hypergraph Convolutional Networks (HGCNs). HGCNs extend the GCN to hypergraphs, which allow a single hyperedge to connect multiple nodes, enabling the modelling of complex higher-order interactions. HGCNs introduce hypergraph-specific operations, such as aggregating node features within each hyperedge and propagating hyperedge-level information to nodes, which makes them suitable for tasks such as multi-user collaboration, product co-purchases, or biological networks where relationships are group-based. For example, Bai et al. [8] propose hypergraph convolution to improve *representation learning* for classification tasks. Although GCN is computationally simpler and more effective for pairwise relationships, HGCN is more expressive and powerful for capturing higher-order dependencies [8], but at the cost of increased computational complexity [122]. The hypergraph convolution operation aggregates information between hyperedges [261].

Hypergraph Attention Networks (HGATs). HGATs extend the GAT mechanism to hypergraphs, which model complex higher-order relationships that involve multiple nodes connected by hyperedges. For example, Chen et al. [20] introduce HGATs to embed high-order data relations in the identification of complex data relationships for object recognition tasks. The HyperGAT model [46] follows the same approach with the objective of improving expressive power to learn text representation. In the same line of research, Kim et al. [118] propose HGATs to resolve the issue of information disparity between different modalities in multi-modal learning tasks. Ding et al. [45] proposed the HyperFormer model to improve representation modelling on feature-sparse data. Using hypergraph instances where nodes represent data instances and hyperedges represent feature values, their model effectively

learns feature representations between different instances, as well as correlations among features. GATs are effective for graphs with pairwise dependencies, while HGATs are more expressive, capturing richer dynamics and higher-order interactions, but with increased computational complexity [104].

2.2.3 Non-Message-Passing Graph Neural Networks

Recent research has applied MLP-based linear layers as an alternative to the message-passing scheme in GNN modelling [125]. These works aim to address issues such as oversmoothing and oversquashing [228]. The use of high-order tensors has also been studied to model HGNNs for tasks such as node classification and link prediction [121]. For example, Maron et al. [177] propose linear operators between tensors of arbitrary order and show that their method is more expressive than several message-passing GNN baselines. Kim et al. [119] model GNNs as independent tokens and embedded as input to transformers, proving that this approach is more expressive than message-passing. In similar studies, the authors of [120] also propose a framework that has better learning performance than message-passing GNN baselines in classification and matching tasks [121].

2.2.4 Hyperbolic Hypergraph Representation Learning

Recently, *representation learning* in hyperbolic spaces has received attention. For example, Nickel et al. [187] show that hyperbolic embeddings outperform Euclidean embeddings in terms of both representation capacity and generalisation ability. In related research [186], they show that the Lorentz model of hyperbolic geometry embedding is effective. Ganea et al. [61] extend embeddings to directed acyclic graphs using hyperbolic entailment cones, introducing hyperbolic neural networks that define core neural network operations in hyperbolic space, such as Mübius addition, Mübius scalar multiplication, exponential and logarithmic maps. Hyperbolic analogues of other algorithms have also been proposed, such as hyperbolic graph attention networks [332] and hyperbolic attention networks [73].

HGNNs generalise traditional graphs by allowing hyperedges to connect multiple nodes simultaneously, thereby capturing higher-order relationships prevalent in domains such as knowledge graphs [152], social networks [342] and biological systems [139]. Embedding hypergraphs in hyperbolic space offers advantages such as exponential volume growth, which is suitable for modelling hierarchical and scale-free structures [192]. The integration of hyperbolic geometry into HGNNs enables effective modelling of complex data by leveraging the geometric properties of hyperbolic space [146], improving the representation of higher-order relationships and dynamic interactions, leading to improved model performance in tasks such as recommendation [319]. This section discusses the mathematical foundations of hyperbolic geometry and hyperbolic HGNNs.

Mathematical Foundations for Hyperbolic Geometry

The n-dimensional hyperbolic space, denoted $\mathbb{H}^n$, is represented using the **Lorentz model** [28]. This model defines $\mathbb{H}^n$ as a subset of the ambient space $\mathbb{R}^{n+1}$:

$$\mathbb{H}^n = \left\{ \mathbf{x} = (x_0, x_1, \ldots, x_n) \in \mathbb{R}^{n+1} : \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1, \quad x_0 > 0 \right\} \tag{2.15}$$

Here, $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ denotes the **Lorentzian inner product** [12], defined for any two vectors $\mathbf{x} = (x_0, x_1, \ldots, x_n)$ and $\mathbf{y} = (y_0, y_1, \ldots, y_n) \in \mathbb{R}^{n+1}$ as

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i, \tag{2.16}$$

where x_0, y_0 are the time-like components, and x_i, y_i for $i = 1, \ldots, n$ are space-like components. The condition $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = -1$ restricts $\mathbf{x}$ to lie on the two-sheeted hyperboloid, while $x_0 > 0$ selects the upper sheet to ensure the manifold is connected and time-orientated. This construction induces a Riemannian manifold [28] with constant negative curvature, fundamental to hyperbolic geometry-based learning.

Exponential and Logarithmic Maps

To enable gradient-based learning in hyperbolic space, the manifold $\mathbb{H}^n$ is mapped to its tangent space $T_{\mathbf{x}}\mathbb{H}^n$ at a point $\mathbf{x} \in \mathbb{H}^n$. This is achieved via exponential and logarithmic maps, which allow computations in the local Euclidean geometry of the tangent space while preserving the global hyperbolic structure [61].

Exponential Map: Given a tangent vector $\mathbf{v} \in T_{\mathbf{x}}\mathbb{H}^n$, the exponential map $\exp_{\mathbf{x}}(\mathbf{v})$ projects $\mathbf{v}$ from the tangent space back onto the hyperbolic manifold:

$$\exp_{\mathbf{x}}(\mathbf{v}) = \cosh(\|\mathbf{v}\|_{\mathcal{L}})\mathbf{x} + \sinh(\|\mathbf{v}\|_{\mathcal{L}})\frac{\mathbf{v}}{\|\mathbf{v}\|_{\mathcal{L}}} \tag{2.17}$$

Here, $\|\mathbf{v}\|_{\mathcal{L}} = \sqrt{\langle \mathbf{v}, \mathbf{v} \rangle_{\mathcal{L}}}$ is the Lorentzian norm of the tangent vector $\mathbf{v}$, and $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ denotes the Lorentzian inner product. This mapping moves along the geodesic defined by $\mathbf{v}$ starting at $\mathbf{x}$.

Logarithmic Map: Given two points $\mathbf{x}, \mathbf{y} \in \mathbb{H}^n$, the logarithmic map $\log_{\mathbf{x}}(\mathbf{y})$ projects $\mathbf{y}$ onto the tangent space at $\mathbf{x}$:

$$\log_{\mathbf{x}}(\mathbf{y}) = \frac{\operatorname{arccosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}})}{\sqrt{\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}^2 - 1}} \left(\mathbf{y} + \langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} \mathbf{x}\right) \tag{2.18}$$

In this expression, $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}$ is the Lorentzian inner product between $\mathbf{x}$ and $\mathbf{y}$, and $\operatorname{arccosh}(\cdot)$ is the inverse hyperbolic cosine function. This operation yields a tangent vector in $\mathbf{x}$ pointing in the direction of $\mathbf{y}$, scaled by the hyperbolic distance between the two points. These maps allow computations in the tangent space, while respecting the intrinsic negative curvature of the manifold [214].

Hyperbolic Hypergraph Neural Networks (H2GNNs)

H2GNNs extend hypergraph learning into hyperbolic space, which have been applied to model hierarchical structures [148]. Unlike traditional HGNNs that operate in flat Euclidean geometry, H2GNNs leverage the negative curvature of hyperbolic space

to more effectively capture high-order dependencies and complex relational patterns such as those encountered in heterogeneous recommendation scenarios [146, 319].

Aggregation via Lorentzian Centroid: In hyperbolic space, direct averaging of vectors as in Euclidean space is not meaningful due to curvature [61]. To aggregate node or hyperedge features, H2GNNs compute the Lorentzian centroid, a generalisation of the Euclidean weighted mean. Given a set of embeddings $\mathbf{x}_i \in \mathbb{H}^n$, the centroid $\mathbf{c} \in \mathbb{H}^n$ is defined as the point that minimises the weighted sum of the squared Lorentzian distances:

$$\mathbf{c} = \arg\min_{\mathbf{z}\in\mathbb{H}^n} \sum_i w_i \, d_{\mathcal{L}}^2(\mathbf{z}, \mathbf{x}_i) \tag{2.19}$$

Here, $\mathbf{x}_i$ are embeddings in the Lorentz model of $\mathbb{H}^n \subset \mathbb{R}^{n+1}$, $w_i > 0$ are scalar weights, and $d_{\mathcal{L}}(\mathbf{z}, \mathbf{x}_i)$ is the Lorentzian distance, defined by:

$$d_{\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \operatorname{arccosh}(-\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}) \tag{2.20}$$

The Lorentzian inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}}$, used to define the distance, is computed as:

$$\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_0 y_0 + \sum_{i=1}^{n} x_i y_i \tag{2.21}$$

where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{n+1}$, and x_0 is the time coordinate. This inner product governs the geometry of the hyperboloid model and ensures distances that maintain the manifold curvature. Since the Lorentzian centroid does not have a closed form solution, it is computed using Riemannian Stochastic Gradient Descent (RSGD), which iteratively updates the centroid by following the Riemannian gradient and projecting updates back onto the manifold using the exponential map. This ensures that the computed centroid remains within $\mathbb{H}^n$ and preserves the geometric structure of the embeddings.

Model Training Using Riemannian Stochastic Gradient Descent

Riemannian Stochastic Gradient Descent (RSGD) [102] is a generalisation of the Stochastic Gradient Descent (SGD) algorithm [4] to Riemannian manifolds [61]. In standard Euclidean space, SGD iteratively updates the parameters by moving in the direction of the negative gradient of the loss function. However, when parameters lie on a curved non-Euclidean space, known as a Riemannian manifold, naively applying Euclidean updates can result in invalid parameter values that no longer reside on the manifold [185]. A *Riemannian manifold* is a smooth and differentiable space equipped with a smoothly varying inner product in its tangent spaces, which allows notions of distance, angles and gradients [28]. In the context of hyperbolic learning, the manifold $\mathbb{H}^n$ is a Riemannian manifold with constant negative curvature. To ensure that the parameters remain on the manifold during training, RSGD computes the *Riemannian gradient*, which is a projection of the Euclidean gradient onto the tangent space of the manifold at the current point. RSGD generalises SGD to Riemannian manifolds. For a parameter $\theta \in \mathbb{H}^n$, the update step is as follows:

$$\theta_{t+1} = \exp_{\theta_t}\left(-\eta_t \cdot \operatorname{grad} f(\theta_t)\right) \tag{2.22}$$

where η_t is the learning rate, $\operatorname{grad} f(\theta_t)$ is the Riemannian gradient, and $\exp_{\theta_t}$ ensures that the updated parameter remains in the manifold. In this way, the model adapts to the geometry of the embedding space to reflect the structure of the data, which is useful when modelling data that has various levels of hierarchy or density [73].

2.2.5 Hypergraph Representation Learning in Recommender Systems

Graph representation learning has been applied to model heterogeneous scenarios [62, 134, 267] and has shown promising results in various RS and information retrieval tasks [229, 261]. However, despite the performance improvement by methods such as graph transformers [275, 276], the high computational cost of these methods

remains a challenge in practical settings. In addressing the limitations of GNNs, *hypergraph representation learning* has been studied to model higher-order user-item interactions [56, 310]. These include self-supervised methods that aim to alleviate noise problems [245, 279] address oversmoothing and skewed data distribution problems in GNNs [278], and learn representations for downstream tasks such as classification and recommendation [65].

2.3 Transformers

The transformer architecture has been applied in applications such as *Natural Language Processing* [76, 130] and *Computer Vision* [160] and RS [54, 133, 172] tasks to improve *representation learning* on complex data. The *self-attention* mechanism [237] weighs the importance of different tokens using *queries*, *keys* and *values* derived from input embeddings, allowing the model to focus on relevant information [47, 84].

Traditional approaches such as recurrent neural networks [91] and convolutional neural networks [25] are limited to local patterns, which restricts their ability to model complex and evolving user preferences [310]. Transformers, on the contrary, provide an expressive and scalable solution by jointly modelling short- and long-term dependencies in user behaviour, capturing contextual information from various signals [133]. Consequently, transformer-based recommender systems such as SASRec [114] and BERT4Rec [224] have achieved state-of-the-art performance, establishing them as strong and widely adopted baselines in recent RS research [223, 307, 321].

2.3.1 Transformer Preliminaries

Transformers are a class of deep learning models designed to handle sequential data using a mechanism known as *self-attention* [169]. Unlike recurrent neural networks (RNNs), which process sequences step by step, transformers operate on the entire input sequence in parallel. This parallelism allows the model to capture long-range

dependencies more effectively [237].

Token Embedding. Given an input sequence of n tokens, each token is mapped to a dense vector representation using an embedding layer. This embedding process transforms discrete tokens into continuous vector spaces, allowing them to be processed by the model. Let $\mathbf{E} \in \mathbb{R}^{d \times n}$ denote the embedding matrix, where d is the embedding dimension and n is the sequence length. The matrix $\mathbf{E}$ is made up of token embedding vectors:

$$\mathbf{E} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n] \tag{2.23}$$

where $\mathbf{e}_i \in \mathbb{R}^d$ represents the embedding of the i th token in the input sequence.

Positional Encoding. The transformer architecture applies positional encoding to capture the order of tokens [117]. Positional encoding injects information on the position of each token in the input sequence, allowing the model to understand their order. Let $\mathbf{P} \in \mathbb{R}^{d \times n}$ be the matrix of positional encodings, where each column $\mathbf{p}_i \in \mathbb{R}^d$ corresponds to the position of the token i. The position-aware input to the transformer is then given by:

$$\mathbf{E}' = \mathbf{E} + \mathbf{P} \tag{2.24}$$

The resulting matrix $\mathbf{E}' \in \mathbb{R}^{d \times n}$ integrates positional information, allowing the model to differentiate between identical tokens that appear at different positions.

Self-Attention Mechanism. The core of the transformer architecture is the *self-attention* mechanism, which enables the model to focus on different parts of the input sequence when encoding a particular token. This is achieved by computing three distinct projections of the input embeddings: *queries*, *keys* and *values*. Given the input $\mathbf{E}' \in \mathbb{R}^{n \times d}$, the *query, key* and *value* matrices are computed as follows:

$$\mathbf{Q} = \mathbf{E}'\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{E}'\mathbf{W}_K, \quad \mathbf{V} = \mathbf{E}'\mathbf{W}_V \tag{2.25}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_k}$ are learnable weight matrices, and $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d_k}$ are the resulting query, key and value matrices. The attention scores between all token pairs are calculated by taking the scaled dot product of queries and keys:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{2.26}$$

This operation produces a weighted sum of the value vectors, where each weight reflects the relevance of one token to another. The scaling factor $\sqrt{d_k}$ stabilises the gradients.

Multi-Head Attention. Instead of computing a single attention function, the transformer uses multiple attention heads to allow the model to jointly attend to information from different representation subspaces. Each head performs its own self-attention operation with independently learned projections:

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \tag{2.27}$$

where $\mathbf{Q}_i = \mathbf{E}'\mathbf{W}_Q^{(i)}$, $\mathbf{K}_i = \mathbf{E}'\mathbf{W}_K^{(i)}$, and $\mathbf{V}_i = \mathbf{E}'\mathbf{W}_V^{(i)}$, for $i = 1, 2, \ldots, h$, with h being the number of attention heads. The outputs of all attention heads are concatenated and projected through a linear output transformation.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O \tag{2.28}$$

where $\mathbf{W}_O \in \mathbb{R}^{hd_k \times d}$ is a learnable output projection matrix. This mechanism enables the model to capture richer patterns and dependencies within the input sequence by considering multiple perspectives simultaneously.

2.3.2 Towards efficient self-attention

One issue widely reported with transformers is the high computational cost (memory and time), which is a bottleneck to scalability in real-world scenarios [7, 116]. The self-attention mechanism of the transformer requires the storage of attention scores for all pairs of nodes, leading to a high memory usage. As a result, the self-attention operation exhibits quadratic computational complexity with respect to the input length [233]. Efficient attention methods focus on a subset of connections or use low-rank approximations to reduce memory requirements [347]. Efficient transformer variants are more effectively parallelised, leading to faster training times. This is particularly advantageous in scenarios where efficient training on large datasets is crucial, such as in recommendation tasks with a large number of user-item interactions [232]. Several transformer variants have been proposed to address the computational inefficiencies of the original transformer architecture, particularly in managing long sequences. Table 2.3 summarises various transformer variants designed to improve computational efficiency.

Recently, State-Space Models have been explored as an efficient method for processing large sequences, with adaptations to graph-based tasks yielding impressive results [154, 165]. However, this thesis focuses on transformer-based architectures, due to their broad empirical validation across various recommendation benchmarks, which makes them a more robust foundation for this research.

2.3.3 Complexity Analysis

To evaluate the efficiency of the models considered in this work, we adopt a consistent framework for analysing computational complexity. We let n represent the number of users or entities, and m the number of items or candidate nodes. The embedding dimension is denoted by d, the neighbourhood size in graph-based or attention-based mechanisms by k, the number of layers in the model by L, and the sequence length by N. All subsequent complexity analyses are expressed using $\mathcal{O}(\cdot)$, which reflects

Model Variant	Computational Complexity
Vanilla Transformer [237]	$O(N^2 \cdot d)$
Sparse Transformer [32]	$O(N \cdot \sqrt{N} \cdot d)$
Longformer [11]	$O(N \cdot d)$
Reformer [123]	$O(N \cdot \log N \cdot d)$
Linformer [249]	$O(N \cdot d^2)$
Performer [34]	$O(N \cdot d \log d)$
Linear Transformer [116]	$O(N \cdot d^2)$
BigBird [323]	$O(N \cdot d)$
Routing Transformer [205]	$O(N \cdot d \log N)$

Table 2.3: Computational complexity of various transformer model variants. The **standard transformer**, with its complete self-attention mechanism, exhibits quadratic complexity $O(N^2 \cdot d)$, where N represents the sequence length and d the dimensionality of the model.

the dominant number of operations required as functions of these parameters.

In line with common practice, we retain the embedding dimension d and the sequence length N, since both contribute significantly to the computational cost in modern architectures. The **standard transformer**, for example, employs a complete self-attention mechanism that requires each token to attend to every other token in the sequence. This results in a quadratic time complexity of $\mathcal{O}(N^2 \cdot d)$, where N is the sequence length and d the dimensionality of the hidden representations. Various transformer variants have been proposed to reduce this cost, either by constraining the attention pattern or approximating the attention matrix, as summarised in Table 2.3, which serves as a reference for discussions of their efficiency trade-offs.

2.3.4 Transformers in Recommender Systems

The growing interest in transformer research has led to their incorporation in models for various tasks, including state-of-the-art recommendation methods [153, 327]. For example, *SASRec*, a self-supervised model, uses transformer-based self-attention for item-item interaction modelling [114] in sequential recommendation. Another sequential recommendation method, *BERT4Rec* [224], uses dynamic user preference modelling based on previous experiences. Despite their potential, these plain

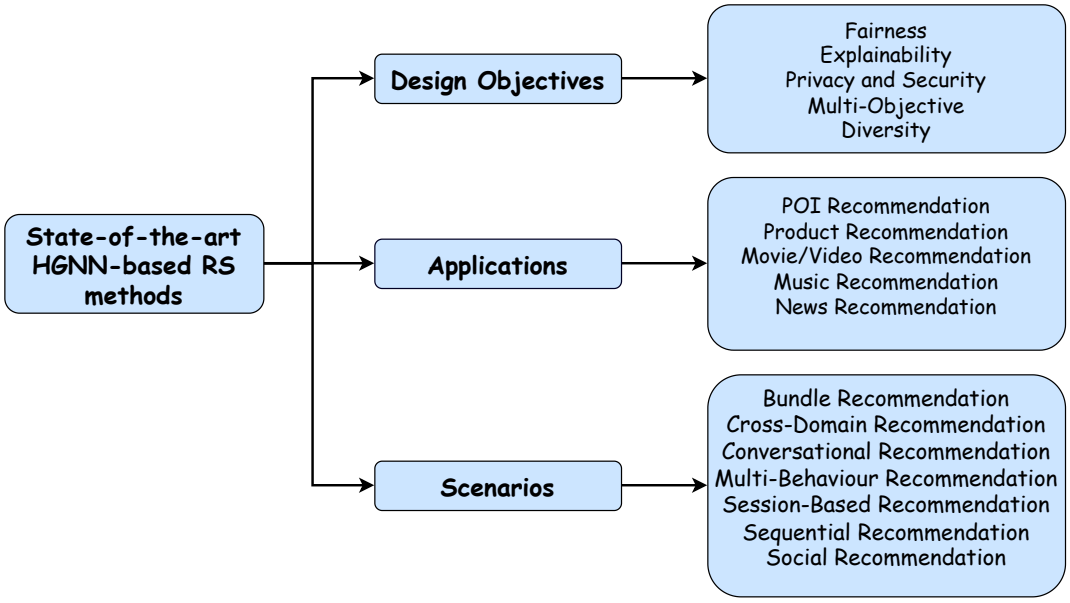

Figure 2.1: Taxonony of HGNN-based RS methods based on applications, design objectives and scenarios.

transformer-based RS methods lag behind state-of-the-art GNN-based methods due to the lack of structural prior [346]. This means that transformers cannot fully exploit the rich connectivity information available in user-item interaction graphs, leading to suboptimal performance compared to GNN-based methods.

2.4 Recommender Systems based on HGNNs

The study of HGNNs with the aim of improving *representation learning* in RSs is an active research area [153, 168, 327]. Taking into account multiple items and users simultaneously, HGNNs have been shown to capture more nuanced preferences and behaviours, leading to better quality recommendations [146, 161]. Other works propose HGNNs to flexibly generate item embeddings from correlated items and then aggregate item representations for downstream recommendation [246], such as encoding and extracting complex user behaviour patterns [210]. HGNNs have also been applied to learn different graph structures by specifying arbitrary weights for the neighbours and inductive learning problems [238]. However, despite their potential, HGNN-based RS methods inherit GNN challenges such as oversmoothing,

oversquashing and scalability issues, especially in large-scale settings [23, 38, 44]. In this section, we provide a literature survey of state-of-the-art HGNN-based RS methods based on applications, design objectives and scenarios, as illustrated in Figure 2.1.

2.4.1 Application Specific HGNN Recommender Systems

To address the GNN methods that are mostly limited to pairwise interactions, HGNN methods have been implemented in various applications [127], citation networks [132, 315], personalised medical recommendation [348], channel recommendation [317] and software code reviewer recommendation [203]. In this section, we provide an overview of the most common applications of HGNN-based RS methods.

Product Recommendation. HGNNs have been widely used in product recommendation, such as recommending products from different categories based on user browsing history, past purchases and interactions with other users or products, cross-selling, up-selling and complementary products [207, 236]. An example is the SREC model [303], which applies a gated HGNN for e-commerce scene-aware product recommendation to predict matching scores among users, scene and merchants. Chen et al. [31] propose the Interactive HGNN (IHGNN) model for personalised product search, which applies a hypergraph representation model constructed from historical user-product-query interactions to improve the representation of the target entity. Despite the potential of these methods, scalability challenges remain due to issues such as oversmoothing and oversquashing [271].

Point-of-Interest Recommendation. Point of Interest (POI) recommendation has drawn growing attention with the widespread popularity of location-based social networks [242]. Although traditional RSs usually struggle with the complex relationships between these different types of data, HGNNs naturally model these complex interactions to incorporate contextual information (such as time of day, weather, or user preferences) directly into the recommendation process. High-order

spatial-temporal relations exist between services, which are not adequately captured by existing methods that rely on ordinary graphs and consider spatial-temporal relations between services as pairwise connections [106]. To incorporate spatial-temporal correlations within user-POI interactions and uncover complex high-order collaborative signals between users [129], several methods propose spatial-temporal HGNN methods for the next POI recommendation, which jointly learn representations from local and global views [140, 295]. Despite the improved expressiveness of these models, high computational cost remains a challenge in large-scale settings, due to their limited computational efficiency [66].

Movie and Video Recommendation. HGNNs have been applied in movie and video recommendation due to their ability to learn complex relationships between different types of entity involved in the movie ecosystem, such as connecting multiple users who watched the same set of movies, or connecting movies with multiple actors, genres and directors simultaneously. HGNNs have been shown to capture these multi-faceted interactions better than traditional methods [317]. HGNNs have also been applied to discover new content and recommend videos with similar themes, creators or genres based on past user interactions [141, 261]. However, despite their potential, issues such as limited scalability and high computational cost limit the scalability of these models in practical scenarios [38].

Music Recommendation. HGNNs have been applied to link users to multiple songs, connecting artists to multiple genres and representing playlists that contain various songs and genres. Recent research has shown the potential of HGNN in music recommendation, particularly in scenarios where various data modalities, such as social media interactions and auditory signals, intersect [324]. For example, Luo et al. [166] propose DWHRec, a "**D**iversified, **W**eighted **H**ypergraph Embedding model for Music Recommendation, that uses a weighted hypergraph to model associations between artists, albums, tags, and tracks in music recommendation. The accuracy of the model is evaluated by measuring the alignment between user pref-

erences and recommended tracks, while diversity is assessed based on the variety of track types included in the recommendation list. Although HGNN-based methods such as DWHRec have shown potential in music recommendation, improving their efficiency is an interesting research direction [208].

News Recommendation. HGNNs have been shown to improve news recommendation by effectively modelling higher-order relationships between users, articles, topics, authors and other related entities. Gharahighehi et al. [67] demonstrate that hypergraph learning handles multi-stakeholder recommendation tasks better than traditional methods. This HGNN-based approach also counters popular bias and generates more fair recommendations compared to existing methods. Moreover, balancing this fairness with efficiency objectives would result in more sustainable multi-objective RSs [262], as discussed in the next paragraph.

2.4.2 Objective-Specific HGNN Recommender Systems

Different RSs have different objectives. Although accuracy is the most commonly used metric to evaluate recommendation performance, recent research has also focused on other objectives such as diversity, explainability, fairness, user privacy and security.

Diversity: Diverse recommendations prevent users from seeing repetitive content and introduce them to new and varied items, which could keep them engaged long-term, avoiding echo chambers where users are only exposed to similar content [57]. Diversity helps break this cycle by introducing novel content. However, in practical scenarios, there is often a trade-off between accuracy and diversity in recommendations. For example, in the music domain, a system may achieve high accuracy by suggesting songs within a user's preferred genres (intra-preference diversity), but this can limit exposure to entirely new genres or styles (inter-preference diversity), reducing overall diversity and the opportunity to broaden the user experience [201].

Explainability: Explainable RSs provide transparent reasoning behind recommendations, thereby enhancing user trust and satisfaction. HGNNs achieve this by applying techniques such as node and hyperedge importance analysis, feature-based interpretability, and visualisation-driven reasoning, which collectively enable transparent and interpretable decision-making [101]. In an e-commerce scenario, for example, an HGNN-based model can justify recommending a **wireless headset** by highlighting a user's prior interactions with *smartphones*, *bluetooth speakers*, and *charging accessories*, represented as interconnected nodes within the hypergraph. This transforms recommendations from opaque 'black-box' outputs into human-understandable insights, thus improving transparency, user trust and engagement [30]. Despite the potential of HGNN-based RSs, achieving an effective balance between computational efficiency and interpretability remains an open challenge for future research [333].

Fairness: HGNNs have been applied to determine whether there is bias against certain groups in recommendation [80]. Bias emerges from various sources, including historical data, user-item interactions, or the way hyperedges and nodes are constructed [108]. Addressing bias in recommender systems is increasingly recognised as an important research direction, since biased models can unfairly favour popular items, misrepresent underserved users or providers, and reinforce long-tail invisibility [108]. For example, surveys identify selection, exposure, and popularity biases as major structural challenges in recommendation data and algorithms [253].

Privacy and Security: With growing concerns regarding user data privacy, federated RS models have recently received attention due to their privacy-preserving capabilities [322]. Recent methods have proposed the use of HGNN methods to address privacy and security issues in RSs. For example, Li et al. [150] propose LASER, a "**L**earning and unlearning model for er**as**abl**e** **R**ecommendation" that partitions data into disjoint and balanced shards using hypergraph embeddings. Zheng et al. [340] also propose a model that blocks model poisoning attacks on RS

users, addressing the risk of sensitive information leakage that exists in RS training in a centralised environment using a distributed computing paradigm that aims to improve privacy [341]. Ethical RS research is an interesting research direction that addresses issues such as privacy and security, ensuring that newly developed methods comply with regulatory frameworks [67, 155].

Scalability: Scalability in modern RSs is a concern due to the high computational cost of most existing methods, particularly when handling large-scale datasets and supporting real-time recommendations [314]. Many state-of-the-art RSs, including GNN and HGNN-based models, process millions of users and items while capturing complex, higher-order interactions [174]. In large-scale settings, HGNN implementations incur prohibitive memory and computation costs, such as hyperedge construction and message-passing that scale with the number of nodes and relations [66]. Even optimised GNNs often struggle to maintain low latency and manageable resource usage beyond millions of nodes, making efficiency a bottleneck for real-world deployment [5]. This motivates the focus of this thesis on enhancing the scalability of HGNN-based RSs.

Multi-Objective RS: Recent work has explored RS methods that aim to simultaneously balance multi-objective RS that include accuracy, diversity, explainability and fairness. For example, Liu et al. [155] propose a knowledge graph-based retrieval method to transfer knowledge derived from multiple auxiliary domains. Jendal et al. [101] introduce HG, an HGNN-based recommendation model that captures high-order connections between users, items and opinions while also incorporating an explainability module that explains the generated predictions. Jiang et al. [110] propose HGRank, a **H**yper**g**raph **rank**ing model that takes into account social trust and reliability in the hypergraph weighting process, in addition to improving the accuracy of the model.

These studies highlight that multi-objective RS research is increasingly based on **hypergraph-based representations** to capture higher-order relationships (for

example, fairness, diversity, explainability) while optimising objectives such as *accuracy, diversity, explainability, and fairness*. The multi-objective perspective aligns with our goal of leveraging *hypergraph representation learning* to enhance recommender system performance. However, our work specifically aims to improve the **efficiency** of HGNN-based recommender models while maintaining high accuracy. Most existing studies focus on *what* objectives are optimised, such as accuracy, fairness, or diversity [175], while our research focuses on *how* to achieve these objectives **at lower computational cost**. We address the often-overlooked trade-off between model accuracy and computational efficiency by proposing HGNN-based methods that reduce both computation overhead without sacrificing performance.

2.4.3 HGNN Recommender System methods based on Specific Scenarios.

Several studies have explored HGNN in various RS scenarios. State-of-the-art methods in the following categories are presented: session-based, social, conversational, cross-domain, bundle, multi-behaviour and sequential recommendation.

Social Recommendation: HGNNs have been applied to model higher-order relationships in social networks, such as the influence on a user's preference of their social influence and past behaviours [290]. In this way, the model learns to aggregate information and provides personalised recommendations that reflect both the individual preferences of the user and the preferences of their social network [316]. For example, Xu et al. [286] propose RecoGCN, a relation-aware GCN-based model for agent-initiated social e-commerce recommendation, which unifies users, items and sales agents in a heterogeneous graph to capture complex interactions in social e-commerce using metapath-based graph convolution network propagation. In related research, Zhu et al. [351] propose SHGCN, a social recommendation model, which applies HGCNs to learn complex triplet social relations, which allows fine-grained modelling of social relations to better capture user preferences, thus

improving recommendation performance. To address noise and data sparsity issues, Wang et al. [243] propose a Dual-objective **C**ontrastive learning **M**ultiple Hypergraph **C**onvolution model for **S**ocial recommendation (DCMHS). Other methods that propagate messages via the graph attention mechanism take into account the differences in the influences of various friends on the social graph and assign them different weights [63, 266, 342]. Although HGNNs naturally model higher-order user-item connections in social recommendation, their application remains challenging in large-scale settings due to scalability issues and the high computational cost incurred when constructing and processing large-scale models [66].

Session-based Recommendation. Session-based recommendation predicts the next interaction or item choice of a user based on the sequence of actions within a single session, without requiring a long-term user history or profiles [269]. HGNNs have been applied to model high-order relations in session-based recommendation [199]. For example, Xia et al. [281] propose a self-supervised "Dual Channel HGCN" model that applies self-supervised learning to improve the recommendation task by leaning mutual information between intra- and inter-sessions. Ma et al. [173] introduce CLHHN, a "**C**ategory-aware **L**ossless Heterogeneous **H**ypergraph **N**eural **N**etwork model for Session-based Recommendation" that incorporates item category and repeated user clicks to alleviate the data sparsity problem. Wang et al. [248] take into account intra- and inter-session dependencies not only from session information but also from side information, to further improve the accuracy of session-base recommendations using a hypergraph learning framework. Although HGNNs are a promising approach for session-based recommendation, their deployment in large-scale settings remains hampered by scalability issues, for example, high memory/compute demands, long training times, which limit their practical usage [38].

Sequential Recommendation. HGNN methods have been applied to model sequential recommendation tasks, whereby hyperedges connect multiple items in the

sequence, offering a richer understanding of the user's behaviour [109]. Recent research efforts apply HGNN to capture higher-order information among items. For example, Chen et al. [25] propose SHCN, a **S**equential **H**ypergraph **C**onvolution **N**etwork model for next-item recommendation. Li et al. [142] also introduce a lightweight HGCN that is applied to refine feature transformations in the hypergraph structure for sequential recommendation. Similarly, Yu et al. [319] propose a unified Hyperbolic Translation-based Sequential Recommendation (HSTR) framework to capture the hierarchical structure in sequential interaction behaviours and model the third-order relationships between users, previously visited and the next item. Despite their potential, applying HGNNs to sequential recommendation remains challenging in practical scenarios involving long user-item interaction sequences. Issues such as oversmoothing [23], where node or embedding representations become indistinguishable, and oversquashing, where distant relational signals are compressed into low-dimensional vectors and consequently lost, tend to dilute useful information and degrade model performance [44].

Bundle/Group Recommendation. Group activities, such as sharing meals with colleagues or watching movies with family members, are common in practical scenarios of human social interactions. Bundle recommendation is common on on-line platforms such as Spotify, Pinterest[10] and Amazon. HGNNs have been applied to model the synergy between items within a bundle, ensuring that the recommended bundles are not just collections of items, but meaningful combinations that appeal to users [328]. Zhao et al. [338] propose DHMAE, a Disentangled **H**ypergraph-**M**asked **A**uto-**E**ncoder model for group recommendation that addresses noise issues in group recommendation. Group recommendation may suffer from data sparsity issues due to sparse group-item interactions [137]. To alleviate this issue, Zhang et al. [327] propose a self-supervised hypergraph learning framework to model complex tuple-wise correlations between users within and outside the groups. Although HGNNs have potential for bundle/group recommendation, their effective deployment at scale

[10]https://www.pinterest.com/

depends on addressing the scalability challenges that arise when constructing and learning large, complex group or bundle representations in large-scale settings [234].

Cross-Domain Recommendation. Cross-Domain Recommendation (CDR) transfer information from the source to the target domain and emphasise mutual utilisation of information from both the source domain and the target domain, and have been applied mainly to address cold-start issues [260]. CDR methods leverage data from multiple domains to improve recommendations. For example, an online retailer that sells books, music and movies can recommend items across these domains. HGNNs have been applied in several CDR methods. For example, H3Trans [291] applies HGNN to extract correlative information from multi-domain user-item feedback to eliminate domain discrepancy in item representations and aggregate user preferences to improve CDR user representations. Han et al. [81] propose an HGCN framework that includes an intra-domain layer and an inter-domain layer in the modelling process. Similarly, Hao et al. [82] introduce MOP, a "**Mo**tif-based **P**rompt learning model for universal Cross-Domain Recommendation" that applies higher-order motif-based embedding to capture domain knowledge, catering to both intra- domain and inter-domain CDR tasks. In related research, Zhang et al. [326] argue that most existing CDR methods share user data across domains, which violates General Data Protection Regulation (GDPR)[11]. They propose FedHCDR, a **Fed**erated **CDR** framework that applies high-pass and low-pass hypergraph filters to decouple domain-exclusive and domain-shared user representations. The application of HGNNs in CDR to handle sparsity and reduce retraining models is gaining traction, such as training an HGNN in a source domain and then fine-tuning for a target domain [309]. However, the focus on reducing computational cost through such HGNN-based transfer learning approaches in RSs remains underexplored [349].

Multi-Behaviour Recommendation. Multi-behaviour recommendation leverages on multiple types of user interactions such as *clicks, views, likes*, *add-to-cart*

[11]https://gdpr-info.eu/.

actions and **purchases** to better predict user preferences [142]. Unlike RSs systems that rely on a single behaviour type (for example, purchases), multi-behaviour models integrate various collaborative signals to provide a more comprehensive understanding of user intent [111]. Despite challenges related to hypergraph construction and computational demands, HGNNs represent a powerful approach to advancing the field of multi-behaviour recommendation, enabling systems to better understand and predict user preferences across a variety of interactions [275]. For example, Yang et al. [297] propose the construction of hyper-metagraphs to model higher-order dependencies for multi-behaviour recommendation. In related research, Li et al. [142] introduce HEML, a **H**ypergraph-Enhanced **M**ulti-interest **L**earning model, which is designed to capture multi-interest and multi-behaviour dependencies for heterogeneous recommendation. Despite their potential, most HGNN-based multi-behaviour recommendation methods remain computationally inefficient in large-scale settings, posing a significant bottleneck for practical deployment [38].

Conversational Recommendation. Conversational Recommender Systems (CRSs) involve user-to-RS interactions where preferences are interactively generated through dialogue and recommended accordingly. HGNNs enable higher-order learning of user-item relationships by unified modelling of various attributes and user intents in the same hypergraph, leading to more nuanced conversational recommendations [64]. For example, Li et al. [137] propose an HGNN-based CRS model for conversational recommendation. Similarly, Zhao et al. [335] propose hierarchical director-actor conversational RS, which trains from weak supervision and a dynamic hypergraph to learn user preferences from high-order relations. Large Language Model (LLM)s have demonstrated their ability in language understanding, reasoning and dialogue [95], and have been applied to CRS [87]. Using LLMs, techniques such as Reinforcement Learning from Human Feedback (RLHF) [230, 284], Chain-of-Thought prompting [334] has been explored for CRS [240]. However, despite their potential, LLM-based RS methods are prone to hallucinations and have a high computational cost, which limits their effectiveness in most real-world RS scenarios [258]. This hap-

pens because LLMs are trained primarily for next-token prediction in broad text corpora, they prioritise generating coherent and contextually plausible responses rather than ensuring factual correctness [331]. In the context of recommendation tasks, this lack of grounding in the actual item catalogue or user–item interaction data means that the model produces recommendations that do not exist, leading to degraded RS model accuracy [258]. At the same time, the computational overhead of LLMs is significantly higher than that of traditional recommendation models [270]. With billions of parameters, they require large GPU memory and incur high inference costs, which presents challenges for practical deployment [212]. Unlike specialised RS methods that use efficient embedding lookups or graph structures, LLMs approach the task as natural language generation, which is slower and resource-intensive. Together, these issues limit the practicality of LLM-based recommenders in real-world scenarios where trustworthiness, scalability, and low latency are essential [193].

2.4.4 Challenges of applying HGNNs to large-scale Recommender Systems

Despite the potential of HGNNs, their application in recommendation methods involves navigating challenges such as graph construction, network design, model optimisation and efficiency [268], which are discussed in this section.

Hypergraph Construction: When constructing a hypergraph, a good hyperedge design should fully consider the density of the hypergraph. A hypergraph that is too dense means increasing the computational demands of embedding propagation for a large number of nodes [145]. To alleviate this high computational cost, methods such as sampling, filtering and pruning have been proposed. However, these also result in model performance degradation in some cases [159]. On the other hand, a too-sparse hypergraph would result in poor embedding propagation due to a limited number of nodes.

Network Design: The propagation layer makes HGNN different from traditional graph learning methods. The choice of the propagation path is essential for modelling high-order similarity in RS. In addition, the propagation is parametric, which assigns different weights to different nodes. The weights refer to the different importance of the items that have historically interacted with each other. The choices of propagation affect the efficiency of the computation [142]. Too-shallow layers pose challenges in modelling the high-order graph structure, while too deep layers would be prone to oversmoothing [330].

2.5 Open Challenges in HGNN-based Recommender Systems

This thesis focuses on addressing open challenges in the HGNN-based recommendation identified in the literature review as summarised in this section. Despite the most recent innovations in different scenarios, objectives and applications, most existing HGNN-based RS methods are still computationally inefficient for large-scale scenarios, and more research is needed for more efficient HGNN-based RSs for practical scenarios, especially in resource-constrained environments [38, 122].

Graph/Sampling partitioning leads to performance losses. Conventional training of HGNNs, when executed in a full batch manner, has been shown to slow down training convergence [300]. Sparsification, sampling and model compression are commonly applied to enable efficient training and inference of GNN [63], especially in large-scale settings [159]. Although these techniques improve efficiency, they often come with drawbacks that negatively impact GNN performance [345]. Sparsification, such as edge or node pruning, results in the loss of information, reducing the model's ability to capture complex interactions [289]. Sampling, though effective for scalability, leads to incomplete representation of node neighbourhoods, limiting the effectiveness of message-passing and resulting in suboptimal *graph rep-*

resentation learning [171]. Similarly, model compression, such as quantisation or pruning, reduces the expressive power of the GNN, affecting their ability to model intricate graph dynamics or long-tail patterns [159]. In **Chapter 3**, addressing **RQ1**, lightweight graph transformer approaches are explored that aim to alleviate these issues without deteriorating the recommendation performance.

Graph Structural Irregularities Pose Efficiency and Scalability Issues. Despite advances in HGNN-based methods and applications, achieving scalable recommendation remains a challenge, particularly in large-scale industrial settings [228]. The irregular structure of the HGNN poses significant challenges in efficiency and scalability [122]. The computational complexity of most state-of-the-art HGNN methods remains a bottleneck for large-scale RS training and inference [63]. Compared to traditional methods, the computational cost of HGNN models is much higher when complex calculations are involved, despite performance gains [5]. Deeper layers of HGNN, needed to capture long-range dependencies, increase complexity, leading to issues such as vanishing gradients and overfitting [289]. In **Chapter 4**, addressing **RQ2.1**, we explore hardware-algorithm co-design as a way to efficiently accelerate large-scale HGNN-based RS models on the GPU architecture.

