THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Sequence Modelling for E-Commerce

Tong Chen

B.E.

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2020

School of Information Technology and Electrical Engineering

Abstract

With the proliferation of electronic commerce (e-commerce), the data generated by both customers and service providers can accumulate at a fast rate. As such, analyzing the rich but subtle patterns within the e-commerce data offers a prominent opportunity of refining user experience and increasing business revenue. Due to the high velocity of e-commerce data, sequence modelling plays a pivotal role in delivering timely predictive analytics and recommendations. Based on the granularity of data, sequence modelling for e-commerce is mainly conducted at two levels, namely macro-level modelling and micro-level modelling. When researching on e-commerce data, macro-level sequence modelling aims to understand the evolution of high-level business trends in order to set the foundation for enterprise strategic planning, e.g., sales prediction for inventory management. Meanwhile, micro-level sequence modelling focuses on learning fine-grained and dynamic user preferences from behavioral data to deliver personalized user experience, e.g., recommendation systems deployed by all major e-commerce platforms. In our research, we aim to effectively tackle sequence modelling in e-commerce scenarios at different levels, and then propose a unified model that allows for both macro- and micro-level sequence modelling, thus supporting a wide range of e-commerce applications. In summary, our research consists of the following three parts.

Firstly, for macro-level sequence modelling, we solve the problem of sales prediction, which is a critical means to achieve a healthy balance between supply and demand in e-commerce. The sales prediction task is formulated as a time series prediction problem which aims to predict the future sales volume for different products with observed influential factors (e.g., brand, season, discount, etc.) and corresponding historical sales records. However, with the development of contemporary commercial markets, the dynamic interactions between influential factors with different semantic meanings become more subtle, causing challenges in fully capturing dependencies among these variables. Besides, though seeking similar trends from the history benefits the accuracy for the prediction, existing methods hardly suit sales prediction tasks because the trends in sales data are more irregular and complex. Hence, we gain insights from the encoder-decoder recurrent neural network (RNN) structure, and propose a novel framework named TADA to carry out trend alignment with dual-attention, multi-task RNNs for sales prediction. In TADA, we innovatively divide the influential factors into internal feature and external feature, which are jointly modelled by a multi-task RNN encoder. In the decoding stage, TADA utilizes two attention mechanisms to compensate for the unknown states of influential factors in the future and adaptively align the upcoming trend with relevant historical trends to ensure precise sales prediction.

Secondly, for micro-level sequence modelling, we investigate sequential top-k recommendation, which infers users' preferences from their sequential behaviors and predicts their next interested items. Though it is important to capture the sequential patterns from the user-item interaction data, existing methods only focus on modelling the sparse item-wise sequential effect in user preference and only consider the homogeneous user interaction behaviors (i.e., a single type of user behavior). As a result, the data sparsity issue inevitably arises and makes the learned sequential patterns fragile and unreliable,

impeding the sequential recommendation performance of existing methods. Hence, in this task, we propose AIR, namely attentional intention-aware recommender systems to predict category-wise future user intention and collectively exploit the rich heterogeneous user interaction behaviors (i.e., multiple types of user behaviors). In AIR, we propose to represent user intention as an action-category tuple to discover category-wise sequential patterns and to capture varied effect of different types of actions for recommendation. A novel attentional recurrent neural network (ARNN) is proposed to model the intention migration effect and infer users' future intention. Besides, an intention-aware factorization machine (ITFM) is developed to perform intention-aware sequential recommendation.

Lastly, we develop a machine learning model that is generalizable to both macro- and micro-level sequence modelling tasks in e-commerce. Specifically, we extend a versatile predictive model, namely factorization machines (FMs) to the sequential setting. In e-commerce, models based on FMs are capable of modelling high-order interactions among features for effective predictive analytics, e.g., targeted advertising and recommendation. However, existing FM-based models assume no temporal orders in the data, and are unable to capture the sequential dependencies or patterns within the dynamic features, impeding the performance and adaptivity of these methods. Hence, we propose a novel sequence-aware factorization machine (SeqFM) for sequential predictive analytics, which models feature interactions by fully investigating the effect of sequential dependencies. As static features (e.g., user gender) and dynamic features (e.g., users' interacted items) express different semantics, we innovatively devise a multi-view self-attention scheme that separately models the effect of static features, dynamic features and the mutual interactions between static and dynamic features in three different views. In SeqFM, we further map the learned representations of feature interactions to the desired output with a shared residual network.

Declaration by Author

This thesis is composed of my original work, and contains no material previously published or written by another person except where due reference has been made in the text. I have clearly stated the contribution by others to jointly-authored works that I have included in my thesis.

I have clearly stated the contribution of others to my thesis as a whole, including statistical assistance, survey design, data analysis, significant technical procedures, professional editorial advice, financial support and any other original research work used or reported in my thesis. The content of my thesis is the result of work I have carried out since the commencement of my higher degree by research candidature and does not include a substantial part of work that has been submitted to qualify for the award of any other degree or diploma in any university or other tertiary institution. I have clearly stated which parts of my thesis, if any, have been submitted to qualify for another award.

I acknowledge that an electronic copy of my thesis must be lodged with the University Library and, subject to the policy and procedures of The University of Queensland, the thesis be made available for research and study in accordance with the Copyright Act 1968 unless a period of embargo has been approved by the Dean of the Graduate School.

I acknowledge that copyright of all material contained in my thesis resides with the copyright holder(s) of that material. Where appropriate I have obtained copyright permission from the copyright holder to reproduce material in this thesis and have sought permission from co-authors for any jointly authored works included in the thesis.

Publications Included in This Thesis

The following publication has been incorporated as Chapter 2.

[1] **Tong Chen**, Hongzhi Yin, Hongxu Chen, Lin Wu, Hao Wang, Xiaofang Zhou, and Xue Li, "TADA: Trend Alignment with Dual-Attention Multi-Task Recurrent Neural Networks for Sales Prediction", *2018 IEEE International Conference on Data Mining (ICDM)*.

Contributor	Statement of contribution	%
Tong Chen	Methodology and experimentation	70
	Paper writing	80
Hongzhi Yin	Methodology and experimentation	10
	Paper writing	10
Hongxu Chen	Methodology and experimentation	15
Lin Wu	Methodology and experimentation	5
Hao Wang	Proofreading	50
Xiaofang Zhou	Proofreading	50
Xue Li	Paper writing	10

The following publication has been incorporated as Chapter 3.

[2] **Tong Chen**, Hongzhi Yin, Hongxu Chen, Rui Yan, Quoc Viet Hung Nguyen, and Xue Li, "AIR: Attentional Intention-Aware Recommender Systems", *2019 IEEE International Conference on Data Engineering (ICDE)*.

Contributor	Statement of contribution	%
Tong Chen	Methodology and experimentation	80
	Paper writing	70
Hongzhi Yin	Methodology and experimentation	10
	Paper writing	15
Hongxu Chen	Methodology and experimentation	10
Rui Yan	Proofreading	50
Quoc Viet Hung Nguyen	Proofreading	50
Xue Li	Paper writing	5

The following publication has been incorporated as Chapter 4.

[3] **Tong Chen**, Hongzhi Yin, Quoc Viet Hung Nguyen, Wen-Chih Peng, Xue Li, and Xiaofang Zhou, "Sequence-Aware Factorization Machines for Temporal Predictive Analytics", *2020 IEEE International Conference on Data Engineering (ICDE)*.

Contributor	Statement of contribution	%
Tong Chen	Methodology and experimentation	90
	Paper writing	80
Hongzhi Yin	Methodology and experimentation	10
	Paper writing	10
Quoc Viet Hung Nguyen	Paper writing	5
Wen-Chih Peng	Proofreading	50
Xue Li	Paper writing	5
Xiaofang Zhou	Proofreading	50

Submitted Manuscripts Included in This Thesis

No submitted manuscripts are included in this thesis.

Other Publications during Candidature

1. [4] **Tong Chen**, Hongzhi Yin, Guanhua Ye, Zi Huang, Yang Wang and Meng Wang, "Try This Instead: Personalized and Interpretable Substitute Recommendation", *2020 International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

2. [5] **Tong Chen**, Hongzhi Yin, Hongxu Chen, Hao Wang, Xiaofang Zhou and Xue Li, "Online Sales Prediction via Trend Alignment-based Multi-Task Recurrent Neural Network", *2019 Knowledge and Information Systems (KAIS)*.

3. [6] Hongxu Chen, Hongzhi Yin, **Tong Chen**, Weiqing Wang, Xue Li and Xia Hu, "Social Boosted Recommendation with Folded Bipartite Network Embedding", *2020 IEEE Transactions on Knowledge and Data Engineering (TKDE)*.

4. [7] Xuhui Ren, Hongzhi Yin, **Tong Chen**, Hao Wang, Quoc Viet Hung Nguyen, Zi Huang and Xiangliang Zhang, "CRSAL: Conversational Recommender Systems with Adversarial Learning", *2020 ACM Transactions on Information Systems (TOIS)*.

5. [8] Qinyong Wang, Hongzhi Yin, **Tong Chen**, Zi Huang, Hao Wang, Yanchang Zhao, Quoc Viet Hung Nguyen, "Next Point-of-Interest Recommendation on Resource-Constrained Mobile Devices", *2020 The Web Conference (WWW)*.

6. [9] Shijie Zhang, Hongzhi Yin, **Tong Chen**, Nguyen Quoc Viet Hung, Zi Huang and Lizhen Cui, "GCN-Based User Representation Learning for Unifying Robust Recommendation and Fraudster Identification", *2020 International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

7. [10] Ruihong Qiu, Hongzhi Yin, Zi Huang and **Tong Chen**, "GAG: Global Attributed Graph Neural Network for Streaming Session-based Recommendation", *2020 International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*.

8. [11] Ke Sun, Tieyun Qian, **Tong Chen**, Yile Liang, Quoc Viet Hung Nguyen, and Hongzhi Yin, "Where to Go Next: Modelling Long- and Short-Term User Preferences for Point-of-Interest Recommendation", *2020 AAAI Conference on Artificial Intelligence (AAAI)*.

9. [12] Hongxu Chen, Hongzhi Yin, **Tong Chen**, Quoc Viet Hung Nguyen, Wen-Chih Peng, and Xue Li, "Exploiting Centrality Information with Graph Convolutions for Network Representation Learning", *2019 IEEE International Conference on Data Engineering (ICDE)*.

10. [13] Lei Guo, Hongzhi Yin, Qinyong Wang, **Tong Chen**, Alexander Zhou, and Quoc Viet Hung Nguyen, "Streaming Session-based Recommendation", *2019 ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*.

11. [14] Shijie Zhang, Hongzhi Yin, Qinyong Wang, **Tong Chen**, Hongxu Chen, and Ouoc Viet Hung Nguyen, "Inferring Substitutable Products with Deep Network Embedding", *2019 International Joint Conference on Artificial Intelligence (IJCAI)*.

12. [15] Ke Sun, Tieyun Qian, Hongzhi Yin, **Tong Chen**, Yiqi Chen, and Ling Chen, "What Can History Tell Us? Identifying Relevant Sessions for Next-Item Recommendation", *2019 ACM International Conference on Information and Knowledge Management (CIKM)*.

Statement of Parts of The Thesis Submitted to Qualify for The Award of Another Degree

No works submitted towards another degree have been included in this thesis.

Research Involving Human or Animal Subjects

No animal or human subjects were involved in this research.

Acknowledgments

During my 3.5-year PhD candidature at The University of Queensland (UQ), I have come across a lot of wonderful people that offered me tremendous help, support, and advice. I would give my supervisors, Dr. Hongzhi Yin and Prof. Xue Li a big thanks for their continuous academic guidance, scholarship support, and patient advisory along my way of research. Dr. Hongzhi Yin has set me a great example of a young scholar with superb visions, capabilities and humbleness, and I have been so lucky to join his research team and contribute to the research goal that we believe will greatly pay back to the society. He is my constant reminder that I still have a lot to learn about what we are working on. Prof. Xue Li is the guide that opened the door of conducting research at UQ for me. Despite his seniority, I can always feel his endless energy and passion for research whenever we have a chat, while his kindness and insistence are the benchmarks I need to bear in mind throughout my lifetime. Apart from my advisory team, I would like to thank many other academic and professional staff at UQ and other institutes, especially A/Prof. Helen Huang, A/Prof. Jun Zhang, Prof. Yang Zhang, Prof. Lin Wu, Tracey Miller, Rosa Armitage, Diana Cassidy, and Kathleen Williamson, who have also made great efforts in offering recognition of my research work, providing knowledgeable insights to my academic pathways, and tackling loads of headaches in the PhD-related administrative tasks.

I have been meeting fantastic friends and colleagues too. I have known my best friend, Dr. Hongxu Chen since the first day I arrive in Australia. We have been through a lot of unforgettable moments, which dates back to working on our first publications with seven coffee shots drank in a row in the late night – by the way the instant coffee was not as good as our much preferred Merlo in the UQ campus. The memory on the intensity of preparing for our confirmation seminars is still fresh, where we practised our 40-minute presentations over and over again and never got tired with them. Of course, the encounter with amazing people does not stop there. I also greatly appreciate Dr. Weiqing Wang, Dr. Xingzhong Du, Dr. Jingwei Ma, Dr. Weitong Chen, Dr. Abdulqader Almars, as well as all members from Dr. Hongzhi Yin's RSBDI lab, who are colleagues with great knowledge and personalities that I feel wonderful to work, chat, and make friend with.

The last paragraph is exclusively reserved for my family. I feel so fortunate and happy to finish this PhD journey with the companion of my wife Yue Xu. It is hard for me to imagine what my life would be without her. For most of the time, Yue is the first person that I share my progress, success, or hardship with. Though her expertise has nothing to do with computer science, she is always willing to be the only audience for my "dreary" presentation rehearsals and "nerdy" conversations, offering me tons of encouragement along the way. Sometimes I do regret for contaminating her brain with computer science terminologies, but what has amazed me the most is that she feels what I feel, and knows what I think. While it is not easy to explain to others why a 10-page article matters to a PhD, I can never forget how happy Yue was when I received my first paper acceptance notification from a top-tier conference. I would like to deeply thank my wife for kindly understanding my long working hours and forgiving my escape from housekeeping duties (well, that is occasional) caused by tight deadlines. Meanwhile, my greatest gratitudes goes to my parents, who has been offering selfless,

unceasing, and immense support from both financial and mental perspectives. Without their support, I would have never been able to pursue a PhD degree overseas. I am very grateful for the fact that they are always with me, regardless of the decisions I make and the life journey I choose to take. When I was still hesitating about the idea of studying overseas, they helped me made up my mind by erasing all my concerns. When I was uncertain about my future career directions, they are the ones that tell me the most important thing is to seek opportunities for self-improvement. They do not interfere with my decision making, but rather, they simply provide the most powerful inspiration and the most sincere praise at every milestone they had witnessed. To this end, I would like to express my deepest love and thankfulness to my family as an indispensable part of my entire PhD candidature.

Financial Support

This project is supported by Australian Research Council. The grant numbers involved are: DE160100308, LP150100671, DP160104075, DP170103954, and DP190101985. This project is also partially supported by the New Staff Research Grant of The University of Queensland (grant number 613134).

Keywords

Machine learning; data mining; recommender systems; sequence modelling; predictive analytics; deep neural networks.

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC Code: 080109, Pattern Recognition and Data Mining, 70%
ANZSRC Code: 080607, Information Engineering and Theory, 30%

Fields of Research (FoR) Classification

FoR Code: 0801, Artificial Intelligence and Image Processing, 50%
FoR Code: 0806, Information Systems, 50%

Contents

Abstract ii

Contents **xi**

List of Figures **xiv**

List of Tables **xv**

1 Introduction **1**

1.1 Background 1

1.2 Sales Prediction 2

1.3 Sequential Recommendation 3

1.4 Temporal Predictive Analytics 3

1.5 Thesis Organization 3

2 Macro-Level Sequence Modelling **7**

2.1 Literature Review: Background and Motivation 7

2.1.1 Evolution of Time Series Prediction Models 7

2.1.2 Trend Alignment for Sales Prediction 8

2.1.3 Motivation and Our Solution 9

2.2 The Proposed Model: TADA 10

2.2.1 Problem Formulation 10

2.2.2 Multi-Task Encoder Structure 11

2.2.3 Dual-Attention Decoder Structure 13

2.2.3.1 Attention for Weighted Decoder Input Mapping 13

2.2.3.2 Attention for Trend Alignment 14

2.2.4 Sales Prediction and Model Learning 16

2.2.5 Time Complexity of TADA 16

2.3 Experiments 17

2.3.1 Datasets and Features 17

2.3.2 Parameters and Experimental Settings 19

2.3.3 Discussion on Effectiveness (RQ1) 21

2.3.4 Importance of Key Components (RQ2) . . . 22
2.3.5 Training Efficiency and Scalability (RQ3) . . . 23
2.4 Summary . . . 24

3 Micro-Level Sequence Modelling 27
3.1 Literature Review: Background and Motivation . . . 27
3.1.1 General Top-*k* Recommendation . . . 27
3.1.2 Sequential Recommender Systems . . . 28
3.1.3 Motivation and Our Solution . . . 28
3.2 The Proposed Model: AIR . . . 30
3.2.1 Preliminaries and Problem Formulations . . . 30
3.2.2 User Intention Prediction . . . 31
3.2.3 Network Structure . . . 31
3.2.4 Modelling Intention Migration via Attention Mechanism . . . 32
3.2.5 User Intention Prediction and Model Learning . . . 33
3.2.6 Intention-Aware Sequential Recommendation . . . 35
3.2.7 Category-wise Aggregation of User Intention . . . 35
3.2.8 ITFM: Intention-Aware Temporal Factorization Machines . . . 35
3.2.9 ITFM Generalizes FM . . . 37
3.2.10 Loss Function for ITFM . . . 37
3.2.11 Model Optimization . . . 38
3.3 Experiments . . . 38
3.3.1 Datasets . . . 39
3.3.2 Evaluation Criteria . . . 40
3.3.3 Baseline Methods . . . 40
3.3.4 Parameter Settings . . . 41
3.3.5 Overall Recommendation Effectiveness (RQ1) . . . 41
3.3.5.1 Comparisons with Baselines . . . 41
3.3.5.2 Sequential Recommendation Simulation . . . 43
3.3.6 Importance of Each Model Component (RQ2) . . . 43
3.3.6.1 Overview . . . 43
3.3.6.2 The Impact of Category-wise User Intention . . . 45
3.3.6.3 ARNN for User Intention Prediction . . . 45
3.3.6.4 ITFM for Intention-Aware Recommendation . . . 45
3.3.7 Analysis on Hyperparameters (RQ3) . . . 46
3.3.8 Model Scalability (RQ4) . . . 47
3.4 Summary . . . 48