Limitations of the GNN message-passing scheme. One of the most popularly adopted GNN architectures is the Message Passing Graph Neural Network (MPGNN), which maintains a representation vector for each node and iteratively updates it by collecting information from neighbouring nodes [68]. Despite its relative simplicity and efficiency, it has several weaknesses that severely limit its performance. (1) **Limited expressive power**, referring to the fact that MPGNNs often fail to distinguish between two structurally different nodes with a similar neighbourhood configuration [119]. (2) **Oversmoothing:** Common HGNNs rely on a message-passing architecture that aggregates information. In each layer, every graph node updates its representation by aggregating the neighbouring nodes [330]. With

the oversmoothing problem, GNNs tend to generate embeddings for all graph nodes that are very similar to each other after several layers of message-passing, resulting in node-specific information being lost, hindering the capture of long-term dependencies between nodes in a graph. Consequently, learning on large graphs remains a persistent challenge [23]. (3) **Oversquashing:** Another issue with message-passing GNN (MPGNN) methods is their limitation in capturing global properties of graphs, which means that they cannot effectively learn long-range interactions within the graph due to oversquashing that occurs as a result of multiple message-passing steps [228]. Oversquashing refers to the limitation of a GNN when transferring information between distant nodes [44]. This issue worsens with an increase in the distance between nodes, which hinders the ability of MGPNNs to model complex behaviours that require long-range interactions. As a result, the long-range dependency learning effectiveness of the MPGNN is reduced. To address these issues, and answer **RQ2.2**, we propose an alternative to message-passing for efficient scalable HGNN-based RSs in **Chapters 4 and 5**.

Dynamic recommendation scenarios are underexplored. Modelling dynamic user preferences is one of the open challenges in practical recommendation scenarios [22]. User preferences and item characteristics change over time, which makes it important for RSs to adapt to these dynamics [312]. Most existing methods focus on static graphs that model the topological structure in a fixed way. However, many real-world scenarios exhibit complex dynamics, making static *representation learning* models inadequate for effectively capturing temporal evolutionary information [53]. Consequently, these models often yield suboptimal performance when applied to dynamic scenarios, which poses challenges to static GNNs [161]. Dynamic changes require frequent updates of the hypergraph topology, including the addition or removal of hyperedges and nodes, and the update of feature embeddings [22]. However, efficiently handling these updates in real time is challenging due to the high computational cost associated with modifying and maintaining the hypergraph structure [167]. This complexity increases with the scale of the data, leading

to increased latency and reduced efficiency [288]. The design of scalable algorithms that accommodate such dynamic changes without significant computational overhead is a challenge for HGNN-based RSs. These issues are addressed in **Chapter 6**, thereby answering **RQ3**.

2.5.1 Baseline Methods.

The proposed methods are compared with the state-of-the-art RS methods that are related to the focus of our study. These are classified as transformer-based, GNN-based, and hybrid methods. The choice of baseline methods is based on the core architecture, encoding, and propagation mechanisms as indicated in Table A.1.

- **SASRec:** A sequential recommendation model that uses the vanilla self-attention mechanism to identify relevant items at each time step for the next-item prediction task [114].
- **BERT4Rec**: A sequential recommendation model that applies bidirectional transformer-based self-attention for recommendation [224].
- **MB-GCN:** A GCN-based model which combines behaviour-aware user-to-item embedding propagation as well as item-to-item embedding propagation layers for multi-behaviour recommendation [111].
- **MB-GMN:** Integrates GNNs with meta-networks for multi-behaviour recommendation. Meta-learners model the behaviour heterogeneity and cross-type dependency [280].
- **KMCLR:** Applies a multi-behaviour learning module to extract user information for user embedding enhancement, and uses a knowledge graph to derive knowledge-aware representations for items. [292].
- **MBHT:** A Multi-Behaviour Hypergraph-enhanced Transformer model that employs a multi-scale transformer to capture local user-item dependencies and a hypergraph convolution module to capture global dependencies [307].

- **MB-STR:** A model that integrates a behaviour transformer layer and a sequential pattern generator module for sequential recommendation [321].
- **HMAR:** Applies masked self-attention and self-attention to items across user behaviours, and encodes the historical frequency of each item behaviour in the input sequence for multi-behaviour recommendation [52].
- **PBAT:** A multi-behaviour recommendation model which applies a personalised behaviour pattern generator in the representation layer and extracts dynamic behaviour patterns for sequential learning. A behaviour-aware collaboration extractor facilitates a behaviour-aware attention mechanism to incorporate behavioural and temporal impacts into collaborative transitions [223].
- **RLMRec** integrates *representation learning* with LLM to capture intricate semantic aspects of user behaviours and preferences. Incorporates auxiliary textual signals, employs LLMs for user/item profiling, and aligns the semantic space of LLMs with collaborative signals through cross-view alignment [200].
- **TRSR** proposes prompt text encompassing user preference summary, recent user interactions, and candidate item information in an LLM-based recommendation, which is fine-tuned to generate recommendations [343].
- **KDA** uses an LLM to obtain language knowledge representations of items that are fed into a latent relation discovery module based on a discrete state variational autoencoder. Self-supervised relation discovery tasks and recommendation tasks are jointly optimised [301].
- **TALLRec:** an LLM tuning framework that structures the recommendation data as instructions to adapt to the recommendation [9].

Chapter 3

Towards Lightweight Hypegraph Transformers for Recommendation

Recent research has applied graph transformers for modelling graph-structured data [94, 215]. This is motivated by the success of transformers in various sequence [239] learning and computer vision tasks [183]. As a result, Graph Transformer (GT)s have emerged as a promising architecture for a variety of *graph representation learning* tasks, including RSs [179]. As discussed in **Chapter 1**, GTs leverage the strengths of both GNNs and transformer architectures, offering a framework that captures complex patterns in recommendation scenarios [162]. However, from the literature survey conducted in **Chapter 2**, despite their potential, it remains challenging to scale graph transformers (GT) to large graphs while maintaining competitive accuracy [215]. In addressing this issue, model compression techniques, such as pruning or quantisation, have been applied to optimise GNN/transformer models with the aim of improving model efficiency [259]. Although these techniques improve efficiency, they lag in terms of accuracy in many practical settings [251]. Graph sampling, for example, though effective for scalability, often leads to incomplete representation of node neighbourhoods, limiting the effectiveness of message passing, and leading to suboptimal *representation learning* and recommendation performance [215]. This brings us to the central research question of this chapter, **RQ1**, as

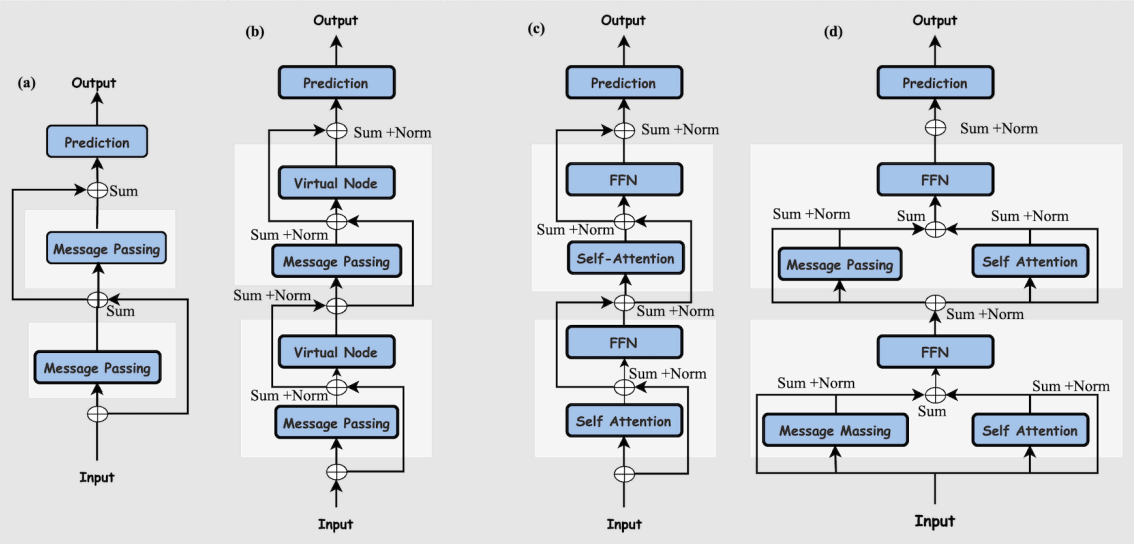

Figure 3.1: An overview of *graph representation learning* architectures, as illustrated in [204]. (a) Graph Neural Networks (GNNs) with Message Passing (MP) are prone to oversmoothing and oversquashing. (b) GNNs augmented with Virtual Nodes (VN) have been proposed to enhance expressivity. (c) Transformer architectures, which utilise Self-Attention (SA) and Feedforward Network(FFN) layers, often underperform in heterogeneous *graph representation learning* due to the lack of structural priors. (d) Hybrid architectures combining MP, SA and FFN layers are computationally expensive in practical scenarios.

outlined in Section 1.3.

> **RQ1:** *Model compression in GNNs, transformers and their hybrid models (GTs) often leads to performance losses. Can we address this issue using lightweight GT-based approaches without degradation of recommendation performance?*

Specifically, the goal is to develop RS methods that do not use excessive computational resources but still achieve high performance. In addressing **RQ1**, two lightweight GT methods, Graph Transformer for Recommendation (GTRec) and Sinkhorn Transformer for Multi-Behaviour Recommendation (STMBR), are proposed to address these highlighted issues of lightweight GT methods.

3.1 GTRec: A Lightweight Graph Transformer for Multi-Behaviour Recommendation

GTs have attracted significant interest in *representation learning* models because of their ability to capture long-range node communication explicitly. This capability allows them to mitigate several limitations associated with message passing GNNs, such as oversmoothing [190], oversquashing [44] and limited expressivity [120, 176]. As shown in Figure 3.1, several approaches have been applied to address these limitations of message passing GNNs. Among them, hybrid GT architectures have shown better performance compared to standalone graph or transformer architectures [215]. However, despite performance gains, most hybrid GT architectures have been shown to be computationally expensive, limiting their practicality for large-scale RSs [352]. One major challenge with GTs is their poor scalability, as the standard self-attention mechanism incurs quadratic time and memory complexity with respect to the number of nodes in the graph [233]. Although this cost is often acceptable for smaller graphs, it is expensive for larger graphs, where GT models often do not fit in memory or require a much smaller batch size, resulting in slower training or the need for high-end GPUs [71]. This problem is further exacerbated in large-scale recommendation settings where multi-hop propagation is necessary, leading to oversmoothing and oversquashing as signals become diluted or compressed over long paths.

To address inefficiencies in traditional transformer architectures, various efficient designs have been proposed. For example, the Performer model [34] reduces computational complexity through kernel-based approximations, while Flash Attention [41] achieves higher efficiency through enhanced parallelism and partitioning strategies. Furthermore, GraphGPS [198], a modular framework that integrates message-passing networks with attention mechanisms and incorporates various structural and positional encodings, has demonstrated state-of-the-art performance in numerous datasets. In this modular framework, the best performance with GTs is usually

achieved by integrating a message-passing network with the full transformer architecture. In contrast, sparse transformer methods tend to underperform [215]. This trend is consistent across various GT models, where attempts to reduce model size - either by pruning, compression, or approximation - leads to significant performance degradation compared to the original models [159]. Given these observations, the key question driving this section is: *Can lightweight GT models achieve superior performance in HGNN-based RSs without compromising efficiency?*

One of the advantages of the modular approach in GTs is the ability to separate local and global attention through distinct roles assigned to the graph and transformer modules, respectively [198]. The motivation behind this design approach is to reduce the average message-passing distance in the GNN module, that is, message-passing is focused mainly on the direct graph connections. This module separation also enables more flexible tuning on both modules in order to determine the pareto-optimal efficiency and performance objectives. To support local and global attention in GT, the concept of virtual nodes has been introduced [215], allowing these external virtual nodes to interact with the graph and facilitate information exchange between existing nodes [268].

Recent advances have also explored the integration of virtual nodes into transformer architectures [215]. For example, Boreeda et al. [14] propose Rehub, a GT-based framework in which virtual nodes communicate with each other and with graph nodes via a hub-and-spoke model, offering a structured way to capture global context.

Inspired by these developments, we propose **GTRec**, a lightweight hub-spoke model for multi-behaviour recommendation. At each layer, nodes (spokes) are dynamically assigned to a fixed number of hyperedges (hubs). This design enables fine-grained local interactions among spokes while maintaining a structured global context through hub-to-spoke communication. By adjusting the hub-to-spoke ratio, **GTRec** we aim to achieve a balance between efficiency and expressivity, thus mitigating the scalability bottlenecks inherent to conventional GTs. In addition, we

integrate differential encoding [325] to improve *representation learning* across layers.

3.1.1 Preliminaries

Virtual Nodes. Virtual nodes are artificial nodes injected into a GNN to model long-range dependencies [194]. They aggregate information from all nodes and redistribute it, effectively serving as a global communication channel or shared memory [215]. By connecting to every node in the graph, virtual nodes allow information to propagate across distant regions in a small number of hops, thus mitigating the oversquashing problem common in deep GNNs [204]. However, since virtual nodes are not semantically grounded in the underlying data, their aggregation is based on simple pooling operations (e.g., summation or averaging), which can dilute important signals and hinder the preservation of task-relevant information, particularly in heterogeneous large-scale graphs [196].

Hub-Spoke Model. The hub-and-spoke paradigm, inspired by domains such as aviation, improves efficiency by centralising communication [15]. A hub aggregates information from multiple nodes (spokes) and redistributes it, while the spokes correspond to the ordinary nodes in the network [188]. This structure reduces the cost of all-to-all communication by routing interactions through a smaller number of hubs, providing both scalability and structured information flow [196].

In graph learning, the hub-spoke model has been applied to integrate local and global contexts: spokes capture neighbourhood-level patterns, while hubs serve as global mediators that consolidate and broadcast information across distant parts of the graph [144, 215]. For example, **Rehub** [14] implements hubs as *virtual nodes*, where each spoke is dynamically assigned to a small fixed number of hubs per layer, with assignments updated using hub–hub similarity. Unlike Rehub, our GTRec approach grounds hubs in hyperedges, allowing efficient capture of higher-order user–item interactions. We hypothesise that this approach enables more expressive global *representation learning* with lower overhead, as each hub corresponds to

a concrete relation, such as a group of users co-purchasing a product, rather than a virtual node.

Differential Encoding. Learning on hypergraphs is challenging due to their inherently non-Euclidean structure [5]. A widely used solution is to integrate a global attention update into each message-passing layer, allowing the model to effectively capture information from distant nodes [220]. Traditional aggregation methods, such as message-passing, often suffer from accumulated information loss across layers, thereby weakening the representational power of embeddings [219]. To address this problem, differential encoding (DE) [325] has been proposed. Instead of summing neighbour features, DE computes differences between the features of a node and those of its neighbours; these differential signals are then aggregated, preserving valuable information that could otherwise be lost. In our GTRec model, we apply DE to improve *representation learning* as illustrated in Figure 3.2.

3.1.2 Methodology

User-item interactions are represented by a hypergraph, with nodes (spokes) for users and items, and hyperedges (hubs) capturing complex relationships. Information is propagated through graph attention for node-hub feature exchange, spoke-to-hyperedge attention to aggregate node features, and hyperedge-to-hyperedge self-attention for long-range communication. Refined hyperedge features are returned to the nodes via hyperedge-to-spoke attention, whereas spoke-to-spoke attention is designed to capture local interactions. The model architecture is shown in Figure 3.2.

Spokes and Hubs

We define N_s spokes, indexed by $i_s = 1, \ldots, N_s$, each with features $\mathbf{s}_{i_s} \in \mathbb{R}^d$, collectively denoted as $\mathbf{s}$. Similarly, we define N_h hubs, indexed by $i_h = 1, \ldots, N_h$, each with features $\mathbf{h}_{i_h} \in \mathbb{R}^d$, collectively denoted as $\mathbf{h}$. The binary hub assignment matrix

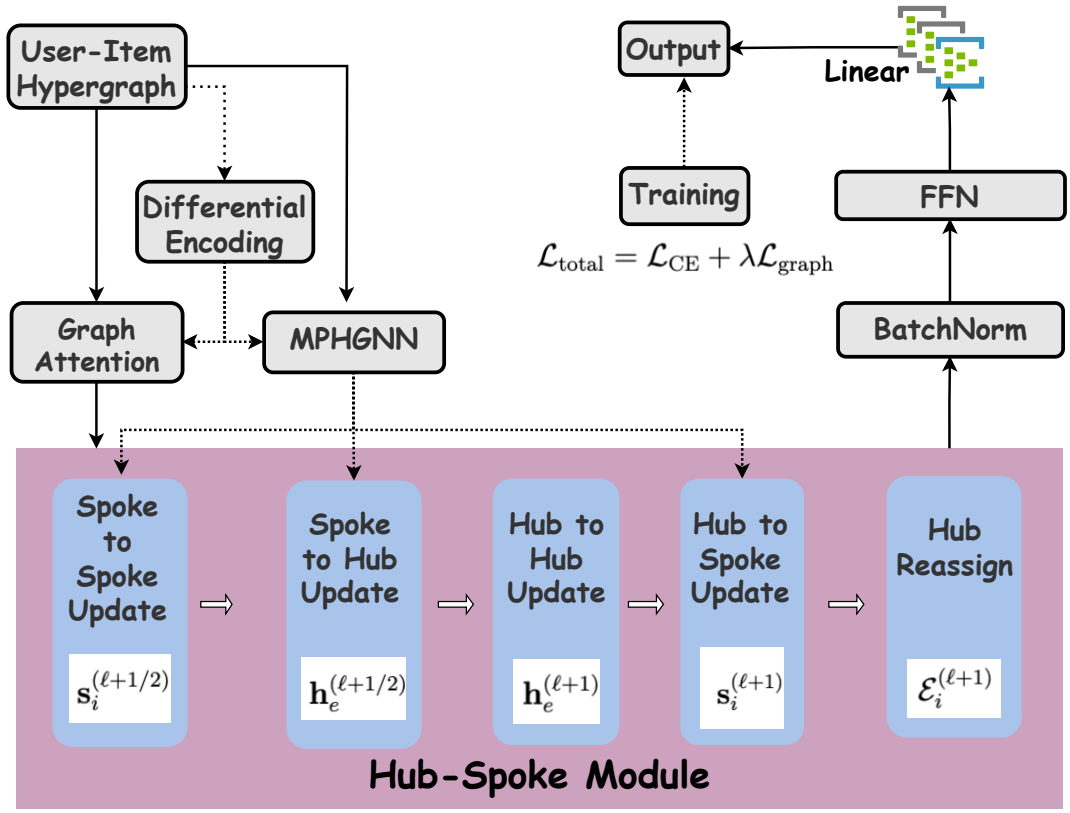

Figure 3.2: Architecture of the proposed GTRec framework. User-item interactions are modelled using a hypergraph, where nodes represent users and items, and hyperedges capture multi-behaviour interactions. Differential encoding is applied to improve *representation learning*. The attention module iteratively updates node and hub features through four steps: (1) Spoke-to-Spoke, refining local node features, (2) Spoke-to-Hub, propagating node features to hubs, (3) Hub-to-Hub, enabling global hub communication and (4) Hub-to-Spoke, propagating refined hub features back to nodes. (5) Hub Reassignment step dynamically updates hub assignments.

$\mathbf{E} \in \{0,1\}^{N_s \times N_h}$ specifies the connections between the spokes and hubs, where:

$$E_{i_s,i_h} = \begin{cases} 1 & \text{if spoke } i_s \text{ is connected to hub } i_h, \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

Graph Attention. In this model, graph attention is used to propagate information between the spokes and the hubs. The attention operation is given by:

$$\mathbf{O}, \mathbf{\Gamma} = \text{Attention}(\mathbf{K}, \mathbf{Q}, \mathbf{E}), \tag{3.2}$$

where: $\mathbf{K} \in \mathbb{R}^{n_k \times d}$ represents the source graph nodes (spokes), $\mathbf{Q} \in \mathbb{R}^{n_q \times d}$ represents the destination graph nodes (hubs), $\mathbf{E} \in \{0,1\}^{n_k \times n_q}$ is the hub assignment matrix. The output of the attention operation is: $\mathbf{O} \in \mathbb{R}^{n_q \times d}$, the updated feature representations for the destination nodes (hubs), $\mathbf{\Gamma} \in \mathbb{R}^{n_k \times n_q}$ is the attention score between the source and destination nodes.

Model Initialisation. The number of hubs is determined as $N_h = r\sqrt{N_s}$, where r is the hub ratio (default $r = 1$) and N_s is the total number of spokes (nodes in the graph). This formula ensures that the number of hubs scales proportionally with the size of the hypergraph, maintaining a balance between computational efficiency and representation capacity. To initialise the hubs, the adjacency matrix $\mathbf{A}$ of the spoke graph into clusters N_h is partitioned using the METIS algorithm [115]. Each cluster C_{i_h} corresponds to a hub i_h, and the features for the hub are initialised by aggregating the features of the spokes within the cluster:

$$\mathbf{h}_{i_h}^0 = \text{Aggregate-Feat}(\{\mathbf{s}_{i_s}^0 : i_s \in C_{i_h}\}), \tag{3.3}$$

where $\mathbf{h}_{i_h}^0$ is the initial feature vector for the hub i_h, $\mathbf{s}_{i_s}^0$ is the initial feature vector of a spoke i_s belonging to the cluster C_{i_h}, and Aggregate-Feat$(\cdot)$ is a function that combines the feature vectors of all spokes in the cluster. Clustering the adjacency matrix ensures that the axes are associated with densely connected spoke groups, preserving local structural information within the graph. Aggregating the features of spokes in each cluster allows the hubs to serve as summary representations of their respective neighbourhood. This approach captures the local context while enabling efficient processing by reducing the number of active elements in the model. Furthermore, the use of the METIS algorithm ensures that this initialisation is scalable and computationally efficient, even for large graphs, thereby providing a foundation for capturing higher-order interactions in subsequent model layers.

Hub-Spoke Update Mechanism

Each spoke is initially assigned to a single cluster hub based on its cluster membership. At each layer ℓ, the updates proceed as follows:

Spoke-to-Spoke Update. Spokes update their features by aggregating information from their neighbouring spokes through a Message Passing Hypergraph Neural Network (MPHGNN). This enables each spoke to incorporate local context:

$$\mathbf{s}_i^{(\ell+1/2)} = \text{MPHGNN}\Big(\mathbf{s}_j^{(\ell)} : j \in \mathcal{N}_i\Big), \quad \forall i. \tag{3.4}$$

where $\mathcal{N}_i$ is the set of neighbours for spoke i, and $\mathbf{s}_j^{(\ell)}$ represents the features of neighbouring spokes in the current layer.

Spoke-to-Hub Update. The updated spoke features are aggregated to refine the connected hub features. This aggregation is performed using the general attention mechanism.

$$\mathbf{h}_e^{(\ell+1/2)} = \mathcal{A}\Big(\{\mathbf{s}_i^{(\ell+1/2)} : i \in \mathcal{E}_e\}\Big), \quad \forall e. \tag{3.5}$$

where $\mathcal{E}_e$ is the set of spokes connected to the hub e. Here, $\mathcal{A}(\cdot)$ denotes a weighted combination of inputs conditioned on their relevance to the hub.

Hub-to-Hub Update. Hubs exchange information with their neighbouring hubs to capture global context in the graph. This update employs the self-attention mechanism.

$$\mathbf{h}_e^{(\ell+1)} = \mathcal{SA}\Big(\mathbf{h}_{e'}^{(\ell+1/2)} : e' \in \mathcal{N}(e)\Big), \quad \forall e. \tag{3.6}$$

Here, $\mathcal{SA}(\cdot)$ is the self-attention mechanism where the queries, keys, and values are all drawn from the same set (the hubs). This allows each hub to selectively integrate information from other hubs based on learned similarity.

Hub-to-Spoke Update. Spokes receive refined information from the hubs they are connected to, incorporating global context into their features:

$$\mathbf{s}_i^{(\ell+1)} = \mathcal{A}\Big(\{\mathbf{h}_e^{(\ell+1)} : e \ni i\}\Big), \quad \forall i. \tag{3.7}$$

where $e \ni i$ denotes the hubs connected to spoke i. As before, $\mathcal{A}(\cdot)$ represents a weighted aggregation of hub features relative to each spoke.

Hub Reassignment. At the end of each layer, the spokes dynamically reassign their connections to the most relevant hubs based on feature similarity. This ensures that connections adapt to evolving feature representations:

$$\mathcal{E}_i^{(\ell+1)} = \arg\max_{e \in \mathcal{E}} \mathrm{Sim}\Big(\mathbf{s}_i^{(\ell+1)}, \mathbf{h}_e^{(\ell+1)}\Big), \quad |\mathcal{E}_i^{(\ell+1)}| = k. \tag{3.8}$$

where Sim is a similarity function that compares the characteristics of the spoke and the hub, and k is the number of hubs to which each spoke is connected.

Complexity Analysis

The model achieves efficient scaling as follows: **Spoke-to-hub** and **hub-to-spoke** updates: $O(N_s \cdot k)$, **hub-to-hub** update: $O(N_h^2)$, with $N_h = O(\sqrt{N_s})$. This ensures linear complexity with respect to N_s, enabling scalability to large graphs. The complexity of local attention steps scales as $O(N_s \cdot d)$, while global attention through multi-hop propagation scales linearly with the number of layers, resulting in an overall complexity of $O(N_s \cdot L \cdot d)$, where L is the number of layers. This design ensures a balance between computational efficiency and expressive power.

3.1.3 GTRec-DE: Model flow with Differential Encoding

In this section, we introduce GTRec-DE, a variant of the GTRec model that incorporates differential encoding [325] in both the MPHGNN and Hub-Spoke modules. In GTRec-DE, the update rules are modified to include differential encoding

terms, which address information loss during the aggregation process and enhance the model's representational power. Specifically, for the *Spoke-to-Spoke* update, the feature vector $\mathbf{s}_i^{(\ell+1/2)}$ of each spoke i is updated using the aggregated message $\mathbf{m}_i^{(\ell)}$, which is computed by summing the messages from neighbouring *spokes*, and additionally, encoding the differential between the feature of *spoke* i and its neighbours using a Multi-Layer Perceptron (MLP):

$$\mathbf{s}_i^{(\ell+1/2)} = \phi_{\text{update}}\left(\mathbf{s}_i^{(\ell)}, \mathbf{m}_i^{(\ell)}\right), \tag{3.9}$$

where the aggregated message $\mathbf{m}_i^{(\ell)}$ is computed as:

$$\mathbf{m}_i^{(\ell)} = \sum_{j \in \mathcal{N}_i} \phi_{\text{msg}}\left(\mathbf{s}_i^{(\ell)}, \mathbf{s}_j^{(\ell)}\right) + \sum_{j \in \mathcal{N}_i} \text{MLP}(\mathbf{s}_j^{(\ell)} - \mathbf{s}_i^{(\ell)}). \tag{3.10}$$

Here, $\mathcal{N}_i$ represents the neighbours of the *spoke* i, and the first term aggregates information from the neighbouring nodes, while the second term, $\text{MLP}(\mathbf{s}_j^{(\ell)} - \mathbf{s}_i^{(\ell)})$, encodes the difference between the features of the *spoke* i and its neighbouring *spoke* j to capture unique information lost during aggregation. During *Spoke-to-Hub* update, each hub's feature is updated using an attention mechanism, and differential encoding is incorporated to preserve information specific to the difference between the spoke's and hub's features:

$$\mathbf{h}_e^{(\ell+1/2)} = \text{Attention}\left(\{\mathbf{s}_i^{(\ell+1/2)} : i \in \mathcal{E}_e\}\right) + \text{MLP}\left(\mathbf{s}_i^{(\ell+1/2)} - \mathbf{h}_e^{(\ell)}\right). \tag{3.11}$$

Here, $\mathcal{E}_e$ denotes the set of spokes connected to the hub e, and $\mathbf{s}_i^{(\ell+1/2)}$ is the updated spoke feature. The term $\text{MLP}(\mathbf{s}_i^{(\ell+1/2)} - \mathbf{h}_e^{(\ell)})$ captures the *spoke* feature and the *hub* feature, encoding this difference using an MLP. During Hub-to-Hub update, the differential encoding between the hub features is included:

$$\mathbf{h}_e^{(\ell+1)} = \mathcal{SA}\left(\{\mathbf{h}_{e'}^{(\ell+1/2)} : e' \in \mathcal{N}(e)\}\right) + \text{MLP}\left(\mathbf{h}_{e'}^{(\ell+1/2)} - \mathbf{h}_e^{(\ell)}\right). \tag{3.12}$$

where $\mathcal{N}(e)$ represents the neighbouring hubs of the hub e, and the term $\text{MLP}(\mathbf{h}_{e'}^{(\ell+1/2)} - \mathbf{h}_e^{(\ell)})$ encodes the difference between features of the neighbouring hubs and the current hub. Lastly, for the *Hub-to-Spoke* update, the feature of each spoke is updated based on connected hubs, incorporating the differential encoding term to maintain distinctions between the *hub* and *spoke* features:

$$\mathbf{s}_i^{(\ell+1)} = \mathcal{A}\left(\{\mathbf{h}_e^{(\ell+1)} : e \ni i\}\right) + \text{MLP}\left(\mathbf{h}_e^{(\ell+1)} - \mathbf{s}_i^{(\ell+1/2)}\right). \tag{3.13}$$

Here, $e \ni i$ indicates the set of hubs connected to the spoke i, and the term $\text{MLP}(\mathbf{h}_e^{(\ell+1)} - \mathbf{s}_i^{(\ell+1/2)})$ encodes the difference between the updated hub feature and the spoke feature. Differential encoding ensures that the model preserves and utilises finer-grained information across layers, reducing the information loss encountered during aggregation steps. We hypothesise that DE improves the ability of the model to represent complex structures and maintain more detailed features, particularly in deeper layers, where information loss is more pronounced [58]. By encoding the differences between the node features (spokes and hubs), the aim is to enhance the ability of the model to differentiate variations.

3.1.4 Model training

The training objective for GTRec combines **cross-entropy loss** for implicit feedback and **hypergraph-based regularisation loss** to preserve the hypergraph structure. Cross-entropy loss predicts the probability of interactions between users and items based on their embeddings. The regularisation loss ensures that similar users and items, according to the hypergraph structure, have similar embeddings. The objective functions are as follows:

Cross-Entropy Loss:

$$\mathcal{L}_{\text{CE}} = -\sum_{(u,i)\in\mathcal{D}} [y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui})] \tag{3.14}$$

where y_{ui} is the true label (0 or 1) indicating interaction, and $\hat{y}_{ui}$ is the predicted probability of interaction.

Regularisation Loss:

$$\mathcal{L}_{\text{hypergraph}} = \sum_{(u,i)\in\mathcal{D}} \|\mathbf{r}_u - \mathbf{r}_i\|^2 \cdot \text{Sim}(\mathbf{r}_u, \mathbf{r}_i) \tag{3.15}$$

where $\mathbf{r}_u$ and $\mathbf{r}_i$ are the embeddings of user u and item i, respectively. The squared Euclidean distance $\|\mathbf{r}_u - \mathbf{r}_i\|^2$ encourages embeddings of related users and items to be close to each other in the latent space. The similarity function $\text{Sim}(\mathbf{r}_u, \mathbf{r}_i)$, derived from the hypergraph structure, weights each pair according to their connectivity—pairs that share many hyperedges or interaction patterns have greater similarity and thus exert a greater influence on the loss, while weakly connected pairs contribute less. This formulation is inspired by *Laplacian hypergraph regularisation* [344], which enforces smoothness of node embeddings across hyperedges, and by HGNN models, where embeddings are updated to reflect hyperedge connectivity [56]. Essentially, $\mathcal{L}_{\text{hypergraph}}$ encourages the learned latent space to *mirror the relational structure* of the hypergraph. Finally, the total loss is calculated as follows:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \lambda\mathcal{L}_{\text{hypergraph}} \tag{3.16}$$

where λ is a regularisation parameter that balances accuracy of the recommendation and the preservation of the graph structure in the final embeddings.

3.1.5 Experiments

In this section, the datasets and baseline methods used to evaluate our proposed methods are introduced. The choice of the datasets is based on their applicability to diverse and heterogeneous real-world scenarios, in particular, the multi-behaviour recommendation task. Our experiments are carried out using the Pytorch RecBole framework [336]. For a fair comparison, we reproduce the baseline methods using

our evaluation framework and also assess their efficiency. In some cases, this results in deviations from the original reported results.

Implementation Details. The AdamW algorithm [164] with a warm-up learning rate schedule over the first 5% of the steps is applied to ensure stable convergence. This means that the learning rate (LR), which controls the magnitude of parameter updates during training, is not set immediately to its maximum value at the beginning of training. Instead, during the warm-up phase (first 5% of steps), the LR starts from a very small value (close to 0) and gradually increases until it reaches the target value. The main motivation for this strategy is twofold: (i) to ensure stable convergence, since large initial learning rates can cause divergence when weights are randomly initialised, and (ii) to achieve better optimisation, as gradual ramp-up allows the optimiser to adapt smoothly to the data distribution without sudden or unstable updates. To monitor progress and avoid overfitting, early stopping is allowed within 5 to 20 epochs. More details of the GTRec hyperparameters are found in Table 3.1.

Hyperparameter	**Value(s)**
Embedding Dimension (d)	64, 128
Number of Spokes (N_s)	Variable (Dataset-dependent)
Number of Hubs (N_h)	$r\sqrt{N_s}$, $r = 1$
Spoke-to-Hub Connections (k)	10
Number of Attention Heads	8
Number of Layers (L)	2
Dropout Rate	0.1
Learning Rate	10^{-3}
Weight Decay	10^{-5}
Optimiser	AdamW

Table 3.1: Hyperparameter settings for the GTRec model after grid search.

Results and Analysis

GTRec achieves the best performance on datasets of various sparsity and sequence lengths, compared to baseline models that include MBHT, PBAT and HMAR. These

Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
SASRec	0.667	0.642	0.681	0.645	0.644
BERT4Rec	0.810	0.732	0.894	0.797	0.692
GRU4Rec	0.642	0.687	0.573	0.585	0.568
MB-GCN	0.840	0.882	0.735	0.761	0.742
MB-GMN	0.851	0.875	0.762	0.835	0.825
GCSAN	0.871	0.881	0.848	0.848	0.846
MB-STR	0.914	0.905	0.928	0.902	0.913
MBHT	0.926	0.927	0.929	0.948	0.928
PBAT	0.932	0.945	0.934	0.952	0.931
GTRec	0.941	0.958	0.947	0.962	0.949
GTRec-DE	**0.950***	**0.961***	**0.956***	**0.964***	**0.957***

Table 3.2: Performance evaluation on the Retailrocket dataset: The best performances indicated in bold show the relative improvement over the best performing baseline, PBAT, at 0.05 significance with paired t-test. Underline indicates the second best performance. Higher HR, NDCG and MRR values indicate better performance.

Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
SASRec	0.257	0.379	0.186	0.227	0.125
BERT4Rec	0.275	0.395	0.197	0.245	0.164
GRU4Rec	0.142	0.207	0.103	0.129	0.121
MB-GCN	0.244	0.349	0.178	0.214	0.147
MB-GMN	0.398	0.496	0.226	0.306	0.153
KMCLR	0.459	0.575	0.298	0.387	0.438
MB-STR	0.694	0.776	0.596	0.616	0.675
MBHT	0.682	0.768	0.594	0.607	0.264
HMAR	0.692	0.819	0.598	0.658	0.673
PBAT	0.737	0.805	0.650	0.676	0.758
GTRec	0.764	**0.838***	0.683	0.697	0.782
GTRec-DE	**0.768***	0.836	**0.688***	**0.685***	**0.798***

Table 3.3: Performance evaluation on the Taobao dataset: Best performances indicated in bold show the improvement over the best performing baselines, PBAT/HMAR, at 0.05 significance with paired t-test. Underline indicates the second best performance.

experimental results position GTRec as a robust framework that effectively models complex user-item interactions, thereby establishing a new benchmark for large-scale multi-behaviour RSs.

Overall evaluation. From the results in Table 3.2, GNN-based methods such as (*MB-GCN*, *MB-GMN*, and *GCSAN*) perform well on the smallest dataset, **Retail-**

Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
SASRec	0.486	0.602	0.356	0.415	0.120
BERT4Rec	0.498	0.625	0.366	0.424	0.230
GRU4Rec	0.139	0.207	0.106	0.115	0.116
MB-GCN	0.463	0.522	0.289	0.339	0.479
MB-GMN	0.463	0.551	0.289	0.352	0.478
KMCLR	0.482	0.574	0.332	0.408	0.492
MB-STR	0.802	0.881	0.695	0.716	0.778
MBHT	0.776	0.857	0.678	0.707	0.772
PBAT	0.874	0.914	<u>0.794</u>	0.799	0.867
GTRec	<u>0.886</u>	<u>0.931</u>	0.791	<u>0.808</u>	<u>0.882</u>
GTRec-DE	**0.889***	**0.938***	**0.811***	**0.817***	**0.896***

Table 3.4: Performance evaluation on the IJCAI dataset: The best performances indicated in bold show the relative improvement over the best performing baseline, PBAT, at 0.05 significance with paired t-test. Underline indicates the second best performance.

Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
GRU4Rec	0.642	0.707	0.573	0.715	0.568
SASRec	0.584	0.606	0.349	0.477	0.527
BERT4Rec	0.594	0.684	0.353	0.492	0.586
MB-GCN	0.625	0.728	0.426	0.505	0.545
MB-GMN	0.655	0.735	0.483	0.539	0.587
KMCLR	0.689	0.737	0.483	0.535	0.632
MB-STR	0.746	0.756	0.496	0.599	0.727
MBHT	0.769	0.782	0.488	0.582	0.714
HMAR	0.775	0.795	0.495	0.618	0.739
PBAT	<u>0.789</u>	<u>0.823</u>	0.587	0.623	0.771
GTRec	0.784	0.819	<u>0.664</u>	<u>0.677</u>	<u>0.786</u>
GTRec-DE	**0.798***	**0.839***	**0.674***	**0.681***	**0.795***

Table 3.5: Performance evaluation on the Tianchi dataset: The best performances indicated in bold show the relative improvement over the best performing baseline, PBAT at 0.05 significance with paired t-test. Underline indicates the second best performance.

rocket, which has the lowest number of interactions and the shortest average sequence length as summarised in Table 2.2. In this dataset, the GNN message-passing mechanism effectively captures user-item interactions with less oversmoothing due to the short average sequence length. The low density also implies limited overlap between user interaction histories, resulting in competitive results by GNN-based models.

However, as the size of the dataset increases, for example, on the **Taobao**, **IJCAI** and **Tianchi**, the performance of GNN-based methods degrades, as shown in Tables 3.3– 3.5 . These datasets generate larger graphs that introduce long-range dependencies that cannot be effectively propagated by message-passing HGNNs [189]. As the number of hops increases, issues such as oversmoothing [190], oversquashing [44], TD and IPS become more pronounced in large-scale HGNNs, thus degrading the model performance. Furthermore, computational and memory bottlenecks become evident in large-scale datasets such as **IJCAI**, with more than 36 million interactions and an average sequence length exceeding 85, where repeated propagation on such large hypergraphs results in significant signal dilution that results in performance losses. In contrast to GNN-based methods, **transformer-based** models such as *SASRec* and *BERT4Rec* excel in datasets with longer average sequence length (e.g., **Taobao** and **Tianchi**), as the self-attention mechanisms effectively captures complex user-item interactions [183]. The ability to capture long user-item interactions enables these models to achieve superior performance in real-world heterogenous datasets.

Sequential models such as *MB-STR*, *MBHT*, *HMAR*, and *PBAT* leverage their advanced architectures to achieve better accuracy compared to transformer-based models in all datasets. However, these models still process sequence and graph signals independently, limiting their ability to jointly reason over temporal and structural dependencies. Our proposed **GTRec** model addresses this gap by unifying graph-based relational reasoning with transformer-based sequence modelling. Through this hybrid design, **GTRec** effectively captures both *local topological relations* in sparse interaction graphs and *global temporal dependencies* in longer behavioural sequences. This balanced representation allows GTRec to maintain high accuracy across varying dataset characteristics. This is validated by the empirical results (Tables 3.2–3.5), which show that **GTRec** consistently outperforms all GNN and transformer baselines, achieving stable generalisation across datasets of diverse properties including size, sparsity, sequence length and other factors such as TD and IPS, as discussed in **Section 2.1.3**. The **GTRec-DE** integrates **differential**

encoding, which results in the model achieving the best overall performance on the four datasets.

Dataset-Specific Insights. The **Retailrocket** dataset exhibits high sparsity and short average sequence lengths. These characteristics are effectively captured by GNN-based models such as *MB-GCN* and *MB-GMN*. However, both **GTRec** and **GTRec-DE** outperform these methods, demonstrating the benefit of combining HGNN-based and attention-based sequence modelling under sparse conditions.

The **Taobao** dataset is larger, denser and has a longer average sequence length than **Retailrocket**. On the **Taobao** dataset, **GTRec** outperforms baseline models such as *HMAR* and *PBAT*, while **GTRec-DE** achieves even higher accuracy than GTRec, due to the effective capture of complexuser-irem interaction through differential encoding. A similar trend is observed on the **IJCAI** dataset, which contains more than 36 million interactions, which poses significant challenges for both GNN- and transformer-based methods. **GTRec-DE** achieves the highest accuracy on this dataset, demonstrating strong robustness in large-scale settings. These results indicate that the GTRec architecture effectively models long-range and multitype interactions, maintaining superior performance compared to baseline methods such as GNN-based (MBGCN, MBGMN), transformer-based (SASRec, BERT4Rec) and advanced hybrid models (MB-STR, MBHT).

The **Tianchi** dataset further highlights the robustness of **GTRec** in modelling long-range user–item interactions. **GTRec** consistently maintains superior performance in this dataset, and similar to the trend observed in the other datasets, the integration of the DE mechanism in GTRec-DE improves its ability to capture complex user-item interactions, resulting in more accurate recommendations, as reflected in HR, NDCG and MRR metrics (Table 3.5). These results show that the architectural design of **GTRec** enables more effective modelling of complex, long-range dependencies, leading to superior recommendation quality.

3.1.6 Ablation Study

	Retailrocket		**Taobao**		**IJCAI**		**Tianchi**	
Model Variant	HR@10	N@10	R@10	N@10	HR@10	N@10	HR@10	N@10
w/o DE	0.849	0.856	0.816	0.773	0.867	0.648	0.795	0.651
w/o Graph Attention	0.786	0.635	0.683	0.537	0.654	0.521	0.665	0.596
w/o HGNN Module	0.768	0.647	0.678	0.607	0.683	0.565	0.684	0.612
w/o Hub/Spoke Module	0.921	0.907	0.777	0.619	0.885	0.755	0.811	0.635
w/o Metis Algorithm	0.929	0.914	0.786	0.632	0.891	0.763	0.824	0.652
GTRec-SA	0.958	0.962	**0.842**	**0.692**	**0.944**	**0.828**	0.839	0.674
GTRec	**0.961**	**0.964**	0.836	0.685	0.931	0.817	**0.839**	**0.683**

Table 3.6: Ablation study results: Performance comparison of the full GTRec model and its variants.

We conduct an ablation study to investigate the contribution of individual components to the performance of the GTRec model. On the GTRec-SA variant, we remove the hub-spoke module and replace it with standard self-attention. From the results in Table 3.6, the removal of model components such as *self attention* and *MPHGNN* results in performance degradation in all datasets. These ablation results highlight the contribution of each architectural component to **GTRec**'s performance across datasets of varying density and sequence length.

Removal of the **graph attention** and **HGNN** modules leads to a performance decline on the four datasets, confirming that this module contributes to improving the overall performance of the model. Similarly, the **hub–spoke** module contributes significantly to the recommendation accuracy of the model. Although its removal leads to a smaller performance drop compared to the HGNN module, the consistent decrease in both HR and NDCG metrics indicates that this component to improved model performance.

The **Metis** partitioning algorithm also contributes to improved performance in GTRec. Its removal produces notable performance degradation on large-scale datasets such as **Taobao** and **IJCAI**. This suggests that partitioning facilitates effective learning between clusters. For example, when the graph becomes too large to process, effective partitioning ensures that message-passing remains effective.

The **DE** module also plays a complementary role in improving the performance

of the model. Removing DE leads to a consistent decline in NDCG scores across all datasets, which underscores its importance in modelling user intent and item attributes. DE proves beneficial for large-scale, especially for large-scale datasets such as **IJCAI** and **Tianchi**, where capturing complex user-item interactions is necessary to improve the ranking accuracy.

Finally, a comparative evaluation between the full **GTRec** model and its transformer-based self-attention variant, **GTRec-SA**, contextualises the design trade-offs. **GTRec-SA** achieves competitive results and even slightly outperforms GTRec on the **Retailrocket** and **Taobao** datasets, where user-item interaction sequences are shorter and moderate, respectively, while GTRec demonstrates superior performance across datasets of varying scale and sparsity. The *hub–spoke* mechanism offers a more computationally efficient alternative to self-attention, achieving comparable or superior accuracy with lower memory overhead and better scalability on the evaluated datasets.

In general, the experimental results show that the superior performance of **GTRec / GTRec-DE** is not due to a single component but from the integration of various modules, such as the *hub-spoke* module, *Metis* clustering and **DE**. This enables **GTRec** to maintain high accuracy and efficiency on datasets that are different in density and size, demonstrating the potential of the models to address scalability issues in practical settings.

Efficiency Analysis

Method	Retailrocket		Tianchi		IJCAI		Taobao	
	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)
PBAT	1386.26	9.23	1688.64	9.78	2065.32	14.82	1990.56	13.67
MB-STR	1790.74	12.04	2125.78	12.96	2589.79	16.62	2457.87	15.98
MBHT	1211.20	8.80	1483.37	9.24	1662.15	11.51	1509.89	12.84
SASRec	1258.77	8.21	1524.50	8.84	1967.52	12.03	1790.60	12.27
BERT4Rec	1330.77	8.56	2021.50	9.67	1983.93	12.42	1859.87	12.76
GTRec	1101.22	8.12	1338.37	8.37	1517.93	11.27	1424.89	11.94
GTRec-DE	1198.41	8.31	1401.56	8.52	1601.65	11.49	1502.18	12.09
GTRec-SA	1620.31	12.28	2023.13	12.49	2612.64	15.76	2451.78	15.82

Table 3.7: Model runtime and GPU memory usage at batch size 16. We highlight the top first, second, and third results per dataset.

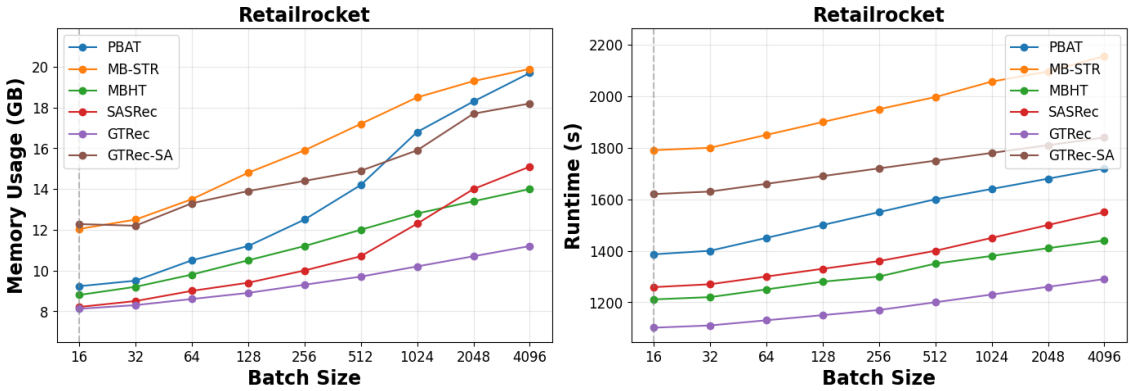

Figure 3.3: Efficiency evaluation of GTRec and baseline methods on the Retailrocket dataset.

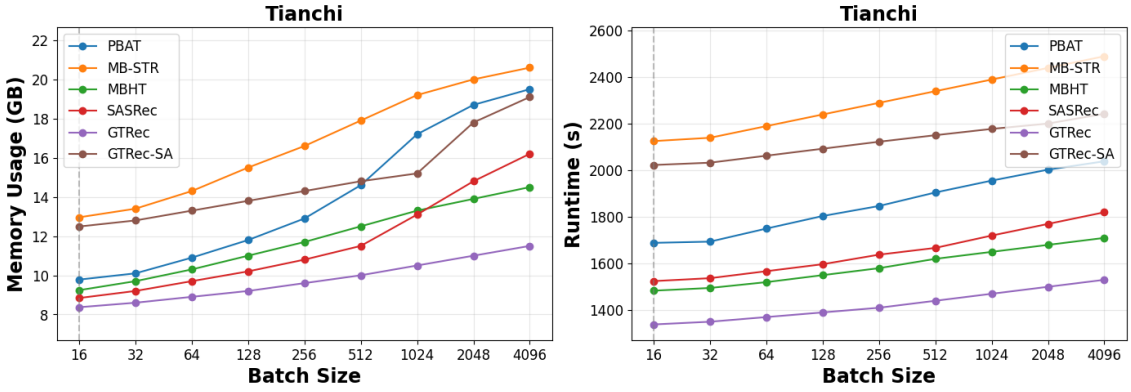

Figure 3.4: Efficiency evaluation of GTRec and baseline methods on the Tianchi dataset.

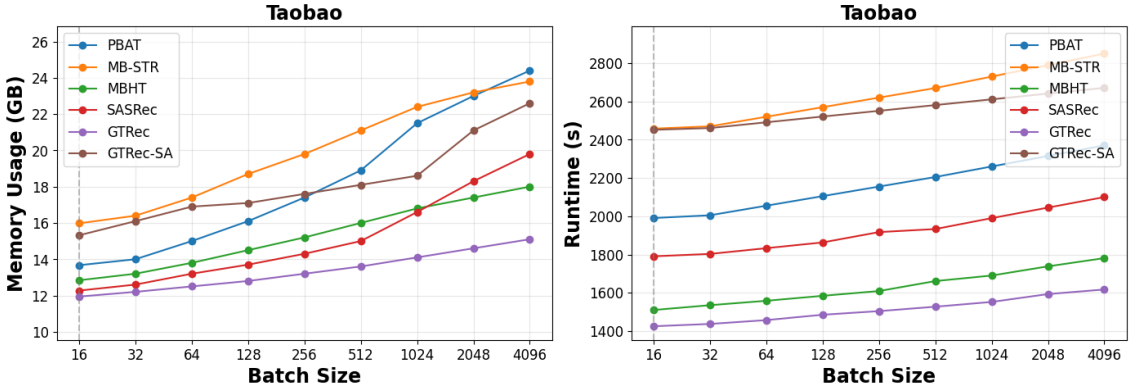

Figure 3.5: Efficiency evaluation of GTRec and baseline methods on the Taobao dataset.

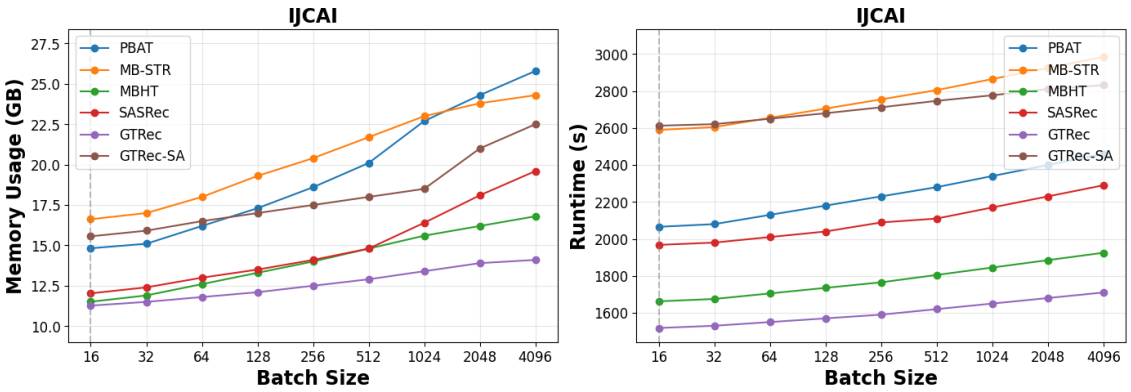

Figure 3.6: Efficiency evaluation of GTRec and baseline methods on the IJCAI dataset.

Runtime and Memory Efficiency Analysis. We evaluate the **runtime** and **memory usage** of all models across the four datasets at a batch size of 16. As shown in Table 3.7 and Figure 3.3, **GTRec** model demonstrates the best overall efficiency, achieving substantially lower runtime per epoch and reduced memory consumption compared to all baselines.

From the experimental results, the advantage of GTRec becomes more evident as the size of the dataset increases. In large-scale datasets such as **Taobao** and **IJCAI**, most baseline models experience a growth in runtime and GPU memory usage. In contrast, GTRec maintains a consistently low memory footprint and a stable convergence speed. This efficiency is attributed to the novel design of the GTRec architecture design model, which effectively shortens the computational path without incurring significant computational cost.

Experimental results in this section show that GTRec scales efficiently across varying dataset sizes. In sparse datasets such as **Retailrocket** and **Tianchi**, where message-passing is inefficient due to limited connectivity, GTRec efficiently propagates messages through the hub–spoke module, preserving both speed and accuracy. In contrast, **GTRec-SA**, which applies the full transformer self-attention mechanism, achieves high competitive accuracy, but incurs higher memory costs and slower training times as the sequence length increases. Similarly, models such as *MBHT* and *SASRec* exhibit good performance in smaller batch sizes but fail to scale efficiently on large-scale datasets such as Taobao and IJCAI.

GTRec achieves the best trade-off between efficiency, scalability, and performance on all datasets and configurations. Its consistent low resource use and stable training dynamics underscore its suitability for large-scale, real-world RS scenarios, where both computational efficiency and robust *representation learning* are essential.

Conclusion. In this section, we present the **GTRec** framework, a unified hybrid architecture that integrates innovative modules to achieve superior recommendation performance while maintaining lower computational costs than state-of-the-art methods such as MBHT and PBAT. **GTRec** introduces a *hub-spoke model* that enables the model to represent complex user-item interactions in multi-behaviour recommendation tasks, thus improving model performance. In addition, GTRec maintains *linear computational complexity* through the **hub-spoke** module, improving the efficiency of the model. Experimental results on four datasets demonstrate that GTRec consistently achieves state-of-the-art performance in both accuracy and efficiency, validating its robustness, scalability, and suitability for real-world recommendation scenarios.

3.2 STMBR: A Sparse Sinkhorn Transformer for Multi-behaviour Recommendation

Real-world user–item interaction data in recommender systems, particularly in domains such as e-commerce and social networks, often exhibit complex higher-order relational structures that go beyond simple pairwise connections [29, 142]. GT-based approaches have shown for modelling such scenarios by integrating graph topology with attention mechanisms [63]. However, despite their potential, most existing GT-based methods face sscalability challenges [288]. The quadratic computational and memory complexity of dense attention operations makes them prohibitively expensive for large-scale interaction graphs commonly found in real-world recommendation scenarios [233].

A further limitation is observed in models that adopt sparse attention mechanisms such as the Performer model [34], although they improve efficiency, the accuracy of the model is often degraded [215]. In contrast, graph-enhanced RS methods better capture complex dependencies but typically introduce substantial computational overhead [38].

To address the highlighted issues, and those discussed in Chapter 2, we introduce the **STMBR** model, another lightweight GT model that combines hypergraph-based structural learning with *sparse sinkhorn attention* [231] for multi-behaviour recommendation. As in Section 3.1, the motivation is to improve the efficiency of the model without degrading the performance of the recommendation. User-item interactions are processed by an HGAT module, which captures structural information through message-passing across hyperedges. The resulting embeddings are further refined using a differentiable sorting network [180], which ranks features based on their relevance to the multi-behaviour recommendation task. Input sequences of user-item interaction tokes are partitioned into blocks and passed to the *sorting* module. This module rearranges the blocks to focus on the most significant interactions, enabling sparse-attention computations that improve performance while reducing complexity [231]. STMBR combines graph-based structural modelling and efficient attention mechanisms, which alleviates scalability challenges while improving recommendation performance. This hybrid approach leverages the strengths of hypergraph learning, differentiable sorting, and sparse attention to deliver state-of-the-art performance in capturing diverse user-item interactions, as illustrated in Figure 3.7.

STMBR balances these trade-offs by combining hypergraph-based structural modelling with efficient sparse attention, enabling both high predictive accuracy and reduced computational cost. In this way, STMBR solves the dual challenge of efficiency and accuracy, delivering a lightweight yet powerful framework suitable for real-world large-scale and real-time recommendation scenarios.

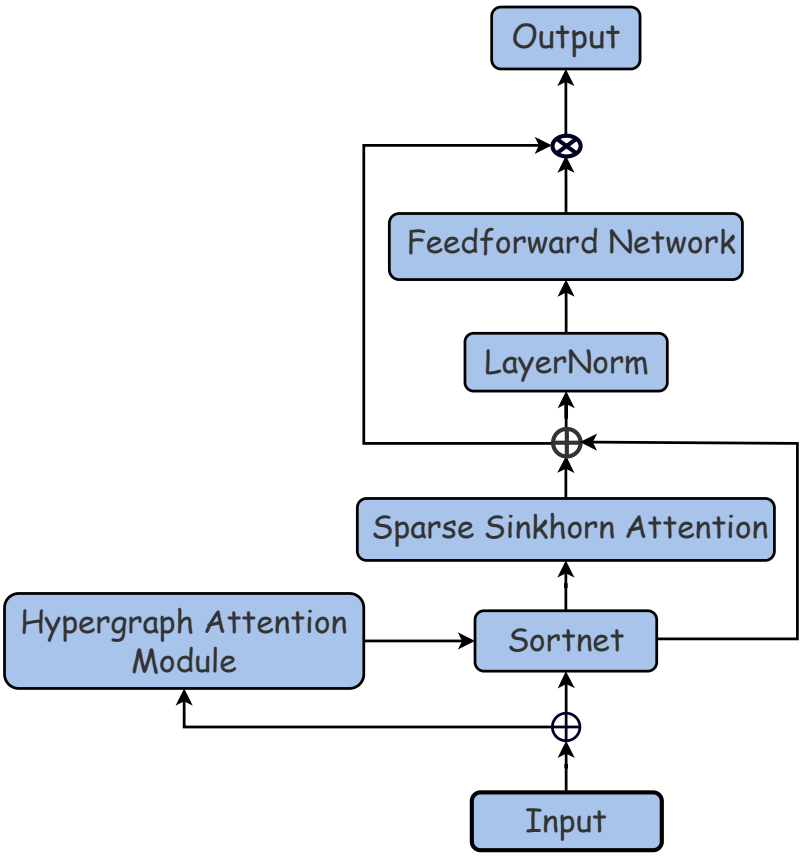

Figure 3.7: Architecture of the STMBR model. User-item interaction embeddings are passed into the Hypergraph Attention module, partitioned into blocks which are then sorted. Re-sorting the blocks facilitates effective *sparse sinkhorn* computations attention to the relevant tokens which may be further apart in the unsorted sequence.

3.2.1 Preliminaries

In this section, we introduce the main concepts applied in the STMBR model: *sinkhorn attention*, *reversible networking* and *sinkhorn normalisation.*

Sparse Sinkhorn Attention

Standard Transformer attention [237] calculates pairwise scores for all query-key pairs with complexity $O(N^2)$, which becomes prohibitive as the sequence length N increases [352]. To address this issue, we employ the *sparse sinkhorn attention* mechanism [231], which learns a blockwise reordering of sequence elements through differentiable sorting. This approach aims to reduce computational cost while main-

taining model performance. The design is motivated by the observation that long sequences often contain only a subset of semantically related tokens; thus, reordering allows the attention module to capture these dependencies more effectively [231]. Let the input token sequence be of length N. We divide it into N_B contiguous blocks of size b:

$$N = N_B \cdot b. \tag{3.17}$$

Blockwise processing reduces computation from $O(N^2)$ to $O(N \cdot b)$ for local attention, while allowing the network to learn global structure through reordering. Each block i is summarised into a vector to reduce the dimensionality of the sorting problem, allowing the sorting network to focus on coarse-grained semantic relations rather than individual tokens. A *sorting network* computes A_{ij} per head, indicating how the block i should be assigned to the position j to improve the model expressivity.

$$A \in \mathbb{R}_{\geq 0}^{N_B \times N_B}. \tag{3.18}$$

Sinkhorn Normalisation

We then apply *Sinkhorn normalisation* [218] to convert A into a soft permutation matrix:

$$P = \text{Sinkhorn}(A), \tag{3.19}$$

which is doubly stochastic, that is, all its rows sum to 1 and all its columns sum to 1, making P a valid 'soft' permutation. The block embeddings are permuted as

$$X'_{\text{blocks}} = P\, X_{\text{blocks}}, \tag{3.20}$$

where X_{blocks} is the sequence of block embeddings. Sinkhorn normalisation ensures a continuous permutation space, making sorting differentiable and allowing end-to-end gradient-based learning. After reordering, attention is calculated only *locally* (within each block or between neighbouring blocks), rather than globally. In this way, the computational cost of local attention is cheaper and, combined with learned

block reordering, capturing long-range dependencies.

Reversible Networking

In reversible neural networks, inputs are reconstructed from outputs, eliminating the need to store all intermediate activations, with the aim of reducing the use of activation memory during model training [69]. This design allows for the removal of most forward activations, recomputing them during the backward pass, thereby reducing memory usage independently of network depth and sequence length. This strategy is beneficial for large sequences and block-based models, where memory is often the main bottleneck [352].

3.2.2 Methodology

The proposed STMBR model uses a hypergraph structure to represent multi-behaviour user-item interactions, which serves as the foundation for the query ($\mathbf{Q}$), key ($\mathbf{K}$) and value ($\mathbf{V}$) vectors. The hypergraph enables the model to capture behaviour-aware embeddings ($\mathbf{b}_j$) and to represent different types of interaction, allowing the attention mechanism to focus on relevant user-item relationships. User-item interactions, represented as a hypergraph, are processed through blockwise sorting operations, followed by *sparse sinkhorn* attention, which selectively attends to the most relevant interactions, improving computational efficiency. The attention mechanism weighs the hypergraph-encoded interactions, prioritising items that align with user preferences. The workflow of the model is depicted in Figure 3.8.

Figure 3.8: Workflow of the STMBR model. The first step, **Hypergraph Representation**, represents user–item interactions and positional information. In the second step, **Sequence Encoding**, the model determines query, key, and value sequences from the hypergraph. In the third step, the **Sorting Network**, orders sequence blocks to capture structural dependencies. ***Sinkhorn Attention*** is then applied to prioritise relevant interactions. Finally, in the **Prediction** step, the model ranks and recommends items based on the computed attention scores.

Hypergraph Representation. Multi-type user-item interactions are modelled as a hypergraph $G = (V, E)$, where V represents nodes (users and items) and E represents hyperedges. A hyperedge connects a user u_i to multiple items $\{v_{j1}, v_{j2}, \ldots, v_{jk}\}$ that they have interacted with under a specific interaction type b (e.g., view, purchase). Each type of interaction b corresponds to a distinct subset of hyperedges E_b, capturing complex, multi-behaviour relationships. For personalised recommendations, a *query* vector q_i is constructed to summarise the user's preferences based on their historical interactions. The *key* vectors k_j encode salient features of the items, such as the attributes of the content or collaborative signals. Meanwhile, the *value* vectors v_j store information used for recommendation predictions. The behaviour-aware interaction representation is computed as:

$$h_j = e_j \oplus p_j \oplus b_j \tag{3.21}$$

where e_j is the item embedding, p_j is the positional encoding, and b_j represents the type of interaction behaviour.

Sequence Encoding

To effectively capture higher-order dependencies among users and items, we introduce a **Hypergraph Attention (HGAT)** module. Unlike GATs, which are limited to modelling pairwise interactions, HGATs allow the modelling of multiple nodes simultaneously, improving the expressivity of the model [20]. For each user u, we define the embedding matrix $\mathbf{H}_u$ that represents all the items the user has interacted with. These embeddings are projected into three distinct spaces to form the *query*, *key*, and *value* sequences:

$$\mathbf{Q}_u = \text{Query}(\mathbf{H}_u), \quad \mathbf{K}_u = \text{Key}(\mathbf{H}_u), \quad \mathbf{V}_u = \text{Value}(\mathbf{H}_u) \tag{3.22}$$

These sequences serve as input to the hypergraph attention mechanism, which computes the relative importance of different item interactions by aligning *queries*

with *keys* and aggregating the corresponding *values*. For notational convenience, we define the combined sequence $\mathbf{S_u}$, used in the attention mechanism $\mathbf{S}_u$ as:

$$\mathbf{S}_u = [\mathbf{Q}_u, \mathbf{K}_u, \mathbf{V}_u]\,. \tag{3.23}$$

where $\mathbf{H}_u$ represents the embeddings of all the items interacted with by user u.

Sorting Network

We apply a sorting module to the user-item interaction sequence $S \in \mathbb{R}^{\ell \times d}$, where ℓ is the sequence length and d is the dimensionality of user/item embeddings. To reduce computational complexity and enable permutation learning, the sequence is partitioned into contiguous blocks of size ℓ_B:

$$N_B = \left\lceil \frac{\ell}{\ell_B} \right\rceil, \tag{3.24}$$

where N_B is the total number of blocks, ℓ_B is the block size, and the ceiling ensures that any remaining items form an additional block if ℓ is not divisible by ℓ_B. Partitioning reduces the effective sequence length from ℓ items to N_B blocks, allowing subsequent operations to scale efficiently. A block-aggregated sequence $S' \in \mathbb{R}^{N_B \times d}$ is computed using a block-aggregation operator ψ_P, where each block S'_i is the sum of its constituent element embeddings:

$$S'_i = \sum_{j=i\cdot\ell_B}^{(i+1)\cdot\ell_B - 1} S_j, \quad i = 0, 1, \ldots, N_B - 1. \tag{3.25}$$

Here, $S_j \in \mathbb{R}^d$ is the embedding of item j in the original sequence. This aggregation is differentiable and permutation-invariant, ensuring that the order of items within a block does not affect its representation. To learn a permutation over blocks, a non-negative score matrix $R \in \mathbb{R}_{\geq 0}^{N_B \times N_B}$ is constructed, where each entry R_{ij} represents the compatibility of assigning block i to position j:

$$R \in \mathbb{R}_{\geq 0}^{N_B \times N_B}. \tag{3.26}$$

This score matrix is converted into a doubly stochastic soft permutation matrix $M \in \mathbb{R}^{N_B \times N_B}$. Each element of R is first exponentiated to ensure positivity:

$$S_{ij}^{(0)}(R) = \exp(R_{ij}), \quad i, j = 0, 1, \ldots, N_B - 1, \tag{3.27}$$

where $S^{(0)}(R)$ is the initialisation matrix for the Sinkhorn iteration. Rows and columns are then iteratively normalised using the row- and column-normalisation operators.

$$F_r(X)_{ij} = \frac{X_{ij}}{\sum_j X_{ij}}, \qquad F_c(X)_{ij} = \frac{X_{ij}}{\sum_i X_{ij}}, \quad X \in \mathbb{R}^{N_B \times N_B}, \tag{3.28}$$

where X is an intermediate matrix in the Sinkhorn iteration, X_{ij} is its element in row i and column j, $F_r(X)$ scales rows to one and $F_c(X)$ scales columns to one. The iterative Sinkhorn update is as follows:

$$S^{(k)}(R) = F_c\Big(F_r(S^{(k-1)}(R))\Big), \quad k = 1, 2, \ldots, \tag{3.29}$$

and it converges in the limit to the soft permutation matrix:

$$M = \lim_{k \to \infty} S^{(k)}(R) \in \mathbb{R}^{N_B \times N_B}. \tag{3.30}$$

Here, M is differentiable, doubly stochastic (each row and column sums to 1), and allows gradients to propagate through the sorting operation. The final sorted block sequence is obtained by multiplying the soft permutation matrix by the block-aggregated sequence:

$$S_{\text{sorted}} = M \cdot S' \in \mathbb{R}^{N_B \times d}. \tag{3.31}$$

This produces a convex combination of the input blocks according to the learned permutation, resulting in a differentiable, softly sorted sequence [2]. Operating at the block level ensures that the attention mechanism focuses on the most relevant user–item interactions. Standard attention has complexity $O(\ell^2)$, while block-based

Sinkhorn sorting reduces this to $O(\ell\ell_B)$ with $\ell_B \ll \ell$, significantly improving computational efficiency.

Attention Module

The attention module assigns importance to interactions using the query vectors ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) derived from hypergraph embeddings. Similarity scores between query and key vectors determine how well each item aligns with the user's preferences, which are used to weight the value vectors.

Attention Computation: For each query vector Q_i, similarity scores with the key vectors K_j are calculated as:

$$\text{Attention}_{i,j} = \begin{cases} Q_i \cdot \psi_P(K)_j^\top + Q_i \cdot K_j^\top, & \text{if } \left\lfloor \frac{j}{\ell_B} \right\rfloor = \left\lfloor \frac{i}{\ell_B} \right\rfloor \\ 0, & \text{otherwise} \end{cases} \tag{3.32}$$

The attention output Y is then derived as:

$$Y = \text{Softmax}(\text{Attention}) \cdot \psi_S(V) \tag{3.33}$$

where ψ_S applies the sorting to the value vectors V, ensuring that only relevant interactions contribute to the attention output. To optimise the memory efficiency of the STMBR model, we apply reversible networking (RN) [69]. During the forward pass, only the final outputs are retained, while the intermediate activations are discarded. A reversible block F takes an input sequence $X = (X_1, X_2)$, where $X_1, X_2 \in \mathbb{R}^{N_B \times d}$ represent partitions of the input sequence, and produces outputs (Y_1, Y_2) as:

$$Y_1 = X_1 + f(X_2), \qquad Y_2 = X_2 + g(Y_1), \tag{3.34}$$

where f and g are differentiable transformations (e.g., attention or feedforward blocks). During backpropagation, the inputs X_1 and X_2 are reconstructed from the

outputs:

$$X_2 = Y_2 - g(Y_1), \qquad X_1 = Y_1 - f(X_2), \tag{3.35}$$

This allows gradients to be computed without storing intermediate activations. For a multi-block sequence $S_{\text{sorted}} \in \mathbb{R}^{N_B \times d}$, reversible blocks are applied recursively to each partitioned block pair $(S_{\text{sorted},i}, S_{\text{sorted},i+1})$, ensuring memory-efficient training:

$$(S_{\text{out},i}, S_{\text{out},i+1}) = F(S_{\text{sorted},i}, S_{\text{sorted},i+1}). \tag{3.36}$$

This design aims to reduce memory usage by discarding intermediate activations, which are recomputed during backpropagation, without significantly increasing the computational cost [69].

3.2.3 Model Training

The training objective is to minimise the cross-entropy loss function, which measures the difference between the predicted probability distribution and the true labels. The model outputs a probability distribution on all items for each user. The predicted probability $\hat{p}_{ij}$ represents the likelihood that user i will interact with item j. The true label y_{ij} is one-hot encoded, where $y_{ij} = 1$ if user i interacts with the item j, and 0 otherwise. The loss function is calculated as follows:

$$\mathcal{L}_{\text{CE}} = -\sum_{i \in \mathcal{U}} \sum_{j \in \mathcal{I}} y_{ij} \log \hat{p}_{ij} \tag{3.37}$$

where: $y_{ij} \in \{0, 1\}$ is the true label vector for user i and item j, $\hat{p}_{ij}$ is the predicted probability of interaction between user i and item j.

3.2.4 Experiments

Implementation details

We use the same datasets and evaluation framework as in the GTRec model. The details of the STMBR hyperparameters are found in the Appendix Table A.2.

3.2.5 Results and Analysis

Dataset	Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
Retailrocket	MB-STR	0.914	0.905	0.928	0.902	0.913
	MBHT	0.926	0.927	0.929	0.948	0.928
	PBAT	0.932	<u>0.945</u>	0.924	0.935	0.931
	GTRec-DE	**0.950***	**0.961***	**0.956***	**0.964***	**0.957***
	STMBR	<u>0.935</u>	0.943	<u>0.948</u>	<u>0.952</u>	<u>0.927</u>
Taobao	MB-STR	0.694	0.776	0.596	0.616	0.675
	MBHT	0.682	0.768	0.594	0.607	0.264
	HMAR	0.692	0.819	0.598	0.658	0.673
	PBAT	0.737	0.805	0.650	0.676	0.758
	GTRec-DE	**0.768***	**0.836***	**0.688***	<u>0.685</u>	**0.798***
	STMBR	<u>0.746</u>	<u>0.822</u>	<u>0.679</u>	**0.689***	<u>0.776</u>
IJCAI	MB-STR	0.802	0.881	0.695	0.716	0.778
	MBHT	0.776	0.857	0.678	0.707	0.772
	PBAT	<u>0.874</u>	0.914	<u>0.794</u>	<u>0.799</u>	0.867
	GTRec-DE	**0.889***	**0.938***	**0.811***	**0.817***	**0.896***
	STMBR	0.872	<u>0.923</u>	0.778	0.790	<u>0.892</u>
Tianchi	MB-STR	0.746	0.756	0.496	0.599	0.727
	MBHT	0.769	0.782	0.488	0.582	0.714
	HMAR	0.775	0.795	0.495	0.618	0.739
	PBAT	0.787	<u>0.823</u>	0.587	0.623	0.771
	GTRec-DE	**0.798***	**0.839***	<u>0.674</u>	<u>0.681</u>	**0.795***
	STMBR	<u>0.792</u>	0.812	**0.679***	**0.686***	<u>0.789</u>

Table 3.8: Performance evaluation. Bold values indicate the best performance, showing a relative improvement over the best-performing baseline, PBAT, at 0.05 significance level with a paired t-test. Underlined values indicate the second-best performance.

As shown in Table 3.8, **STMBR** consistently outperforms the baseline methods (**MBHT**, **PBAT**, and **MB-STR**) while achieving competitive performance with **GTRec-DE**. This demonstrates that the proposed architectural design offers a robust and generalisable *representation learning* framework capable of effectively addressing the multi-behaviour recommendation task across datasets with diverse

levels of sparsity, scale, and sequence length. The dataset-specific performance insights are discussed in the following section.

Dataset-Specific Insights. On the **Retailrocket** dataset, STMBR achieves the best performance among all baselines. This dataset is sparse and contains short user-item interaction sequences, as summarised in Table 2.2. STMBR effectively models the underlying dependencies in such data, yielding competitive results comparable to those of GTRec-DE. This superior performance demonstrates the model's ability to capture complex user–item interactions, highlighting the effectiveness of the STMBR approach in real-world scenarios where challenges such as data sparsity, skewness, and popularity bias are prevalent [178].

On the **Taobao** dataset, which is larger and contains longer user–item interaction sequences, **STMBR** significantly outperforms the baseline models. Although **GTRec-DE** achieves the highest accuracy in terms of HR, NDCG and MRR metrics. **STMBR** shows competitive performance with GTRec, outperforming baseline models such as MBHT, MB-STR and PBAT, demonstrating its ability to capture complex user-item interactions despite its lightweight design. Similarly, on another large-scale dataset, **IJCAI**, which contains more than 36 million user-item interactions, **STMBR** consistently achieves higher accuracy than baseline models such as **MBHT**, **MB-STR** and **PBAT**. These results underscore the robustness of STMBR and its potential for deployment in practical settings.

Lastly, the **Tianchi** dataset, characterised by the longest *sequence length* and high density, as shown in Table 2.2, highlights the strength of **STMBR** in modelling complex dependencies. **STMBR** achieves the highest NDCG@5 and NDCG@10 scores, outperforming **GTRec-DE** and the other baseline models. These results underscore the effectiveness of STMBR in capturing complex user–item interactions, demonstrating its suitability for heterogeneous real-world recommendation tasks.

	Retail		Taobao		IJCAI		Tianchi	
Model Variant	HR@10	N@10	HR@10	N@10	HR@10	N@10	HR@10	N@10
w/o Sorting Network	0.861	0.845	0.872	0.627	0.845	0.631	0.735	0.641
w/o Attention	0.777	0.669	0.685	0.613	0.705	0.591	0.768	0.620
w/o HGAT	0.756	0.634	0.787	0.626	0.735	0.638	0.743	0.607
w/o RN	0.897	0.843	0.836	0.652	0.759	0.660	0.771	0.629
STMBR	**0.937***	**0.932***	**0.822***	**0.689***	**0.923***	**0.679***	**0.812***	**0.686***

Table 3.9: Ablation study results: Performance comparison of the full STMBR model and its variants evaluated on Recall@10 and NDCG@10.

Ablation Study

The ablation study shows the importance of each component on the performance of the STMBR model: *sorting, sparse sinkhorn* attention and HGAT. From the results in Table 3.9, the removal of each of these components leads to a noticeable performance degradation in both HR and NDCG metrics. Specifically, the removal of the attention mechanism and HGAT components reduces HR and NDCG performance compared to the full model. Similarly, excluding the *sorting* module degrades performance, while the full STMBR model consistently achieves the highest performance, confirming the synergistic contribution of the components to improving the recommendation performance.

Impact of the Model Components

In this section, we discuss the impact of key model components on the performance of the STMBR model. The synergy between the HGAT module, structured attention, dynamic sorting, and reversible networking enables the framework to capture both global and local dependencies. This integrated design allows STMBR to maintain high accuracy and robustness on diverse datasets with varying properties that are discussed in Chapter 2.

HGAT Module. The HGAT module plays a role in capturing complex user-item interactions relationships among users and items. Through this module, STMBR effectively models higher-order dependencies, making the model more expressive

than most existing GNN methods that are limited to modelling pairwise user-item interactions [62]. As shown in Table 3.9, removing the HGAT component results in a decrease in HR and NDCG in all datasets, demonstrating its role in capturing intricate dependencies, resulting in improved recommendation performance.

Sorting Network. The sorting network refines the order of the input tokens before they are passed to the sinkhorn attention module. The results in Table 3.9 indicate that removing this module leads to a noticeable performance drop, confirming that sequence ordering plays a role in improving the performance of the STMBR model.

Attention Mechanism. The attention mechanism enables STMBR to model to prioritise relevant tokens during model inference, in order to improve the accuracy of the model. When the attention module is removed, the STMBR model exhibits a decline in both HR and NDCG metrics, particularly on datasets with longer sequences such as Taobao and Tianchi. This highlights the importance of the module in improving the overall performance of the model.

Reversible Networking. The Reversible Networking component facilitates efficient information propagation between layers while maintaining gradient stability and reducing memory consumption. A performance drop is observed when the RN component is removed, confirming its positive impact on model accuracy while improving the computational efficiency of the model, which is indicated by the lower memory usage in Table 3.11.

Impact of Model Parameters

In this section, we evaluate the impact of block size and sorting iterations on the STMBR model.

Effect of Block Size. The size of the sequence block B plays a role on the NDCG performance of STMBR. From experimental evaluation, we deduce that the size of the block determines the granularity with which the permutation matrix reorders the behavioural sequences. For example, setting a very small block size (for example, $B = 16$) leads to a drop in the NDCG performance, as shown in Figure 3.9. In contrast, larger block sizes (e.g., $B = 32$ and $B = 64$) expand the attention context, increasing the model's ability to align and aggregate local and global user–item interactions and improving the model accuracy compared to the NDCG at $B = 16$. The improvements observed with $B = 32$ and $B = 64$ further suggest that larger blocks strike an effective balance between computational feasibility and modelling capacity. However, the adjustment of block size from $B = 32$ to $B = 64$ has contrasting effects on the datasets, with performance improvements observed on Retailrocket and Tianchi and slight declines recorded on Taobao and IJCAI. From this trend, we conclude that although smaller blocks provide finer granularity, which is effective for the two smaller datasets (Retailrocket and Tianchi), they fail to preserve sufficient global context, causing the model to miss important dependencies, and thereby diminishing recommendation performance on the 2 largest datasets, Taobao and IJCAI.

Effect of Sorting Iteration Values. Figure 3.10 shows the impact of the number of sorting iterations on the HR@10 performance of the STMBR model. The number of sorting iterations controls how the doubly stochastic matrix approximates the true permutation. With only two iterations, the approximation remains under-refined, producing a noisy alignment between user behaviours and candidate items. This leads to degraded top-k performance, as reflected in lower accuracy scores. Increasing the iterations to 5 or 10 yields a clear improvement: we deduce that the permutation matrix converges to a more stable and accurate ordering. Interestingly, setting the iterations to 20 does not further improve the results; in fact, it slightly degrades the performance. This phenomenon is attributed to over-refinement, where excessive iterations introduce numerical instability and diminishing returns in the

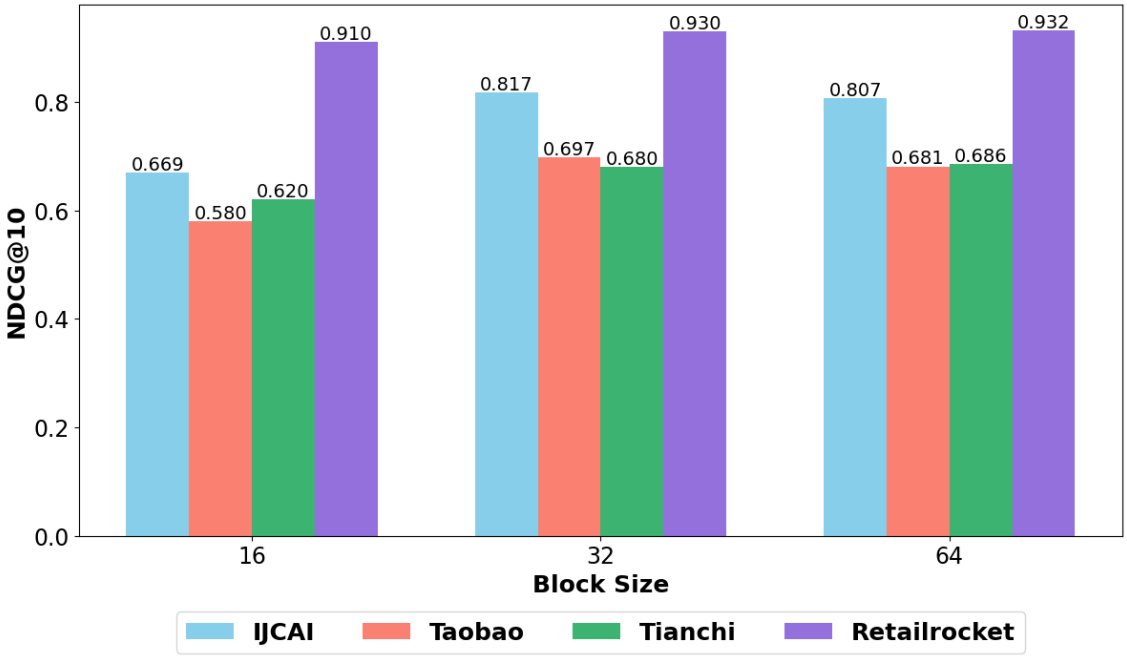

Figure 3.9: Effect of block size on the STMBR model performance.

Sinkhorn normalisation. Thus, a moderate setting (n = 5–10) provides the best balance between alignment accuracy and efficiency.

The dataset-specific insights are also interesting, showing a clear *increase-then-decrease* trend across all datasets. On the Retailrocket dataset, performance increases by 2.52% when moving from 2 to 10 iterations, followed by a slight decrease of 0.43% at 20 iterations. For the IJCAI dataset, performance improves by 9.52% from 2 to 10 iterations, but then decreases by 10.87% at 20 iterations. The Taobao dataset shows the largest relative improvement, with a 19.14% increase from 2 to 10 iterations, followed by a 8. 03% decline at 20 iterations. Similarly, the Tianchi dataset shows a 9.54% increase from 2 to 10 iterations and a minor decrease of 2.09% at 20 iterations. While all datasets follow a similar increase-then-decrease trend, the magnitude of improvement varies, with Retailrocket showing the smallest relative gains and Taobao the largest. These observations reinforce the conclusion that a moderate number of sorting iterations (n = 5-10) offers the best trade-off between computational efficiency and accurate permutation alignment across diverse recommendation scenarios. A summary of these trends is shown in Table 3.10.