4 Uniting Macro- and Micro-Level Sequence Modelling 51
4.1 Literature Review: Background and Motivation . . . 51

4.1.1 Predictive Analytics under Sparsity . . . 51
4.1.2 Evolution of Factorization Machines . . . 52
4.1.3 Motivation and Our Solution . . . 52
4.2 The Proposed Model: SeqFM . . . 54
4.2.1 Preliminaries . . . 54
4.2.2 Sequence-Aware Factorization Machines . . . 55
4.2.3 Embedding Layer . . . 56
4.2.4 Static View with Self-Attention . . . 56
4.2.5 Dynamic View with Self-Attention . . . 57
4.2.6 Cross View with Self-Attention . . . 58
4.2.7 Intra-View Pooling Operation . . . 59
4.2.8 Shared Residual Feed-Forward Network . . . 59
4.2.9 View-Wise Aggregation . . . 60
4.2.10 Output Layer . . . 61
4.2.11 Time Complexity Analysis . . . 61
4.3 Applications and Optimization of SeqFM . . . 61
4.3.1 SeqFM for Ranking . . . 61
4.3.2 SeqFM for Classification . . . 62
4.3.3 SeqFM for Regression . . . 63
4.3.4 Optimization Strategy . . . 63
4.4 Experiments . . . 64
4.4.1 Datasets . . . 64
4.4.2 Baseline Methods . . . 65
4.4.3 Evaluation Metrics . . . 66
4.4.4 Parameter Settings . . . 67
4.4.5 Prediction Performance (RQ1) . . . 67
4.4.6 Impact of Hyperparameters (RQ2) . . . 69
4.4.7 Importance of Key Components (RQ3) . . . 71
4.4.8 Training Efficiency and Scalability (RQ4) . . . 72
4.5 Summary . . . 72

5 Conclusion **75**

Bibliography **77**

List of Figures

1.1 Relationships among the three tasks studied in this thesis. 2

2.1 Illustration of the proposed framework TADA. 9
2.2 The unfolded structure of our proposed multi-task LSTM encoder. 11
2.3 Demonstration of proposed attention mechanism for weighted input mapping. 13
2.4 Demonstration of proposed attention mechanism for trend alignment. 15
2.5 Demonstration of proposed trend alignment scheme in TADA with attention mechanism. 22
2.6 The training time of TADA with varied proportions of training data. 24

3.1 Demonstration of proposed attention mechanism for intention migration modelling. . . . 34
3.2 Recommendation results w.r.t. $Hits@k$ on two datasets. 42
3.3 The relationship between $\mathscr{L}_{ARNN}$, $\mathscr{L}_{ITFM}$ and $Hits@20$. 43
3.4 Relationship between the predicted and aggregated user intention $\widetilde{\boldsymbol{\gamma}}$ (left) and top 5 recommendation results (right) on MovieLens. 44
3.5 The intention migration matrix Θ learned via the proposed attention mechanism in AIR. . 44
3.6 Parameter sensitivity analysis on the number of LSTM layers and the size of hidden dimension. 46
3.7 Training time for AIR w.r.t. varied data ratio. 47

4.1 The differences in feature interaction modelling between traditional FM-based models (upper part) and our proposed SeqFM (lower part). Note that the embedding process of sparse features is omitted to be succinct. 54
4.2 The overall architecture of SeqFM. 57
4.3 Parameter sensitivity analysis. 70
4.4 Training time of SeqFM w.r.t varied data proportions. 72

List of Tables

2.1 Statistics of datasets in use. 17
2.2 Features extracted from datasets. 19
2.3 Sales prediction results under the offline setting. 20

3.1 Description of major notations used in this chapter. 30
3.2 Statistics of datasets in use. 40

4.1 Statistics of datasets in use. 65
4.2 Ranking task (next-POI recommendation) results. 67
4.3 Classification task (CTR prediction) results. 68
4.4 Regression task (rating prediction) results. 69
4.5 Ablation test with different model architectures. 71

Chapter 1

Introduction

1.1 Background

We are now living in a world where data is constantly and rapidly collected, processed, and analyzed. In the recent two decades, with the proliferation of big data, we have witnessed the growth and prosperity of a revolutionary business form – electronic commerce (e-commerce). E-commerce companies like Amazon, eBay, and Alibaba provide online trading platforms for products and services, bringing immense convenience to customers' daily lives. As e-commerce enables online business transactions, generating large-scale and real-time data has never been easier. Consequently, leveraging machine learning techniques to discover the behavioral patterns from online users' transaction data opens up opportunities to understand user preferences, offer personalized experiences, and eventually maximize customer values. In e-commerce, as data is usually gathered in a chronological manner, the immense availability of sequential data has attracted substantial research attention. Generally, there are three main characteristics of sequential data in e-commerce. (1) High velocity: the data generated by both customers and service providers can accumulate at a fast rate. (2) High diversity: in e-commerce, sequential data usually contains multiple variables that are dynamically changing. (3) Rich patterns: sequential dependencies and transition patterns are the key to success for many e-commerce applications. On one hand, for companies and business owners, mining the latent patterns within the e-commerce sequence paves the way for performing timely predictive analytics and increasing business revenue. On the other hand, for individual customers, the outcomes of sequence mining on e-commerce data are essential for delivering accurate personalized recommendations and refining user experience.

Given the multifaceted advantages of sequence modelling for e-commerce, this thesis aims to ***systematically investigate the pathways towards effective and efficient utilization of the sequential e-commerce data for driving various business benefits***. Specifically, in the context of e-commerce, we divide the applications of sequence modelling into two major categories – ***macro-level sequence modelling*** and ***micro-level sequence modelling***. Macro-level sequence modelling is commonly related to quantitative analysis on the dynamic patterns of e-commerce data, and representative tasks

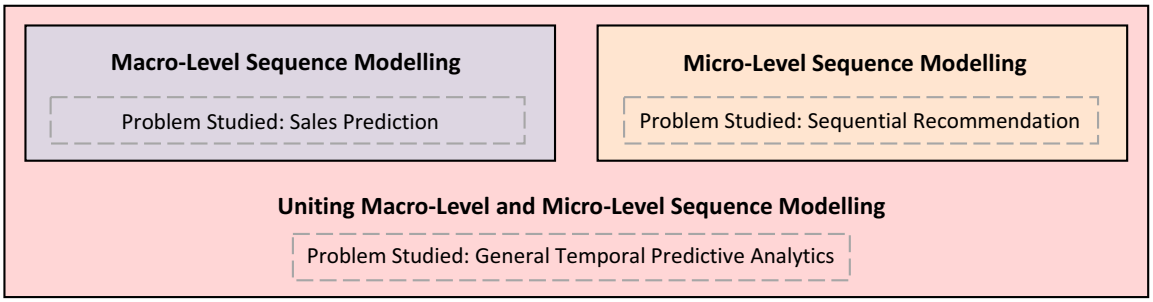


Figure 1.1: Relationships among the three tasks studied in this thesis.

are commonly prediction-related, e.g., sales prediction, stock price forecasting, click-through rate prediction, etc. As a beneficial tool for decision-makers of e-commerce businesses, it is useful for understanding high-level trends and phenomenons, and is indispensable for setting the foundation for strategic planning. In contrast, micro-level sequence modelling concentrates more on learning the subtle and dynamic information. While having a stronger focus on deriving fine-grained business insights, it is usually adopted to advance users' experience, e.g., learning user preferences for personalized recommendation, which eventually yields higher customer satisfaction as well as increased business revenue in return. In light of this, in this thesis, we will firstly focus on two research tasks, namely sales prediction and sequential recommendation, which correspond to sequence modelling at the macro-level and micro-level, respectively. Then, we will thoroughly study the general problem of temporal predictive analytics (including ranking, classification, and regression tasks) by combining the capability of performing both macro- and micro-level sequence modelling with a unified and versatile machine learning model. The inherent relationships among the three proposed tasks are illustrated via Figure 1.1. In what follows, we provide an overview for each individual task, and set the task-specific research goal we would like to achieve in this thesis.

1.2 Sales Prediction

In this thesis, we first solve the problem of sales prediction, which is a typical macro-level sequence modelling task in e-commerce. Keeping a balance between supply and demand is crucial to retailers, and the accurate prediction of sales volume is becoming indispensable for commercial success [16]. Overestimated sales can result in excessive inventory, unhealthy cash flow and even bankruptcy, while the underestimated sales may lead to unfulfilled orders, decreased business reputation and profit [17]. In practice, sales prediction is formulated as a time series forecasting problem, which aims to predict future sales volume based on the observed multivariate time series data which consists of historical sales volume and influential factors including brand, season, discount, etc. Thus, reasonable modelling of the influential factors and historical sales information should be performed to successfully predict sales volume. The research goal in this task is to: (1) review, test, and analyze state-of-the-art time series prediction models in terms of their efficacy in sales prediction; and (2) propose a new model to advance the performance in real-life sales prediction applications.

1.3 Sequential Recommendation

Recommender systems have already demonstrated their strong benefits to both online service platforms and common users as they grant users easier access to preferred resources and help service providers understand their customers better. In the context of top-k recommendation where the goal is to recommend k items that a user is likely to interact with in the near future, approaches like matrix factorization [18,19] and factorization machines [20,21] achieve great success in top-k recommendation with the assumption that user preference is static. However, while the numbers of both users and items are now growing exponentially over time, it is more practical to investigate the problem of sequential top-k recommendation nowadays as the dynamics of the data play a pivotal role in recommender systems. Unlike conventional top-k recommendation, the sequential top-k recommendation approaches model the user behavior as a sequence of items instead of a set of items [22]. The research goal in this task is to: (1) review, test, and analyze the effectiveness of the latest sequential recommendation approaches; and (2) propose a novel sequential recommendation model that achieves state-of-the-art performance under the sequential top-k recommendation setting.

1.4 Temporal Predictive Analytics

Lastly, we make an attempt to unite both macro- and micro-level sequence modelling with a versatile machine learning model, thus enabling effective solutions to a broader range of e-commerce applications. In e-commerce scenarios, as an important supervised learning scheme, temporal predictive analytics play a pivotal role in various tasks, ranging from ranking (e.g., recommender systems [23,24]) to regression (financial analysis [25, 26]) and classification (online advertising [27, 28]). While classic methods like logistic regression and support vector machines tend to fail with the commonly high-dimensional but sparse categorical features in predictive analytics, the factorization machine (FM) [20] is a well-established model for handling sparse features via feature interactions. In recent years, a large body of FM variants are proposed to better capture the effect of feature interactions, and are successfully applied to temporal predictive analytics. Hence, the research goal in this task is to: (1) review, test, and analyze existing FM-based models' adaptivity and effectiveness on temporal predictive analytics; and (2) propose a new, versatile, and high-performance FM-based model that can be generalized to various temporal ranking, classification, and regression tasks.

1.5 Thesis Organization

In the rest part of the thesis, we will present our detailed pathways towards the completion of each research task. Specifically, Chapters 2, 3 and 4 respectively focus on the tasks of sales prediction, sequential recommendation, and temporal predictive analytics. In each of the three chapters, we will thoroughly review the existing literatures in related areas to identify their technical characteristics and shortcomings. We will then introduce our proposed solution to each research task in detail, followed

by extensive experimental studies on large-scale real datasets to verify the efficacy of our technical innovations. It is worth mentioning that, due to the breadth and diversity of research domains relevant to the thesis, we carry out a comprehensive literature review for every single research task to make it easier to follow and keep the thesis organized. At last, we will conclude all the findings and research merits achieved by this thesis in Chapter 5, and discuss possible directions of future research on sequence modelling for e-commerce.

Chapter 2

Macro-Level Sequence Modelling

In this chapter, we investigate a representative macro-level sequence modelling problem in e-commerce – sales prediction. We will start with the research background in relation to the core techniques behind it, which are time series prediction models. Afterwards, we will formally present our proposed model for the sales prediction task named Trend Alignment with Dual-Attention Recurrent Neural Networks (TADA), whose superiority in this task is further proved by experiments on real-life commercial data.

2.1 Literature Review: Background and Motivation

2.1.1 Evolution of Time Series Prediction Models

Sales prediction is essentially a time series forecasting problem, which aims to predict future sales volume based on the observed multivariate time series data which consists of historical sales volume and ***influential factors*** (e.g., brand, season, discount, etc.). In this regard, the techniques can be divided into linear models and non-linear models. While linear models like autoregressive integrated moving average (ARIMA) [29], support vector machine (SVM) [30] and robust regression [31] mostly aim at finding parameterized functions from statistics, non-linear models like Gaussian process [32, 33] and gradient boosting machines [34, 35] can better model complicated dependencies by leveraging machine learning techniques. However, due to the high computational cost and unsatisfying scalability in real applications [36, 37], these approaches are not ideal for sales time series which usually carries high dimensionality and long time range. In addition, these methods mainly rely on carefully designed mapping functions, so sufficient domain knowledge of the data is a prerequisite. To address this issue, recurrent neural network (RNN) [38], along with its two popular variants, namely long short-term memory (LSTM) [39] and gated recurrent unit (GRU) [40] have been proposed to dynamically capture long-range dependencies among the sequential data via a flexible non-linear mapping from the inputs to the outputs.

Attempts on time series modelling using RNNs have demonstrated the efficacy of RNNs in various time series prediction tasks, such as dynamic location prediction [41, 42] and user satisfaction

prediction [43]. In the aforementioned applications, a single RNN is leveraged to learn discriminative hidden states from the raw sequential inputs, and the last hidden state in a sequence is used to generate desired output. As real-life tasks get more complex, the one-step prediction result generated from the last hidden state of a single RNN no longer suit the demand. Consequently, the encoder-decoder network is first proposed in neural machine translation scenarios [40, 44], which further inspires relevant researches on multi-step ahead time series prediction [45–47].

2.1.2 Trend Alignment for Sales Prediction

In recent years, time series prediction algorithms are widely adopted in many areas such as stock price prediction [48, 49] and medical data processing [50, 51]. Among these applications, the discovery of trending events or repeating patterns based on the clues from historical observations has inspired some interesting applications like traffic modelling [37], solar intensity prediction [52] and argument discovery [53]. Undoubtedly, the discovery of recurring trends will greatly benefit the forecast of sales by aligning relative contextual information learned from the influential factors, and this insight is referred to as ***trend alignment*** in this thesis. However, both traditional autoregressive-based methods [29, 54, 55] and recent trend mining models [37, 56] are ineffective for trend alignment in sales prediction. This is because these methods assume the trend in time series data recurs periodically (i.e., distributes with a fixed time period), thus requiring domain knowledge for every application area and carefully chosen parameters based on the data. Hence, existing techniques are unable to align similar trends in sales time series where the sales patterns are much more subtle and irregular due to the effect from complicated real-world situations, and the difficulty increases when there are a large number of different products.

The formation of a trend in sales time series has specific contexts which can be modelled from the interaction among various influential factors. In regards to contextual information learning from raw time series, recurrent neural network (RNN) models have been intensively studied and applied to learn vector representations from sequential inputs [37, 57]. Compared with previous efforts on time series prediction like kernel methods and Gaussian process [32, 58] which are limited by their predefined non-linear form, RNNs show their advantages in flexible yet discriminative non-linear relationship modelling. Moreover, two variants of RNN, namely long short-term memory (LSTM) [39] and gated recurrent unit (GRU) [40] further advance the performance in tasks related to neural machine translation [59] and image captioning [60]. Among these applications, the encoder-decoder RNN architecture leverages two independent RNNs to encode sequential inputs into latent ***contextual vectors*** and decode these contexts into desired interpretations [44, 59, 60]. After showing its superiority in recent time series modelling tasks [49, 51], it is natural to consider encoder-decoder RNNs for sales prediction by leveraging its capability to fully capture the non-linear relationship between the influential factors and the sales volume.

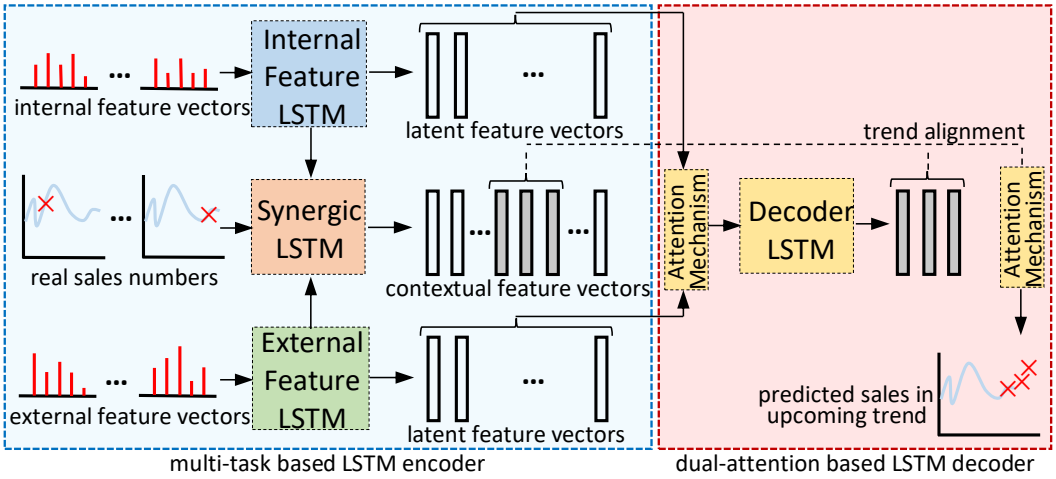


Figure 2.1: Illustration of the proposed framework TADA.

2.1.3 Motivation and Our Solution

However, even with the state-of-the-art encoder-decoder RNN models, sales prediction is still a challenging research problem because when multiple influential factors interact with each other, they have different influences on different products. For instance, temperature has more impact on the sales of down jackets than shirts because shirts are intrinsically cheaper and can be worn all year round. Furthermore, the influential factors are dynamic and unpredictable in many cases, so it is impractical to assume their future availability. For example, though environmental policy significantly affects electrical car sales, and fashion trend dominates the clothing industry, we have very limited prevision on these influential factors. To make things worse, when performing trend alignment using contexts learned from the past, the decoder cannot generate rich contexts with the unknown states of influential factors. Hence, the main challenges in sales prediction are summarized as follows. The first is how to ***fully capture the dynamic dependencies among multiple influential factors***. Secondly, without any prior knowledge of mutative variables in the future, how can we possibly ***glean wisdoms from the past to compensate for the unpredictability of influential factors***. Third, as different sales trends recur irregularly due to complex real-world situations, it is necessary to ***align the upcoming trend with historical sales trends***, thus selectively gather relative contextual information for accurate prediction of sales volume.