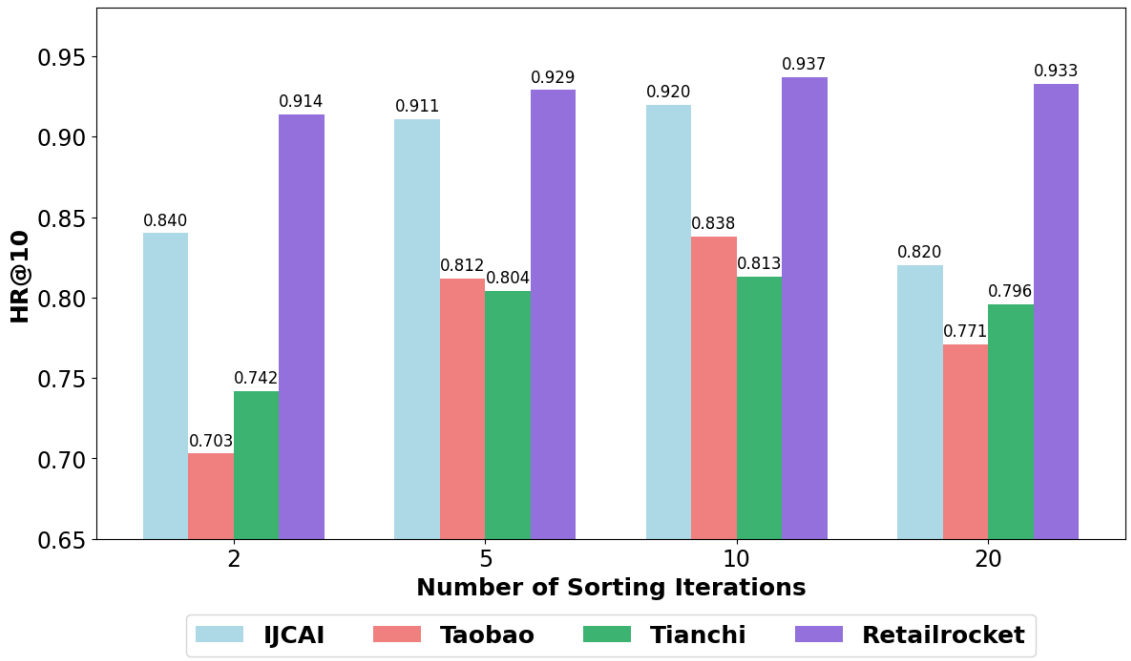

Figure 3.10: Effect of sorting iteration values on the STMBR model performance.

Dataset	2→10 Iterations	10→20 Iterations
Retailrocket	+2.52%	-0.43%
IJCAI	+9.52%	-10.87%
Taobao	+19.14%	-8.03%
Tianchi	+9.54%	-2.09%

Table 3.10: Variations in model performance (HR@10) across different numbers of sorting iterations.

Efficiency Evaluation

In this section, we evaluate the efficiency of the STMBR model.

Method	Retailrocket		Tianchi		IJCAI		Taobao	
	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)
SASRec	1258.77	8.21	1524.50	8.84	1967.52	12.03	1790.60	12.27
BERT4Rec	1330.77	8.56	2021.50	9.67	1983.93	12.42	1859.87	12.76
PBAT	1386.26	9.23	1688.64	9.78	2065.32	14.82	1990.56	13.67
MB-STR	1790.74	12.04	2125.78	12.96	2589.79	16.62	2457.87	15.98
MBHT	1211.20	8.80	1483.37	9.24	1662.15	11.51	1509.89	12.84
GTRec	1101.22	8.12	1338.37	8.37	1517.93	11.27	1424.89	11.94
STMBR	1228.31	8.55	1447.75	8.71	1621.64	11.59	1547.58	11.73

Table 3.11: Runtime and GPU memory usage of the STMBR model. We highlight the top first, second and third results per dataset.

Runtime and GPU Memory Usage. Table 3.11 reports the runtime per epoch and the GPU memory consumption of STMBR, compared to several baseline meth-

ods on four benchmark datasets. The results demonstrate that **STMBR achieves the most efficient performance profile**, consistently obtaining the lowest overall memory usage and the fastest training times on all datasets. The superior efficiency of STMBR is attributed to the novel model design: (i) the use of sparse *sinkhorn* attention that is sub-quadratic, (ii) the doubly stochastic approximation enhances accuracy without introducing significant computational overhead, and (iii) the model's flexibility to adjust block size and iteration count, which allows adaptation to the characteristics of the dataset while preserving model efficiency. In general, STMBR provides superior accuracy compared to baselines while consuming fewer computational resources, making the model suitable for deployment in large-scale, resource-constrained scenarios.

Effect of the sorting module. The sorting module is attributed to improving the computational efficiency and performance in the STMBR model, as shown in Tables 3.8 and 3.11. Partitioning the input sequence into smaller, manageable blocks reduces the computational overhead associated with attention calculations, ensures that attention computations focus only on the most relevant subsets of data, significantly improving model efficiency.

Effect of Sparse *Sinkhorn* Attention. From the results in Table 3.11, there is a significant difference in terms of efficiency between STMBR and the baseline models that use standard self-attention (BERT4Rec, SASRec, MBHT and MBSTR), which compute dense pairwise interactions, while *sparse sinkhorn* attention leverages blockwise sorting and computations to reduce computational cost. The efficiency gains in the STMBR model are also attributed in part to reversible networking, which reduces memory usage by not storing intermediate activations during backpropagation, making the model more efficient.

3.3 Chapter Summary

In this chapter, we propose GT-based GTRec and STMBR frameworks for heterogeneous recommendation. Both methods are based on a combination of lightweight self-attention and MPHGNNs to effectively capture fine-grained user-item interactions. This study is motivated by a performance drop that is synonymous with compression techniques for GNN models such as sampling or sparsification and lightweight self-attention methods. The experimental results show the benefits of these architectural innovations that improve model performance (in terms NDCG, HR and MRR) without incurring significant computational cost (in terms of GPU memory usage and runtime per epoch). Although the methods introduced in this chapter show performance and efficiency improvement, the results in the smaller Tianchi and Retailrocket datasets are consistently better than on the Taobao and IJCAI datasets, which have more user-item interactions. This is because in smaller datasets, the interaction space is relatively sparse, user-item interaction sequences are shorter, and issues such as TD and IPS are less prevalent. As a result, the model converges more quickly and achieves higher accuracy. In contrast, larger datasets present scalability challenges due to their density and heterogeneity [6]. Longer sequences and richer interaction patterns increase the complexity of learning an accurate permutation matrix, as the model must align a much larger set of interactions across diverse candidate items, in addition to other issues such as TD and IPS [301].

The evaluations also show a decrease in the results of the GNN-based methods as the size of the dataset increases. This is attributed in part to oversmoothing and oversquashing issues [330]. To address this issue, we explore two different architectures that do not incorporate the message-passing scheme in Chapters 4 and 5. **In Chapter 4**, we focus on hardware-algorithm co-design followed by an exploration of hyperbolic-geometry modelling in **Chapter 5**.

Chapter 4

Towards Efficient Hypergraph Representation Learning Without Message-Passing

In Chapter 3, two GT-based approaches, GTRec and STMBR, were proposed, leveraging a modular framework in which hypergraph and transformer modules capture local and global user-item interaction dependencies, respectively. Although these models perform well on smaller datasets such as Retailrocket, experimental results reveal a noticeable decline in NDCG performance on larger datasets such as IJCAI and Taobao. This degradation is attributed in part to **oversmoothing** [330] and **oversquashing** [44].

Another factor, as discussed in **Section 2.5**, is **the irregular structure of the hypergraph**. Large-scale hypergraphs often exhibit highly uneven distributions of nodes and hyperedges, creating not only computational bottlenecks and imbalanced workloads, but also complicating dynamic updates to the user-item interaction topology [122]. These structural irregularities hinder efficient model training and inference [66], limiting scalability and ultimately contributing to the performance drop on large datasets, as observed in the evaluation of the GTRec and STMBR models. These two issues bring us to **RQ2.1** which motivates an investigation of

whether the recommendation performance and efficiency of the HGNN-based models can be jointly improved by adapting the algorithm to the hardware structure of the GPU without applying message-passing.

> ***RQ2.1:*** *Hypergraph structure is irregular, leading to memory bottlenecks and computational redundancies. Does hardware-algorithm co-design (aligning the computational patterns of the recommendation algorithm with the structure of the underlying hardware accelerator platform, i.e., GPU), without applying HGNN message-passing, improve performance and efficiency in real-world RS scenarios?*

In addressing **RQ2.1**, we introduce the **F**ast **At**tention **H**ypergraph **(FATH)** framework, a novel *hypergraph representation learning* model for multi-behaviour recommendation, which leverages high-order permutation-invariant layers without message-passing. This *representation learning* approach has been shown to be more expressive compared to the GNN message-passing scheme [121, 228]. Furthermore, efficient higher-order attention mechanisms that approximate standard self-attention with linear time and space complexity are applied, thereby significantly reducing memory usage and achieving a faster model runtime [233].

4.1 Introduction

GNN models often face computational bottlenecks due to the irregular structure of graph-based data. Traditional optimisation techniques, developed for other neural network architectures, often do not meet the specific demands of GNNs [227]. The irregularity of graph-structured data poses significant challenges for efficient computation and memory access, both of which are necessary to achieve high performance on hardware accelerator platforms. Common acceleration platforms such as Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU), are optimised for workloads with regular patterns of computational and memory access, struggle to efficiently accelerate these methods [27].

In GNNs, the input graph often has an unpredictable structure, where each node's connections (edges) may vary significantly in number and type. This irregularity means that memory access patterns are not uniform; instead of processing data in contiguous, predictable chunks, the network must frequently access non-contiguous memory locations. This creates challenges for accelerators such as GPUs and TPUs, which are optimised for workloads with regular access patterns. Because of the irregular connectivity in GNNs, there is less opportunity for efficient parallelisation and memory coalescing (where data from adjacent memory locations is processed together). This leads to inefficient use of memory caches and memory bandwidth, as irregular memory access causes more cache misses, leading to slower processing and increased latency. Moreover, the overhead associated with managing these irregular access patterns reduces the overall efficiency of general-purpose hardware accelerators, making it difficult to achieve the same level of performance as with traditional deep learning models that have regular access patterns. This chapter focuses on overcoming these challenges by introducing innovations that adapt HGNN algorithms to hardware limitations, thereby improving their scalability in real-world recommendation scenarios.

4.1.1 Preliminaries

GPU Memory Architecture. The GPU memory structure comprises High Bandwidth Memory (HBM) and Static Random Access Memory (SRAM). HBM serves as the primary memory, while SRAM acts as a high-speed cache. Since SRAM is faster and more compact, it facilitates rapid access during computations. During model execution, we apply kernels that retrieve input data from HBM, perform computations using SRAM, and store the results back into HBM, which has been shown to reduce latency and improve computational efficiency [41]. In standard self-attention mechanisms, frequent memory transfers between HBM and SRAM increase both memory and time complexity, particularly for large models. In addressing these issues, we develop efficient algorithms that aim to alleviate this high cost by opti-

mising the compute model, leveraging specialised compute units available in GPU hardware.

Memory Hierarchy: The GPU memory hierarchy consists of slower global memory (HBM) and faster on-chip memory (SRAM). **GPU Performance Characteristics:** Modern GPUs, such as the Nvidia A100, integrate specialised units for low-precision matrix multiplication [78]. GPUs execute operations through kernels that run on thousands of threads organised into thread blocks and scheduled on streaming multiprocessors. Threads within a block form warps that communicate via fast shuffle instructions or shared memory, facilitating efficient matrix multiplications. Each kernel loads HBM input data into registers and SRAM, performs computations, and writes the output back to HBM [41]. **Specialised Compute Units:** Modern GPUs and TPUs feature specialised compute units, such as tensor cores in Nvidia GPUs, which accelerate matrix multiplications. For example, on an A100 GPU, half-precision matrix multiplications (FP16) on tensor cores are up to 16 times faster than on standard Compute Unified Device Architecture (CUDA) cores, thus improving training efficiency [79].

4.2 Methodology

Most existing RS models based on GNN and HGNN rely on iterative message passing to propagate information [68]. However, such methods are limited to modelling capture pairwise interactions, whereas real-world scenarios contain higher-order and multi-type interactions [315]. Moreover, repeated message-passing often leads to oversmoothing [330], where node embeddings become indistinguishable as layers deepen, thus reducing discriminative capacity. Another issue is that message-passing approaches suffer from oversquashing [44], since compression of rich, high-order relational signals into fixed-size vectors inevitably discards essential contextual information. In addition, standard self-attention mechanisms that are applied in most existing methods introduce additional computational bottlenecks due to frequent memory transfers between HBM and static SRAM, which increases computational

cost [41].

Our proposed FATH model addresses these challenges by shifting from message-passing to a tensor-based formulation of hypergraph learning. Instead of iteratively aggregating signals across edges, we employ permutation-invariant linear layers [177] and tensors to represent user-item interactions in a higher-order embedding space. This design ensures that the irregularities inherent in hypergraph data are effectively captured while maintaining expressive and precise embeddings [40]. Encoding relations directly in tensor space allows the model to naturally avoid the degenerative effects of oversmoothing and oversquashing: information is neither homogenised through repeated propagation nor compressed into rigid low-dimensional structures.

To further enhance computational efficiency, we propose **FATH-FA**, an FATH-based model that applies the **F**lash **A**ttention mechanism to improve computational efficiency. In the FATH-FA variant, we reorganise the attention computations with the aim of reducing the read/write frequency between HBM and SRAM [40], with the goal of reducing both latency and energy costs [41]. We hypothesise that this model is not only theoretically robust in capturing higher-order user–item dependencies but also practically efficient for deployment in large-scale settings.

Users and items are nodes, and hyperedges connect them based on different interaction types (*view, add-to-cart, purchase*). Personalised recommendations are generated by computing a *query* vector from a user's historical interactions, capturing their preferences across contexts. The *key* vector represents item features (e.g., category, brand), aligning user preferences with item characteristics, while the *value* vectors contain relevant item information for downstream recommendation tasks. The learning process models relationships between these vectors in interaction sequences to understand the dynamics between users and items. The architecture of the FATH model is shown in Figure 4.1.

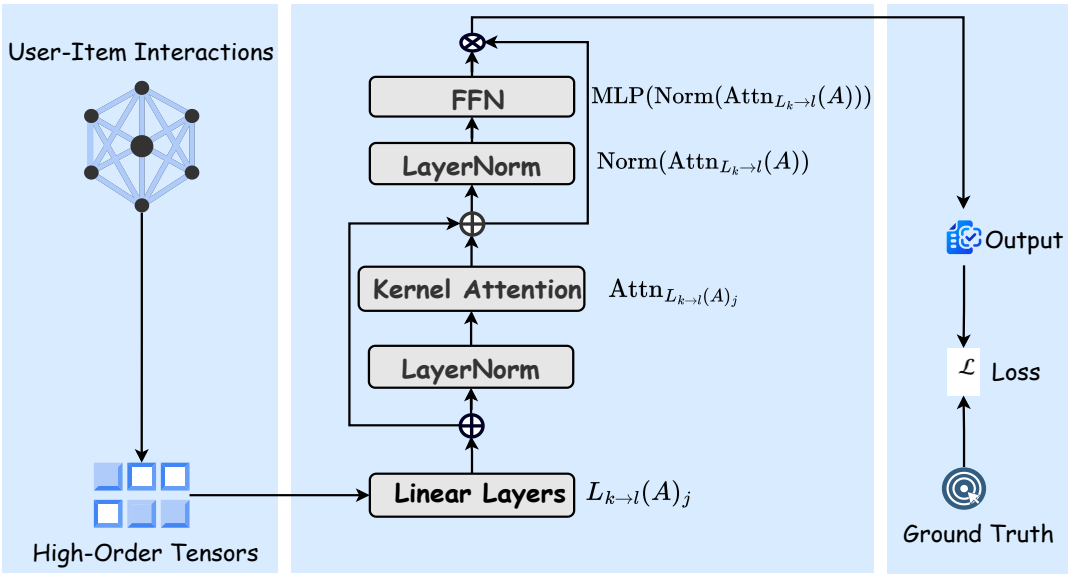

Figure 4.1: Proposed FATH framework. User-item interactions are embedded as higher-order tensor representations. Kernel attention is then applied to capture complex user-item interactions.

4.2.1 FATH: Representation Learning and Kernel Attention

A common challenge in HGNN-based recommendation methods is how to represent higher-order user-item interactions in a manner that is both expressive and computationally efficient [5]. Conventional graph neural networks are designed primarily for pairwise relationships [138], which limits their capacity to model higher-order relations, where interactions naturally involve more than two nodes. To address this limitation, we generalise linear transformations to the tensor setting, allowing the model to directly operate on k-way interactions while preserving permutation invariance. This extension allows us to capture rich relational structures that would otherwise be compressed or homogenised in traditional message-passing frameworks.

Higher-Order Linear Layers. We introduce a higher-order linear layer, $L_{k\to l}$, which projects a k-th-order interaction tensor into a structured feature. Given an input tensor $\mathbf{A} \in \mathbb{R}^{n^k \times d}$, where k is the tensor order and d the feature dimension, the projection is defined as

$$L_{k\to l} : \mathbb{R}^{n^k \times d} \longrightarrow \mathbb{R}^{n^l \times d'}. \tag{4.1}$$

For each output index $j \in \{1, \ldots, n^l\}$, the projection is computed as

$$L_{k \to l}(\mathbf{A})_j = \text{Norm}\left(\sum_{i=1}^{n^k} \mathbf{B}_{i,j}\, \mathbf{A}_i\, \mathbf{W} \right), \tag{4.2}$$

where $\mathbf{W} \in \mathbb{R}^{d \times d'}$ is a learnable weight matrix, $\mathbf{B} \in \{0,1\}^{n^k \times n^l}$ encodes active hyperedges,

$$\mathbf{B}_{i,j} = \begin{cases} 1 & \text{if hyperedge } (i,j) \text{ is active,} \\ 0 & \text{otherwise,} \end{cases}$$

and Norm$(\cdot)$ ensures stable representations. The input dimension d captures raw node or interaction features, while the output dimension d' defines the subspace processed by the attention module. This layer both reduces dimensionality to manage computation and decomposes representations into subspaces naturally suited for multi-head attention, yielding expressive embeddings for downstream recommendation tasks.

Kernel Attention. Higher-order linear layers encode complex relational patterns, which are weighted according to their contextual relevance for each query node. To achieve this, we introduce a kernel-based attention mechanism. For each attention head h, the attention coefficient $\alpha^h_{i,j}$ between node i and query j is computed as:

$$\alpha^h_{i,j} = \frac{\exp\left(\sigma(\mathbf{Q}^h_j, \mathbf{K}^h_i) \cdot S_{i,j} / \sqrt{d'}\right)}{\sum_{i'} \exp\left(\sigma(\mathbf{Q}^h_j, \mathbf{K}^h_{i'}) \cdot S_{i',j} / \sqrt{d'}\right)}, \tag{4.3}$$

where $\mathbf{Q}^h_j$ and $\mathbf{K}^h_i$ are the query and key vectors for head h, $\sigma(\cdot)$ is a similarity function (e.g., scaled dot product) and $S_{i,j}$ is the hypergraph adjacency indicating whether a connection is active. By masking inactive hyperedges, the attention operation focuses on meaningful relationships, thereby preventing oversmoothing and reducing redundant computation. The attention output aggregates information

across all heads, while a residual connection retains the original linear embeddings:

$$\mathrm{Attn}_{k\to l}(\mathbf{A})_j = \sum_{h=1}^{H} \left(\sum_i \alpha_{i,j}^h \Big(L_{k\to l}(\mathbf{A})_i \Big) \right) \mathbf{W}_h^V \mathbf{W}_h^O + \mathrm{Linear}(\mathbf{A}_j), \tag{4.4}$$

where H is the number of attention heads, W_h^V and W_h^O are the *value* and *output* projections, and the residual path stabilises training and preserves the original signals. Each head attends to a distinct subspace of d', allowing the model to capture complementary relational aspects. Importantly, the restriction to active hyperedges reduces complexity from $O(n^2)$ to $O(k \cdot n)$, where $k \ll n$, enabling scalability to large hypergraphs.

$$O(n \cdot d \cdot d') + O(k \cdot n \cdot d') \approx O(k \cdot n \cdot d'). \tag{4.5}$$

4.2.2 FATH-FA: Flash Attention for Enhanced Efficiency.

To further improve the efficiency and scalability of the FATH model, we propose *FATH with Flash Attention* (**FATH-FA**). The core idea of FATH-FA is to accelerate and refine the attention mechanism within hypergraph learning by combining *node-to-node attention*, *node-to-hyperedge attention*, and *Relative Positional Embedding (RPE)*. In addition, the model incorporates **hyperedge-augmented representations**, which enrich the learned embeddings with additional contextual information and strengthen the modelling of higher-order dependencies [346]. The model architecture is shown in Figure 4.2.

Flash Attention-Enhanced Representation Learning. Standard attention mechanisms, despite their expressiveness, incur significant input/output (I/O) computational and memory overheads due to frequent memory transfers between high-bandwidth memory (HBM) and on-chip SRAM. To mitigate this, we extend our FATH model to *FATH with Flash Attention* (**FATH-FA**), which reorganises attention computation to minimise memory traffic [41, 40], while preserving rich relational representations.

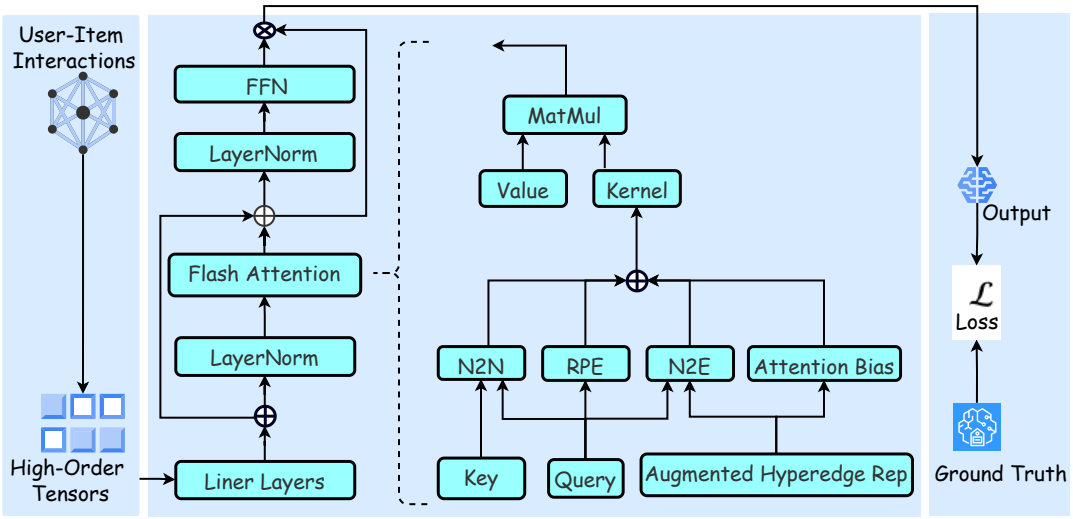

Figure 4.2: FATH-FA model architecture consisting of the embedding layer, attention layer and the prediction layer.

Node-to-Node and Node-to-Edge Attention. FATH-FA incorporates two complementary attention mechanisms. **Node-to-node attention** captures direct connections between nodes.

$$\text{Attn}_{i\to j} = \sum_{h=1}^{H} \phi(\mathbf{Q}_i^h) \cdot \phi(\mathbf{K}_j^h) \cdot \mathbf{A}_i w_h^V, \tag{4.6}$$

where $\phi(\cdot)$ denotes the linear projection into the h-th attention head space, $\mathbf{Q}_i^h$ and $\mathbf{K}_j^h$ are the *query* and *key* representations of nodes i and j, and w_h^V is the value weight. **Node-to-edge attention** captures higher-order interactions by modelling the contribution of a node i to the hyperedge μ it belongs to:

$$\text{Attn}_{i\to\mu} = \sum_{h=1}^{H} \phi(\mathbf{Q}_i^{\mu,h}) \cdot \phi(\mathbf{K}_\mu^h) \cdot \mathbf{A}_i w_{h,\mu}^V, \tag{4.7}$$

where $\mathbf{Q}_i^{\mu,h}$ and $\mathbf{K}_\mu^h$ denote the *query* and *key* of node i and hyperedge μ, and $w_{h,\mu}^V$ is the *value* weight. The attention output aggregates both node-to-node and node-to-edge contributions across all heads:

$$\text{Attn}_{k\to l}(\mathbf{A})_j = \sum_{h=1}^{H} \sum_i \alpha_{i,j}^h \Big(L_{k\to l}(\mathbf{A})_i\Big) W_h^V W_h^O, \tag{4.8}$$

where $L_{k\to l}(\mathbf{A})_i$ is the i-th row of the higher-order linear projection, W_h^V and W_h^O are the value and output projections, and $\alpha_{i,j}^h$ are learned attention coefficients.

Relative Positional Embedding. To capture sequential and structural dependencies beyond absolute indices, we incorporate *relative positional embedding* (RPE) into the attention mechanism. Unlike absolute encodings, which assign a fixed positional vector to each element, relative encodings model the pairwise offset between tokens, hyperedges, or nodes. Formally, the attention score is augmented with a learnable bias r_{i-j}:

$$\mathrm{Attn}(Q,K,V) = \mathrm{Softmax}\left(\frac{QK^\top}{\sqrt{d'}} + R\right)V, \qquad R_{i,j} = r_{i-j}. \tag{4.9}$$

where $Q, K, V \in \mathbb{R}^{n\times d'}$ are the *query, key,* and *value* matrices, and $R \in \mathbb{R}^{n\times n}$ encodes relative positional biases. RPE enables the model to generalise across varying sequence lengths and higher-order structures. In the context of high-order tensors, RPE enriches hyperedge-aware attention by aligning embeddings with both interaction order and relative structure, yielding more expressive and context-sensitive representations.

Chunk-Wise Flash Attention. To further improve memory efficiency and scalability, FATH-FA uses fragment-wise FlashAttention [302]. Instead of materialising the full $n \times n$ attention matrix:

$$\mathrm{Attention}(Q,K,V) = \mathrm{Softmax}\Big(\frac{QK^\top}{\sqrt{d'}}\Big)V, \tag{4.10}$$

attention is computed incrementally on smaller chunks of size C, maintaining numerical stability and reducing the memory footprint. For chunk i, the intermediate

state $S[i]$ and corresponding output $O[i]$ are updated as:

$$S[i+1] = \gamma \cdot S[i] + \sum_{j=iC}^{(i+1)C-1} \mathbf{K}_j \mathbf{V}_j^\top, \tag{4.11}$$

$$O[i+1] = O[i] + \sum_{j=iC}^{(i+1)C-1} \mathbf{Q}_j \left(S[i] \mathbf{W}^O \right), \tag{4.12}$$

where γ is a scaling factor, and $\mathbf{Q}_j$, $\mathbf{K}_j$, $\mathbf{V}_j$ are the *query, key,* and *value* vectors of the j-th element. This reduces memory complexity from $O(n^2 d')$ to $O(C^2 d')$ per chunk. The chunk size C is dynamically determined based on available SRAM:

$$C = \left\lfloor \frac{\text{Available SRAM}}{d' \cdot (2 + H)} \right\rfloor, \tag{4.13}$$

ensuring that computation fits within on-chip memory, while maintaining multi-head attention functionality.

Summary. The representation and attention pipeline in FATH-FA integrates higher-order linear layers for permutation-invariant embeddings, node-to-node and node-to-edge kernel attention for expressive aggregation, relative positional embedding for structure-aware contextualisation, and chunk-wise Flash Attention for scalable, memory-efficient computation. This unified framework preserves rich relational information, captures both direct and higher-order interactions, and scales to large hypergraphs, overcoming limitations of existing HGNN-based RS methods.

4.2.3 Model Training

Our training objective is to minimise the cross-entropy loss. Let $P(j^+|u,i)$ denote the predicted probability that item j is the next item for user u after interacting with item i. The total loss combines cross-entropy with L_2 regularisation:

$$L_{\text{total}} = -\frac{1}{N} \sum_{(u,i)\in\mathcal{D}} \log \frac{\exp(P(j^+|u,i))}{\sum_{j\in\{j^+\}\cup J^-} \exp(P(j|u,i))} + \lambda \sum \|\mathbf{W}\|_2^2, \tag{4.14}$$

where J^- are negative samples, N is the number of user-item pairs in the data $\mathcal{D}$, and $\mathbf{W}$ represents all trainable weight matrices. The predicted probabilities $P(j|u,i)$ are computed using the FATH or FATH-FA models.

Training Algorithms. The training of **FATH** proceeds via a standard forward and backward pass. The forward pass computes attention scores across all nodes and hyperedges using higher-order linear layers and kernel attention (Equations. 4.4), storing intermediate attention weights for backpropagation, and the backward pass then propagates gradients through all parameters.

The training of **FATH-FA** modifies this procedure for memory efficiency. The forward pass is performed in *chunks*, where *queries, keys,* and *values* are processed fragment-wise using *flash attention*, reducing (I/O) and memory overhead (Eq. 4.13). During the backward pass, attention scores are recomputed on demand rather than stored, allowing gradient computation through higher-order linear layers and kernel attention.

Algorithm 1 FATH-FA: Forward Pass with Chunk-Wise FlashAttention

Require: Query Q, Key K, Value V, Mask S, RPE bias R, Weight Matrices W^V, W^O
Ensure: Output O
1: Divide Q, K, V into chunks of size C (Eq. 4.13).
2: Initialise $O = 0$.
3: **for** $i = 1$ to N (chunks) **do**:
4: Load chunk $Q[i], K[i], V[i]$ into shared memory.
5: Compute scores with bias and mask (Eq. 4.9).
6: Apply softmax attention (Eq. 4.10).
7: Compute chunk output $O[i]$ (Eq. 4.12).
8: **end for**
9: Concatenate chunk outputs and project with W^O (Eq. 4.8). **return** O

4.3 Experiments

To evaluate the effectiveness of the FATH and FATH-FA models, we conduct experiments using the datasets introduced in Chapter 2: **Retailrocket, IJCAI, Tianchi**,

Algorithm 2 FATH-FA: Backward Pass with Recomputation

Require: Gradients ∇O, Inputs Q, K, V, Mask S, RPE bias R
Ensure: Gradients $\nabla Q, \nabla K, \nabla V$
1: Divide Q, K, V into chunks of size C (Eq. 4.13).
2: Initialise gradients to zero.
3: **for** chunk $i = 1$ to N **do**:
4: Recompute scores with bias and mask (Eq. 4.9).
5: Recompute attention weights (Eq. 4.10).
6: Backpropagate through values, queries, and keys (Eqs. 4.11–4.12).
7: **end forreturn** $\nabla Q, \nabla K, \nabla V$

and **Taobao**. The hyperparameters for each model are selected by grid search in the ranges specified in Table 4.1.

Hyperparameter	**Search Range**
`Embedding Dim (d)`	64–512
`Chunk Size (C)`	32–128
`Attention Heads (H)`	4–16
`Flash Scaling Factor (`γ`)`	0.5–1.0
`Negative Samples (`$\lvert J^- \rvert$`)`	50–200

Table 4.1: FATH-FA Hyperparameters search table

After grid search over the ranges in Table 4.2, we apply the following hyperparameters in the experiments.

Hyperparameter	**Retailrocket**	**Tianchi**	**Taobao / IJCAI**
`Chunk Size (C)`	256	128	64
`Negative Samples`	100	200	500
`Dropout`	0.2	0.2	256
`Flash Scaling Factor (`γ`)`	0.5	0.7	0.7
`Attention Heads (H)`	4	8	8

Table 4.2: Dataset-specific hyperparameters and attention configurations for the FATH-FA model after grid search.

4.3.1 Results and Analysis

Performance Evaluation. From the results in Table 4.3, both **FATH** and **FATH-FA** consistently outperform all baseline models. This demonstrates the robustness and adaptability of the higher-order tensor framework employed in these models

Dataset	Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
Retailrocket	MB-STR	0.914	0.905	0.928	0.902	0.913
	MBHT	0.926	0.927	0.929	0.948	0.928
	PBAT	<u>0.932</u>	<u>0.945</u>	0.924	0.935	0.931
	FATH	0.923	0.942	<u>0.936</u>	<u>0.951</u>	<u>0.933</u>
	FATH-FA	**0.938***	**0.948***	**0.953***	**0.957***	**0.941***
Taobao	MB-STR	0.694	0.776	0.563	0.616	0.675
	MBHT	0.682	0.768	0.594	0.607	0.264
	HMAR	0.692	0.819	0.598	0.658	0.673
	PBAT	0.737	0.805	0.650	0.676	0.758
	FATH	<u>0.754</u>	<u>0.824</u>	**0.678***	**0.687***	<u>0.775</u>
	FATH-FA	**0.758***	**0.831***	<u>0.674</u>	<u>0.680</u>	**0.778***
IJCAI	MB-STR	0.800	0.881	0.695	0.716	0.778
	MBHT	0.776	0.857	0.678	0.707	0.772
	PBAT	0.874	0.914	0.794	0.799	0.867
	FATH	**0.881***	**0.926***	<u>0.798</u>	**0.811***	<u>0.879</u>
	FATH-FA	<u>0.869</u>	<u>0.917</u>	**0.802***	<u>0.807</u>	**0.886***
Tianchi	MB-STR	0.746	0.756	0.496	0.599	0.727
	MBHT	0.769	0.782	0.488	0.582	0.714
	HMAR	0.775	0.799	0.495	0.618	0.739
	PBAT	0.787	0.823	0.587	0.623	0.771
	FATH	<u>0.804</u>	**0.830***	**0.686***	**0.696***	**0.798***
	FATH-FA	**0.807***	<u>0.826</u>	<u>0.683</u>	<u>0.692</u>	<u>0.787</u>

Table 4.3: Performance evaluation across datasets. Best results (bold) denote statistically significant improvements ($p < 0.05$, paired t-test) over the best-performing baselines (PBAT, HMAR). Underlined values indicate the second-best performance.

across datasets of different interaction densities and sequence lengths. Moreover, the performance of these models varies depending on the characteristics of each dataset, as discussed in the following analysis.

Dataset-Specific Insights. On the **Retailrocket** dataset—the smallest among the evaluated datasets—**FATH-FA** achieves the highest performance across all evaluation metrics, slightly surpassing PBAT, MBHT, and MB-STR. These results suggest that the FATH framework effectively enhances *representation learning* in sparse, short-sequence recommendation settings, leading to improved model accuracy.

On the **Taobao** dataset, which contains dense and long user–item interaction sequences, both **FATH** and **FATH-FA** achieve superior performance compared to baseline models. This demonstrates their effectiveness in modelling long-range dependencies and fine-grained behavioural transitions. The marginal improvement of **FATH-FA** over **FATH** in the HR and MRR metrics highlights the advantage

of modular attention mechanisms (N2E and N2N), which effectively leverage dense user–item interaction signals to improve *representation learning* and overall model performance.

On **IJCAI**, another large-scale dataset, both FATH and FATH-FA achieve superior performance compared to baselines such as PBAT and HMAR. **FATH** achieves the best performance in terms of HR metrics, while **FATH-FA** records slightly higher NDCG and MRR. Similarly to the other datasets, these findings suggest that the base FATH architecture effectively captures global user-item interactions, thus improving recommendation performance.

On the **Tianchi** dataset, which is characterised by moderate sparsity and extremely long user-item interaction sequences, both **FATH** and **FATH-FA** achieve competitive performance, significantly exceeding all baseline models. These results further demonstrate the effectiveness of the proposed FATH-based approaches, achieving higher accuracy than MBHT, MB-STR and PBAT.

In summary, the **FATH** and **FATH-FA** models demonstrate better performance on datasets with varying levels of density and sequence length, compared to baseline models. The integration of HGNN-based reasoning with efficient attention allows the model to effectively balance global relational awareness and sensitivity to local interactions. The consistent improvements achieved by **FATH-FA** further establish it as a high-accuracy and computationally efficient framework for heterogeneous recommendation system settings.

	RetailRocket		**Taobao**		**IJCAI**		**Tianchi**	
Model Variant	HR@10	N@10	HR@10	N@10	HR@10	N@10	HR@10	N@10
w/o RPE	0.896	0.784	0.746	0.562	0.749	0.587	0.770	0.614
w/ N2E attention	0.854	0.832	0.782	0.577	0.734	0.696	0.616	0.581
w/o N2N attention	0.765	0.746	0.718	0.634	0.723	0.555	0.757	0.598
w/o Chunking	0.858	0.866	0.773	0.659	0.731	0.642	0.761	0.677
FATH-FA	**0.947***	**0.958***	<u>0.826</u>	<u>0.682</u>	<u>0.909</u>	<u>0.805</u>	**0.834***	**0.698***
FATH	<u>0.944</u>	<u>0.955</u>	**0.831***	**0.689***	**0.913***	**0.811***	<u>0.830</u>	<u>0.693</u>

Table 4.4: Ablation study results showing the effect of key modules on the FATH-FA model performance. Bold values (*) denote the best result, while underlined values indicate the second best.

	RetailRocket		**Taobao**		**IJCAI**		**Tianchi**	
Model Variant	HR@10	N@10	HR@10	N@10	HR@10	N@10	HR@10	N@10
FATH-MP	0.753	0.747	0.721	0.541	0.718	0.565	0.762	0.593
FATH-SA	**0.953***	<u>0.949</u>	**0.835***	**0.693***	**0.919***	**0.820***	<u>0.825</u>	**0.697***
FATH	<u>0.944</u>	**0.955***	<u>0.831</u>	<u>0.689</u>	<u>0.913</u>	<u>0.811</u>	**0.830***	<u>0.693</u>

Table 4.5: Ablation study results showing the effect of key modules on the FATH model performance. Bold values (*) denote the best result, while underlined values indicate the second best.

Ablation Study

To examine the contribution of each component within the proposed models, we conduct an ablation study comparing the performance of **FATH** and **FATH-FA** with several variants of the model created by selectively removing or modifying specific modules or architectural elements. In the **FATH-MP** variant, the MLP-based hypergraph module is replaced with a standard hypergraph convolutional network to quantify the impact of message-passing. In **FATH-SA**, the kernel-based attention is replaced with full self-attention to explore the trade-off between computational efficiency and representational capacity. The **w/o RPE** variant removes the relative positional encoding (RPE) from **FATH-FA** to assess its effect on sequential modelling. In **w/o N2E attention**, the Node-to-Edge attention mechanism within **FATH-FA** is disabled to evaluate its role in modelling higher-order dependencies. Finally, in **w/o N2N attention**, the Node-to-Node attention pathway is removed to assess its importance for capturing direct user–item interactions. The results, summarised in Tables 4.4 and 4.5, report the performance on the RetailRocket, Taobao, IJCAI and Tianchi datasets.

FATH Ablation. Replacement of the hypergraph MLP module with standard message-passing (**FATH-MP**) results in a significant performance decline in terms of NDCG and HR metrics. In the **FATH-SA** variant, incorporating full self-attention often leads to higher accuracy than the base FATH model, indicating that although kernel attention provides greater computational efficiency, full self-

attention offers enhanced expressiveness of the model.

FATH-FA Ablation. Removing the relative positional encoding (**RPE**) consistently reduces performance across all datasets, confirming its essential role in modelling the temporal and sequential behaviour of users. Similarly, removing the **N2E**, **N2N** attention components, or disabling chunking also leads to decline in model accuracy, indicating that each module contributes complementary information. Specifically, **N2E** attention helps to capture higher-order structural dependencies, while **N2N** attention is essential to preserve direct user-item interaction modelling.

Summary. The ablation results confirm the effectiveness of the architectural choices in both **FATH** and **FATH-FA**. In **FATH**, kernel attention achieves an effective balance between performance and computational efficiency, while full self-attention offers performance gains. For **FATH-FA**, all components **RPE, N2E attention, chunking,** and **N2N attention**, contribute significantly, with the highest performance achieved on the complete model. Thus, the observed improvements in FATH/FATH-FA models arise not from any single component but from the synergistic interaction among these modules.

Efficiency Evaluation

In this section, we evaluate the efficiency of the FATH and FATH-FA models in terms of the GPU memory usage and runtime per epoch.

Method	Retail		Tianchi		IJCAI		Taobao	
	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)	Epoch (s)(↓)	Mem. (GB)(↓)
PBAT	1386.26	9.23	1688.64	9.78	2065.32	14.82	1990.56	13.67
MB-STR	1790.74	12.04	2125.78	12.96	2589.79	16.62	2457.87	15.98
MBHT	1211.20	8.80	1483.37	9.24	1662.15	11.51	1509.89	12.84
FATH-MP	1234.11	11.07	1254.46	11.98	1389.67	13.21	1678.98	13.06
FATH-SA	1830.20	11.56	2146.49	12.24	2125.15	13.73	1845.87	13.65
FATH-FA	**1136.12**	**8.68**	**1253.42**	**8.97**	**1264.17**	**10.71**	**1254.45**	**10.16**
FATH	<u>1189.32</u>	<u>8.76</u>	<u>1279.15</u>	<u>9.07</u>	<u>1365.82</u>	<u>10.76</u>	<u>1276.65</u>	<u>10.26</u>

Table 4.6: Runtime and GPU memory usage per dataset. The best results are indicated in bold, while the second-best results are underlined.

Runtime and GPU Memory Usage. Table 4.6 presents the runtime per epoch and the GPU memory usage for each model on the four benchmark datasets. The results highlight the computational advantages of the proposed FATH and FATH-FA frameworks compared to transformer-based (MB-STR, PBAT) and message-passing (MBHT, FATH-MP) baselines. FATH-FA achieves the most efficient performance, recording the lowest average runtime and memory consumption on all datasets. FATH also maintains comparable efficiency while offering slightly reduced computational savings relative to FATH-FA. This superior efficiency is attributed to the combination of kernel-based attention and the structured hypergraph encoding used in both models. The *kernel* and *flash* attention mechanisms reduce the quadratic complexity of standard self-attention [233]. As a result, FATH-based models preserve expressive power while maintaining a lower computational cost as the size of the dataset increases.

FATH-MP and **FATH-SA**, which replace core architectural components for ablation purposes, show increased runtime and memory demand. **FATH-MP** introduces additional overhead from message-passing in hypergraph convolution, while self-attention in **FATH-SA** significantly increases computational cost due to quadratic computational complexity in sequence length. These results confirm that the proposed architectural refinements, *kernel/flash attention*, and **linear-layer embedding** contribute to the efficiency of FATH/FATH-FA models. MBHT demonstrates relatively higher computational efficiency compared to PBAT and MB-STR, but still lags behind FATH and FATH-FA. The difference is more pronounced on large-scale datasets such as *IJCAI* and *Taobao*, where FATH and FATH-FA maintain stable runtime and memory growth, while the baselines exhibit steeper increases due to heavier attention and aggregation operations.

Overall, these experimental results confirm that the proposed FATH and FATH-FA models successfully improve scalability and computational efficiency without compromising the predictive accuracy of the model. The design principles - *kernel/flash attention* and *linear layer encoding* collectively enable the model to achieve

a balance between accuracy and efficiency, outperforming transformer and message-passing baselines in both speed and memory usage.

Scalability Analysis. As illustrated in Figure 4.3, the evaluation in all four datasets demonstrates that both FATH and FATH-FA exhibit consistently better performance as the dimension of the model increases. This trend indicates that FATH-based models are more effective than baselines in exploiting additional representational capacity, resulting in superior predictive performance. Among the datasets, *RetailRocket* achieves the highest NDCG scores and shows rapid early gains for lower dimensions (64 and 256), suggesting that even modest embedding sizes are sufficient to capture most of the predictive signals. As discussed in Chapter 2, the user–item interaction sequences in the *RetailRocket* dataset are relatively short, and the dataset itself is smaller in scale. It comprises three types of user behaviour, allowing the models to capture relevant behavioural patterns with higher accuracy. In contrast, the other datasets (*Tianchi*, *Taobao*, and *IJCAI*) also exhibit scalability improvements for FATH, FATH-FA, and baseline models, although with diminishing returns at higher dimensions. This trend can be attributed to their larger dataset sizes, longer interaction sequences, and increased behavioural complexity, as each contains four behaviour types rather than three. Performance gains are observed across all datasets, with FATH and FATH-FA models achieving the highest overall scalability due to their more regular embedding structures and computationally efficient attention mechanisms, outperforming both message-passing and transformer-based baselines.

4.4 Chapter Summary

In this chapter, the FATH and FATH-FA models framework for multi-behaviour recommendation are introduced. The permutation-invariant layers ensure that these higher-order embeddings maintain discriminative information across all nodes. As a result, the model achieves robust and expressive representations without the typical

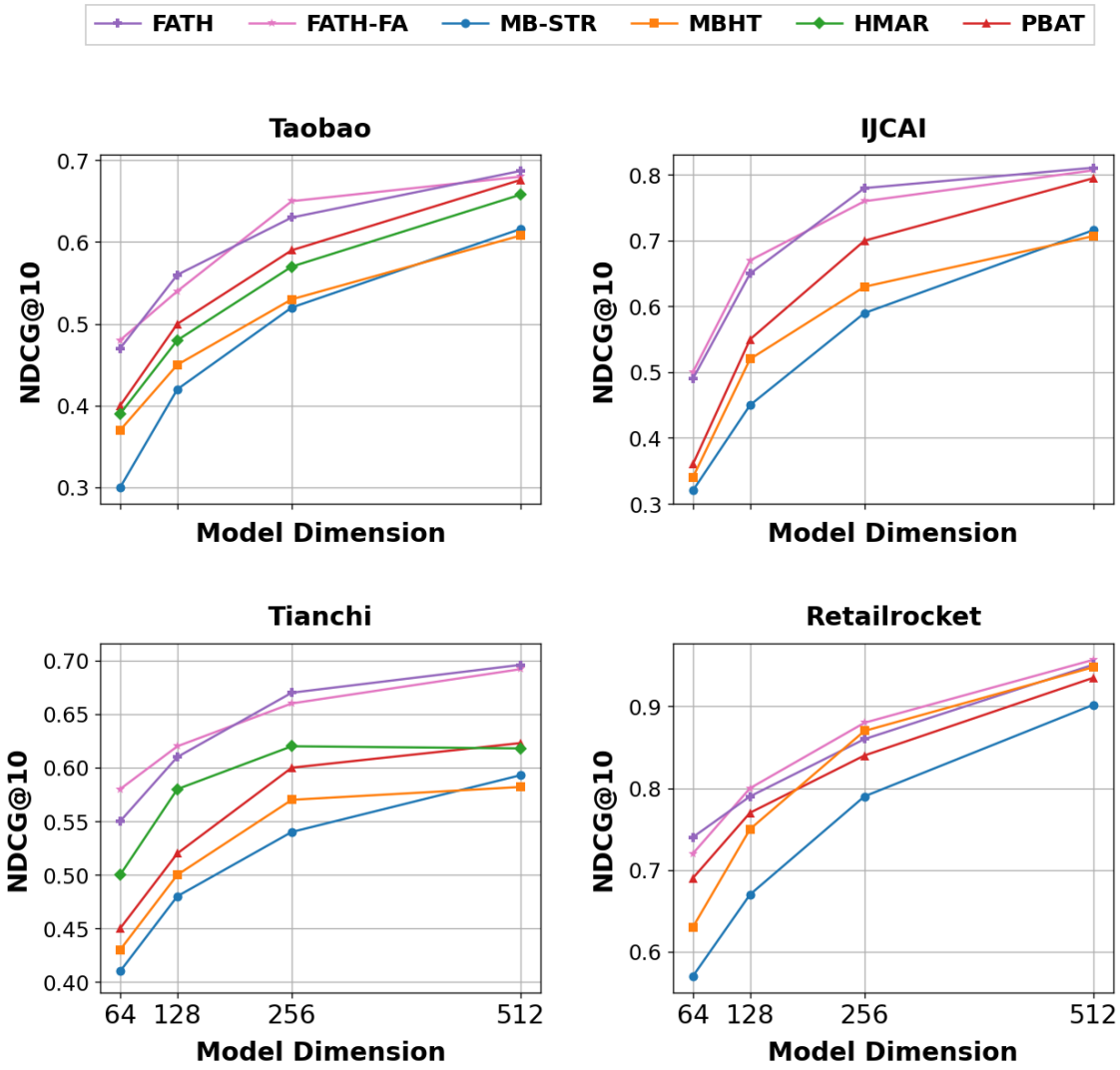

Figure 4.3: Scalability performance evaluation of the FATH model and its variants.

degradation associated with deep graph propagation. The integration of hardware-efficient attention mechanisms, including chunk-wise tiling, has a significant impact on both the scalability and efficiency of the model. This shows that efficient hardware-software integration not only reduces resource usage, but also stabilises the training process by minimising bottlenecks during attention computation. In summary, the FATH and FATH-FA models outperform the baseline models in Section 3.2 in both time and memory efficiency without compromising the model performance. Flash attention (FATH-FA) reduces memory accesses and computational bottlenecks, resulting in faster runtime and more efficient memory usage.

The consistent efficiency of FATH and FATH-FA across datasets underscores

their scalability for real-world applications where computational resources are constrained. The balance between high representational capacity and low training cost makes these models suitable for deployment in production-scale recommendation environments. Furthermore, the relatively modest GPU memory footprint observed in FATH-FA demonstrates its potential for distributed or on-device inference, and real-time RSs that require low-latency processing, robust performance, and lower computational cost.

Chapter 5

Hyperbolic Representation Learning for Multi-Behaviour Recommendation

In **Chapter 3**, we observed that the NDCG performance of the GTRec and STMBR models declines on the larger datasets, IJCAI and Taobao, compared to the smaller Retailrocket dataset. This degradation can primarily be attributed to **oversmoothing**, **oversquashing**, and the **irregular structure of hypergraph data encountered in practical scenarios**. **In Chapter 4**, the FATH and FATH-FA models are proposed to address these three issues, resulting in improvements in accuracy and efficiency compared to the baseline methods introduced in Chapter 2.

In this chapter, we shift focus to the use of hyperbolic geometry for *hypergraph representation learning*. The motivation stems from the observation that real-world recommendation data often exhibits hierarchical structures (e.g., category (Clothing) → subcategory (Jacket) → item (Black Biker Jacket)) [146] . However, traditional message-passing HGNNs repeatedly average information across nodes, causing embeddings to collapse into overly similar vectors, leading to oversmoothing, oversquashing, and ultimately degraded model performance [192].

To address these challenges, we propose *hyperbolic representation learning*, where

embeddings are mapped into a hyperbolic space. This geometry naturally preserves both hierarchical order and relative distances, while Lorentz transformations [85] with learnable curvature ensure that embeddings remain well separated [319]. Furthermore, we incorporate linear attention to improve efficiency, enabling scalable training while maintaining the representational benefits of hyperbolic embeddings. Formally, this aligns with our research question.

> ***RQ2.2:*** *HGNN-message-passing is prone to oversmoothing and oversquashing as the size of the graph increases. Can the limitations of HGNNs be addressed through hyperbolic geometry by better modelling hierarchical data?*

In addressing **RQ 2.2**, *HyperRec*, a novel hyperbolic approach based on the Lorentz model [85] is introduced. HyperRec applies a linear self-attention mechanism to capture complex relationships between multiple user behaviours in a unified framework. Experimental results demonstrate the effectiveness and scalability of HyperRec in large-scale multi-behaviour recommendation.

5.1 Introduction

Real-world data, particularly in domains such as e-commerce and social networks, often exhibits hierarchical structures [29]. For example, a product catalogue may have categories such as *Clothing*, subcategories such as *Jackets*, and individual items such as *Black Biker Jacket*. Traditional HGNNs rely on message-passing to propagate information across nodes [148]. Although effective, repeated averaging in message-passing often leads to **oversmoothing** [330] and **oversquashing** [44], where node embeddings collapse to similar vectors, reducing their discriminative power and degrading model performance [214].

In multi-behaviour RSs, user-item interactions also consist of various behaviours, such as clicks, views, ratings, and purchases [142]. These interactions form a complex dynamic representation of user preferences, often exhibiting hierarchical or tree-

like structures [52]. Hyperbolic geometry has shown potential for modelling such data due to its superior ability to represent hierarchical relationships compared to Euclidean space [146]. As illustrated in Figure 1.1, user behaviours are not only interconnected but also hierarchically organised. This makes hyperbolic geometry a natural choice for modelling multi-behaviour recommendation tasks, as capturing such underlying structures can lead to improved prediction accuracy [158]. Hyperbolic models have shown promise in effectively capturing hierarchical and tree-like structures [306], making them suitable for modelling multi-behaviour interactions in RSs [29]. Various methods leveraging hyperbolic geometry have been developed, achieving competitive performance in domains such as computer vision, NLP, and graph learning [33, 85]. Moreover, self-attention methods in non-Euclidean spaces have been explored [28, 73, 214]. However, these approaches often face challenges related to efficiency and scalability, such as high memory and time complexity [45].

To address these limitations, we propose a **Hyper**bolic **Rec**ommedation model, **HyperRec**, which preserves hierarchical relationships and efficiently captures complex user-item interactions. To enhance the efficiency of the model, we apply frequency domain kernelisation with the Discrete Cosine Transform (DCT) [24], as an alternative to *softmax* operation in the attention mechanism, which is incorporated in another variant of the model: **HyperRec-S**. Experimental results demonstrate that HyperRec achieves competitive efficiency in terms of GPU memory usage compared to baseline methods.

5.1.1 Foundational Concepts

Hyperbolic Geometry: Hyperbolic space provides a continuous curved space that can naturally embed hierarchical and tree-like structures. Unlike Euclidean space, distances in hyperbolic space grow exponentially with the radius, allowing embeddings to remain well separated even for nodes far apart in the hierarchy.

Lorentz Model: We apply the Lorentz model of hyperbolic space, which enables transformations that maintain the relative order and distances between nodes

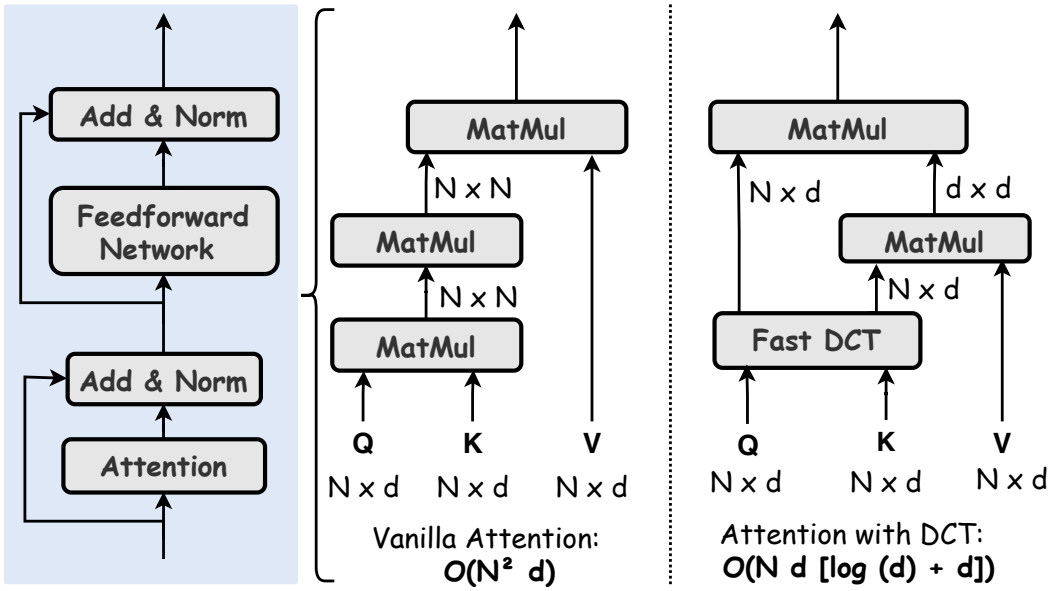

Figure 5.1: Comparison of the standard tranformer self-attention ($O(N^2d)$) and Fast DCT-based attention ($O(Nd[log(d) + d])$) applied in HyperRec-S and HyperRec respectively, showing the difference in computational complexity as illustrated by [24]. N represents the sequence length and d is the embedding dimension.

[139]. This is crucial for preserving hierarchical relationships while avoiding embedding collapse, allowing compact, low-distortion embeddings of hierarchical user-item relationships, allowing more accurate preference modelling and better relevance in recommendations [131].

Linear Attention: Standard attention mechanisms incur *quadratic complexity* with respect to the number of nodes, which is inefficient for large-scale graphs [232]. To address this issue, we apply a *linear attention* mechanism in *HyperRec*, reducing computational complexity from $O(n^2)$ to $O(n)$ while maintaining high-quality representations for downstream recommendation tasks.

5.2 Related Work

Recent research in recommender systems has explored the use of hyperbolic geometry to better model complex user-item interactions [146, 226, 319]. These works show the utility of hyperbolic space in improving representation capacity and preserving latent hierarchical structures that are common in real-world scenarios. For

example, Li et al. [131] introduces HNCR, a **H**yperbolic **N**eural **C**ollaborative **R**ecommendation model that applies a *deep representation learning* framework in hyperbolic space, using semantic neighbour sets to capture mutual semantics between users and items more effectively than Euclidean models. In related research, Yang et al. [299] propose HCRF, a **H**yperbolic **R**egularisation **C**ollaborative **F**iltering model that applies geometry-aware regularisation to address the over-smoothing issue in hyperbolic aggregation, improving the discriminative capacity of user and item representations. In addition, Yang et al. [306] introduce HGSR, a **H**yperbolic **G**raph Learning **So**cial **R**ecommendation model that uses a heterogeneous graph based on user-item interactions and social networks for hyperbolic social recommendation.Another model, HGFormer [304] also addresses the limitations of traditional Euclidean transformers in modelling hierarchical and graph-structured data, using curvature-aware positional encoding and hyperbolic message aggregation in order to preserve both local and global graph structures. Hypergraph-based hyperbolic RSs. Li et al. [148] further demonstrate that hyperbolic geometry outperforms traditional Euclidean baselines on large-scale datasets. For example, Hyperbolic Graph Convolutional Filtering (HGCF) [225] extends graph convolutional approaches to the hyperbolic space, allowing for more expressive modelling of higher-order relationships in recommendation. Similarly, the HyperRec model introduced in this chapter applies *hyperbolic representation learning* to improve higher-order user-item interaction modelling. However, unlike most existing methods, efficiency issues that arise with modelling the complex user-item interactions are addressed without compromising model performance.

5.3 Methodology

The architecture of the HyperRec framework is illustrated in Figure 5.2. User-item interaction data is mapped to the Lorentz model [85] using an exponential map to represent each interaction type. The data is then processed through hyperbolic transformation, positional encoding, LayerNorm and activation functions

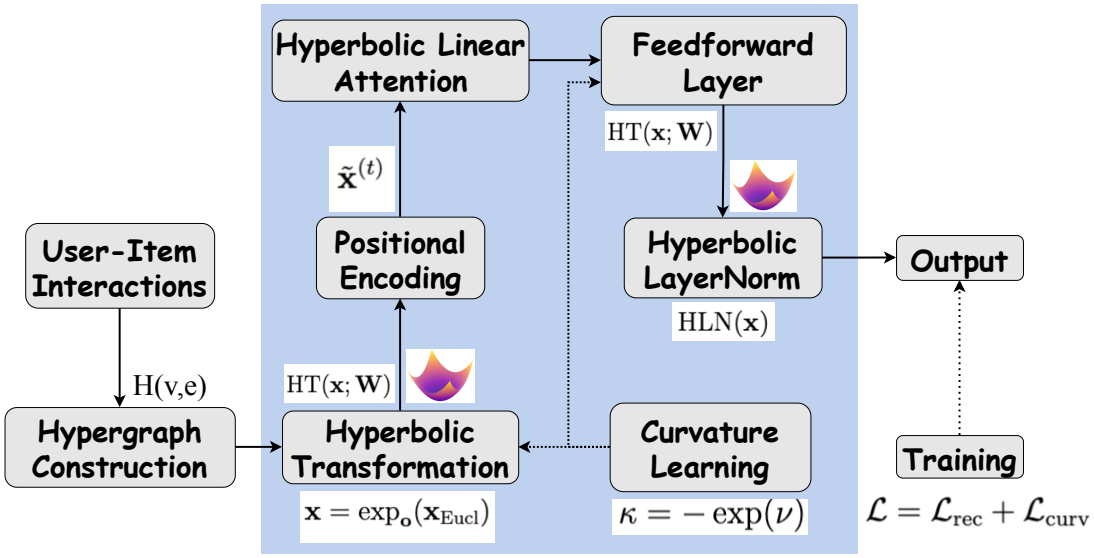

Figure 5.2: Architecture of the HyperRec model. User-item interactions are represented by a hypergraph to capture complex dependencies. These interactions are transformed into hyperbolic space for effective representation of hierarchical structures. Positional encoding is added to enhance context-awareness. An efficient kernel attention mechanism is applied to improve computational efficiency. The outputs from the attention mechanism pass through a feedforward layer and a hyperbolic LayerNorm module, producing the final recommendations.

[123]. To capture hypergraph-based relationships, such as user-user or item-item interactions, an HGNN is integrated into the transformer encoder. The encoder processes user-item interactions with hyperbolic self-attention, applies positional encoding and layer normalisation, and then outputs embeddings that are used directly for prediction. The training loop processes input through the encoder, applies self-attention and normalisation, and then computes the cross-entropy loss.

Inspired by [298], HyperRec integrates *hyperbolic representation learning* with scalable attention mechanisms to address the challenges of HGNNs. **Hyperbolic transformation** facilitates transformations between hyperbolic spaces of different curvatures while preserving the relative distances between the data points. **Hyperbolic linear attention** addresses scalability issues by implementing a linear complexity **encoder-only** self-attention mechanism in hyperbolic space, which allows efficient processing of large-scale graphs.

Hypergraph-Enhanced Lorentz Model

Hypergraph Construction. Hyperedges are constructed to capture multi-behaviour interactions. A hyperedge connects a user and all items interacted and also links items co-interacted by multiple users (e.g. items frequently purchased together). The hypergraph is encoded by the incidence matrix:

$$H(v,e) = \begin{cases} 1, & \text{if node } v \in \text{hyperedge } e \\ 0, & \text{otherwise} \end{cases} \tag{5.1}$$

The incidence matrix $H \in \mathbb{R}^{|V|\times|E|}$ is normalised to avoid degree bias:

$$\tilde{H}(v,e) = \frac{H(v,e)}{\sqrt{\sum_{v'\in e} H(v',e)} \cdot \sqrt{\sum_{e'\ni v} H(v,e')}} \tag{5.2}$$

Lorentzian Hyperbolic Embeddings. Users and items are embedded in the Lorentz hyperbolic space model, denoted by $\mathcal{L}^{d,\kappa}$, which consists of points $\mathbf{x} \in \mathbb{R}^{d+1}$ lying on a hyperboloid defined by the constraint $\langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = \frac{1}{\kappa}$ with the time-like coordinate $x_t > 0$ to select a unique sheet. The Lorentzian inner product $\langle \mathbf{x}, \mathbf{y} \rangle_{\mathcal{L}} = -x_t y_t + \mathbf{x}_s^\top \mathbf{y}_s$ has a signature $(- + \cdots +)$, where x_t is the time-like component and $\mathbf{x}_s$ are the space-like components, inducing hyperbolic geometry in this manifold. The curvature $\kappa < 0$ is parameterised as $\kappa = -\exp(\nu)$ with $\nu \in \mathbb{R}$ learned during training, ensuring negative curvature and allowing the model to capture hierarchical relational structures more efficiently.

$$\mathcal{L}^{d,\kappa} := \left\{ \mathbf{x} \in \mathbb{R}^{d+1} \,\middle|\, \langle \mathbf{x}, \mathbf{x} \rangle_{\mathcal{L}} = \frac{1}{\kappa}, \quad x_t > 0 \right\} \tag{5.3}$$

Users and items are initialised via exponential mapping from Euclidean to hyperbolic space:

$$\mathbf{x} = \exp_{\mathbf{o}}(\mathbf{x}_{\text{Eucl}}) = \left(\sqrt{-\frac{1}{\kappa}} \cosh\left(\|\mathbf{x}_{\text{Eucl}}\| \sqrt{-\kappa} \right), \sqrt{-\frac{1}{\kappa}} \sinh\left(\|\mathbf{x}_{\text{Eucl}}\| \sqrt{-\kappa} \right) \frac{\mathbf{x}_{\text{Eucl}}}{\|\mathbf{x}_{\text{Eucl}}\|} \right) \tag{5.4}$$

where $\mathbf{x}_{\text{Eucl}} \sim \mathcal{N}(0, 0.01)$ and $\mathbf{o} = (\sqrt{-1/\kappa}, \mathbf{0})$ is the origin. The curvature $\kappa = -\exp(\nu)$ is learned via $\nu \in \mathbb{R}$ to adapt to hierarchical structures.

Hyperbolic Linear Transformation. We apply Hyperbolic Transformation (HT) [298] to enable linear transformations in hyperbolic space without frequent mappings to tangent space, reducing the computational overhead. Curvature κ is learned through parameterisation $\kappa = -\exp(\nu)$, with $\nu \in \mathbb{R}$, which allows $\kappa < 0$ to model hierarchical structure effectively. The transformation avoids tangent space projections by operating directly in $\mathcal{L}^{d,\kappa}$:

$$\text{HT}(\mathbf{x}; \mathbf{W}) := \begin{bmatrix} \sqrt{\|\mathbf{W}^\top \mathbf{x}_s + \mathbf{b}\|^2 - \frac{1}{\kappa}} \\ \mathbf{W}^\top \mathbf{x}_s + \mathbf{b} \end{bmatrix}, \quad \text{with } \mathbf{W} \in \mathbb{R}^{d \times d}, \mathbf{b} \in \mathbb{R}^d \tag{5.5}$$

, where$\mathbf{W} \in \mathbb{R}^{d \times d}$ denotes the learnable weight matrix and $\mathbf{b} \in \mathbb{R}^d$ denotes the learnable bias vector of the hyperbolic transformation.

Hyperbolic Attention

Two variants of attention mechanisms are applied in the Lorentzian hyperbolic space for the HyperRec model: *linear attention* and *softmax attention*.

Hyperbolic Linear Attention. The default implementation, used in HyperRec, is hyperbolic linear attention aimed at improving efficiency and scalability in large-scale settings. A linear hyperbolic attention mechanism is applied that is based on the fast discrete cosine transform [24], reducing the complexity to $O(Nd)$, where d is the embedding dimension. Given the embeddings of input tokens $\mathbf{X} \in \mathcal{L}_{N \times d, \kappa_1}$ in the Lorentz space with curvature κ_1, the query, key and value matrices are obtained through hyperbolic transformations such as $\mathbf{Q}_s = \text{HT}(\mathbf{X}; \mathbf{W}_Q)_s$, $\mathbf{K}_s = \text{HT}(\mathbf{X}; \mathbf{W}_K)_s$ and $\mathbf{V}_s = \text{HT}(\mathbf{X}; \mathbf{W}_V)_s$, where $\text{HT}(\cdot; \mathbf{W})_s$ denotes the space-like component of the

Lorentz transformation. The attention output is computed as:

$$\mathbf{Z}_s = \frac{\mathbf{Q}_s(\mathbf{K}_s^\top \mathbf{V}_s)}{\mathbf{Q}_s(\mathbf{K}_s^\top \mathbf{1})}, \tag{5.6}$$

where $\mathbf{Q}_s, \mathbf{K}_s, \mathbf{V}_s \in \mathbb{R}^{N\times d}$ are the space-like query, key and value matrices respectively. The numerator captures the projected value representations modulated by hyperbolic key-query interactions, while the denominator provides normalisation analogous to the *softmax* function. This formulation enables scalable attention in hyperbolic space with $O(Nd)$ time complexity, preserving geometric consistency and efficient processing of long user-item interaction sequences.

Algorithm 3 Hyperbolic Linear Attention

1: **Input**: $\mathbf{X} \in \mathcal{L}^{N\times d,\kappa}$, weights $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V$
2: $\mathbf{Q}_s, \mathbf{K}_s, \mathbf{V}_s \leftarrow \mathrm{HT}(\mathbf{X}; \mathbf{W}_Q)_s, \mathrm{HT}(\mathbf{X}; \mathbf{W}_K)_s, \mathrm{HT}(\mathbf{X}; \mathbf{W}_V)_s$
3: $\mathbf{A} \leftarrow \mathbf{Q}_s(\mathbf{K}_s^\top \mathbf{V}_s)/(\mathbf{Q}_s(\mathbf{K}_s^\top \mathbf{1}) + \epsilon)$, $\epsilon = 1e-5$
4: **Return**: $\mathbf{A}$ (projected back to $\mathcal{L}^{d,\kappa}$ via $\exp_{\mathbf{o}}$)

HyperRec-S: Softmax Attention. Another model variant, **HyperRec-S** is proposed, this computes attention scores based on Lorentzian inner products between transformed query and key vectors, normalised via the *softmax* function:

$$\alpha_{ij} = \mathrm{softmax}\left(\frac{\langle \mathbf{q}_i, \mathbf{k}_j \rangle_{\mathcal{L}}}{\tau}\right), \quad \tau = \sqrt{d} \tag{5.7}$$

where $\mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^{d+1}$ are Lorentz-transformed embeddings and $\langle \cdot, \cdot \rangle_{\mathcal{L}}$ denotes the Lorentzian inner product with temperature τ. Although this *softmax-based* formulation enables a more expressive representation of hierarchical dependencies, it incurs higher time and memory complexity with respect to the number of nodes, which translates into higher computational requirements [352].

Hyperbolic Positional Encoding. In multi-behaviour recommendation tasks, positional encoding helps capture sequential dependencies in user interactions. Po-

sitional vectors $\mathbf{p}_i \in \mathbb{R}^d$ are learned and injected via tangent space addition:

$$\tilde{\mathbf{x}}^{(t)} = \exp_{\mathbf{x}^{(t)}}\left(\epsilon \cdot \log_{\mathbf{x}^{(t)}}(\mathbf{p}_i)\right), \quad \epsilon = 0.01 \tag{5.8}$$

where $\mathbf{o} = \left(\sqrt{-\frac{1}{\kappa}}, \mathbf{0}\right)$ is the Lorentzian origin, $\mathbf{p}$ is the positional encoding vector in the tangent space, and ϵ controls the encoding. This ensures that positional information is added via tangent-space operations, preserving hyperbolic consistency.

LayerNorm and Activation in Hyperbolic Space To ensures Lorentz-norm consistency, layer normalisation and nonlinear activation are adapted to hyperbolic space through a single-curvature hyperbolic normalisation (HLN) that operates on the space-like components:

$$\mathrm{HLN}(\mathbf{x}) = \begin{bmatrix} \sqrt{\|\mathbf{x}'_s\|^2 - \frac{1}{\kappa}} \\ \mathbf{x}'_s \end{bmatrix}, \quad \mathbf{x}'_s = \frac{\mathbf{x}_s - \mu}{\sigma + \epsilon} \odot \gamma + \beta \tag{5.9}$$

where μ, σ are mean and standard deviation, and γ, β are learnable. LN denotes standard layer normalisation applied to the space-like vector $\mathbf{x}_s$, and κ is the single learned curvature. $\mathbf{x}_s$ denotes the input vector to be normalised, representing the features of a single element in a sequence. μ is the mean of the elements in $\mathbf{x}_s$, computed over its feature dimension. σ is the standard deviation of the elements in $\mathbf{x}_s$, also calculated over its feature dimension, with a constant ϵ added for numerical stability. γ is a learnable scaling parameter that has the same shape as $\mathbf{x}_s$ and allows the model to modulate the normalised output. β is a learnable bias parameter, also of the same shape as $\mathbf{x}_s$, used to shift the normalised output.

5.3.1 Model Training

The training objective is to minimise the *cross-entropy loss* combined with a curvature regularisation term on a single learnable curvature κ.

Cross Entropy Loss. For each behaviour $b \in \mathcal{B}$, the model predicts the probability of interaction $\hat{y}^b_{u,i}$ between the user u and the item i. The cross-entropy loss is defined as:

$$\mathcal{L}_{\text{rec}} = -\sum_{b \in \mathcal{B}} w_b \sum_{(u,i) \in \mathcal{D}} y^b_{u,i} \log(\hat{y}^b_{u,i}), \tag{5.10}$$

where w_b balances the importance of different behaviours and $\mathcal{D}$ is the dataset. In our experiments, *purchase* prediction is the target behaviour.

Curvature Regularisation. To stabilise the learned geometry, we apply L_2 regularisation to the curvature parameter ν through $\mathcal{L}_{\text{curv}} = \lambda\nu^2$, preventing degenerate Euclidean solutions ($\kappa \to 0$) while maintaining flexibility for *representation learning*. A regularisation term is added to stabilise the learnable curvature parameter ν from which κ is calculated as $\kappa = -\exp(\nu)$:

$$\mathcal{L}_{\text{curv}} = \lambda\nu^2, \tag{5.11}$$

where λ controls the regularisation. The total loss combines the recommendation loss and curvature regularisation for end-to-end optimisation:

$$\mathcal{L} = \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{curv}}. \tag{5.12}$$

Training Protocol. The model is trained using an optimised configuration determined by grid search, with key hyperparameters including an embedding dimension $d = 256$, batch size of 512, dropout rate of 0.3, and 2 attention layers. We employ negative sampling with a ratio of 5 negative items per positive interaction, uniformly drawn from unobserved pairs. Optimisation is performed using the Riemannian Adam [10], which extends the Adam optimiser to hyperbolic space, with a learning rate of 0.005 and a weight decay of 1×10^{-4} for regularisation.

Algorithm 4 Training Algorithm for HyperRec

Require: Dataset $\mathcal{D}$ of user-item interactions, behaviour set $\mathcal{B}$, learning rate η, epochs T, regularisation coefficient λ.
Ensure: Trained model parameters $\Theta = \{\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V, \mathbf{W}_{\text{FFN}}, \nu\}$.

```
Initialise model parameters Θ.
for t = 1 to T do
    Sample mini-batch B_batch ⊆ D.
    Compute curvature κ = − exp(ν).
    Map batch B_batch to Lorentz space L^{d,κ} via exponential map at origin o.
    Compute encoder outputs using unified hyperbolic transformation HT(·).
    Compute self-attention on space components (Eq. for Q_s, K_s, V_s and Z_s).
    Apply hyperbolic layer normalisation and feed-forward networks.
    Predict interactions ŷ^b_{u,i} using Lorentzian inner products of user and item embeddings.
    Compute L_rec and L_curv.
    Update Θ ← Θ − η∇_Θ L.
end for
return Trained parameters Θ.
```

Hyperparameter	**Value**
Embedding dimension (d)	256
Batch size	512
Dropout rate	0.3
Attention layers	2
Negative samples	5
Learning rate	0.005
Optimiser	AdamW
Weight decay	1×10^{-4}
Curvature reg. (λ)	0.1

Table 5.1: Final hyperparameters after grid search.

5.4 Experiments

We use the same datasets and evaluation framework as in Chapter 3. The HyperRec and HyperRec-S models are evaluated against baseline models that show the highest performance in Chapters 3 and 4: MBHT, MB-STR, HMAR and PBAT.

5.4.1 Results and Analysis

Results and Analysis. Table 5.2 shows that the proposed models, **HyperRec** and its variant **HyperRec-S**, consistently outperform baseline approaches, includ-

Dataset	Model	HR@5(↑)	HR@10(↑)	NDCG@5(↑)	NDCG@10(↑)	MRR(↑)
Retailrocket	MB-STR	0.914	0.905	0.928	0.902	0.913
	MBHT	0.926	0.927	0.929	0.948	0.928
	PBAT	0.932	<u>0.945</u>	0.924	0.935	0.931
	HyperRec	**0.946**	**0.954**	**0.957**	**0.956**	**0.939**
	HyperRec-S	<u>0.935</u>	0.941	<u>0.952</u>	<u>0.938</u>	<u>0.933</u>
Taobao	MB-STR	0.694	0.776	0.596	0.616	0.675
	MBHT	0.682	0.768	0.594	0.607	0.264
	HMAR	0.692	0.819	0.598	0.658	0.673
	PBAT	0.737	0.805	0.650	0.676	0.758
	HyperRec	**0.768***	**0.846***	**0.689***	**0.705***	**0.802***
	HyperRec-S	<u>0.757</u>	<u>0.838</u>	<u>0.682</u>	<u>0.691</u>	<u>0.788</u>
IJCAI	MB-STR	0.802	0.881	0.695	0.716	0.778
	MBHT	0.776	0.857	0.678	0.707	0.772
	PBAT	<u>0.874</u>	0.914	<u>0.794</u>	<u>0.799</u>	0.867
	HyperRec	**0.879***	**0.924***	**0.801***	**0.815***	**0.888***
	HyperRec-S	0.870	<u>0.919</u>	0.792	0.789	<u>0.885</u>
Tianchi	MB-STR	0.746	0.756	0.496	0.599	0.727
	MBHT	0.769	0.782	0.488	0.582	0.714
	HMAR	0.775	0.795	0.495	0.618	0.739
	PBAT	0.787	0.823	0.587	0.623	0.771
	HyperRec	**0.804***	**0.831***	**0.694***	**0.711***	**0.799***
	HyperRec-S	<u>0.798</u>	<u>0.827</u>	<u>0.685</u>	<u>0.702</u>	<u>0.792</u>

Table 5.2: Performance evaluation across datasets. Best performances are indicated in bold, showing relative improvement over the best-performing baseline at a 0.05 significance level with paired t-test. Underlined values indicate the second-best performances.

ing PBAT, MBHT and HMAR. These results highlight the robustness and effectiveness of HyperRec across datasets with varying levels of sparsity and sequence length. The following discussion provides dataset-specific insights into the performance and characteristics of these models.

Dataset-Specific Insights. On the **Retailrocket** dataset, which is relatively sparse and has short sequences, HyperRec achieves the best overall performance, outperforming PBAT by a substantial margin. HyperRec-S also maintains strong performance. This highlights the ability of these models to effectively model user-item interactions in sparse settings through hyperbolic embeddings that capture higher-order dependencies. On the **Taobao** dataset, which has higher density than Retailrocket, HyperRec records significant performance gains compared to baseline models such as HMAR and PBAT. **HyperRec-S** also shows the same trend as on

the **Retailrocket** dataset, closely matching the performance of **HyperRec**. Similarly, on the **IJCAI**, which has more than 36M interactions, HyperRec achieves the highest scores in all metrics, particularly in the NDCG and MRR metrics. These results demonstrate the effectiveness of hyperbolic embedding space in capturing complex user-item interactions, which often include hierarchical user/item relationships and long-range dependencies. Finally, on the **Tianchi** dataset, HyperRec also achieves superior results, improving HR and NDCG scores compared to baseline models.

Discussion. Experimental evaluation of the HyperRec model demonstrates that the integration of a hyperbolic framework with transformer-based attention improves the accuracy of recommendations across datasets of diverse density and sequence length. From these results, we deduce that hyperbolic embeddings enable the model to capture complex user-item interactions more effectively than traditional Euclidean approaches, as indicated by the superior performance of HyperRec compared to the baseline models. HyperRec-S also maintains competitive accuracy and generally outperforms baseline models, but at higher computational cost. These results validate the proposed hyperbolic framework as an effective *representation learning* approach for large-scale RSs.

	Taobao				**IJCAI**			
Model Variant	**HR@5**	**NDCG@5**	**HR@10**	**NDCG@10**	**HR@5**	**NDCG@5**	**HR@10**	**NDCG@10**
w/o HT	0.735	0.664	0.821	0.686	0.770	0.687	0.834	0.733
w/o DCT	0.729	0.651	0.837	0.659	0.758	0.712	0.817	0.755
w/o Positional Encoding	0.712	0.667	0.802	0.689	0.736	0.675	0.779	0.738
w/o Attention	0.706	0.626	0.803	0.634	0.713	0.694	0.796	0.720
HyperRec	**0.768**	**0.689**	**0.846**	**0.705**	**0.879**	**0.801**	**0.924**	**0.815**

Table 5.3: Ablation results of HyperRec on the Taobao and IJCAI datasets. Removal of the HT, DCT and positional encoding components leads to performance degradation compared to the full HyperRec model.

Ablation Study. Tables 5.3 and 5.4 present the ablation results of the HyperRec model. To evaluate the contribution of different architectural components, we remove three key modules: HT, DCT and positional encoding. The results indicate

	Retailrocket				Tianchi			
Model Variant	**HR@5**	**NDCG@5**	**HR@10**	**NDCG@10**	**HR@5**	**NDCG@5**	**HR@10**	**NDCG@10**
w/o HT	0.854	0.860	0.886	0.875	0.745	0.648	0.789	0.662
w/o DCT	0.867	0.836	0.869	0.762	0.721	0.624	0.726	0.639
w/o Positional Encoding	0.859	0.892	0.883	0.867	0.727	0.631	0.795	0.671
w/o Attention	0.831	0.809	0.827	0.725	0.702	0.607	0.711	0.623
HyperRec	**0.946***	**0.957***	**0.954**	**0.956***	**0.8048**	**0.694***	**0.831***	**0.711***

Table 5.4: Ablation results of HyperRec on the Retailrocket and Tianchi datasets. Removing HT, DCT or positional encoding components degrades performance compared to the full HyperRec model.

that each of these components contributes to the overall performance of the model.

Removing the **HT** component results in a consistent performance decline in all metrics. We attribute this decline to the model reverting to the Euclidean embedding space, thus limiting its ability to preserve complex relational information. Similarly, excluding the **DCT** component from the attention module results in a noticeable degradation of model performance, although to a lesser extent than omitting the attention module entirely. Removal of the **attention module** from the HyperRec model results in significant performance degradation, as indicated by a drop in HR and NDCG performance on the four datasets. This underscores the role of the attention mechanism in capturing complex dependencies and interactions within the data. Another observation is that removal of **positional encoding** results in a degradation in the accuracy of the model. The role of positional encoding is to preserve the sequential order of interactions (e.g., *view* $\rightarrow$ *add-to-cart* $\rightarrow$ *purchase*). The complete HyperRec model outperforms all ablated variants, confirming the robustness of these findings across datasets of different scales and sparsity levels.

These results underscore that the effectiveness of HyperRec comes from the complementary contributions of its components. HT provides hierarchical capacity, mitigates oversmoothing and oversquashing, DCT improves efficiency. Their integration allows HyperRec to achieve consistent and substantial performance, validating the design choices of the proposed architecture.

Efficiency Evaluation. We perform additional experiments to assess how the model scales with different batch sizes. As illustrated in Figure 5.3, memory usage

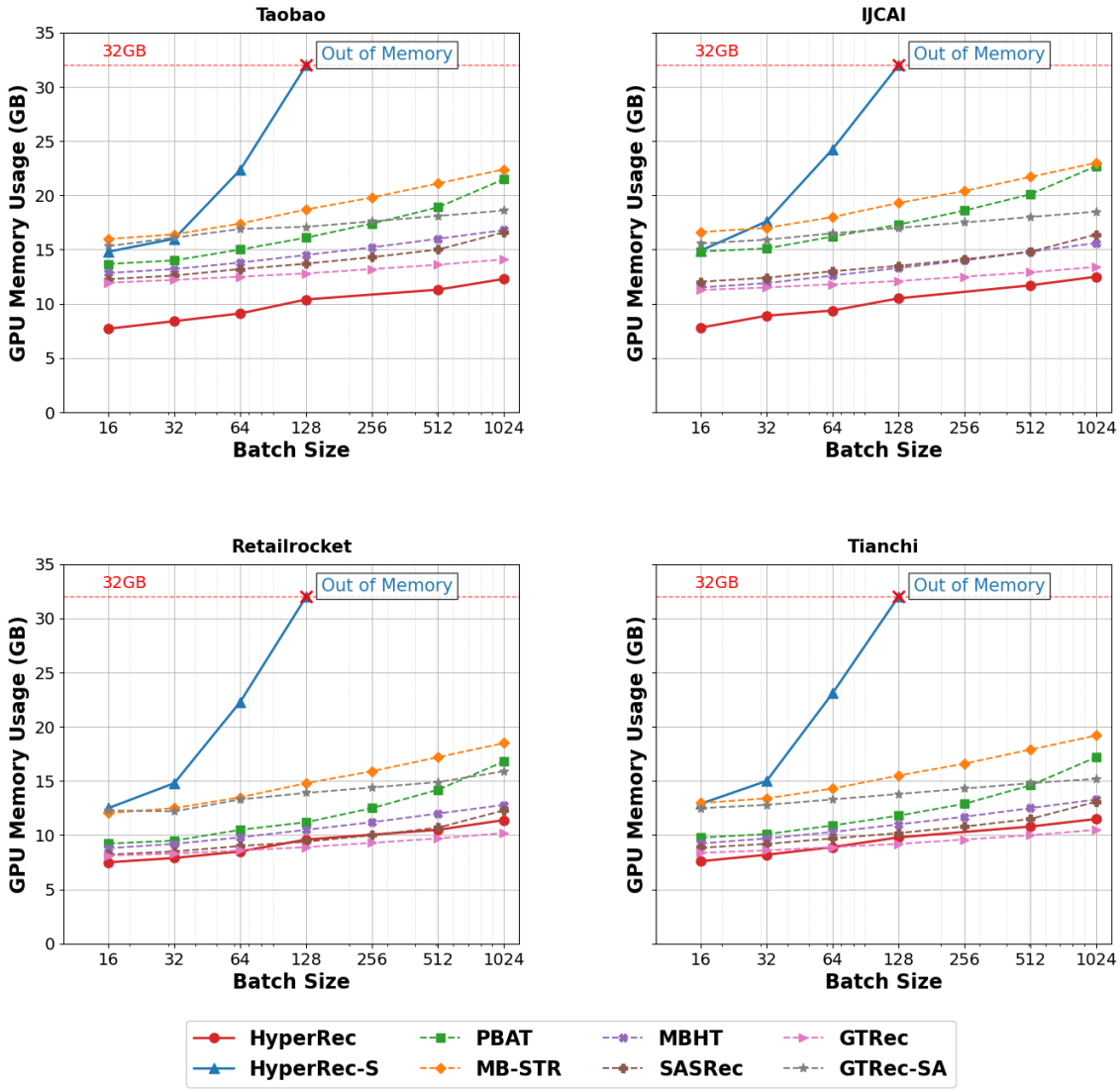

Figure 5.3: Efficiency Evaluation of HyperRec. Memory usage comparison of HyperRec linear and *softmax* attention across batch sizes on the four datasets. Hyperbolic *softmax* attention encounters "Out of Memory" errors.

graphs in the Taobao, IJCAI, Retailrocket and Tianchi datasets provide a comprehensive view of scalability and efficiency between the variants of the HyperRec model and baselines under increasing batch sizes. A clear trend emerges: HyperRec with *softmax* attention consumes significantly more memory compared to HyperRec, and encounters "Out-of-Memory (OOM)" errors. HyperRec scales much better across all datasets, maintaining memory efficiency even at the largest batch sizes tested.