In light of these challenges, we propose a novel sales prediction framework, namely Trend Alignment with Dual-Attention Multi-Task Recurrent Neural Network for Sales Prediction (TADA). TADA consists of two major components: the multi-task LSTM encoder and the dual-attention LSTM decoder, which are illustrated in Figure 4.1. In order to solve the first challenge, we make our own observation on the characteristics of sales time series based on previous discussions. The semantics of influential factors in sales prediction are diverse, which however has been ignored by the conventional time series prediction methods. Specifically, for each product, its influential factors come with its intrinsic properties which are directly related to customers' subjective preference, e.g., brand, category, price, etc. Meanwhile, there are also many factors that objectively affects the sales, e.g., weather,

holiday, promotion, etc. In this work, we categorize the intrinsic properties of a product as its ***internal feature*** and the other influential factors as the ***external feature***. While internal features and external features express different semantic meanings, they both contribute to the fluctuations of the product sales volume at the same time. Hence, compared with predictive models that treat all kinds of features in a unified way [37, 49, 61], we propose a multi-task LSTM encoder to learn contextual vector representations of historical sales time series. As shown in Figure 4.1, to solve the first challenge, we novelly model the internal feature and external feature in parallel via two individual LSTM layers. Then, we use a synergic LSTM layer to simultaneously join these two learned latent representations at each time step. The insight of a multi-task encoder structure is to comprehensively leverage all available resources by modelling internal and external features separately first, and then pose a dynamic interaction between different features to generate contextual representations of historical sales time series.

To address the challenges of trend alignment and unknown influential factors, we propose an innovative dual-attention LSTM decoder to tackle the difficulties. Grasping intuitions from existing attention mechanisms [59,62] which aim to select relevant parts of hidden states learned by the encoder to attend, we develop our simple yet effective attention mechanisms which perfectly blend into the neural network for accurate sales prediction. As illustrated in Figure 4.1, in the decoding stage, the first attention models the effect of unknown influential factors using relevant contextual vectors from the encoder. After new sales contexts are generated within the look-ahead time interval, the second attention gathers contextual information of this upcoming trend, and then actively aligns the new trend with historical ones. Eventually, we combine the representation from the aligned trends to produce a sequence of estimated sales volume in the future.

2.2 The Proposed Model: TADA

In this section, we first mathematically formulate the definition of sales prediction and then we present the technical details of our proposed model TADA. Finally, we introduce the loss function and optimization strategy.

2.2.1 Problem Formulation

The objective of sales prediction is to predict future sales volume according to multivariate observations (e.g., previous sales, weather, price, promotion, etc.) from the past. The formulation of sales prediction is similar to, but different from multivariate time series forecasting and autoregressive models (AR). Formally, for an arbitrary product, the input is defined as its fully observed feature vector set $\{\mathbf{x}_t\}_{t=1}^{T} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T\}$ and the corresponding sales volume $\{y_t\}_{t=1}^{T} = \{y_1, y_2, ..., y_T\}$ at time step t. Here, $\mathbf{x}_t \in \mathbb{R}^n$, $y_t \in \mathbb{R}$ and n is variable according to the feature dimension, while T is the amount of total time steps. The output of sales prediction is the estimated sales volume of following Δ time steps after T, denoted as $\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = \{\hat{y}_{T+1}, \hat{y}_{T+2}, ..., \hat{y}_{T+\Delta}\}$, where Δ is adjustable according to

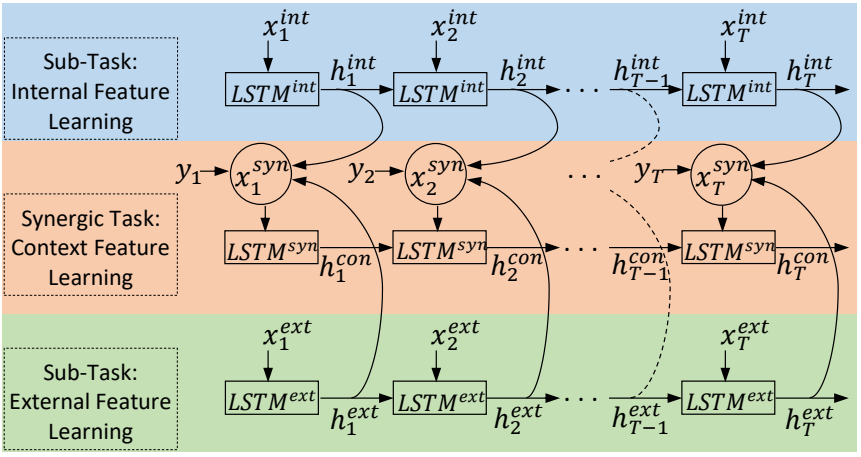


Figure 2.2: The unfolded structure of our proposed multi-task LSTM encoder. Two sub-tasks consist of internal feature learning and external feature learning LSTMs, denoted by $LSTM^{int}$ and $LSTM^{ext}$ respectively. After latent representations of both internal and external features are generated, they are combined with the real sales number $\{\mathbf{y}_t\}_{t=1}^{T}$ to compute the contextual vectors $\{\mathbf{h}_t^{con}\}_{t=1}^{T}$ via the synergic task LSTM ($LSTM^{syn}$).

the business goal. In this work, we assume $\Delta \ll T$ to ensure the prediction accuracy because $\{\mathbf{x}_t\}_{t=T+1}^{T+\Delta}$ is non-available in the prediction stage.

Importantly, compared with multivariate time series forecasting and AR, sales prediction models behave differently. This is because our target is to acquire the one-dimensional scalar representing the sales volume without prior knowledge of the features in the future. Meanwhile, in multivariate time series forecasting, the output is specifically $\{\mathbf{x}_t\}_{t=T+1}^{T+\Delta}$, which has the same form and contextual meaning of its input [37]. Also, the AR assumes $\{\mathbf{x}_t\}_{t=T+1}^{T+\Delta}$ is available when predicting $\{\hat{y}_t\}_{t=T+1}^{T+\Delta}$ [49] because it is designed to model a mapping function between conditions and consequences.

Hence, we formulate sales prediction as a non-linear mapping from time series features $\{\mathbf{x}_t\}_{t=1}^{T}$ and real sales $\{y_t\}_{t=1}^{T}$ in the history to the estimation of sales volume $\{\hat{y}_t\}_{t=T+1}^{T+\Delta}$ with Δ time steps ahead:

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = F\left(\{\mathbf{x}_t\}_{t=1}^{T}, \{y_t\}_{t=1}^{T}\right), \tag{2.1}$$

where $F(\cdot)$ is the non-linear mapping function to learn.

2.2.2 Multi-Task Encoder Structure

Taking a time series $\{\mathbf{x}_t\}_{t=1}^{T}$ as input, recurrent neural network (RNN) encodes $\{\mathbf{x}_t\}_{t=1}^{T}$ into hidden states $\{\mathbf{h}_t\}_{t=1}^{T}$ via $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$, where $f(\cdot)$ is a non-linear mapping function. To capture the long-range dependency, we leverage RNNs with long short-term memory architecture (LSTM) via the

following formulation [39]:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f),\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o),\\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c),\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned} \tag{2.2}$$

where $\odot$ denotes element-wise multiplication and the recurrent activation σ is the *Logistic Sigmoid* function. $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ and $\mathbf{c}$ are respectively the input gate, forget gate, output gate, and cell state vectors. When updating each of them, there are corresponding trainable input-to-hidden and hidden-to-hidden weights $\mathbf{W}$ and $\mathbf{U}$ along with the bias vectors $\mathbf{b}$.

For sales prediction, ***internal feature*** and ***external feature*** are two kinds of features with different semantic meanings in sales time series. We use $\{\mathbf{x}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{x}_t^{ext}\}_{t=1}^{T}$ to denote the feature vectors of internal and external information in sales time series respectively. As we discussed in previous sections, internal features carry information of intrinsic attributes directly linked with the product like store location and item category, while the external features store information of extrinsic attributes viewed as external influential factors like weather condition and holiday. As a result, a single LSTM structure may suffer from loss of contextual information as it maps all raw features into one unified space, as we will reveal in Section 2.3. Hence, we use two LSTMs in parallel to effectively capture the different semantics by treating internal and external feature modelling as two sub-tasks. Correspondingly, we extend the problem formulation in Eq.(2.1) as:

$$\{\hat{y}_t\}_{t=T+1}^{T+\Delta} = F\left(\{\mathbf{x}_t^{int}\}_{t=1}^{T}, \{\mathbf{x}_t^{ext}\}_{t=1}^{T}, \{y_t\}_{t=1}^{T}\right). \tag{2.3}$$

Figure 2.2 demonstrates our proposed encoder architecture. We use $\{\mathbf{h}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{h}_t^{ext}\}_{t=1}^{T}$ to denote the latent representations learned from $\{\mathbf{x}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{x}_t^{ext}\}_{t=1}^{T}$. After the hidden states are learned from both sub-tasks, we simultaneously feed those hidden states into a synergic LSTM layer to learn a joint representation, namely ***contextual vectors*** denoted by $\{\mathbf{h}_t^{con}\}_{t=1}^{T}$ at all T time steps in the sales time series. Furthermore, to enhance the expressive ability of the encoder, instead of adopting $\{y_t\}_{t=1}^{T}$ to calculate the prediction loss, we fuse $\{y_t\}_{t=1}^{T}$ with hidden states from both internal and external encoding LSTMs to calculate the input $\{\mathbf{x}_t^{syn}\}_{t=1}^{T}$ for the synergic layer:

$$\mathbf{x}_t^{syn} = \mathbf{W}_{syn}[\mathbf{h}_t^{int}; \mathbf{h}_t^{ext}; y_t] + \mathbf{b}_{syn}, \tag{2.4}$$

where $[\mathbf{h}_t^{int}; \mathbf{h}_t^{ext}; y_t]$ represents the concatenation of $\mathbf{h}_t^{int}$, $\mathbf{h}_t^{ext}$ and y_t while $\mathbf{W}_{con}$ and $\mathbf{b}_{con}$ are weights and biases to be learned. For notation convenience, we format the multi-task encoder structure into the following equations:

$$\begin{aligned}
\mathbf{h}_t^{int} &= LSTM^{int}(\mathbf{x}_t^{int}, \mathbf{h}_{t-1}^{int}),\\
\mathbf{h}_t^{ext} &= LSTM^{ext}(\mathbf{x}_t^{ext}, \mathbf{h}_{t-1}^{ext}),\\
\mathbf{h}_t^{con} &= LSTM^{syn}(\mathbf{x}_t^{syn}, \mathbf{h}_{t-1}^{con}),
\end{aligned} \tag{2.5}$$

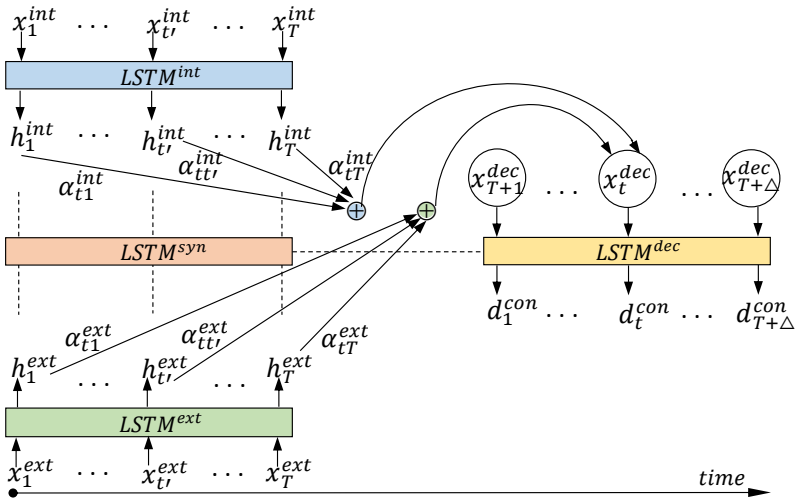


Figure 2.3: Demonstration of proposed attention mechanism for weighted input mapping. The details of $LSTM^{syn}$ are omitted for a clearer view. With the calculated attention weights $\alpha_{tt'}^{int}$ and $\alpha_{tt'}^{ext}$, the latent representations generated by $LSTM^{int}$ and $LSTM^{ext}$ are mapped into the input vectors $\{\mathbf{x}_t^{dec}\}_{t=T+1}^{T+\Delta}$ for the decoder $LSTM^{dec}$.

where $LSTM^{int}(\cdot)$, $LSTM^{ext}(\cdot)$ and $LSTM^{syn}(\cdot)$ denote internal, external and synergic LSTM encoders respectively. Note that the trainable weights are not shared across different LSTM layers in our multi-task encoder structure.

2.2.3 Dual-Attention Decoder Structure

After encoding the entire historical sales time series with the multi-task encoder, we have the ***contextual vectors*** $\{\mathbf{h}_t^{con}\}_{t=1}^{T}$ where each $\mathbf{h}_t^{con}$ carries contextual information of the sales time series at time step t. The latent representations, $\{\mathbf{h}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{h}_t^{ext}\}_{t=1}^{T}$ for internal and external features are also learned. To predict the desired sales volume $\{\hat{y}_t\}_{t=T+1}^{T+\Delta}$, we adopt a LSTM decoder to mimic the ***contextual vectors*** in the following Δ time steps. Similar to Eq.(2.5), when $T < t \leq T+\Delta$, we have:

$$\mathbf{d}_t^{con} = LSTM^{dec}(\mathbf{x}_t^{dec}, \mathbf{d}_{t-1}^{con}), \tag{2.6}$$

where $\mathbf{d}_t^{con} \in \{\mathbf{d}_t^{con}\}_{t=T+1}^{T+\Delta}$ is the contextual vector to learn in the decoding stage at time step t, $LSTM^{dec}(\cdot)$ is the decoder with the same formulation as Eq.(2.2), $\mathbf{x}_t^{dec}$ is the ***attention-weighted*** input for the decoder and $\mathbf{d}_{t-1}^{con}$ is the previous decoder hidden state.

2.2.3.1 Attention for Weighted Decoder Input Mapping

According to the problem formulation, we assume that both $\{\mathbf{x}_t^{int}\}_{t=T+1}^{T+\Delta}$ and $\{\mathbf{x}_t^{ext}\}_{t=T+1}^{T+\Delta}$ are non-available in the decoding stage because both of them contain attributes unknown to the future, such as price as an internal feature and weather as an external feature. Thus, to formulate the decoder input at time $t > T$, we propose an attention mechanism to dynamically select and combine relevant contextual vectors from $\{\mathbf{h}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{h}_t^{ext}\}_{t=1}^{T}$ with:

$$\mathbf{x}_t^{dec} = \mathbf{W}_{dec}\left[\sum_{t'=1}^{T}\alpha_{tt'}^{int}\mathbf{h}_{t'}^{int}; \sum_{t'=1}^{T}\alpha_{tt'}^{ext}\mathbf{h}_{t'}^{ext}\right] + \mathbf{b}_{dec}, \tag{2.7}$$

where $\alpha_{tt'}^{int}$ and $\alpha_{tt'}^{ext}$ denote the attention weights mapped to t'-th hidden states of internal and external feature encoder, respectively. We use Fig.2.3 to illustrate the attention for weighted decoder input mapping process. We enforce $\sum_{t'=1}^{T}\alpha_{tt'}^{int}=\sum_{t'=1}^{T}\alpha_{tt'}^{ext}=1$, so that $[\cdot]$ in Eq.(2.7) can be viewed as the concatenation of two probability expectations from $\{\mathbf{h}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{h}_t^{ext}\}_{t=1}^{T}$. The rationale is that we simulate $\mathbf{x}_t^{dec}$ by summarizing varied influences from all $2T$ historical hidden states of both internal and external feature. The influences are computed through quantifying the relevance between $\mathbf{d}_{t-1}^{con}$ and each $\mathbf{h}_{t'}^{int}$, $\mathbf{h}_{t'}^{ext}$:

$$\begin{aligned} e_{tt'}^{int} &= \mathbf{v}_{int}^{\top}\tanh(\mathbf{M}_{int}\mathbf{d}_{t-1}^{con}+\mathbf{H}_{int}\mathbf{h}_{t'}^{int}),\\ e_{tt'}^{ext} &= \mathbf{v}_{ext}^{\top}\tanh(\mathbf{M}_{int}\mathbf{d}_{t-1}^{con}+\mathbf{H}_{ext}\mathbf{h}_{t'}^{ext}), \end{aligned} \tag{2.8}$$

where $e_{tt'}^{int}$ and $e_{tt'}^{ext}$ are the relevance scores mapped to t'-th hidden states in $\{\mathbf{h}_t^{int}\}_{t=1}^{T}$ and $\{\mathbf{h}_t^{ext}\}_{t=1}^{T}$ for the decoder input at time t, while $\mathbf{v}_{int}$, $\mathbf{v}_{ext}$, $\mathbf{M}_{int}$, $\mathbf{v}_{ext}$, $\mathbf{H}_{int}$ and $\mathbf{H}_{ext}$ are parameters to learn. In particular, Eq.(2.8) compares two hidden states with different semantic meanings. Intuitively, this is a scoring scheme that shows how well two vectors are correlated by projecting them into a common space. Afterwards, we apply $SoftMax$ on both attention weights:

$$\begin{aligned} \alpha_{tt'}^{int} &= \frac{\exp(e_{tt'}^{int})}{\sum_{s=1}^{T}\exp(e_{ts}^{int})},\\ \alpha_{tt'}^{ext} &= \frac{\exp(e_{tt'}^{ext})}{\sum_{s=1}^{T}\exp(e_{ts}^{ext})}, \end{aligned} \tag{2.9}$$

which enforces $\sum_{t'=1}^{T}\alpha_{tt'}^{int}=\sum_{t'=1}^{T}\alpha_{tt'}^{ext}=1$.

2.2.3.2 Attention for Trend Alignment

Ideally, at time t, each acquired contextual vector in $\{\mathbf{h}_t^{con}\}_{t=1}^{T}$ and $\{\mathbf{d}_t^{con}\}_{t=T+1}^{T+\Delta}$ carries contextual information of both time t and previous time steps. However, as discussed in [40, 49], the performance of the encoder-decoder networks decrease significantly when the length of time series grows. To alleviate the problem, traditional attention mechanisms have been designed to align the current output with the targeted input by comparing the current hidden state with the ones generated at previous time steps. Meanwhile, these methods are not applicable as we aim to match similar trends for the prediction period Δ, and we propose a novel attention mechanism for trend alignment. Mathematically, we represent a Δ-step trend in sales time series as the concatenation of Δ successive contextual vectors in $\{\mathbf{h}_t^{con}\}_{t=1}^{T}$:

$$\mathbf{p}_i=[\mathbf{h}_i^{con};\mathbf{h}_{i+1}^{con};...;\mathbf{h}_{i+\Delta-1}^{con}],\quad 1\le i\le T-\Delta+1 \tag{2.10}$$

where $\mathbf{p}_i$ denotes the i-th trend in T with a timespan of Δ. Similarly, we represent the upcoming trend $\widetilde{\mathbf{p}}$ in the $[T+1,T+\Delta]$ time interval via the concatenation of all contextual vectors in $\{\mathbf{d}_t^{con}\}_{t=T+1}^{T+\Delta}$:

$$\widetilde{\mathbf{p}}=[\mathbf{d}_{T+1}^{con};\mathbf{d}_{T+2}^{con};...;\mathbf{d}_{T+\Delta}^{con}]. \tag{2.11}$$

We explain the workflow of attention for trend alignment in Fig.2.4. As demonstrated in Fig.2.4, when the trend index i increases from 1 to $T-\Delta+1$, $\mathbf{p}_i$ can be viewed as a sliding window that

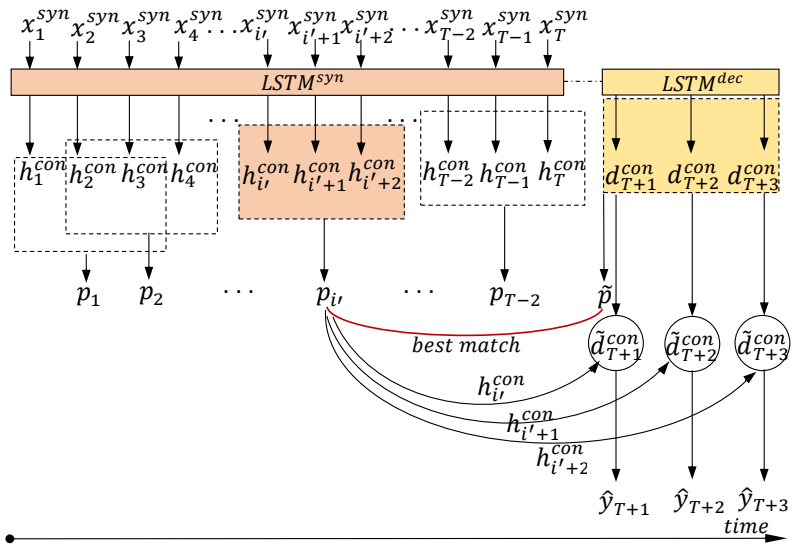


Figure 2.4: Demonstration of proposed attention mechanism for trend alignment. The process for generating output label is included as well. We omit $LSTM^{int}$ and $LSTM^{ext}$ to be succinct. Note that we assume $\Delta = 3$ in this figure for better readability. The essence is to find a best match denoted by $\mathbf{p}_{i'}$ for the current trend $\widetilde{\mathbf{p}}$. Afterwards, we sequentially join the aligned contextual vector pairs within two trends to produce the final contextual vectors $\{\widetilde{\mathbf{d}}_t^{con}\}_{t=T+1}^{T+\Delta}$ and then predict the upcoming sales $\{\hat{y}_t\}_{T+1}^{T+\Delta}$.

dynamically captures temporary contextual information learned from existing sales time series with respective step and window size as 1 and Δ. Hence, we compute the relevance score between $\widetilde{\mathbf{p}}$ and each $\mathbf{p}_i \in \{\mathbf{p}_i\}_{i=1}^{T-\Delta+1}$, with:

$$e_i^{trd} = \mathbf{p}_i^\top \widetilde{\mathbf{p}}, \tag{2.12}$$

and then find out the best match of $\widetilde{\mathbf{p}}$:

$$i' = \text{argmax}(e_i^{trd}, e_{i+1}^{trd}, ..., e_{T+\Delta-1}^{trd}), \tag{2.13}$$

where e_i^{trd} denotes the relevance between $\widetilde{\mathbf{p}}$ and $\mathbf{p}_i$, while i' indicates the i'-th trend in $\{\mathbf{p}_i\}_{i=1}^{T-\Delta+1}$ is the most relevant to $\widetilde{\mathbf{p}}$. Because $\widetilde{\mathbf{p}}$ and $\mathbf{p}_i$ express similar contextual semantics with the same dimensionality, we don't use the scoring scheme in Eq.(2.8) but adopt the dot product to be computational efficient. Intuitively, the closer $\widetilde{\mathbf{p}}$ and $\mathbf{p}_i$ are, a larger e_i^{trd} will be generated and vice versa ($e_i^{trd} = 0$ when orthogonal), so we can align the upcoming trend $\widetilde{\mathbf{p}}$ with its best match $\mathbf{p}_{i'} = [\mathbf{h}_{i'}^{con}; \mathbf{h}_{i'+1}^{con}; ...; \mathbf{h}_{i'+\Delta-1}^{con}]$.

More importantly, now the contextual vectors within both trends, i.e., $\{\mathbf{d}_t^{con}\}_{t=T+1}^{T+\Delta}$ and $\{\mathbf{h}_t^{con}\}_{t=i'}^{i'+\Delta-1}$ are also aligned as trend components instead of individual hidden states. With the upcoming sales trend $\widetilde{\mathbf{p}}$ aligned with the i'-th historical trend, we merge each pair of contextual vector in $\{\mathbf{d}_t^{con}\}_{t=T+1}^{T+\Delta}$ and $\{\mathbf{h}_t^{con}\}_{t=i'}^{i'+\Delta-1}$ into the aligned representation of contextual vectors:

$$\begin{aligned} \widetilde{\mathbf{d}}_t^{con} &= \mathbf{W}_{ali}[\mathbf{d}_j^{con}; \mathbf{h}_k^{con}] + \mathbf{b}_{ali}, \\ T+1 \le j \le T, &\quad i' \le k \le i'+\Delta-1, \end{aligned} \tag{2.14}$$

where $\widetilde{\mathbf{d}}_t^{con}$ is the aligned contextual vectors at time t, $\mathbf{W}_{ali}$ and $\mathbf{b}_{ali}$ are parameters to learn, $[\mathbf{d}_j^{con}; \mathbf{h}_k^{con}]$ is the concatenation of aligned contextual vector pair. We use the following algorithm to acquire the full set of aligned contextual vectors for sales prediction:

Algorithm 1 Generate Aligned Contextual Vectors

Input: prediction time steps Δ; aligned trend index i'; encoded time length T; sales contextual vectors $\{\mathbf{d}_t^{con}\}_{t=T+1}^{T+\Delta}$ and $\{\mathbf{h}_t^{con}\}_{t=i'}^{i'+\Delta-1}$
Output: aligned representations of contextual vectors $\{\widetilde{\mathbf{d}}_t^{con}\}_{t=T+1}^{T+\Delta}$
initialize with $j = T+1$, $k = i'$;
while $j \leq T+\Delta$ **and** $k \leq i'+\Delta-1$ **do**
    update $\widetilde{\mathbf{d}}_t^{con}$ via Eq.(2.14);
    $j++$;
    $k++$;
end

Here, $\{\widetilde{\mathbf{d}}_t^{con}\}_{t=T+1}^{T+\Delta} = \{\widetilde{\mathbf{d}}_{T+1}^{con}, \widetilde{\mathbf{d}}_{T+2}^{con}, ..., \widetilde{\mathbf{d}}_{T+\Delta}^{con}\}$ contains the final latent representation at each upcoming time step in the simulated sales context.

2.2.4 Sales Prediction and Model Learning

With the aligned contextual vectors $\{\widetilde{\mathbf{d}}_t^{con}\}_{t=T+1}^{T+\Delta}$ generated, we approximate the future sales with regression:

$$\hat{y}_t = \mathbf{v}_y^\top \widetilde{\mathbf{d}}_t^{con} + b_y, \tag{2.15}$$

where $\hat{y}_t \in \{\hat{y}_t\}_{t=T+1}^{T+\Delta}$ denotes the predicted sales at time t, $\mathbf{v}_y^\top$ and b_y are parameters to learn.

For model learning, we apply the simple yet effective mean squared error coupled with *L2* regularization (to prevent overfitting) on model parameters:

$$\mathscr{L}_F = \frac{1}{N}\left(\sum_{n=1}^{N}\sum_{t=T+1}^{T+\Delta}(\hat{y}_{nt} - y_{nt})^2\right) + \lambda\sum_{l}^{L}\theta_l^2, \tag{2.16}$$

where $n \leq N$ is the number of training samples, $l \leq L$ is the index of model parameters, y_{nt} is the actual label of sales at t-th time step, θ_l is the model parameter, and λ is the weight decay coefficient that needs to be tuned.

In the training procedure, we leverage mini-batch Stochastic Gradient Decent (SGD) algorithm, namely Adam [63] optimizer. Specifically, we set the batch size as 128 according to device capacity and the start learning rate as 0.001 which is reduced by 10% after each 10,000 iterations. We iterate the whole training process until the loss converges.

2.2.5 Time Complexity of TADA

Because the proposed multi-task, dual-attention RNN model is heavily associated with multiple parameters, here we discuss its time complexity in detail. We prove that like a standard LSTM system, with the model parameters fixed, the asymptotic time complexity of TADA is linear to the size of data.

For a basic LSTM cell in Eq.(2.2), we denote the number of hidden dimensions as q (i.e., $\mathbf{h} \in \mathbb{R}^{q\times 1}$). According to [39, 64], ignoring the biases, a single-task LSTM with T time steps has the complexity of $O(q^2T)$. Similarly, we formulate the time complexity for our encoder-decoder structure. Assuming all LSTMs in TADA have q hidden dimensions, and the multi-task encoder structure with $LSTM^{int}$,

Table 2.1: Statistics of datasets in use.

Dataset	Time Series	Granularity	Time Range	Variables
Favorita	11,536	1 day	365 days	13
OSW	1,585	1 week	106 weeks	11

$LSTM^{ext}$ and $LSTM^{syn}$ are deployed in parallel, the time complexity is $O\big(q^2(T+\Delta)\big)$, which is identical to a basic encoder-decoder LSTM structure.

Then, we focus on the dual-attention mechanism. Since Eq.(2.8) can be viewed as two parallel feed-forward networks, the complexity is $O(q^2)$ for each time step. Coupled with Eq.(2.7), the time complexity of attention mechanism in Section 2.2.3.1 is $O(q^2T\Delta + q^2\Delta) = O\big(q^2(T+1)\Delta\big) \simeq O(q^2T\Delta)$. According to [62], dot product-based attention mechanism Eq.(2.12) has the complexity of $O\big(q\Delta(T-\Delta+1)\big) \simeq O(qT\Delta - q\Delta^2)$. Combining with Eq.(2.14), the overall complexity of attention mechanism in Section 2.2.3.2 is $O(q^2\Delta + qT\Delta - q\Delta^2)$.

With the complexity of encoder-decoder and dual-attention mechanism sorted, we aggregate the complexity for generating the aligned contextual vectors $\{\widetilde{\mathbf{d}}_t^{con}\}_{t=T+1}^{T+\Delta}$. Note that the complexity of Eq.(2.4) throughout time T is $O(q^2T)$, and the complexity of Eq.(2.15) throughout time Δ is $q\Delta^2$. Finally, the overall complexity of TADA comes to $O\big(2q^2(T+\Delta) + qT(q+\Delta)\big)$. In practice, we have $\Delta \ll T$ and $\Delta \ll q$, so T and q are dominating in dimensionality. Therefore, we simplify the final time complexity as $O(3q^2T) \to O(q^2T)$. For a dataset with N samples (time series), it takes $O(Nq^2T)$ to go through the entire dataset once. In summary, when the hidden dimension q and total time step T is fixed, the time complexity of TADA is linearly associated with the scale of the data.

2.3 Experiments

In this section, we conduct experiments on real commercial datasets to showcase the advantage of TADA in the task of sales prediction. In particular, we aim to answer the following research questions via the experiments:

RQ1 How effectively and accurately TADA can predict continuous sales volume with observed sales time series from the past?

RQ2 How can TADA benefit from each component of the proposed structure for sales prediction?

RQ3 How efficiently can TADA be trained when handling training data with different sizes?

2.3.1 Datasets and Features

To validate the performance of TADA, we use two real-life commercial datasets shown in Table 2.1, namely **Favorita** and **OSW**. Here we briefly introduce the properties of these two datasets below:

- **Favorita**: It contains the daily features and sales volume of all products in 54 Ecuadorian-based grocery stores. Note that the original Favorita dataset covers the time range from 1 January 2013 to 15 August 2017, but we only use the portion from 15 August 2016 to 15 August 2017 (365 days) due to two reasons: (1) a magnitude 7.8 earthquake struck Ecuador on 16 April 2016, which exerted abnormal sales patterns in the following few weeks[1]; (2) shorter time series suits the real-life conditions better as it is faster for the model to learn.

- **OSW**: One Stop Warehouse[2] is one of the largest solar energy appliance suppliers in Australia. The dataset covers 12 warehouses' weekly sales volume of various products (e.g., solar panels, batteries, etc.) from 22 February 2016 to 4 March 2017 (106 weeks). Empirically, sales prediction on OSW dataset is more challenging from two perspectives: (1) the sales volume of solar energy appliances is more dependent on external causes (e.g., policy, electricity price, promotion, etc.), which are unavailable in this dataset; (2) the sales volume in OSW dataset fluctuates more significantly than Favorita.

The features we used from the datasets are listed in Table 2.2. Features consist of binary (represented as 1 or 0), categorical (represented via one-hot vectors) and numerical data, which are marked by superscripts of b, c and n respectively. To accelerate the training process, we process all the numerical features by performing $\log_{10}$ transfer (a small bias of 0.001 was added to all numerics to avoid the case of 0). In addition, we leverage embedding to reduce the original dimensionality of categorical data, and combine all these features together as the model input. As suggested by the Tensorflow research team from Google[3], we set the embedding dimension of each categorical feature by taking the 4th root of the total amount of categories. In Table 2.2, numbers with '$*$' mean the dimension of embedding for categorical features.

In both datasets, each time series is actually a log file for a specific product. Hence, we don't split different products up for training and test because it means many products are totally new to the model during test, which is not realistic in real business. So, we first randomly take 3,000 and 400 time series out of Favorita and OSW dataset for validation. Then, given time series with the total time steps of M (365 for Favorita and 106 for OSW) and Δ steps to predict, we apply the 'walk-forward' split strategy on the remaining data. For training, we encode the information with $t \in [1, M-2\Delta]$ and predict sales with $t \in [M-2\Delta+1, M-\Delta]$. For evaluation, we encode the information with $t \in [\Delta+1, M-\Delta]$ and predict sales with $t \in [M-\Delta+1, M]$ to test the accuracy. This test strategy has more practical meaning in the real world, where most businesses tend to predict future sales volume according to previous records.

[1] https://www.kaggle.com/c/favorita-grocery-sales-forecasting/data
[2] https://www.onestopwarehouse.com.au
[3] https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html

Table 2.2: Features extracted from datasets.

Dataset	Type	Feature	Dimension	
Favorita	internal feature	city of store^c	3*	17
		state of store^c	2*	
		store type^c	2*	
		store group^c	2*	
		item family^c	3*	
		item class^c	5*	
	external feature	promotion state^b	1	11
		date^c	5*	
		store transaction^n	1	
		oil price^n	1	
		local holiday^b	1	
		national holiday^b	1	
		pay day^b	1	
OSW	internal feature	item index^c	5*	16
		city of store^c	2*	
		item category^c	5*	
		battery type^c	3*	
		item price^n	1	
	external feature	week number^c	4*	9
		discontinued state^b	1	
		solar exposure^n	1	
		temperature^n	1	
		week(s) after last holiday^n	1	
		week(s) to next holiday^n	1	

2.3.2 Parameters and Experimental Settings

In TADA, we apply the same size to the hidden states of all LSTM systems to maintain the consistency of contextual feature dimension. That is to say, there are only two hyperparameters in TADA to be determined, namely the size of hidden states and the weight decay penalty λ. We conduct grid search for the number of hidden states and λ over $\{32, 64, 128, 256, 512\}$ and $\{0.001, 0.01, 0.1, 1, 10\}$ respectively. The settings with the best performance on the validation set ($\lambda = 0.01$ on Favorita, $\lambda = 0.1$ on OSW, and 128 hidden states for both datasets) are used in the test.

We conduct experiments against the following state-of-the-art predictive methods:

- **Random Forest (RF)**: We implement a widely-used, predictive decision tree-based model, random forest to predict sales from the observed features.
- **XGBoost**: It stands for extreme gradient boosting, proposed by Chen *et al.* [35]. It is a state-of-the-art, gradient boosted regression tree approach based on the gradient boosting machine (GBM) [34].
- **SAE-LSTM**: From the cutting edge of economics research, we adpot the stacked autoencoder with LSTM (SAE-LSTM) [45] which is a neural network-based model proposed for financial time series prediction.

Table 2.3: Sales prediction results under the offline setting. Numbers in boldface are the best results within each column.

Dataset	Method	$\Delta = 2$		$\Delta = 4$		$\Delta = 8$	
		MAE	SMAPE(%)	MAE	SMAPE(%)	MAE	SMAPE(%)
Favorita	RF	32.483	200(max)	35.507	200(max)	41.329	200(max)
	XGBoost [35]	16.705	87.433	19.833	91.230	22.547	158.461
	SAE-LSTM [45]	7.364	39.447	8.033	44.384	8.116	46.932
	A-RNN [59]	11.610	60.781	12.226	62.397	13.005	65.812
	DA-RNN [49]	7.816	43.859	8.234	44.704	8.566	46.281
	LSTNet [37]	7.419	43.523	7.982	45.662	8.729	48.469
	TADA-SE	9.995	58.715	11.076	60.332	10.955	60.257
	TADA-SA$_1$	8.152	46.732	8.273	43.951	8.968	49.079
	TADA-SA$_2$	7.635	42.883	8.247	44.942	8.626	48.609
	TADA	**6.955**	**38.770**	**7.323**	**40.588**	**7.422**	**43.675**
OSW	RF	29.147	89.482	35.576	137.892	43.096	200(max)
	XGBoost [35]	21.496	49.556	24.916	53.243	30.322	82.633
	SAE-LSTM [45]	17.828	44.241	19.805	46.887	20.823	49.873
	A-RNN [59]	17.391	44.635	18.823	44.603	22.129	49.180
	DA-RNN [49]	17.634	44.215	19.578	47.139	20.693	48.365
	LSTNet [37]	16.625	42.317	18.989	45.782	21.246	49.191
	TADA-SE	19.635	53.017	20.884	49.370	21.687	51.685
	TADA-SA$_1$	16.585	42.620	18.624	44.331	21.699	51.195
	TADA-SA$_2$	17.087	42.199	18.643	45.219	21.190	49.825
	TADA	**15.418**	**41.354**	**17.572**	**43.265**	**19.618**	**47.782**

- **A-RNN**: Attention RNN (A-RNN) was originally designed by Bahdanau *et al.* for machine translation tasks [59], with the output of a probability distribution over the word dictionary. We modify the output layer by mapping the learned hidden states into scalar values and use the loss function in Eq.(2.16) for sales prediction task.