These results imply that linear attention not only improves scalability but also makes the model more efficient for large-scale recommendation tasks where efficiency in terms of computational resources is a priority. HyperRec achieves a balance be-

tween ranking performance and efficiency, which is beneficial in real-world scenarios that require high accuracy and resource efficiency. Moreover, the efficiency advantage paves the way for extending HyperRec to streaming RSs, where low-latency inference and adaptability to continuous user interaction are essential.

5.5 Chapter Summary

In this chapter, the challenge of learning effective representations of hierarchical data, such as *category–subcategory–item* structures, is addressed. MPHGNN-based RS methods often face challenges such as oversmoothing and oversquashing in large-scale settings, as discussed in Chapters 1 and 2. We attribute the performance decline of the GTRec and STMBR models on the larger IJCAI and Taobao datasets, compared to the smaller Retailrocket dataset, in part to these limitations. To address these challenges, **a non-message-passing model, HyperRec**, is introduced. HyperRec leverages hyperbolic representations together with a linear attention mechanism, enabling it to scale effectively while maintaining representational quality. Furthermore, HyperRec's hyperbolic embedding space provides a natural way to capture hierarchical relationships with low distortion [28], which allows compact representation of complex hierarchies in hypergraph data,involving higher-order dependencies [146]. In addition, curvature learning enables adaptive geometry that fits the data, a flexibility that is absent in standard HGNN methods that operate in fixed Euclidean spaces [139]. Finally, the unified encoder-only architecture integrates multi-behaviour hypergraph information seamlessly without requiring separate propagation for each relation type, simplifying the model architecture.

Chapter 6

Towards Efficient Hypergraph and MoA-Enhanced Recommender Systems

User preferences and item characteristics change over time, making it important for RSs to adapt to these dynamics [312]. From the literature survey in **Chapter 2**, it is identified that most existing HGNN-based methods model user-item interactions as static. However, many real-world scenarios exhibit complex dynamics, making static *representation learning* models inadequate for effectively capturing temporal evolutionary information [53]. Consequently, these models, including the ones introduced in **Chapters 3, 4** and **5** of this thesis, often yield suboptimal performance when applied to these practical scenarios [161]. Most existing works which aim to address this issue apply generative methods such as LLM-based RS models [59]. However, despite their abilities, LLM-based methods are computationally expensive and prone to hallucination, often generating inaccurate or misleading recommendations [331]. This brings us to **RQ3**, which is addressed in this chapter.

> ***RQ3****: Most real-world recommendation scenarios exhibit complex and evolving user-item interaction, which makes static HGNN-based models insufficient to capture temporal patterns effectively. Recent methods that*

attempt to address this limitation are often computationally intensive. This raises the question: Can a hypergraph-enhanced MoA solution offer a more effective alternative to model temporal dynamics in RS?

To address **RQ3**, **HGLMRec**, an MoA enhanced RS is introduced, that incorporates a hypergraph encoder designed to capture complex dynamic relationships between users and items. The encoder works in conjunction with an MoA framework. Instead of feeding raw, unstructured user-item interaction data directly to the MoA framework, only the most relevant nodes and hyperedges are retrieved, thereby reducing computational overhead while providing an enriched retrieval context.

6.1 Introduction

In real-world recommendation scenarios, user preferences evolve continuously over time, leading to complex and dynamic interaction patterns [163]. Static *representation learning* models, such as HGNN-based models, often struggle to capture these dynamics as they assume a static structure and ignore the time-dependent nature of user-item interactions [339]. With the recent surge in LLMs, a new paradigm in RSs has emerged that combines information retrieval with LLMs to produce contextually relevant recommendations [26, 140, 184, 216]. Generative recommendation models have shown benefits, such as semantic understanding and interactive reasoning, which can improve the relevance and quality of recommendations by generating output that aligns with user preferences [143, 250].

Despite these advances, current LLM-based recommendation approaches face two significant limitations. First, hallucination, where the model generates inaccurate or misleading recommendations, can compromise the reliability of the system [87, 97, 103]. Secondly, the high computational cost, resulting from the need to search through large vocabularies or fine-tune LLMs on domain-specific data, renders these methods impractical for real-time or large-scale deployment [49, 257, 308]. Although pretraining or fine-tuning LLMs in recommendation-specific datasets can

improve performance, these strategies require substantial computational resources [149], domain expertise, and large volumes of high-quality data, further complicating real-world implementation [347]. Consequently, efficient recommendation models are needed that can adapt to evolving user preferences and dynamic interaction patterns while maintaining high accuracy [83, 273].

Motivated by these challenges, we explore whether *hypergraph representation learning* can be harnessed to improve recommendation performance in dynamic, multi-behaviour scenarios. We propose **HGLMRec**, a novel framework that integrates an HGNN encoder with an MoA architecture. The central idea of HGLMRec is to enhance modelling of user-item interactions, allowing the model to capture higher-order dependencies across multiple behaviours. Hyperedges in this representation connect a user with multiple items and behavioural types, generating dense token embeddings that encode both local and global preference patterns. These embeddings are then processed by the MoA framework, which employs multiple specialised agents to refine recommendations.

Our **contributions** are as follows. We apply an HGNN encoder to capture complex, higher-order interactions that static pairwise GNNs often miss, and integrate embeddings across multiple behaviours to preserve the richness of user-item interactions. Inspired by the work in [247], we pass the HGNN and user prompt tokens through an MoA framework that learns dynamic patterns in user behaviour. We hypothesise that this combination allows HGLMRec to achieve expressive *representation learning* while remaining computationally efficient, making it suitable for large-scale deployment. Experiments on three real-world datasets demonstrate that our proposed model, HGLMRec, outperforms state-of-the-art baselines, including several models based on LLMs, despite using a lightweight architecture.

6.2 Related Work

Graph and LLM Integration. The integration of graph structures with LLMs has gained significant attention because of their complementary strengths. Graphs

explicitly encode relationships between entities, offering a structured representation that enhances reasoning and reduces the risk of hallucination in LLMs [55]. LLMs, on the other hand, excel in understanding and generating unstructured text, but lack mechanisms for directly incorporating graph-based knowledge [273]. Previous studies have explored the use of knowledge graphs with LLM for tasks such as node classification, link prediction and question answering [174]. These approaches often rely on fine-tuned LLM instruction to align graph-based reasoning with task objectives [339]. However, they are limited by their reliance on static graphs and their inability to take advantage of the rich relational information captured by hypergraphs [167]. In addressing this issue, Fatemi et al. [55] propose GraphToken, a parameter-efficient technique that augments textual prompts with learned tokens that represent graph structures, to encode structured data for LLM. Instead of fine-tuning the entire LLM, GraphToken trains a small set of additional parameters that encode the structure of the input data, allowing the LLM to better understand the data without extensive fine-tuning [193]. This enables efficient integration of structured data with LLMs, preserving their reasoning capabilities and expanding their applicability to graph-based tasks [331]. To address the limitations of HGNN-based RS methods in dynamic scenarios, our HGLMRec methodology extends this concept to RS settings [193].

6.3 Methodology

HGLMRec integrates three key concepts,*higher-order modelling, MoA*and *iterative refinement* to model complex user-item interactions.

Higher-Order Modelling. HGLMRec applies HGNN encoding to capture historical user-item interactions. Unlike traditional graphs, hypergraphs connect multiple nodes through hyperedges, enabling higher-order interaction modelling across items and behaviours. This overcomes the pairwise limitations of standard GNNs [142]. User–item interactions are converted into token embeddings that preserve both se-

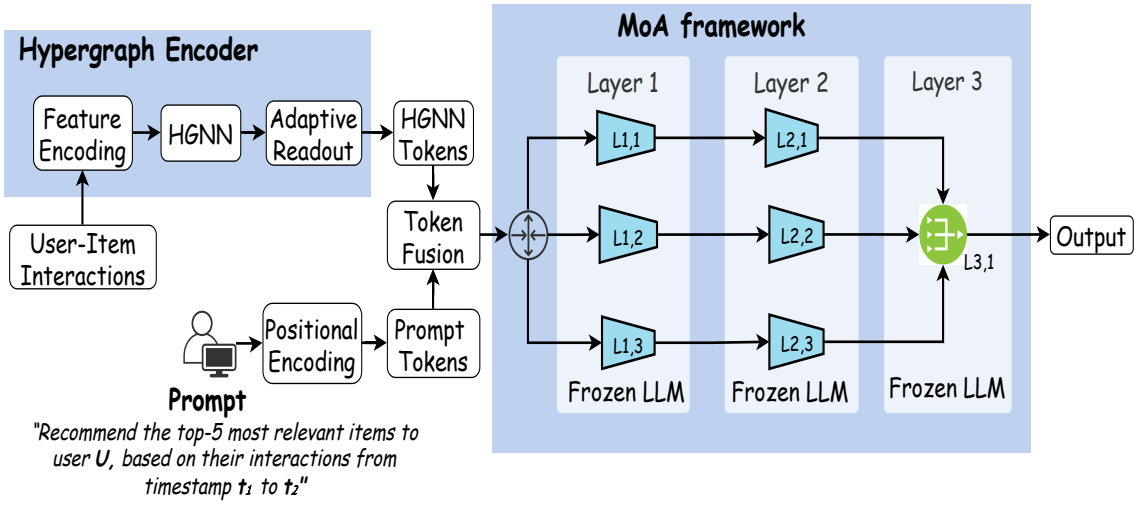

Figure 6.1: End-to-end architecture of HGLMRec. The model processes user-item interactions through a hypergraph encoder, fuses graph tokens with task prompts, and refines recommendations via a 3-layer MoA framework.

mantic and structural information, providing compact expressive representations for downstream processing.

Mixture of Agents. Inspired by [247], MoA employs multiple specialised agents to independently refine token embeddings. This hierarchical design aims to capture both fine-grained and high-level user preferences, improve recommendation accuracy and mitigate hallucinations.

Interactive Refinement and Adaptability. HGLMRec is designed to adapt to evolving user behaviours through its multi-agent refinement process. Combined with hypergraph encoding, the model aims to capture diverse, multi-behaviour interactions, with a goal to generate context-aware recommendations that address limitations of GNNs [176].

6.3.1 HGLMRec Architecture

The architecture of HGLMRec, illustrated in Figure 6.1, consists of three interconnected stages. The first stage, hypergraph encoding, constructs hypergraph representations of multi-behaviour interactions and generates initial token embeddings.

These embeddings are then passed through the token fusion stage, which integrates information across behaviours to produce unified representations that encode both fine-grained and global patterns of user-item interactions. Finally, hierarchical MoA processing refines these fused embeddings through multiple specialised agents, with dynamic weighting to produce the final recommendation scores.

HGLMRec is designed to overcome the limitations of static HGNNs and effectively models complex multi-behaviour dependencies. The hierarchical MoA framework refines embeddings through specialised agents, reducing hallucinations commonly observed in generative LLM-based approaches and improving recommendation reliability. Moreover, the lightweight design of both the hypergraph encoder and the MoA components allows efficient training and inference, offering a computationally feasible alternative to large-scale LLM-based recommendation models. Figure 6.1 illustrates the architecture, which consists of three key stages: (1) Hypergraph Encoding, (2) Token Fusion, and (3) Hierarchical MoA Processing.

6.3.2 Hypergraph Encoder

Given a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the node set $\mathcal{V} = U \cup I$ consists of users U and items I, and the hyperedges $\mathcal{E}$ capture multi-behaviour interactions such as views, cart additions and purchases, the hypergraph encoder produces compact user-item representations through the following steps:

Feature Initialisation. Each node $v \in \mathcal{V}$ is initialised with a learnable embedding vector:

$$\mathbf{h}_v^{(0)} = \begin{cases} \mathbf{E}_u[v] \in \mathbb{R}^d & \text{if } v \in U, \\ \mathbf{E}_i[v] \in \mathbb{R}^d & \text{if } v \in I, \end{cases} \tag{6.1}$$

where $\mathbf{E}_u$ and $\mathbf{E}_i$ are embedding lookup tables for users and items, respectively, and d is the embedding dimension.

Hypergraph Convolution. To capture higher-order user-item interactions, we apply two layers of hypergraph convolution. At each layer $l \in \{0, 1\}$, the node features are updated by aggregating the normalised messages from all hyperedges:

$$\mathbf{h}_v^{(l+1)} = \text{LayerNorm}\left(\sigma\left(\sum_{e \in \mathcal{E}(v)} \frac{1}{|e|} \sum_{u \in e} \mathbf{h}_u^{(l)} \mathbf{W}^{(l)}\right)\right), \tag{6.2}$$

where $\mathcal{E}(v)$ is the set of hyperedges containing node v, $|e|$ is the size of hyperedge e, $\mathbf{W}^{(l)} \in \mathbb{R}^{d \times d}$ is a learnable weight matrix, σ is the ReLU activation function, and *LayerNorm* stabilises training.

Adaptive Readout. To improve the *representation learning* of user-item interactions from the HGNN module, we apply adaptive readout [16], a function that aggregates the HGNN embeddings [287]. In HGLMRec, readout is applied using an attention-weighted grouping (Equation 6.3) that retains expressive hypergraph summaries tailored to the particular interactions for the particular iteraction.

Token Generation. After two convolutional layers, node embeddings $\{\mathbf{h}_v^{(l)}\}_{v \in \mathcal{V}}$ are pooled by leveraging *adaptive readout* to form compact graph tokens:

$$\alpha_v = \frac{\exp\left(\mathbf{a}^\top \tanh\left(\mathbf{W}_a \, \mathbf{h}_v^{(l)}\right)\right)}{\sum_{u \in \mathcal{V}} \exp\left(\mathbf{a}^\top \tanh\left(\mathbf{W}_a \, \mathbf{h}_u^{(l)}\right)\right)}, \quad \mathbf{G} = \text{MLP}\Big(\sum_{v \in \mathcal{V}} \alpha_v \, \mathbf{h}_v^{(l)}\Big) \tag{6.3}$$

where α_v are attention weights learned via $\mathbf{W}_a$ and $\mathbf{a}$, and the MLP ensures flexible mapping of aggregated features.

6.3.3 Token Fusion

To align graph-based and prompt signals, HGLMRec fuses $\mathbf{G}$ with a tokenised task prompt. We apply token fusion by concatenation [135] to the model that combines structured graph tokens with the recommendation task prompt. Specifically, a prompt such as "*Recommend the top-5 most relevant items for user U, based on their interactions from timestamp t2 to t1*", is tokenised, which converts the text

into a sequence of token embeddings $\mathbf{P} \in \mathbb{R}^{m \times d}$, where m is the number of prompt tokens and d is the embedding dimension. These prompt embeddings $\mathbf{P}$ are then concatenated with hypergraph tokens $\mathbf{G} \in \mathbb{R}^{k \times d}$, which summarise user-item interactions learned from the hypergraph encoder. To retain the position information of the tokens in the combined sequence, the positional encoding $\mathbf{P}_{\text{pos}} \in \mathbb{R}^{(k+m) \times d}$ is applied. The fused input tokens are then passed into the downstream MoA agents, which aligns HGNN and the recommendation task representations in a shared embedding space with positional context.

$$\mathbf{x}_1 = \text{Concat}(\mathbf{G}, \mathbf{P}) + \mathbf{P}_{\text{pos}} \tag{6.4}$$

.

Equation (6.4) fuses HGNN tokens $\mathbf{G} \in \mathbb{R}^{k \times d}$ with prompt tokens $\mathbf{P} \in \mathbb{R}^{m \times d}$ to obtain unified token embeddings $\mathbf{x}_1 \in \mathbb{R}^{(k+m) \times d}$. Concatenation preserves all token-level information from both sources while maintaining the embedding dimension d consistent for downstream processing. Positional encoding $\mathbf{P}_{\text{pos}} \in \mathbb{R}^{(k+m) \times d}$ ensures that the model retains information about the relative positions of the tokens within the HGNN and prompt sequences, allowing for downstream MoA tasks.

6.3.4 MoA Framework

The MoA module processes the fused input tokens to iteratively refine the recommendation predictions. Specifically, the MoA consists of multiple layers, each containing several agents denoted by $A_{i,j}$, where i indexes the layer and j indexes the agent within that layer. At each layer i, the input token representation $x_i \in \mathbb{R}^{m \times d}$ is processed in parallel by all n agents $A_{i,1}, \ldots, A_{i,n}$. Each agent $A_{i,j}(\cdot)$ is a "frozen" LLM that produces refined token embeddings. The outputs of all agents in the layer are combined using a cross-agent attention-based aggregation operator, denoted by $\bigoplus$. This aggregated output is then combined with the initial fused input tokens $\mathbf{x}_1$ through a residual connection, producing the intermediate representation y_i, which

serves as input x_{i+1} for the next layer.

$$y_i = \bigoplus_{j=1}^{n} A_{i,j}(x_i) + \mathbf{x}_1, \quad \text{and} \quad x_{i+1} = y_i. \tag{6.5}$$

Hypergraph Encoding (Algorithm 5). User-item interactions are into hypergraph tokens using a two-layer HGNN.

Algorithm 5 Hypergraph Encoder

Require: Hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, user/item sets U, I
Ensure: Graph tokens $\mathbf{G} \in \mathbb{R}^{k \times d}$
Initialise embeddings: $\mathbf{E}_u \in \mathbb{R}^{|U| \times d}$, $\mathbf{E}_i \in \mathbb{R}^{|I| \times d}$
for each node $v \in \mathcal{V}$ **do**
    $\mathbf{h}_v^{(0)} \leftarrow \mathbf{E}_u[v]$ if $v \in U$; else $\mathbf{E}_i[v]$
end for
for $l = 0$ to 1 **do**
    for each node $v \in \mathcal{V}$ **do**
        $\mathbf{z}_v \leftarrow \sum_{e \in \mathcal{E}(v)} \frac{1}{|e|} \sum_{u \in e} \mathbf{h}_u^{(l)} \mathbf{W}^{(l)}$
        $\mathbf{h}_v^{(l+1)} \leftarrow \text{LayerNorm}(\text{ReLU}(\mathbf{z}_v))$
    end for
end for
$\mathbf{G} \leftarrow \text{MLP}(\text{MeanPool}(\{\mathbf{h}_v^{(2)}\}_{v \in \mathcal{V}}))$
return G

6.3.5 Training Procedure

The training objective is to minimise the cross-entropy loss, which measures the discrepancy between predicted model predictions and ground truth. The loss function is defined as:

$$\mathcal{L} = -\frac{1}{N} \sum_{u=1}^{N} \sum_{b \in B} \sum_{i=1}^{I} r_{ui}^b \log \hat{r}_{ui}^b + \lambda \|\Theta\|_2^2 \tag{6.6}$$

where, N denotes the total number of users, I is the number of items, and B is the set of interaction behaviours such as *view*, *cart*, or *purchase*. The term $r_{ui}^b \in \{0, 1\}$ represents the ground truth label, while $\hat{r}_{ui}^b \in (0, 1)$ is the prediction of the model. The parameter set Θ includes all trainable components of the model - such as user and item embeddings, HGNN weights, and MLP parameters - and λ is a regularisation coefficient that controls the penalty L_2 $\|\Theta\|_2^2$ to prevent overfitting.

Prompt-Guided Reasoning (Algorithm 6). Fuses graph tokens with prompt embeddings and passes them through a three-layer MoA architecture.

Algorithm 6 MoA Framework

Require: Graph tokens $\mathbf{G}$, prompt T, agents $\{A_{i,j}\}$
Ensure: Predicted scores $\mathbf{y}_{\text{final}} \in \mathbb{R}^{|I|}$

$\mathbf{P} \leftarrow \text{MoA\_Tokenizer}(T)$
$\mathbf{x}_1 \leftarrow \text{Concat}(\mathbf{G}, \mathbf{P}) + \text{PE}()$
for $i = 1$ to 3 **do**
  $n \leftarrow \{3, 3, 1\}[i]$
  for $j = 1$ to n **do**
    $\mathbf{o}_j \leftarrow A_{i,j}(\mathbf{x}_i)$
  end for
  if $i < 3$ **then**
    $\mathbf{y}_i \leftarrow \text{MultiHeadAttention}(\mathbf{x}_i, \{\mathbf{o}_j\}) + \mathbf{x}_1$
    $\mathbf{x}_{i+1} \leftarrow \mathbf{y}_i$
  else
    $\mathbf{y}_{\text{final}} \leftarrow \text{Softmax}(\text{MLP}(\mathbf{o}_1))$
  end if
end for
return $\mathbf{y}_{\text{final}}$

Optimisation (Algorithm 7). Hypergraph Encoder Training.

Algorithm 7 Encoder Training

Require: Dataset $\mathcal{D}$, epochs E, batch size B, learning rate η
Ensure: Trained parameters Θ

Initialise AdamW optimiser with linear warmup
for epoch $= 1$ to E **do**
  for each batch $(G, T, A) \in \mathcal{D}$ **do**
    $\mathbf{G} \leftarrow$ Algorithm 5(G)
    $\hat{A} \leftarrow$ Algorithm 6$(\mathbf{G}, T)$
    Compute loss: $\mathcal{L}$
    Update: $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}$
  end for
  Log validation metrics
end for

6.4 Experiments

Datasets. For experiments, we use three real-world datasets that contain four types of interactions , *purchase, add-to-favourites, add-to-cart*, and *view*: **Taobao**

IJCAI and **Tianchi.** These datasets are chosen for evaluation because they directly reflect the challenges outlined in **RQ3**. More details on the datasets are provided in **Chapter 2**.

Implementation Details. In our experiments, the MoA model is configured with Qwen2-7B [296] for the intermediate agent layers and LLaMA-3-8B [50] for the final aggregation layer. Using different large-language models (LLMs) in the intermediate and final layers, we can leverage the strengths of each model, which in turn helps produce more coherent output, an essential factor for generating high-quality recommendations [247]. For the **HGLM-SM** variant, we employ **LLaMA-3-70B** [50] and **GPT-4o** [1], selected for their reasoning abilities [331]. This configuration allows us to evaluate the MoA framework's ability to generate accurate recommendations efficiently with smaller agents, while providing a meaningful baseline comparison against high-capacity models capable of handling more complex interaction patterns.

Hyperparameters. For the **HGNN** encoder, the feature dimension is set to 128, balancing expressiveness and computational efficiency. The number of hypergraph convolution layers is set to 2 to capture high-order dependencies. We apply the AdamW optimiser for training with a learning rate of 5×10^{-4}. Warm-up steps are set to 500 to facilitate stable convergence. The details of the key hyperparameters are found in Table 6.1.

6.4.1 Results and Analysis

Table 6.2 presents the comparison of HGLMRec performance results against several baseline models. Generative methods RLMRec, TRSR, KDA and TALLRec show better performance compared to sequential models MBHT and PBAT, which is attributed to their generative capabilities. HGLMRec consistently outperforms baseline models in the three datasets, which is attributed to the combined benefits of the HGNN and MoA modules. The encoder provides a better capture of the

Hyperparameter	**Value**
HGNN Layers (l)	2
Embedding Dim (d)	128
HGNN Tokens (k)	10
MoA Layers	3
Learning Rate	10^{-3}
Weight Decay	10^{-5}
Optimiser	AdamW
Dropout Rate	0.2
Warmup Steps	500

Table 6.1: HGLMRec hyperparameters applied after grid search

Dataset	**Metric**	MBHT	PBAT	RLMRec	TRSR	KDA	TALLRec	HGLMRec
Taobao	HR@5(↑)	0.687	0.735	0.745	0.784	0.757	<u>0.793</u>	**0.832***
	NDCG@5(↑)	0.590	0.651	0.657	0.668	0.641	<u>0.685</u>	**0.697***
	HR@10(↑)	0.764	0.802	0.828	0.844	0.835	<u>0.849</u>	**0.865***
	NDCG@10(↑)	0.615	0.673	0.676	0.689	0.681	<u>0.708</u>	**0.726***
IJCAI	HR@5 (↑)	0.774	0.871	0.882	<u>0.914</u>	0.878	0.892	**0.919***
	NDCG@5(↑)	0.675	0.781	0.790	0.812	0.784	<u>0.797</u>	**0.814***
	HR@10 (↑)	0.852	<u>0.924</u>	0.913	0.921	0.906	0.917	**0.932***
	NDCG@10(↑)	0.701	<u>0.798</u>	0.791	0.786	0.795	0.790	**0.812***
Tianchi	HR@5 (↑)	0.718	0.739	0.764	<u>0.786</u>	0.773	0.766	**0.795***
	NDCG@5 (↑)	0.536	0.568	0.579	<u>0.584</u>	0.581	0.574	**0.606***
	HR@10 (↑)	0.743	0.784	0.776	<u>0.797</u>	0.799	0.783	**0.814***
	NDCG@10(↑)	0.556	0.573	0.580	0.587	<u>0.601</u>	0.585	**0.613***

Table 6.2: Performance comparison: The best performances indicated in bold show the relative improvement over the best performing baseline at 0.05 significance with paired t-test. Underlined values indicate second best performance.

user-item interactions, while the MoA module improves the ability of the model to capture diverse and complex patterns in user behaviour to enhance the recommendation. The results also show that the MoA model outperforms the models with a single LLM configuration. This performance gain is attributed to the mixture of the MoA agents, each partitioning the cognitive load to improve the accuracy of the recommendations [247].

In dense datasets such as Taobao and Tianchi, HGLMRec achieves substantial performance gains over baseline models. The richer interaction graphs in these datasets provide more relational and behavioural cues, which our hypergraph-based and MoA-enhanced approach can effectively exploit. Consequently, the benefits of higher-order modelling and multi-agent embedding refinement are more fully realised

in these settings. In sparse datasets such as IJCAI, where much of the user–item space remains unobserved, baseline models often struggle with missing information. Here, HGLMRec still yields consistent improvements, albeit in more modest increments. However, the fact that our model does not degrade performance in these sparse settings indicates its robustness to data scarcity.

The MoA framework outperforms single-LLM methods. Table 6.3 shows that MoA provides better recommendation performance compared to configurations using one large LLM. Distributing both the computational and cognitive work among multiple smaller agents enables the MoA module to capture user-item interactions more effectively. Smaller agents also reduce computational overhead, improve scalability, and enable real-time deployment. Our evaluation confirms that the distributed workload in MoA yields better results than a single-LLM configuration.

	Taobao		**IJCAI**		**Tianchi**	
Model	HR@10	N@10	HR@10	N@10	HR@10	N@10
w/o Hypergraph Encoder	0.782	0.645	0.816	0.762	0.749	0.568
HGLM-SM	0.777	0.624	0.782	0.673	0.734	0.571
HGLMRec (1 Layer)	0.712	0.608	0.765	0.614	0.717	0.538
HGLMRec (2 Layers)	0.832	0.679	0.796	0.681	0.775	0.591
HGLMRec (3 Layers)	**0.865***	**0.726***	**0.932***	**0.812***	**0.814***	**0.613***

Table 6.3: Ablation study results: Performance comparison of the full HGLMRec model and its variants.

Ablation Study

The significance of the components of the HGLMRec model is evaluated by conducting experiments that determine their impact on NDCG and HR. The model variant configurations are outlined as follows: *HGLM-SM:* The MoA module is replaced with a single LLM model, *w/o Hypergraph Encoder:* the hypergraph encoder module is removed from the full model. The results summarised in Table 6.3 show that the removal of the indicated components leads to a noticeable performance degra-

dation, highlighting the contribution of each module to the model performance, and the complete model consistently achieves the best results on the three datasets. In the model variant without the hypergraph encoder, performance is degraded across all datasets, which indicates that the module contributes to improving model performance. In the HGLM-SM model variant, performance also decreases compared to the HGLMRec model.

Hypergraph encoder benefits. The encoder in HGLMRec effectively captures heterogeneous user-item interactions and facilitates richer contextual reasoning of the model, as shown by the results in Table 6.3. In addition, the hypergraph representation enhances the MoA module's ability to learn interactions between users and items, allowing for better understanding of the overall context of user behaviour [272, 331]. The connected structure provided by the hypergraph leads to improved model learning of the user's intent, both of which are crucial for effective recommendations.

Performance against the number of agent layers. The performance of HGLMRec is evaluated by varying the number of layers of LLM agents. As shown in Table 6.3, the results show a clear trend: the NDCG score improves as the number of layers increases. Specifically, the model performance at 3 layers is better than variants with 1 or 2 layers. This indicates that adding more layers enhances the model's ability to capture complex patterns in user behaviour, leading to better recommendations.

Efficiency Evaluation

A cost comparison of HGLMRec is evaluated against other baseline models. The cost is calculated based on the pricing information available from API providers such as OpenAI[1] and Together.ai[2]. From the results in Figure 6.2, HGLMRec consistently

[1]https://openai.com/api/pricing.
[2]https://api.together.ai/models

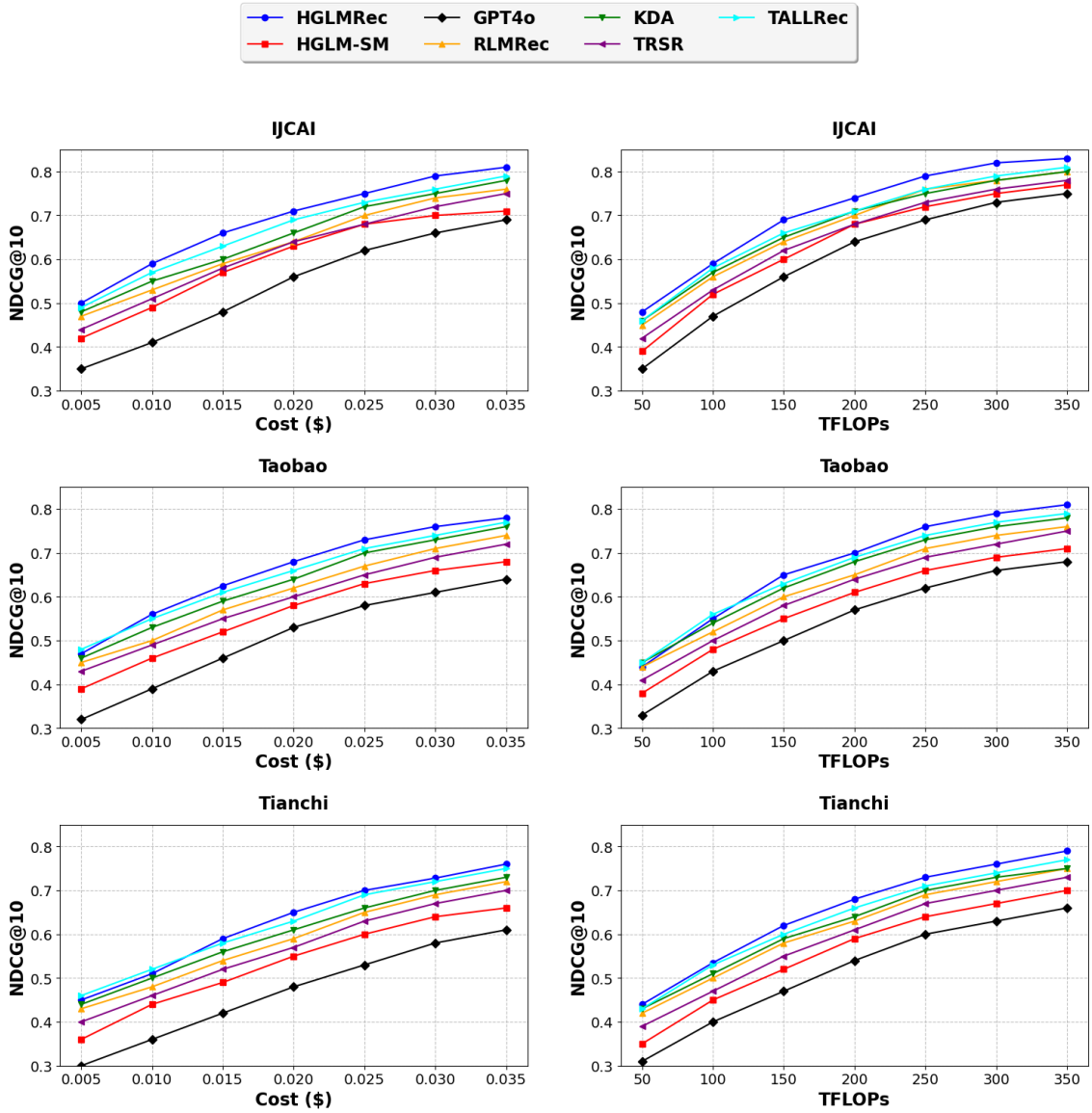

Figure 6.2: Cost and efficiency evaluation of the HGLMRec model. The cost is calculated based on pricing information available on API provider websites.

achieves the best performance and efficiency, achieving the highest NDCG scores at the lowest computational cost. TALLRec and KDA also exhibit strong and stable performance trends, closely following HGLMRec, especially in the mid-to-high resource range. In contrast, GPT4o shows the weakest performance in all datasets and resource levels, indicating limited effectiveness despite increased computational input. HGLMRec outperforms GPT-4o and HGLM-SM in terms of both API cost and TFLOPs, achieving the highest NDCG values at every resource level, indicating superior efficiency at lower computational cost compared to methods with a single LLM configuration.

6.5 Comparative Analysis of The Proposed Models

The proposed models - **GTRec**, **GTRec-DE**, **STMBR**, **FATH**, **FATH-FA**, **HyperRec**, and **HGLMRec** - consistently outperform baseline models (including **MBHT**, **PBAT**, and **HMAR**) across datasets of varying sparsity and sequence length.

GTRec employs a lightweight hypergraph transformer architecture that achieves robust performance with low computational complexity. Its enhanced variant, **GTRec-DE**, integrates differential encoding to further improve accuracy while maintaining efficiency. Another GT-based approach, **STMBR**, introduced in Chapter 3, shows competitive performance with **GTRec** and **GTRec-DE** on metrics such as HR, MRR and NDCG, at lower computational cost. In Chapter 4, the **FATH** and **FATH-FA FATH** models achieve performance comparable to **GTRec** and **STMBR** across all datasets, with greater computational efficiency in terms of runtime and memory usage. Chapter 5 presents the **HyperRec** model, which uses hyperbolic embeddings and linear attention to capture complex hierarchical data. Its variant, **HyperRec-S**, further enhances accuracy at a higher computational cost.

Despite the performance and efficiency advantages demonstrated by the methods proposed in Chapters 3, 4 and 5 - **GTRec, GTRec-DE, STMBR, FATH** and **FATH-FA**, these models remain limited in real-world settings where user preferences continuously evolve [339]. To overcome these limitations, HGLMRec is introduced as an interactive recommendation framework capable of adapting to dynamic user behaviours and issues such as TD, as discussed in Chapter 2. HGLMRec consistently achieves superior accuracy on diverse datasets, including dense (*Taobao*), large-scale (*IJCAI*), and long-sequence (*Tianchi*) settings. These results confirm that the integration of structural reasoning with agentic inference yields enhanced contextual understanding and improved model accuracy. Table 6.4 summarises the qualitative advantages of the methods introduced in this thesis, in terms of their computational efficiency and accuracy across datasets with varying sparsity, density and sequence lengths.

Dataset Name	**Dataset Characteristics**	GTRec	GTRec-DE	STMBR	FATH	FATH-FA	HyperRec	HGLMRec
Retailrocket	Sparse, Short Sequences	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑	–
IJCAI	Sparse, Long Sequences	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑↑
Taobao	Dense, Medium Sequences	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑↑
Tianchi	Dense, Long Sequences	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑	↑↑↑

Table 6.4: Relative performance improvement of the proposed models across datasets of varying sparsity, density, and sequence length. Arrows are colour-coded: ↑ = computation efficiency, ↑ = accuracy, ↑ = dynamic adaptability.

6.6 Chapter Summary

In this chapter, the limitation in modelling dynamic scenarios faced by most HGNN-based RS methods, as discussed in **Chapter 2**, is addressed. The interactions between users and items in the real world continue to evolve, making it important for RSs to adapt to these dynamics [312]. However, most state-of-the-art HGNN-based RS methods, **including those introduced in Chapters 3, 4 and 5**, are limited in effectively modelling these dynamic scenarios due to issues such as **time drift**. To address this issue, HGLMRec is proposed, an RS model that integrates a hypergraph encoder with an MoA framework to learn complex user-item interac-

tions. HGLMRec demonstrates the advantages of combining structured user-item interaction data with "frozen" LLM agents, opening up new possibilities for future research and applications in this area. Experimental results show that HGLMRec outperforms state-of-the-art baselines, achieving better performance at lower computational cost. In summary, HGLMRec presents a novel and practical approach to interactive RSs, capturing complex user-item interactions with high accuracy and lower computational (API, TFLOPs) cost. Finally, Table 6.4 summarises the qualitative benefits of the methods introduced in this thesis.

Chapter 7

Conclusion

7.1 Revisiting Research Questions and Findings

In this chapter, the contributions of this thesis are summarised. **Chapter 2** introduces graph neural networks, hyperbolic geometry, transformers and the useful background knowledge of RSs that is required to follow the thesis. **Chapter 3** introduces two lightweight graph transformer-based methods for RSs **GTRec** and **STMBR**, which enable effective modelling of heterogeneous user-item interactions while improving computational efficiency. This chapter addresses the first research question of this thesis (**RQ1**). In **Chapter 4**, the problem of computational bottlenecks (memory and runtime) caused by the irregular structure of user-item interactions (**RQ2.1**) is addressed. A hardware-algorithm co-design approach is proposed for HGNN-based RSs, with the goal to improve efficiency, scalability and performance in real-world settings. The **FATH** framework, that does not apply the GNN message-passing scheme in *representation learning*, is introduced. In **Chapter 5**, the application of hyperbolic geometry as another alternative to the message-passing scheme is studied for heterogeneous recommendation, and the **HyperRec** model is proposed, which shows competitive performance and efficiency compared to state-of-the-art methods, addressing **RQ2.2**. In **Chapter 6**, the **HGLMRec**, a model that collaboratively applies hypergraph and MoA frameworks, is proposed to address efficiency issues in practical recommendation scenarios. Empirical evaluation shows

that this architecture outperforms single large LLM configurations in recommendation tasks with significant computational cost benefits. This chapter addresses **RQ3**. The summary of the thesis contributions is shown in Table 7.1.

Challenge(s)	**Proposed Solution(s)**	**Finding(s)**
RQ1: Model compression in GNN / transformer - based models often leads to performance losses. Can we address this issue using lightweight graph transformer approaches without degradation of recommendation performance?	**Chapter 3**: Lightweight graph transformers for recommendation performance and efficiency improvement. **GTRec:** applies a **hub and spoke** mechanism and **STMBR:** combines **Sortnet** and **Sparse Sinkhorn** algorithms.	Learnable pattern embedding with modular local and global attention improves performance while maintaining computational feasibility. However, GNN methods show degraded performance as the graph size increases, which is attributed to oversmoothing and oversquashing issues in large-scale settings.
RQ2.1: Graph structure is irregular, which leads to memory bottlenecks. Can we optimise the computational pipeline by adapting to the GPU memory architecture?	**Chapter 4**: **FATH framework:** Hardware-algorithm co-design to regularise the input structure with the goal of optimizing memory usage and reducing runtime.	Hardware-algorithm co-design using flash attention improves efficiency with competitive model performance against state-of-the-art baselines. FATH performance gains are attributed to node-to-node and node-to-edge attention.
RQ2.2: Message-passing GNN methods are prone to oversmoothing and oversquashing in large-scale settings as the size of the graph increases. Can we address this issue using non-message-passing methods?	**Chapter 5: HypRec:** Hypergraph representation learning using hyperbolic geometry as an alternative to GNN message-passing. Softmax and DCT attention mechanisms are applied comparatively.	Hyperbolic representation learning with linear DCT attention improves model performance with significant efficiency gains as the dataset size increases. Integration of softmax attention, however, increases computational cost.
RQ3: Static HGNN-based RS methods struggle to effectively capture dynamic scenarios. Most LLM-based RS methods that aim to address this issue have high computational cost (of pre-training, fine-tuning and API pricing) which is prohibitive in practical scenarios.	**Chapter 6**: **HGLM-Rec:** Hypergraph-enhanced MoA RS framework that collaborates with a hypergraph encoder designed to address halucination issues and capture complex dynamic user-item interactions.	HGLMRec shows significant computational savings with competitive performance. Most RS methods that use LLM, however, remain computationally expensive compared to predictive methods, requiring further research to improve their efficiency.

Table 7.1: Summary of thesis contributions which include challenges addressed, proposed solutions and findings.

7.2 Application Areas and Industrial Impact

The methods proposed in this thesis offer practical benefits, particularly in e-commerce, where personalised recommendations enhance user engagement and drive sales. In addition to performance, efficiency is also essential in real-world RS applications to reduce cloud computing costs and infrastructure demands [182]. Reducing computational costs has a positive environmental impact [128]. Large-scale AI models deployed in data centres consume vast amounts of energy, contributing to significant greenhouse gas emissions [222]. Optimising efficiency aligns with the industry's growing focus on sustainable practices [262], helping to minimise the environmental footprint of AI infrastructure deployment [128]. Some of the methods introduced in this thesis have been implemented in industrial settings[1], resulting in the following research output:

> **Tendai Mukande**, Dimiter Dinkov, Riccardo Superbo and Noel E. O'Connor. **"Accelerating Workflows in Video Game Localisation: A Recommender System for Review and Post-Edit Assignments."** *ACM Transactions on Recommender Systems: Special Issue on Recommender Systems in Industry.* 2025.