- **DA-RNN**: This is a non-linear autoregressor (AR) with attentions in both encoder and decoder RNNs [49]. Compared with A-RNN, the proposed encoder attention in DA-RNN assumes the inputs must be correlated along the time, which is not always true in sales time series.

- **LSTNet**: It is a deep learning framework (long- and short-term time series network) designed for multivariate time series prediction [37]. This method combines a convolutional neural network with a recurrent-skip network to capture both short-term and long-term trending patterns of the time series.

Furthermore, to fully study the performance gain from each component of our proposed model, we implement three degraded versions of TADA:

- **TADA-SE**: We replace the multi-task based encoder with a single-task, 1-layer LSTM encoder. The internal and external feature vectors are concatenated as its input.

- **TADA-SA$_1$**: We remove the attention mechanism for decoder input mapping to build a single-attention variant.

- **TADA-SA$_2$**: We remove the attention mechanism for trend alignment to build another single-attention variant.

To measure the effectiveness of all the methods in sales prediction, we adopt two evaluation metrics, namely mean absolute error (MAE) and symmetric mean absolute percentage error (SMAPE). Mathematically, they are defined as follows:

$$\begin{aligned} \textbf{MAE} &= \frac{1}{N \times \Delta} \sum_{n=1}^{N} \sum_{t=T+1}^{T+\Delta} |y_t - \hat{y}_t|, \\ \textbf{SMAPE} &= \frac{100\%}{N \times \Delta} \sum_{n=1}^{N} \sum_{t=T+1}^{T+\Delta} \left(\begin{array}{c} 0, \ \ if \ y_t = \hat{y}_t = 0 \\ \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2}, otherwise \end{array} \right), \end{aligned} \tag{2.17}$$

where y_t and $\hat{y}_t$ denote real and predicted sales volume respectively. We choose them because MAE is scale-dependent while SMAPE is not, so MAE is suitable for comparison of different methods on the same dataset and SMAPE suits comparison across different datasets. We test all methods on two datasets with $\Delta \in \{2, 4, 8\}$ to showcase their robustness in multiple sales prediction scenarios.

2.3.3 Discussion on Effectiveness (RQ1)

We report the results of all tested methods on all Δ settings in Table 2.3, where the best performance is highlighted with bold face. MAE measures the error with the deviation between predicted and real sales volume, and SMAPE quantifies such error with a proportional perspective.

It is as expected that all neural network-based predictive models outperform decision tree-based models (RF and XGBoost) by a significant margin in both datasets. Hence, we can empirically suggest that deep neural networks better suit the task of sales prediction in the real-world scenario. Apparently, the performance of all methods starts to drop when we gradually increase the time range for sales prediction with $\Delta \in \{2, 4, 8\}$. However, among this observation, TADA demonstrates the least negative impact from the increasing Δ and presents the dominating prediction performance against all state-of-the-art baselines. In other words, the trend alignment scheme from TADA can practically meet the requirement of sales prediction when merchants are trying to look ahead at more upcoming time steps. When comparing with other deep neural network-based approaches (SAE-LSTM, A-RNN, DA-RNN and LSTNet) the results also support the superiority of TADA. This is because: (1) the multi-task encoder in TADA is better at capturing the interactive effect from both internal and external features to the real sales than modelling all influential factors in the unified way; (2) the dual attention architecture in TADA successfully captures latent trends from the past which are similar to the upcoming one, especially when comparing with existing attention mechanisms (A-RNN and DA-RNN) and periodic trend modelling method (LSTNet). The effectiveness of each proposed component in TADA is initially revealed in Table 2.3 by its degraded versions, which we will further discuss in the following section.

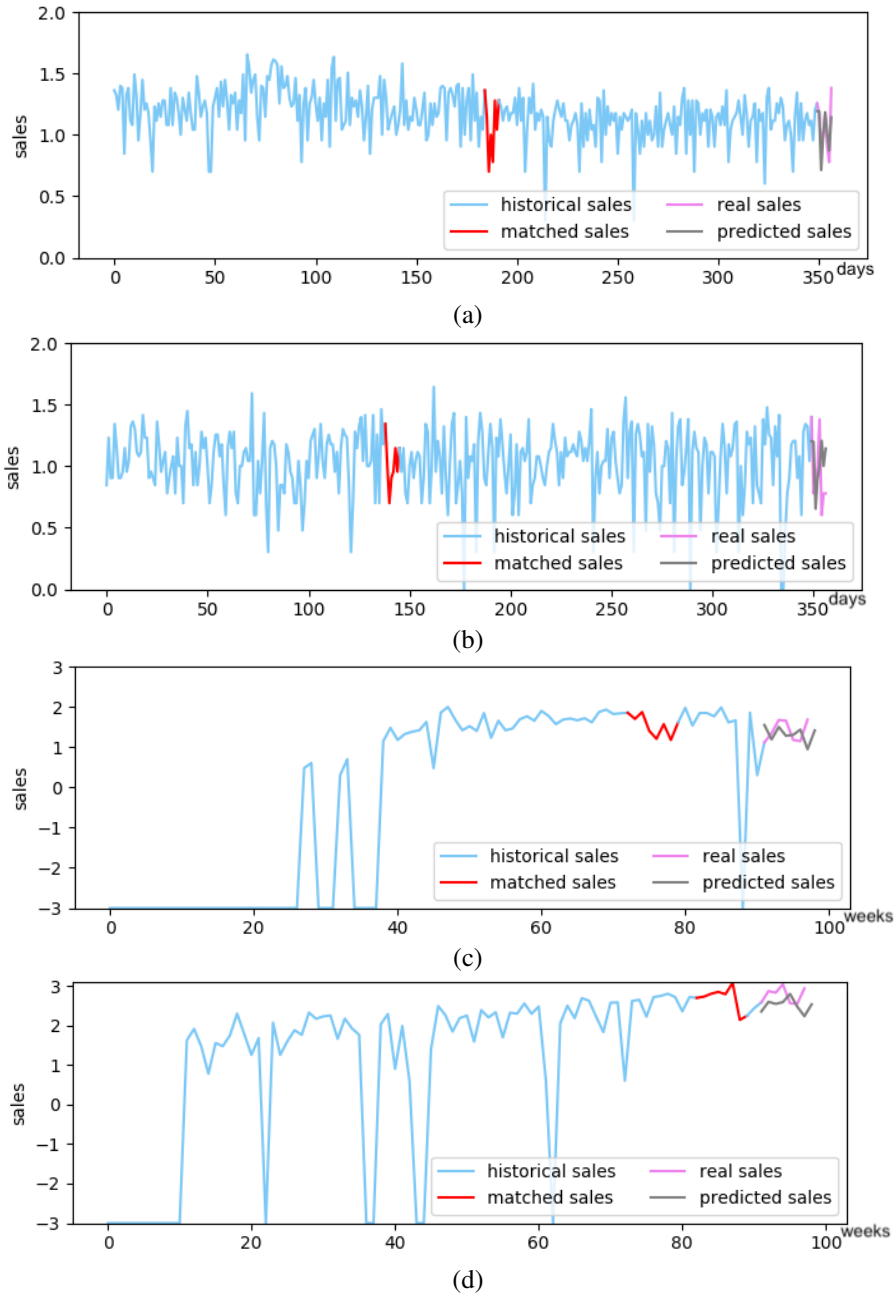


Figure 2.5: Demonstration of proposed trend alignment scheme in TADA with attention mechanism. Among these four visualizations, (a) and (b) are selected from Favorita, while (c) and (d) are selected from OSW. The sales axis is rescaled via $\log_{10}$ transfer on each dataset for better readability. Apparently, there are no obvious recurring trends in all these sales records, but TADA successfully selects the most relevant one to assist the prediction. The figures illustrate that aligned trends in sales time series not only share similar contexts, but also have close sales volume.

2.3.4 Importance of Key Components (RQ2)

We implement three variants of our proposed model TADA, namely TADA-SE, TADA-SA_1 and TADA-SA_2, by removing one of the key components each time. With the degraded versions of TADA, we carry out the ablation study on the performance gain from every proposed component within TADA. As shown in Table 2.3, the evaluation results on two real datasets indicate that these variants suffer from noticeable drops in the prediction performance. Specifically, TADA-SE shows more obvious

infection. This provides evidence for our assertion that by dividing the influential factors in sales time series into semantically different internal and external features, the multi-task encoder structure can extract more latent contextual information related to the real sales volume. In TADA-SE, the dynamic interaction of internal and external features are no longer modelled, causing insufficient performance accuracy.

According to Table 2.3, when we remove each one of the two proposed attention mechanisms in TADA-SA_1 and TADA-SA_2, the prediction performance both drops. Combining their performance on both datasets, the performance reduction is similar when either part of the dual attention mechanism is blocked. So, we draw the observation that both attentions contribute positively and almost equally, and they are indispensable to each other for precise sales prediction. Thus, after the contextual vectors are learned from the encoder, it is crucial to leverage the dual-attention decoder to mimic the contextual information in the future as well as aligning the upcoming trend with historical ones to enhance the prediction of sales. Furthermore, as the attention mechanism provides TADA (full version) with a better interpretability, we visualize the intermediate results of aligned trends in the predicting (decoding) stage, along with the predicted sales. Fig.(2.5) visualizes the results of trend alignment from samples selected from both Favorita and OSW datasets by highlighting the sales trend with the highest attention weight. As a result, we find that similar sales contexts lead to similar sales volume, which confirms the rationale of performing trend alignment for sales prediction and the effectiveness of all components in TADA.

2.3.5 Training Efficiency and Scalability (RQ3)

Due to the importance of practicality in real-life applications, we validate the scalability of TADA. As we proved in Section 2.2.5, when all the parameters in the network are fixed (in our case, the dimension for all hidden states is 128, and T is determined according to Δ), the training time for TADA is only associated with the number of training samples. Ideally, the training time for TADA should increase linearly as we enlarge the scale of the training data. Note that we set $\Delta = 8$ ($T = 349$ correspondingly) for this validation.

We test the training efficiency and scalability of TADA by using different proportions of the whole training set from Favorita, and then report the corresponding training time (excluding I/O). The growth of training time along with the data size is shown in Fig.2.6. When the ratio of training data gradually extends from 0.2 to 1.0, the training time for TADA increases from 3.54×10^3 seconds to 22.15×10^3 seconds. It shows that the link between training time and the data scale is approximately linear. Hence, we conclude that since its linear time complexity can ensure high scalability, TADA can be efficiently trained with large-scale datasets.

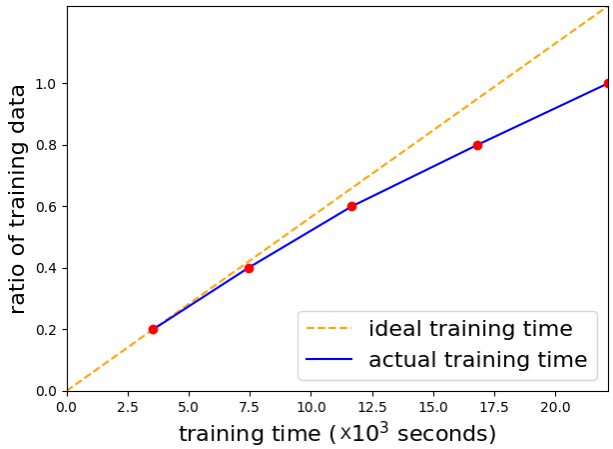


Figure 2.6: The training time of TADA with varied proportions of training data.

2.4 Summary

Sales prediction is a significant yet unsolved problem due to the subtle influential patterns among different factors and the irregular sales trends triggered by complex real-life situations. In this chapter, we propose TADA, a novel model that performs trend alignment with dual-attention, multi-task recurrent neural networks to predict sales volume in real-life commercial scenarios. With TADA, we first model the internal and external features within the influential factors in the sales time series in a multi-task fashion, thus maintaining their unique semantic meanings when timely modelling their mutual influences to the sales. Besides, we propose a dual-attention decoder to simulate the sales contextual information in the future, and then align the generated representation of the upcoming trend with the most relevant one from the past. By this means, TADA conquers existing challenges in the sales prediction task and outperforms the state-of-the-art baselines in two real datasets. We summarize the specific contributions of this work as follows:

- We are the first to categorize the influential factors in sales time series into internal features and external features, and innovatively model these two aspects with multi-task LSTM encoder. We also adopt a synergic LSTM layer to model the dynamic interaction between different types of influential factors.

- We propose TADA, a dual-attention multi-task recurrent neural network to tackle the aforementioned challenges in sales prediction. The novel approach allows the encoder-decoder structure to comprehensively model variables with different semantic meanings. Also, the embedded dual-attention increases both the interpretability and accuracy of the model by simulating unknown states of future contexts and aligning the upcoming sales trend with the most relevant one from the past.

- We conduct extensive experiments on two real-life commercial datasets. The results showcase the superiority of our approach in sales prediction by outperforming a group of state-of-the-art predictive models. We validate the vigorous contribution of each component in TADA via

ablation tests and visualizations. Additional experiments on training efficiency further show promising scalability of TADA.

Chapter 3

Micro-Level Sequence Modelling

As an important means to maximize customer values in e-commerce, recommender systems have already demonstrated their strong benefits to both online service platforms and common users as they grant users easier access to preferred resources and help service providers understand their customers better. In this chapter, we will study the problem of micro-level sequence modelling in e-commerce from the view of sequential recommendation.

3.1 Literature Review: Background and Motivation

3.1.1 General Top-k Recommendation

General top-k recommendation tasks aim to recommend k items that a user is likely to interact with in the near future. In this context, early works mainly adopt collaborative filtering (CF) methods e.g., matrix factorization to utilize historical interactions to infer the links between users and items [18,19,65,66]. Approaches like matrix factorization [18,19] and factorization machines [20,21] achieve great success in top-k recommendation with the assumption that user preference is static. Usually, the predominant problem is matrix completion where we are given a user-item interaction matrix for and the goal is to predict the missing values [67]. Such methods seek to uncover latent dimensions to represent user preference and item properties, and predict interactions by ranking the inner product of the user and item embeddings [18]. Another widely adopted method is factorization machines (FM) [20] which effectively mimics the effect of CF by modelling the high-order feature interactions and allowing the use of side information. Because FMs show promising results in regression problems and can serve as pairwise scoring functions in top-k recommendation tasks [68, 69], recent variants of FMs further leverage deep neural networks to enhance the performance [69–71] with a compromise in model complexity. However, compared with our prosed ITFM, existing FMs lack the consideration of sequential dependencies when predicting the intensity of user-item interactions.

3.1.2 Sequential Recommender Systems

While the numbers of both users and items are now growing exponentially over time, it is more practical to investigate the problem of sequential top-k recommendation nowadays as the dynamics of the data play a pivotal role in recommender systems.

Unlike conventional top-k recommendation, the sequential top-k recommendation approaches model the user behavior as a sequence of items instead of a set of items [22]. In order to capture the sequential dependency of user-item interactions, recommendation techniques based on Markov Chain (MC) and Recurrent Neural Networks (RNNs) have been developed. MC-based approaches [72–74] model sequential user interactions by learning a transition graph over items that is used to predict the users' next interested items. Although they are reported to achieve good performance, these methods tend to fail to capture the intricate dynamics of complex scenarios [23], and lack sufficient representation capability for user and item modelling. Recently, with the success of RNNs in a wide range of sequence modelling tasks like machine translation [59] and sales prediction [1], RNN-based models have attracted the attention from many sequential recommendation researchers [75–78]. For instance, being capable of learning deep representations from the sequence, RNNs have been proved applicable for sequential session-based recommendation tasks [75]. Furthermore, [76] points out the characteristics of both users and items vary over time, and a twin RNN scheme is designed to capture the dynamics of both users and items in parallel for movie recommendations.

Another line of deep neural network-based sequential recommender systems is recurrent neural networks (RNNs) which are effective in modelling sequential user actions [75, 79, 80]. For instance, the recurrent recommender networks developed by [80] captures user preference drifts via two parallel RNNs over time. Meanwhile, as a rising technique, the applications of attention mechanisms coupled with neural recommender systems [71, 81–83] do contribute to the performance improvement. In summary, the main disadvantage of existing sequential recommender systems is that they only model item-wise user preference instead of category-wise user preference, and they neglect the heterogeneity of different user actions, resulting in severe data sparsity. In what follows, we will elaborate on the practical issues caused by the shortcomings of existing sequential recommenders, and present our solution accordingly.

3.1.3 Motivation and Our Solution

However, to ensure the model expressiveness, the state-of-the-art deep neural networks require large-scale and dense training data [23], rendering it difficult to fully generalize when the observable user preference information is highly sparse. One main cause of the sparsity is that ***the user preference studied in existing methods is item-wise rather than category-wise*** (i.e., they directly model user-item pairs). In our evaluation dataset Tmall (see Section 3.3.1), there are over 100 million possible user-item interactions, of which only 0.1% are observed and can be used to infer such item-wise user preference. The learned item-to-item sequential patterns are inevitably unreliable and unstable due to the large amount of items and extreme data sparsity. To address the data sparsity issue, a promising

idea is to focus on category-level user interaction data and learn category-wise sequential patterns, which can significantly narrow down the prediction space and allow RNN-based models to give full play to their advantages with denser data. For instance, under a sequential setting, a user may consider buying a phone case after purchasing a new cellphone. While the migration from a specific cellphone to a specific phone case (e.g., from iPhone X Space Grey to *Louis Vuitton* Case for iPhone X) tend to be subtle and unstable, the general category-to-category migration (e.g., from *cellphone* to *phone case*) presents a more reliable and stable sequential pattern for the recommender system. As a result, neglecting the category-wise interaction sets an obstacle to learning useful patterns from the sequence and capturing user dynamics for sequential top-k recommendation.

In addition, most existing approaches on sequential recommendation focus on only the homogeneous user interaction behaviors. However, user interaction data on a single type of user behavior/action (e.g., purchase) is extremely rare and sparse, therefore it is essential to ***collectively exploit the rich heterogeneous behavior data*** to enhance the sequential recommendation. For example, there are at least four types of user actions in the typical e-commerce platforms (e.g., Amazon or Tmall): click, add-to-cart, add-to-favorite and purchase. To predict what product a user would purchase next, it may be helpful to pay attention to not only what the user has purchased previously, but also what this user has added to favorite list in the past. Furthermore, different types of user actions imply different semantics and have different contributions to the final recommendation. For instance, add-to-cart actions provide stronger signals for user purchase behavior prediction than click actions. It is arguably the most challenging to exploit the sequential correlations among various user behavior/action types and capture their varied effect on the users' decision-making process.

In light of the aforementioned challenges in sequential top-k recommendation, we propose to model a sequence of category-wise ***user intention*** instead of item-wise user preference in this thesis. Specifically, we formulate user intention as a tuple of action type (e.g., click, purchase, etc.) and product category tag, i.e., $(action, category)$. Intuitively, user intention carries two aspects of information: the category of products that a user tends to interact with; and the way the user wants to perform the interaction, which indicates the strength of willingness. Furthermore, in contrast to modelling the dynamics of user preference drift on items (i.e., the item-level sequential effect), we investigate ***user intention migration*** which focuses on category-wise user demand dynamics for sequential dependency learning. To this end, we develop ***AIR***, namely ***Attentional Intention-Aware Recommender Systems*** as a solution to sequential top-k recommendation. The workflow of AIR consists of two main stages. First, we design an Attentional Recurrent Neural Network (ARNN) to predict future user intention based on the history; then, a novel Intention-Aware Temporal Factorization Machine (ITFM) is presented as the pairwise scoring function for top-k item recommendation based on the awareness of user intention. By deriving a sequence of user intention from the user transaction data, we propose to firstly predict what type of item is in demand, then accordingly recommend the exact item to each user. The distinct advantages of our approach against existing methods are: (1) while users' timely preference on exact items has much randomness because of complicated real-world situations (e.g., promotions), their category-wise intention reflects the intrinsic demand of users and tend to

Table 3.1: Description of major notations used in this chapter.

Notation	Description
a_m	a user action indexed by m
$\mathscr{A}$	a set of all actions
c_n	an item category indexed by n
$\mathscr{C}$	a set of all categories
u_i	a user account indexed by i
$\mathscr{U}$	a set of all users
v_j	an item indexed by j
$\mathscr{V}$	a set of items
$\pi_s^{m,n}$	a user intention in the form of (a_m, c_n), indexed by s
Π	a set of user intention candidates
$\boldsymbol{\gamma}$	an intention vector
$\lvert\cdot\rvert$	the size of an arbitrary set
$\theta_{s'}^{s}$	the probability that the s-th intention will migrate to the s'-th intention
Θ	the intention migration effect matrix

show stronger sequential dependencies; (2) when recommending exact items for users, the correctly predicted user intention effectively lays more emphasis on preferable item categories, thus improving the recommendation performance.

3.2 The Proposed Model: AIR

3.2.1 Preliminaries and Problem Formulations

We first introduce the key notations and definitions used throughout the chapter. The major notations are listed in Table 3.1, and the essential concepts are defined as follows.

Definition 1: User Transaction. *For an arbitrary e-commerce dataset, assuming item v_i only belongs to one category c_n, we record each user's T actions as a user transaction sequence $\{(a_{m(t)}, c_{n(t)}, v_{i(t)})\}_{t=1}^{T} = \{(a_{m_1(1)}, c_{n_1(1)}, v_{i_1(1)}), ..., (a_{m_T(T)}, c_{n_T(T)}, v_{i_T(T)})\}$, where $a_{m(t)} \in A$, $c_{n(t)} \in C$ and $v_{i(t)} \in I$ imply one user conducted action a_m on item v_i from category c_n at time step t.*

As stated in previous discussions, compared with item-wise user preference, the prediction space of category-wise user intention is significantly shrunken, making it possible for the model to learn substantial dynamic patterns. Definition 2 - 5 mathematically describe the concept of user intention, intention candidate set, intention sequence, and intention migration.

Definition 2: User Intention. *Given a dataset having a set of actions $\mathscr{A}$ and a set of item categories $\mathscr{C}$, the user intention is defined as an arbitrary pair of action and category: $\pi^{m,n} = (a_m, c_n)$, where $a_m \in \mathscr{A}$ and $c_n \in \mathscr{C}$.*

Definition 3: Intention Candidate Set. *Following Definition 2, with the action set $\mathscr{A}$ and category set $\mathscr{C}$, the intention candidate set is defined as their Cartesian product: $\Pi = \mathscr{A} \times \mathscr{C} = \{\pi_s^{m,n} | a_m \in \mathscr{A}, c_n \in \mathscr{C}\}$, where $s \leq |\Pi|$ is the index of intention candidates in set Π.*

Definition 4: Intention Sequence. *Following Definition 1-3, from the observed user transaction*

$\{(a_{m(t)}, c_{n(t)}, v_{i(t)})\}_{t=\overline{1}}^{T} \{(\pi_{s(t)}^{m,n}, v_{i(t)})\}_{t=1}^{T}$, *the intention sequence* $\{\pi_{s(t)}^{m,n}\}_{t=1}^{T} = \{\pi_{s_1(1)}^{m_1,n_1}, ..., \pi_{s_T(T)}^{m_T,n_T}\}$ *can be derived, representing a user has conducted intention* $\pi_s^{m,n} \in \Pi$ *at time step* t. *The subscription of* (t) *carries the temporal property of user intention.*

Definition 5: Intention Migration. *In an intention sequence, conditioned on the current user intention* $\pi_{s(t)}^{m,n}$, *there is a chance that* $\pi_{s(t)}^{m,n}$ *will migrate to* $\pi_{s'(t')}^{m',n'}$ *where* $t' > t$. *Such effect is called intention migration, and* $\theta_{s'}^{s}$ *is the probability that the s-th intention will migrate to* s'*-th intention.*

For an arbitrary user, based on the intention sequence $\{\pi_{s(t)}^{m,n}\}_{t=1}^{T}$ observed from T time steps, we aim to predict the likelihood of conducting each intention $\pi_s^{m,n} \in \Pi$ in the future. Such prediction outcome is represented via an intention vector defined below.

Definition 6: Intention Vector. *The future user intention is represented using a* $|\Pi|$*-dimensional intention vector* $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_{|\Pi|}]$ *with each element* $\gamma_s = \mathbb{P}(\pi_s^{m,n})$ *denoting the independent probability of the s-th intention candidate. Specifically, we force* $\sum_{s=1}^{S} \gamma_s = 1$, *i.e.,* $\boldsymbol{\gamma}$ *is a probability distribution over all* $|\Pi|$ *intention candidates.*

Intuitively, the probability distribution in the user intention vector $\boldsymbol{\gamma}$ reflects how each item category is of interest to a user. In recommendation tasks, a larger $\gamma_s \in \boldsymbol{\gamma}$ represents a stronger user intention, so $\boldsymbol{\gamma}$ carries each user's varied preference on different categories and can be further used to generate a ranked item list. Hence, with the concept of user intention defined, we respectively formulate the task of user intention prediction and intention-aware recommendation as follows.

Problem 1: User Intention Prediction. *For an arbitrary user, given an observed intention sequence* $\{\pi_{s(t)}^{m,n}\}_{t=1}^{T}$ *conducted in the past* T *time steps, the* ***user intention prediction*** *task predicts the probability distribution of this user's future intention, represented by an intention vector* $\boldsymbol{\gamma} = \{\gamma_1, \gamma_2, ..., \gamma_{|\Pi|}\}$ *where* $\gamma_s = \mathbb{P}(\pi_s^{m,n}) \in \boldsymbol{\gamma}$ *and* $\sum_{s=1}^{S} \gamma_s = 1$.

Problem 2: Sequential Top-k Recommendation. *For a user* $u_i \in \mathscr{U}$, *given the item set* $\mathscr{V}$, *the user transaction sequence* $\{(\pi_{s(t)}^{m,n}, v_{i(t)})\}_{t=1}^{T}$ *and intention vector* $\boldsymbol{\gamma}$, *the target of sequential top-k recommendation is to predict top k items that* u_i *will interact with.*

3.2.2 User Intention Prediction

Now, we solve the problem of user intention prediction by proposing our attentional recurrent neural network (ARNN).

3.2.3 Network Structure

Taking a vector sequence $\{\mathbf{x}_t\}_{t=1}^{T}$ as input, the recurrent neural network (RNN) encodes $\{\mathbf{x}_t\}_{t=1}^{T}$ into hidden states $\{\mathbf{h}_t\}_{t=1}^{T}$ via $\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$, where $f(\cdot)$ is a non-linear mapping function. To capture the long-range dependency, we leverage RNNs with long short-term memory architecture (LSTM) via

the following formulation [39]:

$$\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i\mathbf{h}_{t-1} + \mathbf{b}_i),\\
\mathbf{f}_t &= \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f\mathbf{h}_{t-1} + \mathbf{b}_f),\\
\mathbf{o}_t &= \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o\mathbf{h}_{t-1} + \mathbf{b}_o),\\
\mathbf{c}_t &= \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \odot \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c\mathbf{h}_{t-1} + \mathbf{b}_c),\\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t),
\end{aligned} \tag{3.1}$$

where $\odot$ denotes element-wise multiplication and the recurrent activation σ is the *Logistic Sigmoid* function. $\mathbf{i}$, $\mathbf{f}$, $\mathbf{o}$ and $\mathbf{c}$ are respectively the input gate, forget gate, output gate, and cell state vectors. When updating each of them, there are corresponding trainable input-to-hidden and hidden-to-hidden weights $\mathbf{W}$ and $\mathbf{U}$ along with the bias vectors $\mathbf{b}$. For notation convenience, we simplify the LSTM system in Eq.(3.1) with the following function:

$$\mathbf{h}_t = LSTM(\mathbf{x}_t, \mathbf{h}_{t-1}). \tag{3.2}$$

In the intention sequence $\{\pi^{m,n}_{s(t)}\}^T_{t=1}$ for each user, we represent the t-th intention $\pi^{m,n}_{s(t)} \in \Pi$ using a $|\Pi|$-dimensional one-hot vector $\boldsymbol{\pi}^{m,n}_{s(t)} = [0,...,1,...,0]$ with 1 at the s-th element. Then, we convert each $\boldsymbol{\pi}^{m,n}_{s(t)}$ into a dense input vector $\mathbf{x}^{\pi_s}_t$ for Eq.(3.2) with an embedding layer:

$$\mathbf{x}^{\pi_s}_t = \mathbf{W}_b^\top \boldsymbol{\pi}^{m,n}_{s(t)}, \tag{3.3}$$

where $\mathbf{W}_b$ is the trainable look-up matrix in the embedding layer. With Eq.(3.3), each $\mathbf{h}_t$ in Eq.(3.2) is actually a latent representation in correspondence with the input intention $\pi^{m,n}_{s(t)}$. Therefore, we fuse the embedding layer with our LSTM framework and reformulate Eq.(3.2) into:

$$\mathbf{h}^{\pi_s}_t = LSTM(\boldsymbol{\pi}^{m,n}_{s(t)}, \mathbf{h}^{\pi_{s'}}_{t-1}), \tag{3.4}$$

where we use $\mathbf{h}^{\pi_s}_t$ to denote the t-th hidden state learned from the corresponding intention input $\boldsymbol{\pi}^{m,n}_{s(t)}$.

3.2.4 Modelling Intention Migration via Attention Mechanism

Because of the sequential characteristics of user intention, it is necessary to model the migrations between different user intention in order to fully capture the dynamics of user transaction sequences. To start with, for all the intention candidates in set Π, we formulate the non-negative intention migration matrix Θ as follows:

$$\Theta = \begin{bmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \vdots \\ \boldsymbol{\theta}_{|\Pi|} \end{bmatrix} = \begin{bmatrix} \theta^1_1 & \theta^1_2 & \dots & \theta^1_{|\Pi|} \\ \theta^2_1 & \theta^2_2 & \dots & \theta^2_{|\Pi|} \\ \vdots & \vdots & \ddots & \vdots \\ \theta^{|\Pi|}_1 & \theta^{|\Pi|}_2 & \dots & \theta^{|\Pi|}_{|\Pi|} \end{bmatrix} = ReLU(\mathbf{W}_\Theta) \tag{3.5}$$

where $\mathbf{W}_\Theta \in \mathbb{R}^{|\Pi|\times|\Pi|}$ is the learnable weight matrix to generate Θ via an element-wise *Rectified Linear Unit*. Each entry $\theta^s_{s'} \in \Theta$ denotes the independent probability that intention $\pi^{m,n}_s$ will migrate to $\pi^{m',n'}_{s'}$.

Noticeably, matrix Θ is asymmetric, i.e., when $s \neq s'$, $\theta^s_{s'}$ and $\theta^{s'}_s$ can be unequal, thus enabling the migration probability between different intention candidates to be directional. For example, when a user purchases a new printer, some additional accessories are needed. It is highly possible for this user to purchase a pack of printing paper shortly after because the printer does not come with paper. In contrast, the purchase of printing paper can hardly encourage a quick purchase of a new printer because the user is very likely to have a printer in use already. As a result, modelling intention migration with directions better simulates the influence from previous intention to the current one due to the sequential characteristics of user intention. After the representation $\mathbf{h}^{\pi_s}_t$ is learned at time step t, to capture its migration effect on all intention candidates, the one-hot vector $\boldsymbol{\pi}^{m,n}_{s(t)}$ for the s-th intention is used to select the s-th row of Θ, denoted by a vector $\boldsymbol{\theta}_s = [\theta^s_1, ..., \theta^s_{|\Pi|}]$:

$$\boldsymbol{\theta}_s = \Theta^\top \boldsymbol{\pi}^{m,n}_{s(t)}. \tag{3.6}$$

Therefore, $\boldsymbol{\theta}_s$ can be viewed as a set of weights indicating the likelihood of having the current intention $\pi^{a_m,c_n}_{s(t)}$ migrated to each intention candidate in Π. Afterwards, we leverage an outer product of $\boldsymbol{\theta}_s$ and $\mathbf{h}^{s_t}_t$ to instantiate the migration effect from the current state $\mathbf{h}^{\pi_s}_t$ to each intention candidate in Π:

$$\mathbf{H}_t = \boldsymbol{\theta}_s \mathbf{h}^{s_t\top}_t = \begin{bmatrix} \theta^s_1 \mathbf{h}^{\pi_s}_t \\ \theta^s_2 \mathbf{h}^{\pi_s}_t \\ \vdots \\ \theta^s_{|\Pi|} \mathbf{h}^{\pi_s}_t \end{bmatrix}, \tag{3.7}$$

where matrix $\mathbf{H}_t$ collects totally $|\Pi|$ weighted representations of $\mathbf{h}^{\pi_s}_t$ for all intention candidates as its row elements.

As shown in Figure 3.1, once we compute $\mathbf{H}_t$ for all T time steps, we can produce the final feature matrix $\widetilde{\mathbf{H}} = [\widetilde{\mathbf{h}}_1, ..., \widetilde{\mathbf{h}}_{|\Pi|}]$ with each row $\widetilde{\mathbf{h}}_s \in \widetilde{\mathbf{H}}$ stacking the final representation for the s-th intention candidate. Specifically, $\widetilde{\mathbf{H}}$ is calculated by performing element-wise aggregation for the weighted features in $\mathbf{H}_t$ for all $t \leq T$:

$$\widetilde{\mathbf{H}} = \sum_{t=1}^{T} \mathbf{H}_t = \begin{bmatrix} \widetilde{\mathbf{h}}_1 \\ \widetilde{\mathbf{h}}_2 \\ \vdots \\ \widetilde{\mathbf{h}}_{|\Pi|} \end{bmatrix} = \begin{bmatrix} \sum_{t=1}^{T} \theta^s_1 \mathbf{h}^{\pi_s}_t \\ \sum_{t=1}^{T} \theta^s_2 \mathbf{h}^{\pi_s}_t \\ \vdots \\ \sum_{t=1}^{T} \theta^s_{|\Pi|} \mathbf{h}^{\pi_s}_t \end{bmatrix}, \tag{3.8}$$

where each representation $\widetilde{\mathbf{h}}_s \in \widetilde{\mathbf{H}}$ for intention s can be viewed as the compression of two factors: (1) all the hidden states $\{\mathbf{h}^{\pi_s}_t\}^T_{t=1}$ learned from the user intention sequence $\{\pi^{m,n}_s\}^T_{t=1}$; (2) the migration effect from each hidden state to the s-th intention candidate.