7.3 Limitations

One limitation of this research lies in the computational resources used for experimentation. All experiments were conducted using single GPU instances, which, while sufficient for the scope of this study, impose constraints on the scalability and complexity of the real-world models that can be explored. Moreover, single-GPU environments limit the exploration of distributed training methods, which are increasingly important in modern RSs [77], particularly for handling large-scale data and real-time inference requirements [283]. Although the findings in this thesis pro-

[1] This work was conducted during industrial placement at KantanAI (`https://www.kantanai.io/`)

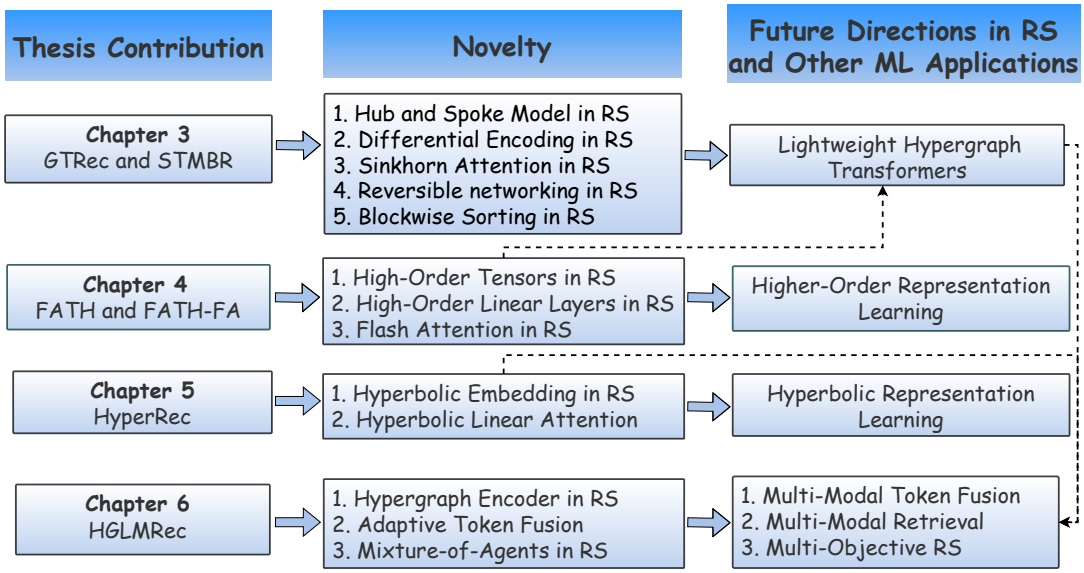

Figure 7.1: Overview of the possible future research directions based on contributions proposed in this thesis.

vide insight within the context of the available resources, future work could address this limitation by the use of multi-GPU setups or cloud-based distributed computing platforms that enable more comprehensive experimentation. These advances could further validate the scalability and efficiency of the proposed methods in real-world high-computational demand scenarios.

7.4 Future Work

This research lays the foundation for numerous avenues of future exploration and development. Building on the proposed methods, future work could focus on addressing the identified limitations and extending the models to other RS or machine learning (ML) applications, as illustrated in Figure 7.1. Other possible research directions include exploring different use cases and addressing ethical issues as discussed in the next paragraphs.

Diversity and Serendipity. One aspect of improving the user experience in RSs lies in fostering diversity and serendipity [201, 354]. Although relevance ensures that recommendations align with user preferences, diversity expands the range of

suggestions to include items from different categories or interests [166, 195]. This prevents users from being trapped in a "filter bubble", where recommendations become too narrow over time. Serendipity, on the other hand, introduces an element of surprise by suggesting items that users may not have actively sought, but are likely to find enjoyable or beneficial [51, 113]. Future work could focus on developing hybrid models that prioritise both exploration and exploitation.

Explainability. Explainability is increasingly recognised as important in RSs, as it promotes transparency, trust and user satisfaction [30, 333]. Users are more likely to accept and act on recommendations when they understand the reasoning behind them. Explainable systems provide insight into why specific items are suggested, which helps users discover new content while feeling confident about the intentions of the platform [241, 294]. To achieve this, the development of interpretable algorithms that integrate user-friendly interfaces to present explanations in a clear and concise manner is essential [30, 282].

Privacy and Security. As personalisation methods become more sophisticated, ensuring user privacy and data security is a concern [155]. RSs often rely on sensitive user data. A balance between personalisation and privacy requires robust encryption methods, anonymisation and adherence to data protection regulations such as GDPR [100, 221]. Privacy-preserving learning approaches, such as federated learning and differential privacy, offer promising directions for future research [93]. These methods allow models to learn from decentralised data sources without compromising the privacy of individual users [341]. In addition, improving security protocols to prevent unauthorised access and data breaches is crucial as RSs become more integral to daily life [92, 100].

Addressing Ethical Issues. Ethical issues in RSs are becoming more significant, especially in large-scale RS settings [112]. For example, addressing bias is an essential research direction, as algorithms can amplify existing inequalities by over-

representing certain groups or perspectives [170, 178]. Ensuring fairness requires continuous evaluation and adjustments to prevent discrimination and promote equitable outcomes for diverse user populations [67, 88]. Moreover, there is a growing need to protect against manipulative practices, such as exploiting users' vulnerabilities for profit [92]. Ethical frameworks should guide the design and implementation of RSs, focusing on values such as inclusivity and accountability [126]. Accountability mechanisms are essential to ensure compliance with relevant data privacy and protection regulations such as GDPR[2] and other applicable laws. In conclusion, collaboration with stakeholders, including users, researchers and policy-makers is necessary to address ethical lapses or unintended consequences. Future research could explore algorithmic auditing tools, fairness-based models and collaborative governance structures to address these challenges.

7.5 Concluding Remarks

In this thesis, HGNN-based RS methods that incorporate efficient self-attention, hyperbolic geometry and hypergraph-enhanced MoA are proposed to capture localised and global patterns, improve efficiency, scalability and recommendation performance. These innovations and provide a foundation for new reseach directions in the RSs landscape, where training and deploying higher-quality models with fewer resources becomes more widely possible.

[2]https://gdpr-info.eu/.

Bibliography

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. "GPT-4 Technical Report". In: *arXiv preprint arXiv:2303.08774* (2023).

[2] Ryan Prescott Adams and Richard S Zemel. "Ranking via Sinkhorn Propagation". In: *arXiv preprint arXiv:1106.1925* (2011).

[3] M Mehdi Afsar, Trafford Crump, and Behrouz Far. "Reinforcement Learning Based Recommender Systems: A Survey". In: *ACM Computing Surveys* 55.7 (2022), pp. 1–38.

[4] Shun-ichi Amari. "Backpropagation and Stochastic Gradient Descent Method". In: *Neurocomputing* 5.4–5 (1993), pp. 185–196.

[5] Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. "A Survey on Hypergraph Representation Learning". In: *ACM Computing Surveys* 56.1 (2023), pp. 1–38.

[6] Ardalan Arabzadeh, Tobias Vente, and Joeran Beel. "Green Recommender Systems: Optimizing Dataset Size for Energy-Efficient Algorithm Performance". In: *International Workshop on Recommender Systems for Sustainability and Social Good*. Springer. 2024, pp. 73–82.

[7] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Henry Mao, Gary Cottrell, and Julian McAuley. "ReZero is All You Need: Fast Convergence at

Large Depth". In: *Uncertainty in Artificial Intelligence*. PMLR. 2021, pp. 1352–1361.

[8] Song Bai, Feihu Zhang, and Philip HS Torr. "Hypergraph Convolution and Hypergraph Attention". In: *Pattern Recognition* 110 (2021), p. 107637.

[9] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. "TallRec: An Effective and Efficient Tuning Framework to Align Large Language Model with Recommendation". In: *arXiv preprint arXiv:2305.00447* (2023).

[10] Gary Bécigneul and Octavian-Eugen Ganea. "Riemannian Adaptive Optimization Methods". In: *arXiv preprint arXiv:1810.00760* (2018).

[11] Iz Beltagy, Matthew E Peters, and Arman Cohan. "Longformer: The Long-Document Transformer". In: *arXiv preprint arXiv:2004.05150* (2020).

[12] Graciela S Birman and Katsumi Nomizu. "Trigonometry in Lorentzian Geometry". In: *The American Mathematical Monthly* 91.9 (1984), pp. 543–549.

[13] Tesfaye Fenta Boka, Zhendong Niu, and Rama Bastola Neupane. "A Survey of Sequential Recommendation Systems: Techniques, Evaluation, and Future Directions". In: *Information Systems* (2024), p. 102427.

[14] Tomer Borreda, Daniel Freedman, and Or Litany. "ReHub: Linear Complexity Graph Transformers with Adaptive Hub-Spoke Reassignment". In: *arXiv preprint arXiv:2412.01519* (2024).

[15] Deborah L Bryan and Morton E O'Kelly. "Hub-and-Spoke Networks in Air Transportation: An Analytical Review". In: *Journal of Regional Science* 39.2 (1999), pp. 275–295.

[16] David Buterez, Jon Paul Janet, Steven J Kiddle, Dino Oglic, and Pietro Liò. "Graph Neural Networks with Adaptive Readouts". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 19746–19758.

[17] Filippo Carnovalini, Antonio Rodà, and Geraint A Wiggins. "Popularity Bias in Recommender Systems: The Search for Fairness in the Long Tail". In: *Information* 16.2 (2025), p. 151.

[18] Mario Casillo, Francesco Colace, Dajana Conte, Marco Lombardi, Domenico Santaniello, and Carmine Valentino. "Context-Aware Recommender Systems and Cultural Heritage: A Survey". In: *Journal of Ambient Intelligence and Humanized Computing* (2021), pp. 1–19.

[19] Huihui Chai, Xiumei Wei, Haoxiang Ma, and Xuesong Jiang. "Knowledge-Enhanced Graph Transformer Network for Multi-Behavior and Item-Knowledge Session-Based Recommendation". In: *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2022, pp. 3421–3426.

[20] Chaofan Chen, Zelei Cheng, Zuotian Li, and Manyi Wang. "Hypergraph Attention Networks". In: *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE. 2020, pp. 1560–1565.

[21] Chong Chen, Weizhi Ma, Min Zhang, Zhaowei Wang, Xiuqiang He, Chenyang Wang, Yiqun Liu, and Shaoping Ma. "Graph Heterogeneous Multi-Relational Recommendation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5. 2021, pp. 3958–3966.

[22] Chonghao Chen, Fei Cai, Wanyu Chen, Jianming Zheng, Xin Zhang, and Aimin Luo. "BP-MoE: Behavior Pattern-Aware Mixture-of-Experts for Temporal Graph Representation Learning". In: *Knowledge-Based Systems* (2024), p. 112056.

[23] Guanzi Chen, Jiying Zhang, Xi Xiao, and Yang Li. "Preventing Over-Smoothing for Hypergraph Neural Networks". In: *arXiv preprint arXiv:2203.17159* (2022).

[24] Hanting Chen, Zhicheng Liu, Xutao Wang, Yuchuan Tian, and Yunhe Wang. "DiJiang: Efficient Large Language Models Through Compact Kernelization". In: *arXiv preprint arXiv:2403.19928* (2024).

[25] Jiaxing Chen, Hongzhi Liu, Yingpeng Du, Zekai Wang, Yang Song, and Zhonghai Wu. "Sequential Hypergraph Convolution Network for Next Item Recommendation". In: *International Conference on Database Systems for Advanced Applications*. Springer. 2023, pp. 389–405.

[26] Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, et al. "When Large Language Models Meet Personalization: Perspectives of Challenges and Opportunities". In: *World Wide Web* 27.4 (2024), p. 42.

[27] Shi Chen, Jingyu Liu, and Li Shen. "A Survey on Graph Neural Network Acceleration: A Hardware Perspective". In: *Chinese Journal of Electronics* 33.3 (2024), pp. 601–622.

[28] Weize Chen, Xu Han, Yankai Lin, Hexu Zhao, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. "Fully Hyperbolic Neural Networks". In: *arXiv preprint arXiv:2105.14686* (2021).

[29] Yankai Chen, Menglin Yang, Yingxue Zhang, Mengchen Zhao, Ziqiao Meng, Jianye Hao, and Irwin King. "Modeling Scale-Free Graphs With Hyperbolic Geometry for Knowledge-Aware Recommendation". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 94–102.

[30] Zirui Chen, Xin Wang, Chenxu Wang, and Jianxin Li. "Explainable Link Prediction in Knowledge Hypergraphs". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 262–271.

[31] Dian Cheng, Jiawei Chen, Wenjun Peng, Wenqin Ye, Fuyu Lv, Tao Zhuang, Xiaoyi Zeng, and Xiangnan He. "IHGNN: Interactive Hypergraph Neural Network for Personalized Product Search". In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 256–265.

[32] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. "Generating Long Sequences with Sparse Transformers". In: *arXiv preprint arXiv:1904.10509* (2019).

[33] Sungjun Cho, Seunghyuk Cho, Sungwoo Park, Hankook Lee, Honglak Lee, and Moontae Lee. "Curve Your Attention: Mixed-Curvature Transformers for Graph Representation Learning". In: *arXiv preprint arXiv:2309.04082* (2023).

[34] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. "Rethinking Attention with Performers". In: *arXiv preprint arXiv:2009.14794* (2020).

[35] Zhixuan Chu, Yan Wang, Qing Cui, Longfei Li, Wenqing Chen, Sheng Li, Zhan Qin, and Kui Ren. "LLM-Guided Multi-View Hypergraph Learning for Human-Centric Explainable Recommendation". In: *arXiv preprint arXiv:2401.08217* (2024).

[36] Israel Cohen, Yiteng Huang, Jingdong Chen, and Jacob Benesty. "Pearson Correlation Coefficient". In: *Noise Reduction in Speech Processing* (2009), pp. 1–4.

[37] Paolo Cremonesi, Roberto Turrin, Eugenio Lentini, and Matteo Matteucci. "An Evaluation Methodology for Collaborative Recommender Systems". In: *2008 International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*. IEEE. 2008, pp. 224–231.

[38] Qionghai Dai and Yue Gao. "Large Scale Hypergraph Computation". In: *Hypergraph Computation*. Springer, 2023, pp. 145–157.

[39] Khalil Damak, Sami Khenissi, and Olfa Nasraoui. "Debiasing the Cloze Task in Sequential Recommendation with Bidirectional Transformers". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 273–282.

[40] Tri Dao. "Flashattention-2: Faster attention with better parallelism and work partitioning". In: *arXiv preprint arXiv:2307.08691* (2023).

[41] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. "FlashAttention: Fast and Memory-Efficient Exact Attention With IO-Awareness". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 16344–16359.

[42] Yashar Deldjoo, Tommaso Di Noia, and Felice Antonio Merra. "Adversarial Machine Learning in Recommender Systems (AML-Recsys)". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020, pp. 869–872.

[43] Yicheng Di, Hongjian Shi, Ruhui Ma, Honghao Gao, Yuan Liu, and Weiyu Wang. "FedRL: A Reinforcement Learning Federated Recommender System for Efficient Communication Using Reinforcement Selector and Hypernet Generator". In: *ACM Transactions on Recommender Systems* (2024).

[44] Francesco Di Giovanni, Lorenzo Giusti, Federico Barbero, Giulia Luise, Pietro Lio, and Michael M Bronstein. "On Over-Squashing in Message Passing Neural Networks: The Impact of Width, Depth, and Topology". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 7865–7885.

[45] Kaize Ding, Albert Jiongqian Liang, Bryan Perozzi, Ting Chen, Ruoxi Wang, Lichan Hong, Ed H Chi, Huan Liu, and Derek Zhiyuan Cheng. "HyperFormer: Learning Expressive Sparse Feature Representations via Hypergraph Transformer". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 2062–2066.

[46] Kaize Ding, Jianling Wang, Jundong Li, Dingcheng Li, and Huan Liu. "Be More with Less: Hypergraph Attention Networks for Inductive Text Classification". In: *arXiv preprint arXiv:2011.00387* (2020).

[47] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer,

Georg Heigold, Sylvain Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[48] Boxin Du, Changhe Yuan, Robert Barton, Tal Neiman, and Hanghang Tong. "Self-Supervised Hypergraph Representation Learning". In: *2022 IEEE International Conference on Big Data (Big Data)*. IEEE. 2022, pp. 505–514.

[49] Yingpeng Du, Ziyan Wang, Zhu Sun, Haoyan Chua, Hongzhi Liu, Zhonghai Wu, Yining Ma, Jie Zhang, and Youchen Sun. "Large Language Model with Graph Convolution for Recommendation". In: *arXiv preprint arXiv:2402.08859* (2024).

[50] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. "The LLaMA 3 Herd of Models". In: *arXiv preprint arXiv:2407.21783* (2024).

[51] Tomislav Duricic, Dominik Kowald, Emanuel Lacic, and Elisabeth Lex. "Beyond-Accuracy: A Review on Diversity, Serendipity, and Fairness in Recommender Systems Based on Graph Neural Networks". In: *Frontiers in Big Data* 6 (2023), p. 1251072.

[52] Shereen Elsayed, Ahmed Rashed, and Lars Schmidt-Thieme. "HMAR: Hierarchical Masked Attention for Multi-Behaviour Recommendation". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer. 2024, pp. 131–143.

[53] Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S Yu. "Continuous-Time Sequential Recommendation with Temporal Graph Collaborative Transformer". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 433–442.

[54] Jun Fang. "Session-Based Recommendation With Self-Attention Networks". In: *arXiv preprint arXiv:2102.01922* (2021).

[55] Bahare Fatemi, Jonathan Halcrow, and Bryan Perozzi. "Talk Like a Graph: Encoding Graphs for Large Language Models". In: *arXiv preprint arXiv:2310.04560* (2023).

[56] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. "Hypergraph Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 3558–3565.

[57] Antonino Ferraro, Antonio Galli, Valerio La Gatta, Vincenzo Moscato, Marco Postiglione, Giancarlo Sperlı, and Flora Amato. "HEMR: Hypergraph Embeddings for Music Recommendation". In: *SEBD*. 2023, pp. 317–325.

[58] Saman Forouzandeh, Dr Sajad Ahmadian, Dr Parham Moradi, and Mahdi Jalili. "DHDHL: Deep Heterogeneous Dynamic Hypergraph Learning for Recommender Systems". In: *Available at SSRN 4872577* (2024).

[59] Luke Friedman, Sameer Ahuja, David Allen, Terry Tan, Hakim Sidahmed, Changbo Long, Jun Xie, Gabriel Schubiner, Ajay Patel, Harsh Lara, et al. "Leveraging Large Language Models in Conversational Recommender Systems". In: *arXiv preprint arXiv:2305.07961* (2023).

[60] Jiarun Fu, Rong Gao, Yonghong Yu, Jia Wu, Jing Li, Donghua Liu, and Zhiwei Ye. "Contrastive Graph Learning Long and Short-Term Interests for POI Recommendation". In: *Expert Systems with Applications* 238 (2024), p. 121931.

[61] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. "Hyperbolic Neural Networks". In: *Advances in Neural Information Processing Systems* 31 (2018).

[62] Chen Gao, Xiang Wang, Xiangnan He, and Yong Li. "Graph Neural Networks for Recommender System". In: *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*. 2022, pp. 1623–1625.

[63] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. "A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions". In: *ACM Transactions on Recommender Systems* 1.1 (2023), pp. 1–51.

[64] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. "Advances and Challenges in Conversational Recommender Systems: A Survey". In: *AI Open* 2 (2021), pp. 100–126.

[65] Yue Gao, Yifan Feng, Shuyi Ji, and Rongrong Ji. "HGNN+: General Hypergraph Neural Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.3 (2022), pp. 3181–3199.

[66] Yue Gao, Shuyi Ji, Xiangmin Han, and Qionghai Dai. "Hypergraph Computation". In: *Engineering* (2024).

[67] Alireza Gharahighehi, Celine Vens, and Konstantinos Pliakos. "Fair Multi-Stakeholder News Recommender System with Hypergraph Ranking". In: *Information Processing & Management* 58.5 (2021), p. 102663.

[68] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. "Neural Message Passing for Quantum Chemistry". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 1263–1272.

[69] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. "The Reversible Residual Network: Backpropagation Without Storing Activations". In: *Advances in Neural Information Processing Systems* 30 (2017).

[70] Carlos A Gomez-Uribe and Neil Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Transactions on Management Information Systems (TMIS)* 6.4 (2015), pp. 1–19.

[71] Florian Grötschla, Jiaqing Xie, and Roger Wattenhofer. "Benchmarking Positional Encodings for GNNs and Graph Transformers". In: *arXiv preprint arXiv:2411.12732* (2024).

[72] Fangda Gu, Heng Chang, Wenwu Zhu, Somayeh Sojoudi, and Laurent El Ghaoui. "Implicit Graph Neural Networks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 11984–11995.

[73] Caglar Gulcehre, Misha Denil, Mateusz Malinowski, Ali Razavi, Razvan Pascanu, Karl Moritz Hermann, Peter Battaglia, Victor Bapst, David Raposo, Adam Santoro, et al. "Hyperbolic Attention Networks". In: *arXiv preprint arXiv:1805.09786* (2018).

[74] Asela Gunawardana, Guy Shani, and Sivan Yogev. "Evaluating Recommender Systems". In: *Recommender Systems Handbook*. Springer, 2012, pp. 547–601.

[75] Lei Guo, Hongzhi Yin, Tong Chen, Xiangliang Zhang, and Kai Zheng. "Hierarchical Hyperedge Embedding-Based Representation Learning for Group Recommendation". In: *ACM Transactions on Information Systems (TOIS)* 40.1 (2021), pp. 1–27.

[76] Maosheng Guo, Yu Zhang, and Ting Liu. "Gaussian Transformer: A Lightweight Approach for Natural Language Inference". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6489–6496.

[77] Vipul Gupta, Dhruv Choudhary, Peter Tang, Xiaohan Wei, Xing Wang, Yuzhen Huang, Arun Kejariwal, Kannan Ramchandran, and Michael W Mahoney. "Training Recommender Systems at Scale: Communication-Efficient Model and Data Parallelism". In: *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2021, pp. 2928–2936.

[78] Dongho Ha, Yunan Zhang, Chen-Chien Kao, Christopher J Hughes, Won Woo Ro, and Hung-Wei Tseng. "M3 XU: Achieving High-Precision and Complex Matrix Multiplication with Low-Precision MXUs". In: *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2024, pp. 1–16.

[79] Ruobing Han, Jun Chen, Bhanu Garg, Jeffrey Young, Jaewoong Sim, and Hyesoon Kim. "CuPBoP: CUDA for Parallelized and Broad-Range Processors". In: *arXiv preprint arXiv:2206.07896* (2022).

[80] Zhongxuan Han, Chaochao Chen, Xiaolin Zheng, Li Zhang, and Yuyuan Li. "Hypergraph Convolutional Network for User-Oriented Fairness in Recommender Systems". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 903–913.

[81] Zhongxuan Han, Xiaolin Zheng, Chaochao Chen, Wenjie Cheng, and Yang Yao. "Intra and Inter Domain Hypergraph Convolutional Network for Cross-Domain Recommendation". In: *Proceedings of the ACM Web Conference 2023*. 2023, pp. 449–459.

[82] Bowen Hao, Chaoqun Yang, Lei Guo, Junliang Yu, and Hongzhi Yin. "Motif-Based Prompt Learning for Universal Cross-Domain Recommendation". In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 2024, pp. 257–265.

[83] Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. "Leveraging Large Language Models for Sequential Recommendation". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 1096–1102.

[84] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. "Transformer Language Models Without Positional Encodings Still Learn Positional Information". In: *arXiv preprint arXiv:2203.16634* (2022).

[85] Neil He, Menglin Yang, and Rex Ying. "Lorentzian Residual Neural Networks". In: *arXiv preprint arXiv:2412.14695* (2024).

[86] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. "Neural Collaborative Filtering". In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 173–182.

[87] Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. "Large Language Models as Zero-Shot Conversational Recommenders". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 720–730.

[88] Natali Helberger, Kari Karppinen, and Lucia D'acunto. "Exposure Diversity as a Design Principle for Recommender Systems". In: *Information, Communication & Society* 21.2 (2018), pp. 191–207.

[89] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. "Evaluating Collaborative Filtering Recommender Systems". In: *ACM Transactions on Information Systems (TOIS)* 22.1 (2004), pp. 5–53.

[90] Lukman Heryawan, Reza Pulungan, et al. "Trust Decay-Based Temporal Learning for Dynamic Recommender Systems with Concept Drift Adaptation". In: *IEEE Access* (2025).

[91] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. "Session-Based Recommendations With Recurrent Neural Networks". In: *arXiv preprint arXiv:1511.06939* (2015).

[92] Yassine Himeur, Shahab Saquib Sohail, Faycal Bensaali, Abbes Amira, and Mamoun Alazab. "Latest Trends of Security and Privacy in Recommender Systems: A Comprehensive Review and Future Perspectives". In: *Computers & Security* 118 (2022), p. 102746.

[93] Wentao Hu and Hui Fang. "Towards Differential Privacy in Sequential Recommendation: A Noisy Graph Neural Network Approach". In: *ACM Transactions on Knowledge Discovery from Data* 18.5 (2024), pp. 1–21.

[94] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. "Heterogeneous Graph Transformer". In: *Proceedings of The Web Conference 2020*. 2020, pp. 2704–2710.

[95] Hanyao Huang, Ou Zheng, Dongdong Wang, Jiayi Yin, Zijin Wang, Shengxuan Ding, Heng Yin, Chuan Xu, Renjie Yang, Qian Zheng, et al. "ChatGPT for Shaping the Future of Dentistry: The Potential of Multi-Modal Large Language Model". In: *International Journal of Oral Science* 15.1 (2023), p. 29.

[96] Kun-Yi Huang, Chung-Hsien Wu, Tsung-Hsien Yang, Ming-Hsiang Su, and Jia-Hui Chou. "Speech Emotion Recognition Using Autoencoder Bottleneck Features and LSTM". In: *2016 International Conference on Orange Technologies (ICOT)*. IEEE. 2016, pp. 1–4.

[97] Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. "Recommender AI Agent: Integrating Large Language Models for Interactive Recommendations". In: *arXiv preprint arXiv:2308.16505* (2023).

[98] Dietmar Jannach and Gediminas Adomavicius. "Recommendations With a Purpose". In: *Proceedings of the 10th ACM Conference on Recommender Systems*. 2016, pp. 7–10.

[99] Dietmar Jannach and Michael Jugovac. "Measuring the Business Value of Recommender Systems". In: *ACM Transactions on Management Information Systems (TMIS)* 10.4 (2019), pp. 1–23.

[100] Danish Javeed, Muhammad Shahid Saeed, Prabhat Kumar, Alireza Jolfaei, Shareeful Islam, and AKM Najmul Islam. "Federated Learning-Based Personalized Recommendation Systems: An Overview on Security and Privacy Challenges". In: *IEEE Transactions on Consumer Electronics* (2023).

[101] Theis E Jendal, Trung-Hoang Le, Hady W Lauw, Matteo Lissandrini, Peter Dolog, and Katja Hose. "Hypergraphs with Attention on Reviews for Explainable Recommendation". In: *European Conference on Information Retrieval*. Springer. 2024, pp. 230–246.

[102] Chunlin Ji, Yuhao Fu, and Ping He. "Adaptive Riemannian Stochastic Gradient Descent and Reparameterization for Gaussian Mixture Model Fitting". In: *Asian Conference on Machine Learning*. PMLR. 2024, pp. 534–549.

[103] Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. "GenRec: Large Language Model for Generative Recommendation". In: *European Conference on Information Retrieval*. Springer. 2024, pp. 494–502.

[104] Junzhong Ji, Yating Ren, and Minglong Lei. "FC–HAT: Hypergraph Attention Network for Functional Brain Network Classification". In: *Information Sciences* 608 (2022), pp. 1301–1316.

[105] Daixi Jia, Hang Gao, Xingzhe Su, Fengge Wu, and Junsuo Zhao. "Introducing Semantic-Based Receptive Field into Semantic Segmentation via Graph Neural Networks". In: *International Conference on Neural Information Processing*. Springer. 2023, pp. 434–451.

[106] Zhixuan Jia, Yushun Fan, Chunyu Wei, and Ruyu Yan. "A Spatial-Temporal Hypergraph Based Method for Service Recommendation in the Mobile Internet of Things-Enabled Service Platform". In: *Advanced Engineering Informatics* 57 (2023), p. 102038.

[107] Meng Jian, Langchen Lang, Jingjing Guo, Zun Li, Tuo Wang, and Lifang Wu. "Light Dual Hypergraph Convolution for Collaborative Filtering". In: *Pattern Recognition* 154 (2024), p. 110596.

[108] Meng Jiang, Keqin Bao, Jizhi Zhang, Wenjie Wang, Zhengyi Yang, Fuli Feng, and Xiangnan He. "Item-Side Fairness of Large Language Model-Based Recommendation System". In: *Proceedings of the ACM on Web Conference 2024*. 2024, pp. 4717–4726.

[109] Xiaolong Jiang, Heli Sun, and Liang He. "Novel Behavior-Enhanced Long-and Short-Term Interest Model for Sequential Recommendation". In: *Information Sciences* (2024), p. 121127.

[110] Yanbin Jiang, Huifang Ma, Yuhang Liu, and Zhixin Li. "Exploring User Trust and Reliability for Recommendation: A Hypergraph Ranking Approach". In: *Neural Information Processing: 27th International Conference, ICONIP*

2020, Bangkok, Thailand, November 23–27, 2020, Proceedings, Part II 27. Springer. 2020, pp. 333–344.

[111] Bowen Jin, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. "Multi-Behavior Recommendation With Graph Convolutional Networks". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 659–668.

[112] Anna Jobin, Marcello Ienca, and Effy Vayena. "The Global Landscape of AI Ethics Guidelines". In: *Nature Machine Intelligence* 1.9 (2019), pp. 389–399.

[113] Marius Kaminskas and Derek Bridge. "Diversity, Serendipity, Novelty, and Coverage: A Survey and Empirical Analysis of Beyond-Accuracy Objectives in Recommender Systems". In: *ACM Transactions on Interactive Intelligent Systems (TiiS)* 7.1 (2016), pp. 1–42.

[114] Wang-Cheng Kang and Julian McAuley. "Self-Attentive Sequential Recommendation". In: *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2018, pp. 197–206.

[115] George Karypis. "hMETIS 1.5: A Hypergraph Partitioning Package". In: *http://www.cs.umn.edu/ metis* (1998).

[116] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. "Transformers Are RNNs: Fast Autoregressive Transformers with Linear Attention". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5156–5165.

[117] Guolin Ke, Di He, and Tie-Yan Liu. "Rethinking Positional Encoding in Language Pre-Training". In: *arXiv preprint arXiv:2006.15595* (2020).

[118] Eun-Sol Kim, Woo Young Kang, Kyoung-Woon On, Yu-Jung Heo, and Byoung-Tak Zhang. "Hypergraph Attention Networks for Multimodal Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14581–14590.

[119] Jinwoo Kim, Dat Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. "Pure Transformers are Powerful Graph Learners". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14582–14595.

[120] Jinwoo Kim, Saeyoon Oh, Sungjun Cho, and Seunghoon Hong. "Equivariant Hypergraph Neural Networks". In: *European Conference on Computer Vision*. Springer. 2022, pp. 86–103.

[121] Jinwoo Kim, Saeyoon Oh, and Seunghoon Hong. "Transformers Generalize DeepSets and Can Be Extended to Graphs & Hypergraphs". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28016–28028.

[122] Sunwoo Kim, Soo Yong Lee, Yue Gao, Alessia Antelmi, Mirko Polato, and Kijung Shin. "A Survey on Hypergraph Neural Networks: An In-Depth and Step-By-Step Guide". In: *arXiv preprint arXiv:2404.01039* (2024).

[123] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. "Reformer: The Efficient Transformer". In: *arXiv preprint arXiv:2001.04451* (2020).

[124] Florian Klimm, Charlotte M Deane, and Gesine Reinert. "Hypergraphs for Predicting Essential Genes Using Multiprotein Complex Data". In: *Journal of Complex Networks* 9.2 (2021), cnaa028.

[125] Risi Kondor, Hy Truong Son, Horace Pan, Brandon Anderson, and Shubhendu Trivedi. "Covariant Compositional Networks for Learning Graphs". In: *arXiv preprint arXiv:1801.02144* (2018).

[126] Madhav Kumar, Pedro Silva, Ashudeep Singh, and Abhay Varmaraja. "Inclusive Recommendations and User Engagement: Experimental Evidence from Pinterest". In: *Proceedings of the 25th ACM Conference on Economics and Computation*. 2024, pp. 1189–1191.

[127] Valerio La Gatta, Vincenzo Moscato, Mirko Pennone, Marco Postiglione, and Giancarlo Sperlí. "Music Recommendation via Hypergraph Embedding". In:

IEEE Transactions on Neural Networks and Learning Systems 34.10 (2022), pp. 7887–7899.

[128] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. "Quantifying the Carbon Emissions of Machine Learning". In: *arXiv preprint arXiv:1910.09700* (2019).

[129] Yantong Lai, Yijun Su, Lingwei Wei, Gaode Chen, Tianci Wang, and Daren Zha. "Multi-View Spatial-Temporal Enhanced Hypergraph Network for Next POI Recommendation". In: *International Conference on Database Systems for Advanced Applications*. Springer. 2023, pp. 237–252.

[130] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. "ALBERT: A Lite BERT for Self-Supervised Learning of Language Representations". In: *arXiv preprint arXiv:1909.11942* (2019).

[131] Anchen Li, Bo Yang, Huan Huo, Hongxu Chen, Guandong Xu, and Zhen Wang. "Hyperbolic Neural Collaborative Recommender". In: *IEEE Transactions on Knowledge and Data Engineering* 35.9 (2022), pp. 9114–9127.

[132] Anchen Li, Bo Yang, Huan Huo, and Farookh Hussain. "Hypercomplex Graph Collaborative Filtering". In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 1914–1922.

[133] Chaoliu Li, Lianghao Xia, Xubin Ren, Yaowen Ye, Yong Xu, and Chao Huang. "Graph Transformer for Recommendation". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 1680–1689.

[134] Chen Li, Xutan Peng, Yuhang Niu, Shanghang Zhang, Hao Peng, Chuan Zhou, and Jianxin Li. "Learning Graph Attention-Aware Knowledge Graph Embedding". In: *Neurocomputing* 461 (2021), pp. 516–529.

[135] Jia Li, Xiangguo Sun, Yuhan Li, Zhixun Li, Hong Cheng, and Jeffrey Xu Yu. "Graph Intelligence with Large Language Models and Prompt Learning". In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 6545–6554.

[136] Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. "GPT4Rec: A Generative Framework for Personalized Recommendation and User Interests Interpretation". In: *arXiv preprint arXiv:2304.03879* (2023).

[137] Jiuqiang Li and Shilei Zhu. "Multi-View Interactive Compromise Learning for Group Recommendation". In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2024, pp. 9396–9400.

[138] Mengran Li, Yong Zhang, Xiaoyong Li, Yuchen Zhang, and Baocai Yin. "Hypergraph Transformer Neural Networks". In: *ACM Transactions on Knowledge Discovery from Data* 17.5 (2023), pp. 1–22.

[139] Nan Li, Zhihao Yang, Yumeng Yang, Jian Wang, and Hongfei Lin. "Hyperbolic Hierarchical Knowledge Graph Embeddings for Biological Entities". In: *Journal of Biomedical Informatics* (2023), p. 104503.

[140] Peibo Li, Maarten de Rijke, Hao Xue, Shuang Ao, Yang Song, and Flora D Salim. "Large Language Models for Next Point-of-Interest Recommendation". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 1463–1472.

[141] Qian Li, Lixin Su, Jiashu Zhao, Long Xia, Hengyi Cai, Suqi Cheng, Hengzhu Tang, Junfeng Wang, and Dawei Yin. "Text-Video Retrieval via Multi-Modal Hypergraph Networks". In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 2024, pp. 369–377.

[142] Qingfeng Li, Huifang Ma, Wangyu Jin, Yugang Ji, and Zhixin Li. "Hypergraph-Enhanced Multi-Interest Learning for Multi-Behavior Sequential Recommendation". In: *Expert Systems with Applications* (2024), p. 124497.

[143] Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. "From Matching to Generation: A Survey on Generative Information Retrieval". In: *arXiv preprint arXiv:2404.14851* (2024).

[144] Xuan Li, Zhanke Zhou, Jiangchao Yao, Yu Rong, Lu Zhang, and Bo Han. "Neural Atoms: Propagating Long-Range Interaction in Molecular Graphs Through Efficient Communication Channel". In: *arXiv preprint arXiv:2311.01276* (2023).

[145] Yang Li, Kangbo Liu, Ranjan Satapathy, Suhang Wang, and Erik Cambria. "Recent Developments in Recommender Systems: A Survey". In: *IEEE Computational Intelligence Magazine* 19.2 (2024), pp. 78–95.

[146] Yicong Li, Hongxu Chen, Xiangguo Sun, Zhenchao Sun, Lin Li, Lizhen Cui, Philip S Yu, and Guandong Xu. "Hyperbolic Hypergraphs for Sequential Recommendation". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 988–997.

[147] Yinfeng Li, Chen Gao, Hengliang Luo, Depeng Jin, and Yong Li. "Enhancing Hypergraph Neural Networks with Intent Disentanglement for Session-Based Recommendation". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1997–2002.

[148] Yongkang Li, Zipei Fan, and Xuan Song. "Heterogeneous Hyperbolic Hypergraph Neural Network for Friend Recommendation in Location-Based Social Networks". In: *ACM Transactions on Knowledge Discovery from Data* 19.3 (2025), pp. 1–29.

[149] Yongqi Li, Xinyu Lin, Wenjie Wang, Fuli Feng, Liang Pang, Wenjie Li, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. "A Survey of Generative

Search and Recommendation in the Era of Large Language Models". In: *arXiv preprint arXiv:2404.16924* (2024).

[150] Yuyuan Li, Chaochao Chen, Xiaolin Zheng, Junlin Liu, and Jun Wang. "Making Recommender Systems Forget: Learning and Unlearning for Erasable Recommendation". In: *Knowledge-Based Systems* 283 (2024), p. 111124.

[151] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. "LLARA: Large Language-Recommendation Assistant". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 1785–1795.

[152] Binghao Liu, Pengpeng Zhao, Fuzhen Zhuang, Xuefeng Xian, Yanchi Liu, and Victor S Sheng. "Knowledge-Aware Hypergraph Neural Network for Recommender Systems". In: *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part III*. Springer. 2021, pp. 132–147.

[153] Bingqian Liu, Duantengchuan Li, Jian Wang, Zhihao Wang, Bing Li, and Cheng Zeng. "Integrating User Short-Term Intentions and Long-Term Preferences in Heterogeneous Hypergraph Networks for Sequential Recommendation". In: *Information Processing & Management* 61.3 (2024), p. 103680.

[154] Chengkai Liu, Jianghao Lin, Jianling Wang, Hanzhou Liu, and James Caverlee. "Mamba4Rec: Towards Efficient Sequential Recommendation with Selective State Space Models". In: *arXiv preprint arXiv:2403.03900* (2024).

[155] Jing Liu, Litao Shang, Yuting Su, Weizhi Nie, Xin Wen, and Anan Liu. "Privacy-Preserving Multi-Source Cross-Domain Recommendation Based on Knowledge Graph". In: *ACM Transactions on Multimedia Computing, Communications and Applications* 20.5 (2024), pp. 1–18.

[156] Meng Liu, Jianjun Li, Guohui Li, and Peng Pan. "Cross-Domain Recommendation via Bi-Directional Transfer Graph Collaborative Filtering Networks".

In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 885–894.

[157] Tie-Yan Liu et al. "Learning to Rank for Information Retrieval". In: *Foundations and Trends® in Information Retrieval* 3.3 (2009), pp. 225–331.

[158] Tongcun Liu, Xukai Bao, Jiaxin Zhang, Kai Fang, and Hailin Feng. "Enhancing Session-Based Recommendation with Multi-Interest Hyperbolic Representation Networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[159] Xin Liu, Mingyu Yan, Lei Deng, Guoqi Li, Xiaochun Ye, Dongrui Fan, Shirui Pan, and Yuan Xie. "Survey on Graph Neural Network Acceleration: An Algorithmic Perspective". In: *arXiv preprint arXiv:2202.04822* (2022).

[160] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. "EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14420–14430.

[161] Yi Liu, Hongrui Xuan, Bohan Li, Meng Wang, Tong Chen, and Hongzhi Yin. "Self-Supervised Dynamic Hypergraph Recommendation Based on Hyper-Relational Knowledge Graph". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 1617–1626.

[162] Yong Liu, Susen Yang, Chenyi Lei, Guoxin Wang, Haihong Tang, Juyong Zhang, Aixin Sun, and Chunyan Miao. "Pre-Training Graph Transformer with Multimodal Side Information for Recommendation". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 2853–2861.

[163] Zihan Liu, Yupeng Hou, and Julian McAuley. "Multi-Behavior Generative Recommendation". In: *arXiv preprint arXiv:2405.16871* (2024).

[164] I Loshchilov. "Decoupled Weight Decay Regularization". In: *arXiv preprint arXiv:1711.05101* (2017).

[165] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. "Structured State Space Models for In-Context Reinforcement Learning". In: *Advances in Neural Information Processing Systems* 36 (2024).

[166] Chaoguang Luo, Liuying Wen, Yong Qin, Liangwei Yang, Zhineng Hu, and Philip S Yu. "Against Filter Bubbles: Diversified Music Recommendation via Weighted Hypergraph Embedding Learning". In: *arXiv preprint arXiv:2402.16299* (2024).

[167] Wenjuan Luo, Han Zhang, Xiaodi Yang, Lin Bo, Xiaoqing Yang, Zang Li, Xiaohu Qie, and Jieping Ye. "Dynamic Heterogeneous Graph Neural Network for Real-Time Event Prediction". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3213–3223.

[168] Ang Ma, Yanhua Yu, Chuan Shi, Zirui Guo, and Tat-Seng Chua. "Cross-View Hypergraph Contrastive Learning for Attribute-Aware Recommendation". In: *Information Processing & Management* 61.4 (2024), p. 103701.

[169] Chao Ma, Minjie Wan, Jian Wu, Xiaofang Kong, Ajun Shao, Fan Wang, Qian Chen, and Guohua Gu. "Light Self-Gaussian-Attention Vision Transformer for Hyperspectral Image Classification". In: *IEEE Transactions on Instrumentation and Measurement* 72 (2023), pp. 1–12.

[170] Liheng Ma, Chen Lin, Derek Lim, Adriana Romero-Soriano, Puneet K Dokania, Mark Coates, Philip Torr, and Ser-Nam Lim. "Graph Inductive Biases in Transformers Without Message Passing". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 23321–23337.

[171] Lu Ma, Zeang Sheng, Xunkai Li, Xinyi Gao, Zhezheng Hao, Ling Yang, Wentao Zhang, and Bin Cui. "Acceleration Algorithms in GNNs: A Survey". In: *arXiv preprint arXiv:2405.04114* (2024).

[172] Muyang Ma, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Huasheng Liang, Jun Ma, and Maarten De Rijke. "Improving Transformer-Based Sequential Recommenders Through Preference Editing". In: *ACM Transactions on Information Systems* 41.3 (2023), pp. 1–24.

[173] Yutao Ma, Zesheng Wang, Liwei Huang, and Jian Wang. "CLHHN: Category-Aware Lossless Heterogeneous Hypergraph Neural Network for Session-Based Recommendation". In: *ACM Transactions on the Web* 18.1 (2023), pp. 1–37.