3.2.5 User Intention Prediction and Model Learning

With $\widetilde{\mathbf{H}}$ calculated, we can now predict the probability distribution of all $|\Pi|$ user intention candidates. We first project each $\widetilde{\mathbf{h}}_s \in \widetilde{\mathbf{H}}$ onto a 1-dimensional space:

$$\mathbf{e} = \widetilde{\mathbf{H}} \mathbf{w}_e, \tag{3.9}$$

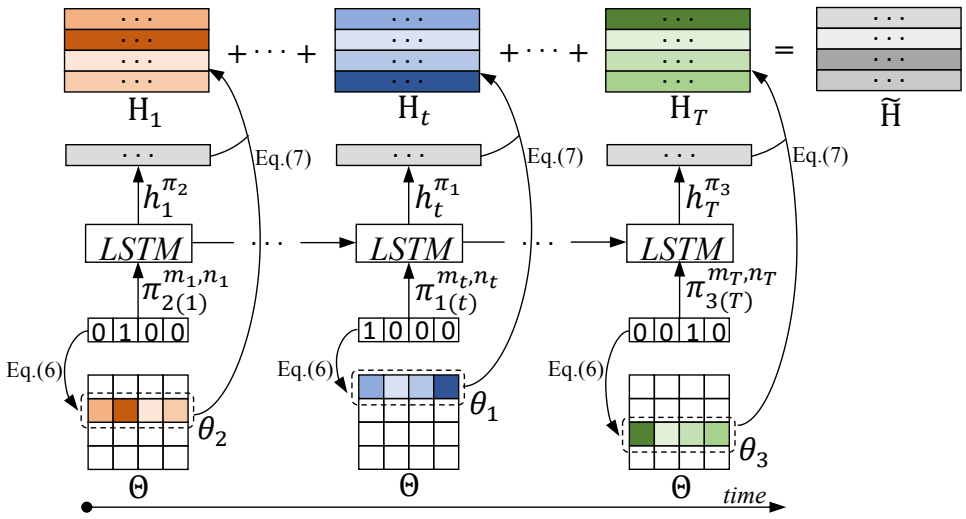


Figure 3.1: Demonstration of proposed attention mechanism for intention migration modelling. We omit the LSTM embedding layer to be succinct. In this example, we set the total number of intention candidates as 4 (i.e., Θ has the size of 4×4) and assume $s = 2, 1, 3$ at time step $1, t, T$ respectively for illustration purpose. At each time step t, the one-hot vector $\boldsymbol{\pi}_{s(t)}^{m,n}$ selects a row $\boldsymbol{\theta_s}$ from Θ, and then an outer product of $\boldsymbol{\theta_s}$ and $\mathbf{h}_t^{\pi_s}$ is performed to compute the feature stack $\mathbf{H}_t$. Afterwards, all $\mathbf{H}_t$ from T time steps are aggregated to produce the final feature matrix $\widetilde{\mathbf{H}}$.

where $\mathbf{w}_e$ is the projection weights to learn, and each $e_s \in \mathbf{e}$ is a compressed scalar representation of the s-th intention candidate. Afterwards, to obtain the predicted intention vector $\widehat{\boldsymbol{\gamma}}$, we apply $SoftMax$ to each $e_s \in \mathbf{e}$ to generate a probability distribution over $|\Pi|$ intention candidates:

$$\begin{aligned} \widehat{\boldsymbol{\gamma}} &= [\widehat{\gamma}_1, ..., \widehat{\gamma}_{|\Pi|}], \\ \widehat{\gamma}_s &= \frac{\exp(e_s)}{\sum_{s'=1}^{|\Pi|} \exp(e_{s'})}, \quad s \leq |\Pi|. \end{aligned} \tag{3.10}$$

Hence, the predicted user intention vector $\widehat{\boldsymbol{\gamma}}$ is produced, which indicates the possibility of conducting each intention $\pi_s^{m,n} \in \Pi$ for a particular user in the future. When training our model for an accurate prediction of $\widehat{\boldsymbol{\gamma}}$, we need to obtain the real probability distribution over all intention candidates in the future Δ time steps after T. Consequently, we represent the real distribution (i.e., ground truth) with $\boldsymbol{\gamma} = [\gamma_1, ..., \gamma_{|\Pi|}]$ where each $\gamma_s \in \boldsymbol{\gamma}$ denotes the frequency of intention s observed from Δ time steps after T. For model training, we apply cross-entropy to the predicted intention $\widehat{\boldsymbol{\gamma}}$. Apart from this, confronting the attention mechanism for intention migration modelling, we pose an additional constraint on the intention migration matrix Θ, so that $\sum_{s'=1}^{|\Pi|} \theta_{s'}^s \approx 1$ for every $s \leq |S|$ and each row of Θ implies a probability distribution:

$$\mathscr{L}_{ARNN} = -\frac{1}{D} \sum_{d=1}^{D} \sum_{s=1}^{|\Pi|} (\gamma_{d,s} \log \widehat{\gamma}_{d,s}) + \lambda_1 \sum_{s=1}^{|\Pi|} (1 - \sum_{s'=1}^{|\Pi|} \theta_{s'}^s)^2, \tag{3.11}$$

where D is the number of training samples, $\gamma_s \in \boldsymbol{\gamma}$ and $\widehat{\gamma}_s \in \widehat{\boldsymbol{\gamma}}$ are respectively the predicted and real probability distributions for the s-th intention candidate, while λ_1 is the weight decay coefficient to be tuned.

3.2.6 Intention-Aware Sequential Recommendation

In this section, we introduce the proposed intention-aware temporal factorization machine (ITFM) in detail.

3.2.7 Category-wise Aggregation of User Intention

If user $u_i \in U$ has intention on several item categories in the same time period, it yields difficulties in recommending the most suitable items because u_i may dynamically interact with items in different categories. Since it is common for items to significantly outnumber their total categories, e.g., $|\mathscr{V}| \gg |\mathscr{C}|$, instead of directly scoring all the items for each user using factorization models or embeddings, we propose an intention-aware recommendation approach to tackle this problem. First, assuming each item $v_j \in \mathscr{V}$ belongs to only one category $c_n \in \mathscr{C}$, we aggregate a user's intention of performing all $|\mathscr{A}|$ actions on each category $c_n \in \mathscr{C}$ to summarize the intensity of user intention in each category. Then, when developing the pairwise scoring function for user-item pairs, we extensively leverage the category information within $\widehat{\boldsymbol{\gamma}} = [\mathbb{P}(\pi_1^{m_1,n_1}), ..., \mathbb{P}(\pi_{|\Pi|}^{m_{|\Pi|},n_{|\Pi|}})]$ to selectively assign larger weights to items in certain categories. In the first step, the category-wise intention aggregation for a user is denoted by a $|\mathscr{C}|$-dimensional vector $\widetilde{\boldsymbol{\gamma}} = [\widetilde{\gamma}_1, ..., \widetilde{\gamma}_{|\mathscr{C}|}]$. Each $\widetilde{\gamma}_n \in \widetilde{\boldsymbol{\gamma}}$ is the quantified user intention on category $c_n \in \mathscr{C}$. Algorithm 2 accomplishes the category-wise aggregation of user intention.

Algorithm 2 Category-wise Aggregation of User Intention

Input: the intention vector $\widehat{\boldsymbol{\gamma}}$ for an arbitrary user; the corresponding user intention sequence $\{\pi_{s(t)}^{m,n}\}_{t=1}^{T}$;
Output: aggregated $|\mathscr{C}|$-dimensional intention representation $\widetilde{\boldsymbol{\gamma}}$
initialize with $\widetilde{\gamma}_n = 0$ for each $\widetilde{\gamma}_n \in \widetilde{\boldsymbol{\gamma}}$ and $n = 1$;
while $n \leq |\mathscr{C}|$ **do**
    set $m = 1$;
    while $m \leq |\mathscr{A}|$ **do**
        compute the times of occurrence of intention $\pi_s^{m,n}$ in sequence $\{\pi_{s(t)}^{m,n}\}_{t=1}^{T}$, denoted by $\#\pi_s^{m,n}$;
        $m++$;
    update $\widetilde{\gamma}_n \in \widetilde{\boldsymbol{\gamma}}$ with
    $\widetilde{\gamma}_n = \sum_{m'=1}^{|\mathscr{A}|} \frac{\sum_{m'=1}^{|\mathscr{A}|} \#\pi_{s'}^{m',n}}{\#\pi_s^{m,n}} \mathbb{P}(\pi_s^{m,n})$
    where $m, m' \leq |\mathscr{A}|$, $n \leq |\mathscr{C}|$ and $\mathbb{P}(\pi_s^{m,n}) = \widehat{\gamma}_s \in \widehat{\boldsymbol{\gamma}}$;
    $n++$;
end

3.2.8 ITFM: Intention-Aware Temporal Factorization Machines

We start with the basic form of Factorization Machines (FM) [20] which are originally proposed for collaborative recommendation. Specifically, given a P-dimensional, real valued feature vector $\mathbf{r}$, FMs are linear regression models that estimate the desired output by modelling all interactions between

each pair of features within the vector $\mathbf{r}$:

$$\hat{y}_{FM}(\mathbf{r}) = b_0 + \sum_{p=1}^{P} w_p r_p + \sum_{p=1}^{P} \sum_{p'=p+1}^{P} \mathbf{z}_p^\top \mathbf{z}_{p'} \cdot r_p r_{p'}, \tag{3.12}$$

where b_0 is the global bias, w_p is the weight assigned to p-th feature. $\mathbf{r}_p$, $\mathbf{r}_{p'} \in \mathbb{R}^z$ are corresponding embedding vectors for the p-th and p'-th feature, while z is the embedding dimension (i.e., dimensionality of the factorization [20]). Thus, the first two terms in Eq.(3.12) can be viewed as a linear regression scheme, while the third term Eq.(3.12) models the effect of pair-wise feature interaction [71]. In previous studies related to FM-based recommender systems [20, 21, 68, 84], the input vector $\mathbf{r}$ is commonly formulated as the concatenation of feature vectors representing the exact user u_i, all interacted items in the past T time steps and the targeted item $v_j \in \mathscr{V}$. Then, for user $u_i \in \mathscr{U}$, FMs are used to produce a ranking score for each item in $\mathscr{V}$. That is to say, though u_i may interact with a different item at each time step $t \leq T$, all the historical interactions are gathered together without any temporal order, and this severely restricts the model's capability of performing sequential recommendation.

As a result, we propose a novel variant of FM, namely ITFM (Intention-Aware Temporal Factorization Machine) to thoroughly capture the temporal characteristics of user-item interactions by utilising the aggregated user intention $\widetilde{\boldsymbol{\gamma}}$, the learned intention migration matrix Θ and the category information of the interacted items from the user transaction $\{(\pi^{m,n}_{s(t)}, v_{j(t)})\}_{t=1}^{T}$. Specifically, for user u_i, a ranking score for v_j in category c_n can be produced via:

$$\begin{aligned}\hat{y}_{ITFM}(\Omega_{ijn}) = b_0 + b_i + \frac{1}{T}\sum_{t'=1}^{T} w_{j'(t')}\widetilde{\gamma}_{n'(t')} + w_j\widetilde{\gamma}_n + \mathbf{v}_j^\top \mathbf{u}_i \widetilde{\gamma}_n + \frac{1}{T}\sum_{t'=1}^{T} \mathbf{v}_{j'(t')}^\top \mathbf{u}_i \widetilde{\gamma}_{n'(t')} \\ + \frac{1}{T|\mathscr{A}|}\sum_{t'=1}^{T}\sum_{s\in\mathscr{S}} \mathbf{v}_{j'(t')}^\top \mathbf{v}_j \theta_s^{s'} + \frac{1}{\sum_{\eta=1}^{T-1}\eta}\sum_{t'=1}^{T}\sum_{t^*=t'+1}^{T} \mathbf{v}_{j'(t')}^\top \mathbf{v}_{j^*(t^*)} \theta_{s^*}^{s'},\end{aligned} \tag{3.13}$$

where $\Omega_{ijn} = \left\{\{(\pi^{m,n}_{s(t)}, v_{j(t)})\}_{t=1}^{T}, \widetilde{\boldsymbol{\gamma}}, \Theta\right\}$ is the input set of ITFM containing the transaction and intention $\widetilde{\boldsymbol{\gamma}}$ for ***each specific user*** u_i along with the learned intention migration matrix Θ, while the index set $\mathscr{S}$ is constructed as follows:

$$\forall s \in \mathscr{S}, \pi_s^{m,n} \in \{\pi_s^{m',n} |\ m' = 1, ..., |\mathscr{A}|\}. \tag{3.14}$$

Note that we keep the superscripts for indexes j, n, s consistent with t to indicate a user's interaction details at the same time step, e.g., j', n', s' are all observed from the t'-th time step. As for the trainable parameters in ITFM, b_0 is the global bias, b_i is the local bias for user u_i, w_j is the weight assigned to item v_j, $\mathbf{u}_i$ and $\mathbf{v}_j$ are latent feature vectors for u_i and v_j respectively. ITFM can be viewed as a fine-grained version of FM that infers the ranking score $\hat{y}_{ITFM}(\Omega_{ijn})$ for any user-item pair (u_i, v_j) with corresponding input set Ω_{ijn}. Intuitively, ITFM factorizes the pairwise user-item and item-item interactions with: (1) awareness of user's future intention, represented by the intention strength $\widetilde{\gamma}_n \in \widetilde{\boldsymbol{\gamma}}$ on item category c_n; (2) sequential dependencies represented by the intention migration effect $\theta_{s'}^{s} \in \Theta$ on temporal interactions.

3.2.9 ITFM Generalizes FM

Compared with ITFM, FM is a static and shallow model because it can be viewed as a special case of ITFM by ignoring both the variations in user intention and the sequential dependencies in user-item interactions. To prove this, we start with a re-formulation of FM. According to [20], in FMs, assuming user u_i has interacted with T items in the past, the input feature vector $\mathbf{r}$ is constructed by a vector concatenation. Specifically, $\mathbf{r} = [\mathbf{r}_{u_i}, \mathbf{r}_{v_j}, \mathbf{r}_{trans}, \mathbf{r}_{side}]$, where $\mathbf{r}_{u_i}$ and $\mathbf{r}_{v_j}$ are respectively one-hot vectors for targeted user u_i and item v_j, $\mathbf{r}_{trans} = [0, ..., \frac{1}{T}, ..., \frac{1}{T}, ..., 0]$ with $\frac{1}{T}$ (i.e., averaged weight) at all interacted item indexes in the user transaction, and $\mathbf{r}_{side}$ is the feature vector for additional side information. Here, ignoring the side information, if we split $\mathbf{r}_{u_i}$, $\mathbf{r}_{v_j}$, $\mathbf{r}_{trans}$ from $\mathbf{r}$ and then use Eq.(3.12) to factorize the element-wise interaction among these three inputs, the elements with value 0 will be skipped. Consequently, the FM scheme in Eq.(3.12) becomes:

$$\begin{aligned}\hat{y}_{FM}(\mathbf{r}_{u_i}, \mathbf{r}_{v_j}, \mathbf{r}_{trans}) = b_0 + w_i + \frac{1}{T}\sum_{g=1}^{T} w_g + w_j + \mathbf{z}_{v_j}^{\top}\mathbf{z}_{u_i} + \frac{1}{T}\sum_{g=1}^{T}\mathbf{z}_{v_g}^{\top}\mathbf{z}_{u_i} \\ + \frac{1}{T}\sum_{g=1}^{T}\mathbf{z}_{v_g}^{\top}\mathbf{z}_{v_j} + \frac{1}{\sum_{\eta=1}^{T-1}\eta}\sum_{g=1}^{T}\sum_{g'=g+1}^{T}\mathbf{z}_{v_g}^{\top}\mathbf{z}_{v_{g'+1}},\end{aligned} \tag{3.15}$$

where $g \leq T$ is the index of each interacted item in the user's transaction, w_i is the weight for u_i and can be considered as a user bias term in this case, while $\mathbf{z}_v$ and $\mathbf{z}_u$ are respectively latent vectors. In ITFM, to discard both user intention and sequential dependencies, we can treat each $\widetilde{\gamma}_c \in \widetilde{\boldsymbol{\gamma}}$ and $\theta^s_{s'} \in \Theta$ equally by setting all the elements in $\widetilde{\boldsymbol{\gamma}}$ and Θ as 1. By doing so, we can recover Eq.(3.15) from Eq.(3.13) with interchangeable representations between the parameters w_i, $\mathbf{z}_u$,$\mathbf{z}_v$ in FM and b_i, $\mathbf{u}_i$,$\mathbf{v}_j$ in ITFM. Apparently, this validates that ITFM has generalized FM in a fine-grained and temporal way.

3.2.10 Loss Function for ITFM

In various recommendation tasks, user's explicit actions (e.g., ratings) on different items can offer important information for model learning. Hence, we score these actions according to the interaction intensity reflected by these actions, and then use the scores as the ground truth to learn ITFM as a pairwise ranking scheme. For example, in movie rating datasets, the exact ratings can be used as ground truth [85]; and there are widely-applied examples to rate user actions for model training, e.g., click, add-to-cart and purchase represent three ascending interaction intensities, and thus can be scored as 1, 2 and 3 respectively. We use y_{ijn} to denote the real quantified interaction that user u_i has performed on item v_j from category c_n. After computing the ranking score $\hat{y}_{ITFM}(\Omega_{ijn})$, we use the mean squared error to quantify the loss of ITFM:

$$\mathscr{L}_{ITFM} = \frac{1}{D}\sum_{d=1}^{D}\Big(y_{ijn} - \hat{y}_{ITFM}(\Omega_{ijn})\Big)^2, \tag{3.16}$$

where D is the total number of training samples.

3.2.11 Model Optimization

Following Section 3.2.2 and Section 3.2.6, with the major components of AIR formally introduced, here we discuss our optimization strategy to train the parameters in AIR. The entire workflow of AIR can be divided into two subtasks: (1) user intention prediction with ARNN; (2) intention-aware recommendation with ITFM. With the loss for user intention prediction defined in Section 3.2.2, we can easily leverage Stochastic Gradient Decent (SGD) algorithm to optimize the proposed ARNN. At the same time, in Section 3.2.6, we have demonstrated that our ITFM is a generalized version of the traditional factorization machines. Consequently, similar to previous work based on factorization machines [20, 21] and their variants [69, 71, 84], SGD can also be utilized to effectively learn the parameters in ITFM. Hence, we propose to learn all parameters AIR in a unified way instead of training these two components separately. We quantify the model loss of AIR by combining the loss functions for its components:

$$\mathscr{L} = \mathscr{L}_{ARNN} + \mathscr{L}_{ITFM} + \mathscr{L}_{Reg}, \tag{3.17}$$

where $\mathscr{L}_{Reg}$ represents the $L2$ regularization term posed on all the parameters within AIR to prevent overfitting:

$$\mathscr{L}_{Reg} = \lambda_2 \sum_{l=1}^{L} \phi_l^2, \tag{3.18}$$

where λ_2 is the weight decay coefficient on $\mathscr{L}_{Reg}$, ϕ_l represents each parameter, and L is the number of model parameters. It is worth mentioning that λ_1 for the attention constraint in $\mathscr{L}_{ARNN}$ and λ_2 are both hyperparameters to be tuned.

Though the ultimate goal of AIR is the accurate item recommendations from ITFM, the rationale of incorporating the loss functions for both ARNN and ITFM is that we expect the whole model to be optimized for both tasks in AIR (i.e., $\mathscr{L}_{ARNN}$ for user intention prediction and $\mathscr{L}_{ITFM}$ for recommendation), thus maximizing the performance gain from our proposed intention-aware recommendation scheme. Also, because ARNN and ITFM share a series of parameters and intermediate results, the joint loss greatly helps with the training efficiency, especially when being compared with 2-stage training paradigms.

In the training procedure, we leverage a mini-batch SGD algorithm, namely Adam [63] optimizer. Specifically, we set the batch size as 256 according to device capacity and the learning rate as 0.001. We iterate the whole training process until the loss converges.

3.3 Experiments

In this section, we conduct experiments on real-life datasets to showcase the advantage of AIR in the task of recommendation. At the same time, we also investigate the effectiveness of different proposed components of AIR. In particular, we aim to answer the following research questions (RQs) via the experiments:

RQ1 How effectively and accurately does AIR recommend desired items to each user?

RQ2 How AIR benefits from each component (i.e., ARNN and ITFM) in the proposed structure for intention-aware sequential top-k recommendation?

RQ3 How are the hyperparameters in AIR tuned to optimize both model effectiveness and training efficiency?

RQ4 How is the scalability of AIR when handling large-scale datasets in real-life recommendation scenarios?