[174] Zi-Feng Mai, Chang-Dong Wang, Zhongjie Zeng, Ya Li, Jiaquan Chen, and Philip S Yu. "Hypergraph Enhanced Knowledge Tree Prompt Learning for Next-Basket Recommendation". In: *arXiv preprint arXiv:2312.15851* (2023).

[175] Mingsong Mao, Jie Lu, Jialin Han, and Guangquan Zhang. "Multiobjective E-Commerce Recommendations Based on Hypergraph Ranking". In: *Information Sciences* 471 (2019), pp. 269–287.

[176] Haggai Maron, Heli Ben-Hamu, Hadar Serviansky, and Yaron Lipman. "Provably Powerful Graph Networks". In: *Advances in Neural Information Processing Systems* 32 (2019).

[177] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. "Invariant and Equivariant Graph Networks". In: *arXiv preprint arXiv:1812.09902* (2018).

[178] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. "A Survey on Bias and Fairness in Machine Learning". In: *ACM Computing Surveys (CSUR)* 54.6 (2021), pp. 1–35.

[179] Xin Mei, Xiaoyan Cai, Libin Yang, and Nanxin Wang. "Graph Transformer Networks Based Text Representation". In: *Neurocomputing* 463 (2021), pp. 91–100.

[180] Gonzalo Mena, David Belanger, Gonzalo Munoz, and Jasper Snoek. "Sinkhorn Networks: Using Optimal Transport Techniques to Learn Permutations". In: *NIPS Workshop in Optimal Transport and Machine Learning*. Vol. 3. 2017.

[181] ChunYan Meng and Hooman Motevalli. "Link Prediction in Social Networks Using Hyper-Motif Representation on Hypergraph". In: *Multimedia Systems* 30.3 (2024), p. 123.

[182] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. "Towards Efficient Generative Large Language Model Serving: A Survey from Algorithms to Systems". In: *arXiv preprint arXiv:2312.15234* (2023).

[183] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. "Transformer for Graphs: An Overview from Architecture Perspective". In: *arXiv preprint arXiv:2202.08455* (2022).

[184] Sheshera Mysore, Andrew McCallum, and Hamed Zamani. "Large Language Model Augmented Narrative Driven Recommendations". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 777–783.

[185] David Newton, Farzad Yousefian, and Raghu Pasupathy. "Stochastic Gradient Descent: Recent Trends". In: *Recent Advances in Optimization and Modeling of Contemporary Problems* (2018), pp. 193–220.

[186] Maximillian Nickel and Douwe Kiela. "Learning Continuous Hierarchies in the Lorentz Model of Hyperbolic Geometry". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3779–3788.

[187] Maximillian Nickel and Douwe Kiela. "Poincaré Embeddings for Learning Hierarchical Representations". In: *Advances in Neural Information Processing Systems* 30 (2017).

[188] Morton E O'Kelly. "A Geographer's Analysis of Hub-and-Spoke Networks". In: *Journal of Transport Geography* 6.3 (1998), pp. 171–186.

[189] Kenta Oono and Taiji Suzuki. "Graph Neural Networks Exponentially Lose Expressive Power for Node Classification". In: *arXiv preprint arXiv:1905.10947* (2019).

[190] Kenta Oono and Taiji Suzuki. "Optimization and Generalization Analysis of Transduction Through Gradient Boosting and Application to Multi-Scale Graph Neural Networks". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 18917–18930.

[191] Andreas Peintner. "Sequential Recommendation Models: A Graph-Based Perspective". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 1295–1299.

[192] Wei Peng, Tuomas Varanka, Abdelrahman Mostafa, Henglin Shi, and Guoying Zhao. "Hyperbolic Deep Neural Networks: A Survey". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.12 (2021), pp. 10023–10044.

[193] Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. "Let Your Graph Do the Talking: Encoding Structured Data for LLMs". In: *arXiv preprint arXiv:2402.05862* (2024).

[194] Trang Pham, Truyen Tran, Hoa Dam, and Svetha Venkatesh. "Graph Classification via Deep Learning with Virtual Nodes". In: *arXiv preprint arXiv:1708.04357* (2017).

[195] Yanni Ping, Yang Li, and Jiaxin Zhu. "Beyond Accuracy Measures: The Effect of Diversity, Novelty and Serendipity in Recommender Systems on User Engagement". In: *Electronic Commerce Research* (2024), pp. 1–28.

[196] Chendi Qian, Andrei Manolache, Christopher Morris, and Mathias Niepert. "Probabilistic Graph Rewiring via Virtual Nodes". In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 28359–28392.

[197] Ruihong Qiu, Hongzhi Yin, Zi Huang, and Tong Chen. "GAG: Global Attributed Graph Neural Network for Streaming Session-Based Recommendation". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 669–678.

[198] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. "Recipe for a General, Powerful, Scalable Graph Transformer". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 14501–14515.

[199] Yunbo Rao, Tongze Mu, Shaoning Zeng, Junming Xue, and Jinhua Liu. "Multi-Session Aware Hypergraph Neural Network for Session-Based Recommendation". In: *Multimedia Tools and Applications* 83.5 (2024), pp. 12757–12774.

[200] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. "Representation Learning with Large Language Models for Recommendation". In: *Proceedings of the ACM on Web Conference 2024*. 2024, pp. 3464–3475.

[201] Kyle Robinson, Dan Brown, and Markus Schedl. "User Insights on Diversity in Music Recommendation Lists". In: *ISMIR*. 2020, pp. 446–453.

[202] David Rolnick, Priya L Donti, Lynn H Kaack, Kelly Kochanski, Alexandre Lacoste, Kris Sankaran, Andrew Slavin Ross, Nikola Milojevic-Dupont, Natasha Jaques, Anna Waldman-Brown, et al. "Tackling Climate Change with Machine Learning". In: *ACM Computing Surveys (CSUR)* 55.2 (2022), pp. 1–96.

[203] Guoping Rong, Yifan Zhang, Lanxin Yang, Fuli Zhang, Hongyu Kuang, and He Zhang. "Modeling Review History for Reviewer Recommendation: A Hypergraph Approach". In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 1381–1392.

[204] Eran Rosenbluth, Jan Tönshoff, Martin Ritzert, Berke Kisin, and Martin Grohe. "Distinguished in Uniform: Self Attention vs. Virtual Nodes". In: *arXiv preprint arXiv:2405.11951* (2024).

[205] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. "Efficient Content-Based Sparse Attention With Routing Transformers". In:

Transactions of the Association for Computational Linguistics 9 (2021), pp. 53–68.

[206] Deepjyoti Roy and Mala Dutta. "A Systematic Review and Research Perspective on Recommender Systems". In: *Journal of Big Data* 9.1 (2022), p. 59.

[207] Fateme Saghafi, Sayeh Mirzaei, and Ali Fahim. "Product Recommender System Using Hypergraph Neural Networks". In: *2023 9th International Conference on Web Research (ICWR)*. IEEE. 2023, pp. 189–195.

[208] Darnbi Sakong, Viet Hung Vu, Thanh Trung Huynh, Phi Le Nguyen, Hongzhi Yin, Quoc Viet Hung Nguyen, and Thanh Tam Nguyen. "Higher-Order Knowledge-Enhanced Recommendation with Heterogeneous Hypergraph Multi-Attention". In: *Information Sciences* 680 (2024), p. 121165.

[209] Clayton Sanford, Bahare Fatemi, Ethan Hall, Anton Tsitsulin, Mehran Kazemi, Jonathan Halcrow, Bryan Perozzi, and Vahab Mirrokni. "Understanding Transformer Reasoning Capabilities via Graph Algorithms". In: *arXiv preprint arXiv:2405.18512* (2024).

[210] Zhufeng Shao, Shoujin Wang, Wenpeng Lu, Weiyu Zhang, Hongjiao Guan, and Long Zhao. "Filter-Enhanced Hypergraph Transformer for Multi-Behavior Sequential Recommendation". In: *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2024, pp. 6575–6579.

[211] Kartik Sharma, Yeon-Chang Lee, Sivagami Nambi, Aditya Salian, Shlok Shah, Sang-Wook Kim, and Srijan Kumar. "A Survey of Graph Neural Networks for Social Recommender Systems". In: *ACM Computing Surveys* 56.10 (2024), pp. 1–34.

[212] Quan Z Sheng, Wei Emma Zhang, Salma Abdalla Hamad, Nguyen Lu Dang Khoa, Nguyen H Tran, et al. "Deep Conversational Recommender Systems: Challenges and Opportunities". In: *Computer* 55.4 (2022), pp. 30–39.

[213] Chuan Shi, Xiao Wang, and S Yu Philip. *Heterogeneous Graph Representation Learning and Applications*. 2022.

[214] Ryohei Shimizu, Yusuke Mukuta, and Tatsuya Harada. "Hyperbolic Neural Networks++". In: *arXiv preprint arXiv:2006.08210* (2020).

[215] Hamed Shirzad, Ameya Velingker, Balaji Venkatachalam, Danica J Sutherland, and Ali Kemal Sinop. "Exphormer: Sparse Transformers for Graphs". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31613–31632.

[216] Damien Sileo, Wout Vossen, and Robbe Raymaekers. "Zero-Shot Recommendation as Language Modeling". In: *European Conference on Information Retrieval*. Springer. 2022, pp. 223–230.

[217] Pradeep Kumar Singh, Pijush Kanti Dutta Pramanik, Avick Kumar Dey, and Prasenjit Choudhury. "Recommender Systems: An Overview, Research Trends, and Future Directions". In: *International Journal of Business and Systems Research* 15.1 (2021), pp. 14–52.

[218] Richard Sinkhorn. "A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices". In: *The Annals of Mathematical Statistics* 35.2 (1964), pp. 876–879.

[219] Sakhinana Sagar Srinivas, Rajat Kumar Sarkar, and Venkataramana Runkana. "Hypergraph Learning-Based Recommender System for Anomaly Detection, Control and Optimization". In: *2022 IEEE International Conference on Big Data (Big Data)*. IEEE. 2022, pp. 1922–1929.

[220] Balasubramaniam Srinivasan, Da Zheng, and George Karypis. "Learning Over Families of Sets: Hypergraph Representation Learning for Higher-Order Tasks". In: *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM. 2021, pp. 756–764.

[221] Christoph Stach and Frank Steimle. "Recommender-Based Privacy Requirements Elicitation-EPICUREAN: An Approach to Simplify Privacy Settings in IoT Applications With Respect to the GDPR". In: *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*. 2019, pp. 1500–1507.

[222] Emma Strubell, Ananya Ganesh, and Andrew McCallum. "Energy and Policy Considerations for Deep Learning in NLP". In: *arXiv preprint arXiv:1906.02243* (2019).

[223] Jiajie Su, Chaochao Chen, Zibin Lin, Xi Li, Weiming Liu, and Xiaolin Zheng. "Personalized Behavior-Aware Transformer for Multi-Behavior Sequential Recommendation". In: *Proceedings of the 31st ACM International Conference on Multimedia*. 2023, pp. 6321–6331.

[224] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. "BERT4Rec: Sequential Recommendation With Bidirectional Encoder Representations From Transformer". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 1441–1450.

[225] Jianing Sun, Zhaoyue Cheng, Saba Zuberi, Felipe Pérez, and Maksims Volkovs. "HGCF: Hyperbolic Graph Convolution Networks for Collaborative Filtering". In: *Proceedings of the Web Conference 2021*. 2021, pp. 593–601.

[226] Zequn Sun, Muhao Chen, Wei Hu, Chengming Wang, Jian Dai, and Wei Zhang. "Knowledge Association with Hyperbolic Knowledge Graph Embeddings". In: *arXiv preprint arXiv:2010.02162* (2020).

[227] Shyam A Tailor, René De Jong, Tiago Azevedo, Matthew Mattina, and Partha Maji. "Towards Efficient Point Cloud Graph Neural Networks Through Architectural Simplification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 2095–2104.

[228] Bohan Tang, Siheng Chen, and Xiaowen Dong. "Hypergraph-MLP: Learning on Hypergraphs without Message Passing". In: *ICASSP 2024-2024 IEEE In-*

ternational Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE. 2024, pp. 13476–13480.

[229] Jiaxi Tang and Ke Wang. "Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding". In: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 2018, pp. 565–573.

[230] Shaohua Tao, Runhe Qiu, Yuan Ping, and Hui Ma. "Multi-Modal Knowledge-Aware Reinforcement Learning Network for Explainable Recommendation". In: *Knowledge-Based Systems* 227 (2021), p. 107217.

[231] Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. "Sparse Sinkhorn Attention". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9438–9447.

[232] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. "Long Range Arena: A Benchmark for Efficient Transformers". In: *arXiv preprint arXiv:2011.04006* (2020).

[233] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. "Efficient Transformers: A Survey". In: *ACM Computing Surveys* 55.6 (2022), pp. 1–28.

[234] Zhiqiang Tian, Yezheng Liu, Jianshan Sun, Yuanchun Jiang, and Mingyue Zhu. "Exploiting Group Information for Personalized Recommendation With Graph Neural Networks". In: *ACM Transactions on Information Systems (TOIS)* 40.2 (2021), pp. 1–23.

[235] Roshni Vachhani and Suncica Hadzidedic. "Responsible AI in Personalised Movie Recommender Systems for the Hearing Impaired Community". In: *Intelligent Systems Conference*. Springer. 2024, pp. 195–214.

[236] Flavian Vasile, Elena Smirnova, and Alexis Conneau. "Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation". In: *Proceed-*

ings of the 10th ACM Conference on Recommender Systems. 2016, pp. 225–232.

[237] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need". In: *Advances in Neural Information Processing Systems* 30 (2017).

[238] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. "Graph Attention Networks". In: *stat* 1050.20 (2017), pp. 10–48550.

[239] M Vijaikumar, Deepesh Hada, and Shirish Shevade. "Hypertenet: Hypergraph and Transformer-Based Neural Network for Personalized List Continuation". In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 1210–1215.

[240] Sruthi Viswanathan, Fabien Guillot, and Antonietta Maria Grasso. "What is Natural? Challenges and Opportunities for Conversational Recommender Systems". In: *Proceedings of the 2nd Conference on Conversational User Interfaces*. 2020, pp. 1–4.

[241] Alexandra Vultureanu-Albişi and Costin Bădică. "Recommender Systems: An Explainable AI Perspective". In: *2021 International Conference on Innovations in Intelligent Systems and Applications (INISTA)*. IEEE. 2021, pp. 1–6.

[242] Chen Wang, Mengting Yuan, Rui Zhang, Kai Peng, and Ling Liu. "Efficient Point-of-Interest Recommendation Services with Heterogeneous Hypergraph Embedding". In: *IEEE Transactions on Services Computing* 16.2 (2022), pp. 1132–1143.

[243] Hongyu Wang, Wei Zhou, Junhao Wen, and Shutong Qiao. "Multiple Hypergraph Convolutional Network Social Recommendation Using Dual Contrastive Learning". In: *Data Mining and Knowledge Discovery* (2024), pp. 1–29.

[244] Huanwen Wang, Yawen Zeng, Jianguo Chen, Ning Han, and Hao Chen. "Interval-Enhanced Graph Transformer Solution for Session-Based Recommendation". In: *Expert Systems With Applications* 213 (2023), p. 118970.

[245] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. "Next-Item Recommendation With Sequential Hypergraphs". In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020, pp. 1101–1110.

[246] Jianling Wang, Kaize Ding, Ziwei Zhu, and James Caverlee. "Session-Based Recommendation with Hypergraph Attention Networks". In: *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*. SIAM. 2021, pp. 82–90.

[247] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. "Mixture-of-Agents Enhances Large Language Model Capabilities". In: *arXiv preprint arXiv:2406.04692* (2024).

[248] Nan Wang, Shoujin Wang, Yan Wang, Quan Z Sheng, and Mehmet A Orgun. "Exploiting Intra- and Inter-Session Dependencies for Session-Based Recommendations". In: *World Wide Web* 25.1 (2022), pp. 425–443.

[249] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. "Linformer: Self-Attention with Linear Complexity". In: *arXiv preprint arXiv:2006.04768* (2020).

[250] Wenjie Wang, Xinyu Lin, Fuli Feng, Xiangnan He, and Tat-Seng Chua. "Generative Recommendation: Towards Next-Generation Recommender Paradigm". In: *arXiv preprint arXiv:2304.03516* (2023).

[251] Wenxiao Wang, Wei Chen, Yicong Luo, Yongliu Long, Zhengkai Lin, Liye Zhang, Binbin Lin, Deng Cai, and Xiaofei He. "Model Compression and Efficient Inference for Large Language Models: A Survey". In: *arXiv preprint arXiv:2402.09748* (2024).

[252] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. "Heterogeneous Graph Attention Network". In: *The World Wide Web Conference*. 2019, pp. 2022–2032.

[253] Yifan Wang, Weizhi Ma, Min Zhang, Yiqun Liu, and Shaoping Ma. "A Survey on the Fairness of Recommender Systems". In: *ACM Transactions on Information Systems* 41.3 (2023), pp. 1–43.

[254] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, and Tie-Yan Liu. "A Theoretical Analysis of NDCG Type Ranking Measures". In: *Conference on Learning Theory*. PMLR. 2013, pp. 25–54.

[255] Zhihui Wang, Jianrui Chen, Fernando E Rosas, and Tingting Zhu. "A Hypergraph-Based Framework for Personalized Recommendations via User Preference and Dynamics Clustering". In: *Expert Systems with Applications* 204 (2022), p. 117552.

[256] Chunyu Wei, Jian Liang, Bing Bai, and Di Liu. "Dynamic Hypergraph Learning for Collaborative Filtering". In: *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2022, pp. 2108–2117.

[257] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 24824–24837.

[258] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. "LLMRec: Large Language Models with Graph Augmentation for Recommendation". In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 2024, pp. 806–815.

[259] Yinwei Wei, Wenqi Liu, Fan Liu, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. "LightGT: A Light Graph Transformer for Multimedia Recommenda-

tion". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 1508–1517.

[260] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. "Contrastive Learning for Cold-Start Recommendation". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 5382–5390.

[261] Yinwei Wei, Xiang Wang, Liqiang Nie, Xiangnan He, Richang Hong, and Tat-Seng Chua. "MMGCN: Multi-Modal Graph Convolution Network for Personalized Recommendation of Micro-Video". In: *Proceedings of the 27th ACM International Conference on Multimedia*. 2019, pp. 1437–1445.

[262] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. "Sustainable AI: Environmental Implications, Challenges and Opportunities". In: *Proceedings of Machine Learning and Systems* 4 (2022), pp. 795–813.

[263] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. "Simplifying Graph Convolutional Networks". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 6861–6871.

[264] Hanrui Wu, Jinyi Long, Nuosi Li, Dahai Yu, and Michael K Ng. "Adversarial Auto-Encoder Domain Adaptation for Cold-Start Recommendation with Positive and Negative Hypergraphs". In: *ACM Transactions on Information Systems* 41.2 (2022), pp. 1–25.

[265] Jialun Wu, Kai He, Rui Mao, Chen Li, and Erik Cambria. "MEGACare: Knowledge-Guided Multi-View Hypergraph Predictive Framework for Healthcare". In: *Information Fusion* 100 (2023), p. 101939.

[266] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. "DiffNet++: A Neural Influence and Interest Diffusion Network for Social

Recommendation". In: *IEEE Transactions on Knowledge and Data Engineering* 34.10 (2020), pp. 4753–4766.

[267] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. "Graph Neural Networks in Recommender Systems: A Survey". In: *arXiv preprint arXiv:2011.02260* (2020).

[268] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. "Graph Neural Networks in Recommender Systems: A Survey". In: *ACM Computing Surveys* 55.5 (2022), pp. 1–37.

[269] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. "Session-Based Recommendation With Graph Neural Networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 346–353.

[270] Yaxiong Wu, Craig Macdonald, and Iadh Ounis. "Goal-Oriented Multi-Modal Interactive Recommendation with Verbal and Non-Verbal Relevance Feedback". In: *Proceedings of the 17th ACM Conference on Recommender Systems*. 2023, pp. 362–373.

[271] Yuxia Wu, Yuan Fang, and Lizi Liao. "On the Feasibility of Simple Transformer for Dynamic Graph Modeling". In: *Proceedings of the ACM on Web Conference 2024*. 2024, pp. 870–880.

[272] Yuxia Wu, Lizi Liao, Gangyi Zhang, Wenqiang Lei, Guoshuai Zhao, Xueming Qian, and Tat-Seng Chua. "State Graph Reasoning for Multimodal Conversational Recommendation". In: *IEEE Transactions on Multimedia* (2022).

[273] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, Rui Zhang, et al. "Towards Open-World Recommendation with Knowledge Augmentation from Large Language Models". In: *arXiv preprint arXiv:2306.10933* (2023).

[274] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Mengyin Lu, and Liefeng Bo. "Multi-Behavior Enhanced Recommendation with Cross-Interaction Collaborative Relation Modeling". In: *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE. 2021, pp. 1931–1936.

[275] Lianghao Xia, Chao Huang, Yong Xu, Peng Dai, Xiyue Zhang, Hongsheng Yang, Jian Pei, and Liefeng Bo. "Knowledge-Enhanced Hierarchical Graph Transformer Network for Multi-Behavior Recommendation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5. 2021, pp. 4486–4493.

[276] Lianghao Xia, Chao Huang, Yong Xu, and Jian Pei. "Multi-Behavior Sequential Recommendation With Temporal Graph Transformer". In: *IEEE Transactions on Knowledge and Data Engineering* (2022).

[277] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. "Hypergraph Contrastive Collaborative Filtering". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 70–79.

[278] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Xiangji Huang. "Hypergraph Contrastive Collaborative Filtering". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2022, Madrid, Spain, July 11-15, 2022*. 2022.

[279] Lianghao Xia, Chao Huang, and Chuxu Zhang. "Self-Supervised Hypergraph Transformer for Recommender Systems". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 2100–2109.

[280] Lianghao Xia, Yong Xu, Chao Huang, Peng Dai, and Liefeng Bo. "Graph Meta Network for Multi-Behavior Recommendation". In: *Proceedings of the*

44th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2021, pp. 757–766.

[281] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. "Self-Supervised Hypergraph Convolutional Networks for Session-Based Recommendation". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5. 2021, pp. 4503–4511.

[282] Yikun Xian, Zuohui Fu, Shan Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. "Reinforcement Knowledge Graph Reasoning for Explainable Recommendation". In: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 285–294.

[283] Youshao Xiao, Shangchun Zhao, Zhenglei Zhou, Zhaoxin Huan, Lin Ju, Xiaolu Zhang, Lin Wang, and Jun Zhou. "G-Meta: Distributed Meta Learning in GPU Clusters for Large-Scale Recommender Systems". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 4365–4369.

[284] Xin Xin, Tiago Pimentel, Alexandros Karatzoglou, Pengjie Ren, Konstantina Christakopoulou, and Zhaochun Ren. "Rethinking Reinforcement Learning for Recommendation: A Prompt Perspective". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1347–1357.

[285] Chengfeng Xu, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Fuzhen Zhuang, Junhua Fang, and Xiaofang Zhou. "Graph Contextualized Self-Attention Network for Session-Based Recommendation". In: *IJCAI*. Vol. 19. 2019, pp. 3940–3946.

[286] Fengli Xu, Jianxun Lian, Zhenyu Han, Yong Li, Yujian Xu, and Xing Xie. "Relation-Aware Graph Convolutional Networks for Agent-Initiated Social E-

Commerce Recommendation". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2019, pp. 529–538.

[287] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In: *arXiv preprint arXiv:1810.00826* (2018).

[288] Mengjia Xu, Apoorva Vikram Singh, and George Em Karniadakis. "Dyng2G: An Efficient Stochastic Graph Embedding Method for Temporal Graphs". In: *IEEE Transactions on Neural Networks and Learning Systems* 35.1 (2022), pp. 985–998.

[289] Tuo Xu and Lei Zou. "Rethinking Higher-Order Representation Learning with Graph Neural Networks". In: *Learning on Graphs Conference*. PMLR. 2024, pp. 38–1.

[290] Xiangdong Xu, Krzysztof Przystupa, and Orest Kochan. "Social Recommendation Algorithm Based on Self-Supervised Hypergraph Attention". In: *Electronics* 12.4 (2023), p. 906.

[291] Zixuan Xu, Penghui Wei, Shaoguo Liu, Weimin Zhang, Liang Wang, and Bo Zheng. "Correlative Preference Transfer with Hierarchical Hypergraph Network for Multi-Domain Recommendation". In: *Proceedings of the ACM Web Conference 2023*. 2023, pp. 983–991.

[292] Hongrui Xuan, Yi Liu, Bohan Li, and Hongzhi Yin. "Knowledge Enhancement for Contrastive Multi-Behavior Recommendation". In: *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 2023, pp. 195–203.

[293] Hong-Jian Xue, Xinyu Dai, Jianbing Zhang, Shujian Huang, and Jiajun Chen. "Deep Matrix Factorization Models for Recommender Systems". In: *IJCAI*. Vol. 17. Melbourne, Australia. 2017, pp. 3203–3209.

[294] An Yan, Zhankui He, Jiacheng Li, Tianyang Zhang, and Julian McAuley. "Personalized Showcases: Generating Multi-Modal Explanations for Recommendations". In: *Proceedings of the 46th International ACM SIGIR Confer-*

ence on Research and Development in Information Retrieval. 2023, pp. 2251–2255.

[295] Xiaodong Yan, Tengwei Song, Yifeng Jiao, Jianshan He, Jiaotuan Wang, Ruopeng Li, and Wei Chu. "Spatio-Temporal Hypergraph Learning for Next POI Recommendation". In: *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2023, pp. 403–412.

[296] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. "Qwen2.5 Technical Report". In: *arXiv preprint arXiv:2412.15115* (2024).

[297] Haoran Yang, Hongxu Chen, Lin Li, S Yu Philip, and Guandong Xu. "Hyper Meta-Path Contrastive Learning for Multi-Behavior Recommendation". In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 787–796.

[298] Menglin Yang, Harshit Verma, Delvin Ce Zhang, Jiahong Liu, Irwin King, and Rex Ying. "Hypformer: Exploring Efficient Transformer Fully in Hyperbolic Space". In: *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024, pp. 3770–3781.

[299] Menglin Yang, Min Zhou, Jiahong Liu, Defu Lian, and Irwin King. "HRCF: Enhancing Collaborative Filtering via Hyperbolic Geometric Regularization". In: *Proceedings of the ACM Web Conference 2022*. 2022, pp. 2462–2471.

[300] Mingdai Yang, Zhiwei Liu, Liangwei Yang, Xiaolong Liu, Chen Wang, Hao Peng, and Philip S Yu. "Unified Pretraining for Recommendation via Task Hypergraphs". In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 2024, pp. 891–900.

[301] Shenghao Yang, Weizhi Ma, Peijie Sun, Qingyao Ai, Yiqun Liu, Mingchen Cai, and Min Zhang. "Sequential Recommendation with Latent Relations Based on Large Language Model". In: *Proceedings of the 47th International*

ACM SIGIR Conference on Research and Development in Information Retrieval. 2024, pp. 335–344.

[302] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. "Gated Linear Attention Transformers with Hardware-Efficient Training". In: *arXiv preprint arXiv:2312.06635* (2023).

[303] Tianchi Yang, Luhao Zhang, Chuan Shi, Cheng Yang, Siyong Xu, Ruiyu Fang, Maodi Hu, Huaijun Liu, Tao Li, and Dong Wang. "Gated Hypergraph Neural Network for Scene-Aware Recommendation". In: *International Conference on Database Systems for Advanced Applications*. Springer. 2022, pp. 199–215.

[304] Xin Yang, Xingrun Li, Heng Chang, Jinze Yang, Xihong Yang, Shengyu Tao, Ningkang Chang, Maiko Shigeno, Junfeng Wang, Dawei Yin, et al. "HGFormer: Hyperbolic Graph Transformer for Recommendation". In: *arXiv preprint arXiv:2502.15693* (2024).

[305] Yang Yang, Yi Zhu, and Yun Li. "Personalized Recommendation with Knowledge Graph via Dual-Autoencoder". In: *Applied Intelligence* (2022), pp. 1–12.

[306] Yonghui Yang, Le Wu, Kun Zhang, Richang Hong, Hailin Zhou, Zhiqiang Zhang, Jun Zhou, and Meng Wang. "Hyperbolic Graph Learning for Social Recommendation". In: *IEEE Transactions on Knowledge and Data Engineering* 36.12 (2023), pp. 8488–8501.

[307] Yuhao Yang, Chao Huang, Lianghao Xia, Yuxuan Liang, Yanwei Yu, and Chenliang Li. "Multi-Behavior Hypergraph-Enhanced Transformer for Sequential Recommendation". In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 2263–2274.

[308] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. "Tree of Thoughts: Deliberate Problem Solving

with Large Language Models". In: *Advances in Neural Information Processing Systems* 36 (2024).

[309] Yuan Yao, Zhiyuan Liu, Yankai Lin, and Maosong Sun. "Cross-Modal Representation Learning". In: *Representation Learning for Natural Language Processing*. Springer Nature Singapore Singapore, 2023, pp. 211–240.

[310] Jaehyuk Yi and Jinkyoo Park. "Hypergraph Convolutional Recurrent Neural Network". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020, pp. 3366–3376.

[311] Jing Yi and Zhenzhong Chen. "Multi-Modal Variational Graph Auto-Encoder for Recommendation Systems". In: *IEEE Transactions on Multimedia* 24 (2021), pp. 1067–1079.

[312] Nan Yin, Fuli Feng, Zhigang Luo, Xiang Zhang, Wenjie Wang, Xiao Luo, Chong Chen, and Xian-Sheng Hua. "Dynamic Hypergraph Convolutional Network". In: *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE. 2022, pp. 1621–1634.

[313] Ying Yin, Li-Xin Ji, Jian-Peng Zhang, and Yu-Long Pei. "DHNE: Network Representation Learning Method for Dynamic Heterogeneous Networks". In: *IEEE Access* 7 (2019), pp. 134782–134792.

[314] Chia-An Yu, Ching-Lun Tai, Tak-Shing Chan, and Yi-Hsuan Yang. "Modeling Multi-Way Relations with Hypergraph Embedding". In: *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 2018, pp. 1707–1710.

[315] Jie Yu, Junchen He, and Lingyu Xu. "Hypergraph-Based Academic Paper Recommendation". In: *International Conference on Knowledge Science, Engineering and Management*. Springer. 2022, pp. 449–462.

[316] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. "Self-Supervised Multi-Channel Hypergraph

Convolutional Network for Social Recommendation". In: *Proceedings of the Web Conference 2021*. 2021, pp. 413–424.

[317] Li Yu, Wei Gong, and Dongsong Zhang. "Live Streaming Channel Recommendation Based on Viewers' Interaction Behavior: A Hypergraph Approach". In: *Decision Support Systems* 184 (2024), p. 114272.

[318] Qiying Yu, Yudi Zhang, Yuyan Ni, Shikun Feng, Yanyan Lan, Hao Zhou, and Jingjing Liu. "Multimodal Molecular Pretraining via Modality Blending". In: *The Twelfth International Conference on Learning Representations*. 2024.

[319] Yonghong Yu, Aoran Zhang, Li Zhang, Rong Gao, Shang Gao, and Hongzhi Yin. "Hyperbolic Translation-Based Sequential Recommendation". In: *IEEE Transactions on Computational Social Systems* (2024).

[320] Zhouxin Yu, Jintang Li, Liang Chen, and Zibin Zheng. "Unifying Multi-Associations through Hypergraph for Bundle Recommendation". In: *Knowledge-Based Systems* 255 (2022), p. 109755.

[321] Enming Yuan, Wei Guo, Zhicheng He, Huifeng Guo, Chengkai Liu, and Ruiming Tang. "Multi-Behavior Sequential Transformer Recommender". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2022, pp. 1642–1652.

[322] Wei Yuan, Chaoqun Yang, Liang Qu, Quoc Viet Hung Nguyen, Jianxin Li, and Hongzhi Yin. "Hide Your Model: A Parameter Transmission-Free Federated Recommender System". In: *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE. 2024, pp. 611–624.

[323] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. "Big Bird: Transformers for Longer Sequences". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 17283–17297.

[324] Zhaodian Zeng, Yining Wang, Yanbin Zhao, and Wenxuan Shi. "A Survey of Music Recommendation Systems". In: *Proceedings of the 5th International Conference on Computer Information and Big Data Applications*. 2024, pp. 507–519.

[325] Haimin Zhang, Jiahao Xia, and Min Xu. "Differential Encoding for Improved Representation Learning over Graphs". In: *arXiv preprint arXiv:2407.02758* (2024).

[326] Hongyu Zhang, Dongyi Zheng, Lin Zhong, Xu Yang, Jiyuan Feng, Yunqing Feng, and Qing Liao. "FedHCDR: Federated Cross-Domain Recommendation with Hypergraph Signal Decoupling". In: *arXiv preprint arXiv:2403.02630* (2024).

[327] Junwei Zhang, Min Gao, Junliang Yu, Lei Guo, Jundong Li, and Hongzhi Yin. "Double-Scale Self-Supervised Hypergraph Learning for Group Recommendation". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 2557–2567.

[328] Peng Zhang, Zhendong Niu, Ru Ma, and Fuzhi Zhang. "DHCL-BR: Dual Hypergraph Contrastive Learning for Bundle Recommendation". In: *The Computer Journal* (2024), bxae056.

[329] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. "Deep Learning Based Recommender System: A Survey and New Perspectives". In: *ACM Computing Surveys (CSUR)* 52.1 (2019), pp. 1–38.

[330] Xu Zhang, Yonghui Xu, Wei He, Wei Guo, and Lizhen Cui. "A Comprehensive Review of the Oversmoothing in Graph Neural Networks". In: *CCF Conference on Computer Supported Cooperative Work and Social Computing*. Springer. 2023, pp. 451–465.

[331] Yadong Zhang, Shaoguang Mao, Tao Ge, Xun Wang, Adrian de Wynter, Yan Xia, Wenshan Wu, Ting Song, Man Lan, and Furu Wei. "LLM as a

Mastermind: A Survey of Strategic Reasoning with Large Language Models". In: *arXiv preprint arXiv:2404.01230* (2024).

[332] Yiding Zhang, Xiao Wang, Chuan Shi, Xunqiang Jiang, and Yanfang Ye. "Hyperbolic Graph Attention Network". In: *IEEE Transactions on Big Data* 8.6 (2021), pp. 1690–1701.

[333] Yongfeng Zhang, Xu Chen, et al. "Explainable Recommendation: A Survey and New Perspectives". In: *Foundations and Trends® in Information Retrieval* 14.1 (2020), pp. 1–101.

[334] Zhuosheng Zhang, Aston Zhang, Mu Li, Hai Zhao, George Karypis, and Alex Smola. "Multimodal Chain-of-Thought Reasoning in Language Models". In: *arXiv preprint arXiv:2302.00923* (2023).

[335] Sen Zhao, Wei Wei, Yifan Liu, Ziyang Wang, Wendi Li, Xian-Ling Mao, Shuai Zhu, Minghui Yang, and Zujie Wen. "Towards Hierarchical Policy Learning for Conversational Recommendation with Hypergraph-Based Reinforcement Learning". In: *arXiv preprint arXiv:2305.02575* (2023).

[336] Wayne Xin Zhao, Shanlei Mu, Yupeng Hou, Zihan Lin, Yushuo Chen, Xingyu Pan, Kaiyuan Li, Yujie Lu, Hui Wang, Changxin Tian, et al. "RecBole: Towards a Unified, Comprehensive and Efficient Framework for Recommendation Algorithms". In: *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 2021, pp. 4653–4664.

[337] Xiangyu Zhao, Maolin Wang, Xinjian Zhao, Jiansheng Li, Shucheng Zhou, Dawei Yin, Qing Li, Jiliang Tang, and Ruocheng Guo. "Embedding in Recommender Systems: A Survey". In: *arXiv preprint arXiv:2310.18608* (2023).

[338] Yingqi Zhao, Haiwei Zhang, Qijie Bai, Changli Nie, and Xiaojie Yuan. "DHMAE: A Disentangled Hypergraph Masked Autoencoder for Group Recommendation". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 914–923.

[339] Ziwei Zhao, Fake Lin, Xi Zhu, Zhi Zheng, Tong Xu, Shitian Shen, Xueying Li, Zikai Yin, and Enhong Chen. "DynLLM: When Large Language Models Meet Dynamic Graph Recommendation". In: *arXiv preprint arXiv:2405.07580* (2024).

[340] Ruiqi Zheng, Liang Qu, Tong Chen, Kai Zheng, Yuhui Shi, and Hongzhi Yin. "Poisoning Decentralized Collaborative Recommender System and Its Countermeasures". In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2024, pp. 1712–1721.

[341] Xiaolin Zheng, Zhongyu Wang, Chaochao Chen, Jiashu Qian, and Yao Yang. "Decentralized Graph Neural Network for Privacy-Preserving Recommendation". In: *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2023, pp. 3494–3504.

[342] Xiaoyao Zheng, Yonglong Luo, Liping Sun, Xintao Ding, and Ji Zhang. "A Novel Social Network Hybrid Recommender System Based on Hypergraph Topologic Structure". In: *World Wide Web* 21 (2018), pp. 985–1013.

[343] Zhi Zheng, Wenshuo Chao, Zhaopeng Qiu, Hengshu Zhu, and Hui Xiong. "Harnessing Large Language Models for Text-Rich Sequential Recommendation". In: *Proceedings of the ACM on Web Conference 2024*. 2024, pp. 3207–3216.

[344] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. "Learning with Hypergraphs: Clustering, Classification, and Embedding". In: *Advances in Neural Information Processing Systems* 19 (2006).

[345] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph Neural Networks: A Review of Methods and Applications". In: *AI Open* 1 (2020), pp. 57–81.

[346] Yuxuan Zhou, Chao Li, Zhi-Qi Cheng, Yifeng Geng, Xuansong Xie, and Margret Keuper. "Hypergraph Transformer for Skeleton-Based Action Recognition". In: *arXiv preprint arXiv:2211.09590* (2022).

[347] Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. "A Survey on Efficient Inference for Large Language Models". In: *arXiv preprint arXiv:2404.14294* (2024).

[348] Fanglin Zhu, Shunyu Chen, Yonghui Xu, Wei He, Fuqiang Yu, Xu Zhang, and Lizhen Cui. "Temporal Hypergraph for Personalized Clinical Pathway Recommendation". In: *2022 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. IEEE. 2022, pp. 718–725.

[349] Feng Zhu, Yan Wang, Chaochao Chen, Jun Zhou, Longfei Li, and Guanfeng Liu. "Cross-Domain Recommendation: Challenges, Progress, and Prospects". In: *arXiv preprint arXiv:2103.01696* (2021).

[350] Jinghua Zhu, Yanchang Cui, Zhuohao Zhang, and Heran Xi. "Knowledge Graph Transformer for Sequential Recommendation". In: *International Conference on Artificial Neural Networks*. Springer. 2023, pp. 459–471.

[351] Zirui Zhu, Chen Gao, Xu Chen, Nian Li, Depeng Jin, and Yong Li. "Inhomogeneous Social Recommendation with Hypergraph Convolutional Networks". In: *arXiv preprint arXiv:2111.03344* (2021).

[352] Bohan Zhuang, Jing Liu, Zizheng Pan, Haoyu He, Yuetian Weng, and Chunhua Shen. "A Survey on Efficient Training of Transformers". In: *arXiv preprint arXiv:2302.01107* (2023).

[353] Xingrui Zhuo, Shengsheng Qian, Jun Hu, Fuxin Dai, Kangyi Lin, and Gongqing Wu. "Multi-Hop Multi-View Memory Transformer for Session-Based Recommendation". In: *ACM Transactions on Information Systems* (2024).

[354] Keneilwe Zuva and Tranos Zuva. "Diversity and Serendipity in Recommender Systems". In: *Proceedings of the International Conference on Big Data and Internet of Things*. 2017, pp. 120–124.

Appendix A

Appendix

A.1 Baseline Models

Model	Core Architecture	Sequence Encoder	Propagation Mechanism
Transformer-based			
SASRec	Transformer	Self-attention	Feed-Forward
BERT4Rec	Bi-directional Transformer	Self-attention	Feed-Forward
PBAT	Transformer	Self-attention	Feed-Forward
MB-STR	Transformer	Self-attention	Feed-Forward
HMAR	Transformer	Self-attention	Feed-Forward
GNN-based			
GRU4Rec	RNN	GRU	Recurrent Propagation
MB-GCN	GNN	GCN	Message Passing
MB-GMN	GNN	GCN	Message Passing
GCSAN	GNN	GAT	Message Passing
Hybrid			
KMCLR	Transformer	Self-attention	Contrastive Learning
MBHT	Transformer	Self-attention	HGNN

Table A.1: Classification of baseline models by architecture.

A.2 STMBR

A.2.1 Hyperparameters

Hyperparameter	**Value(s)**
Embedding Dimension (d)	64, 128, 256, 512
Batch Size (B)	64, 128, 256
Learning Rate	0.001, 0.003
Optimizer	AdamW
Dropout	0.1, 0.2, 0.3
Number of Attention Heads	4, 8, 12
Sequence Length (ℓ)	64, 256, 512
Block Size (ℓ_B)	16, 32, 64
Number of Sorting Iterations	5, 10, 15, 20

Table A.2: Hyperparameter search table for the STMBR model.

A.3 HGLMRec

A.3.1 Adaptive Readout

Readout is defined as a permutation-invariant function that aggregates the final node embeddings $\{\mathbf{h}_v^{(K)}\}$ into a single graph-level vector:

$$\mathbf{h}_{\mathcal{G}} = \text{READOUT}\Big(\{\mathbf{h}_v^{(K)}\}\Big), \tag{A.1}$$

where the GNN attains maximum expressivity - matching the 1 Weisfeiler-Lehman isomorphism test using injective sum pooling at each layer and concatenating across all layers ensuring both permutation invariance and multi-scale expressivity [287]:

$$\mathbf{h}_{\mathcal{G}} = \text{CONCAT}\Big(\sum_v \mathbf{h}_v^{(0)}, \ldots, \sum_v \mathbf{h}_v^{(K)}\Big), \tag{A.2}$$

.

A.3.2 Hyperparameters

Table A.3 details key hyperparameters, selected via grid search on validation data.

Hyperparameter	**Value(s)**
Hypergraph Encoder (HGNN)	
Learning Rate	5×10^{-4}
Layers (L)	2
Hidden Dimension (d_l)	128
Dropout	0.2
Weight Decay (λ)	10^{-5}
Mixture-of-Agents	
MoA Layers	$\{1, 2, 3\}$
Training	
Batch Size	$\{64, 128, 256\}$
Warm-up Steps	500

Table A.3: Hyperparameter settings for HGLMRec.