3.3.1 Datasets

To validate the performance of AIR, we use two real-life datasets shown in Table 3.2, namely **MovieLens** and **Tmall**. The notation of # in Table 3.2 represents the number of entities. We briefly introduce the properties of these two datasets below:

- **MovieLens**: MovieLens[1] is a widely used dataset for recommender systems. The dataset which we conduct experiments on contains more than 1 million movie rating interactions. There are 18 movie category tags for different items and we treat explicit ratings (1, 2, 3, 4, 5) as the actions performed by users. For movies with multiple genre tags, we only use the most frequent one in the dataset. According to the average sequence length (165.60 time steps per user), we limit the length of input sequences to 200 time steps due to hardware capacity.

- **Tmall**: Tmall is a publicly available e-commerce dataset provided by Alibaba[2]. The original dataset contains approximately 12 million user transactions generated by 10,000 users. It contains four types of user behaviors: click, like, add-to-cart and purchase which are already assigned the interaction score of 1, 2, 3 and 4 by the annotators. In the preprocessing stage, similar to the construction strategy of MovieLens [86], we filter out both items and users having less than 20 interaction records. The statistics of the final Tmall dataset are listed in Table 3.2. Note that due to the large number of items in Tmall dataset, the sparsity is significant, making it increasingly difficult to generate accurate recommendations.

Given a collection of transactions $\mathscr{D}_i$ for each user u_i, we first sort all interactions according to their timestamps. Then, following [22, 61], for each transaction sequence, we use the 80% ratio as the cut-off point so that the records before this time point will be used for training while the rest are for evaluation. We denote these two subsets as $\mathscr{D}_i^{train}$ and $\mathscr{D}_i^{test}$ respectively. In the training phase of AIR, we use the first 70% of $\mathscr{D}_i^{train}$ as model input and the following 25% as labels, while the remaining 5% will be used for validation. In the evaluation phase, we use $\mathscr{D}_i^{train}$ as the input of AIR and test the accuracy on $\mathscr{D}_i^{test}$.

[1] http://grouplens.org/datasets/movielens/1m/
[2] https://tianchi.aliyun.com

Table 3.2: Statistics of datasets in use.

Dataset	#Interaction	#User	#Item	#Category	#Action	Sparsity
MovieLens	1,000,209	6,040	3,592	18	5	95.39%
Tmall	116,780	2,567	42,135	10	4	99.89%

3.3.2 Evaluation Criteria

To evaluate all recommendation models, we adopt the widely applied Hits Ratio at Rank k ($Hits@k$) which is commonly used in recommender system research [61, 85, 87]. Specifically, for each item $v_j \in \mathscr{D}_i^{test}$, we first randomly choose J items on which user u_i has never performed action and form J negative samples. Following [82, 88], J is set as 100 and 1,000 for MovieLens and Tmall respectively based on the scale of item variety. Secondly, we compute a score for v_j as well as the J negative samples via AIR. Thirdly, We form a ranked list by sorting these $J+1$ samples according to their scores in a descending order, where we use $rank(v_j)$ to denote the position of item v_j in the ranking list. Finally, we form a top-k item list by picking the k top-ranked items from the list, and if $rank(v_j) \leq k$, we have a ***hit***; otherwise, we have a miss. The computation of $Hits@k$ proceeds as follows. For a single test case $v_j \in \mathscr{D}_i^{test}$, its $hit@k$ is either 1 (for a hit) or 0 (for a miss). To avoid biases from users having a very large $\mathscr{D}_i^{test}$, the overall $Hits@k$ is calculated by firstly computing the hit ratio on all the test cases for each user u_i and then computing the mean for all users:

$$Hits@k = \frac{1}{|\mathscr{U}|} \sum_{i=1}^{|\mathscr{U}|} \frac{\#hit_i@k}{|\mathscr{D}_i^{test}|}, \tag{3.19}$$

where $\#hit_i@k$ is the number of hits in the test set for each user. In our experiments, we use the popular setting of $k = 5, 10, 20$ for validation [61, 87].

3.3.3 Baseline Methods

We conduct experiments against the following state-of-the-art recommendation frameworks:

- **BPR**: It is the widely-adopted Bayesian Personalized Ranking [19] matrix factorization model.
- **SPTF**: Scalable Probabilistic Tensor Factorization [61] was originally proposed to predict user behaviors and it also suits item recommendation scenarios well.
- **NFM**: This is the state-of-the-art, Neural Network-based Factorization Machine [69]. We first train it with the squared loss and then use it as the pairwise scoring function for top-k recommendation.
- **RRN**: Recurrent Recommender Networks [76] use LSTM as an autoregressive model to predict future behavioral trajectories of users by capturing temporal properties of user interests.
- **NARM**: It represents the Neural Attentive Recommendation Machine [89]. NARM performs session-based item recommendation using two parallel gated recurrent units (GRUs) with

attention mechanism for the joint modelling of sequential user behaviors and current user purposes.

- **Caser**: The Convolutional Sequence Embedding Recommendation Model [22] leverages convolutional neural networks to capture both users' general preferences and sequential patterns for sequential top-k recommendation.
- **SASRec**: The Self-Attententive Sequential Recommendation Model [23] leverages self-attention mechanisms to adaptively learn the context from both long-term and recent activities.
- **GES**: Graph Embedding with Side Information [90] is the state-of-the-art sequential recommendation approach based on graph embedding. It is worth mentioning that we only use the base version of GES because the category is the only product side information available but the enhanced GES requires multiple side information.

Furthermore, to fully study the performance gain from each component of our proposed model, we implement three variants of AIR as follows:

- **AIR-IW**: We replace the category-wise user intention with item-wise user preference by treating every user action equally (i.e., $|\mathscr{A}| = 1$) and assuming each item belongs to a unique category (i.e., $|\mathscr{C}| = |\mathscr{V}|$).
- **AIR-NA**: We remove the attention unit for intention migration modelling to build a non-attentional variant.
- **AIR-FM**: We replace the ITFM in AIR with a traditional factorization machine as Eq.(3.12). The aggregated intention vector $\widetilde{\boldsymbol{\lambda}}$ is directly concatenated with the feature vector $\mathbf{r}$ as input.

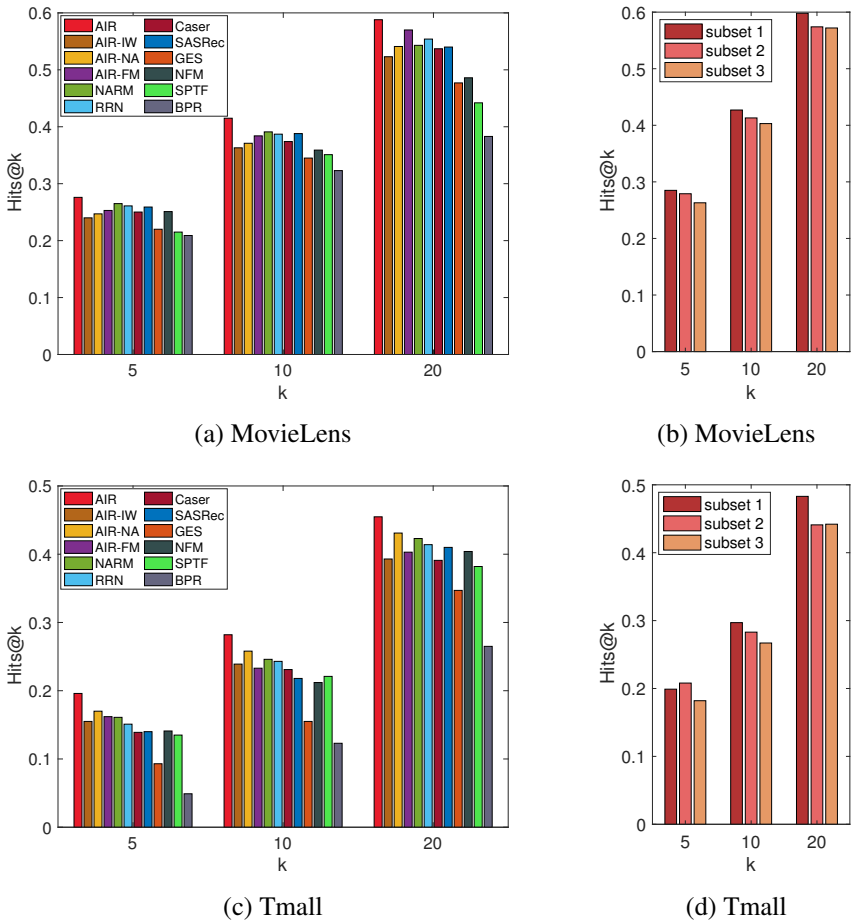
3.3.4 Parameter Settings

In AIR, we adopt a 3-layer LSTM structure with the dimension of hidden states $\mathbf{h}$ set to 128. For consistency, we also apply the same hidden dimension for the intention embedding $\mathbf{x}$, the user feature vector $\mathbf{u}$ and the item feature vector $\mathbf{v}$. The number of LSTM layers and dimension size are respectively tuned via the grid search over $\{1,2,3,4,5\}$ and $\{32,64,128,256,512\}$. We will further discuss the process of obtaining a trade-off between model performance and efficiency in Section 3.3.7. Besides, to determine the weight decay penalties, namely λ_1 and λ_2, we also conduct grid search over $\{0.0001, 0.001, 0.01, 0.1, 1\}$. The settings with the best performance on the validation set ($\lambda_1 = 0.01$ and $\lambda_2 = 0.001$ on MovieLens, $\lambda_1 = \lambda_2 = 0.001$ on Tmall) are used in the test.

3.3.5 Overall Recommendation Effectiveness (RQ1)

3.3.5.1 Comparisons with Baselines

We report the $Hits@k$ results generated by all the tested methods with $k = 5, 10, 20$ in Figure 3.2.(a) and 3.2.(c). Clearly, on both datasets, our proposed AIR constantly outperforms all competitors

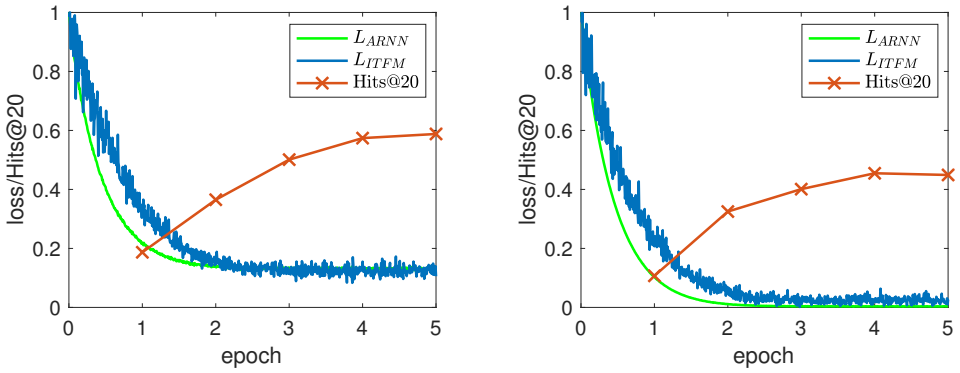


(a) MovieLens (b) MovieLens

(c) Tmall (d) Tmall

Figure 3.2: Recommendation results w.r.t. $Hits@k$ on two datasets. (a) and (c) are the comparisons with baselines, (b) and (d) are the sequential simulation results.

with statistically significant margins ($p < 0.01$). Specifically, AIR scores 0.276, 0.415 and 0.588 for $Hits@5$, $Hits@10$, $Hits@20$ on MovieLens, while the corresponding results are 0.196, 0.282 and 0.455 on Tmall. Additionally, we have made several observations based on the effectiveness results. First, apart from AIR and its variants, it is as expected that all deep neural network-based models (i.e., NARM, RRN, Caser, SASRec and NFM) yield better effectiveness in recommendation tasks than factorization models (i.e., BPR, SPTF, NFM) and network embedding models (i.e., GES). This is because deep neural network-based models are capable of modelling complex non-linear relationships between different factors extracted from the data, which are effective when tackling sparse data and limited side information. Second, we find that sequential recommender systems tend to have more accurate top-k recommendation results compared with static recommender systems (i.e., SPTF and BPR) on both datasets. That is to say, when large amounts of sequential user transactions are available, if we can successfully capture the dynamics within the data, there is expected to be a considerable improvement on the recommendation performance. Third, as shown in Figure 3.2.(a) and 3.2.(c), the most significant improvement is observed when k is relatively small, especially for $k = 10$ on MovieLens dataset and $k = 5$ on Tmall. This indicates the joint effect of ARNN for user prediction and ITFM for intention-aware recommendation greatly enhances the modelling of both users and items in a unified way.



(a) MovieLens (b) Tmall

Figure 3.3: The relationship between $\mathscr{L}_{ARNN}$, $\mathscr{L}_{ITFM}$ and $Hits@20$. Both losses are linearly rescaled to $[0, 1]$ for better readability.

3.3.5.2 Sequential Recommendation Simulation

To demonstrate the practicality of AIR for timely recommendation, we further test AIR by evenly dividing the test set $\mathscr{D}_i^{test}$ for each user u_i by time into three consecutive subsets: ${}^1\mathscr{D}_i^{test}$, ${}^2\mathscr{D}_i^{test}$ and ${}^3\mathscr{D}_i^{test}$. We test the performance of AIR on these three consecutive subsets in a sequential manner ***without retraining***. To be specific, when testing on ${}^1\mathscr{D}_i^{test}$, ${}^2\mathscr{D}_i^{test}$ and ${}^3\mathscr{D}_i^{test}$, the inputs of AIR are respectively $\mathscr{D}_i^{train}$, $\mathscr{D}_i^{train} \cup {}^1\mathscr{D}_i^{test}$ and $\mathscr{D}_i^{train} \cup {}^1\mathscr{D}_i^{test} \cup {}^2\mathscr{D}_i^{test}$, and this mimics the real-life scenario of sequential recommendation. The results on each subset are reported in Figure 3.2.(b) and 3.2.(d). Generally, without retraining on new input data, some minor performance drop can be observed from the second and the third subset, but AIR maintains its recommendation performance within a stable range. This implies the dynamic patterns learned by AIR are robust and stable because AIR can effectively leverage the category and behavior information within sequential user transactions to model timely user intention for accurate recommendation.

3.3.6 Importance of Each Model Component (RQ2)

In this section, we thoroughly examine the performance gain from the two main components (i.e., ARNN and ITFM) in AIR via a series of ablation tests and intuitive visualizations.

3.3.6.1 Overview

We start with our observation during the training process of AIR. According to our loss function in Eq.(3.17), if we ignore the regularization term, we can actually split the total loss into two individual parts, i.e., $\mathscr{L}_{ARNN}$ and $\mathscr{L}_{ITFM}$. To seize a systematic understanding of the joint contribution of ARNN and ITFM to AIR, we propose to keep a track of both individual losses during the training process as well as the model performance after each training epoch (i.e., a whole iteration through all the training data). By this means, we visualize the converge process of $\mathscr{L}_{ARNN}$ and $\mathscr{L}_{ITFM}$ as well as the fluctuations in recommendation performance with Figure 3.3. The loss values are recorded every 10 batch training steps, and we evaluate the recommendation performance of AIR in terms of $Hits@20$

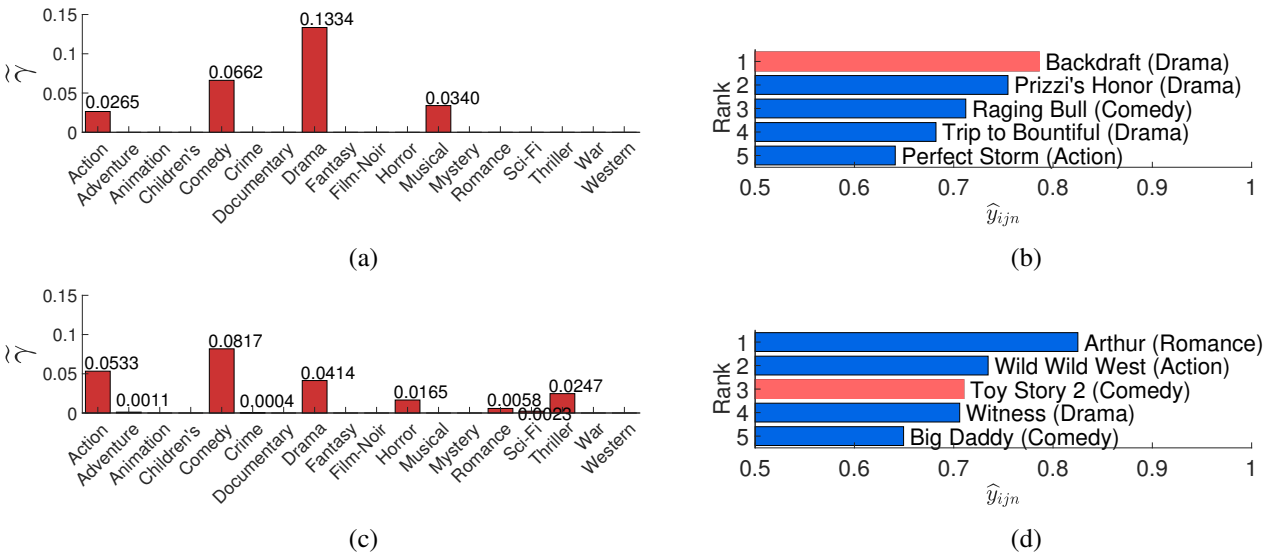


Figure 3.4: Relationship between the predicted and aggregated user intention $\widetilde{\boldsymbol{\gamma}}$ (left) and top 5 recommendation results (right) on MovieLens. (a)-(b) and (c)-(d) are respectively sampled from two different users. In (a) and (c), the horizontal axis represents different dimensions for item categories (i.e., movie genres in this case) while the vertical axis represents the corresponding aggregated user intention value. In (b) and (d), we use red bars for the ground truth, while the horizontal axis represents the personalized ranking score (linearly rescaled to $[0, 1]$) for different items and the vertical axis represents the rank of each item.

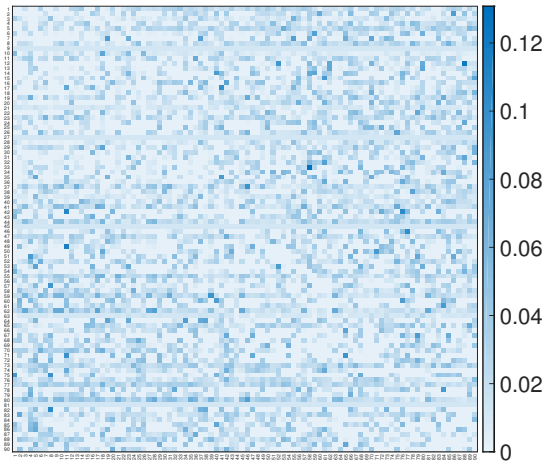


Figure 3.5: The intention migration matrix Θ learned via the proposed attention mechanism in AIR. Each grid represents a $\theta^s_{s'} \in \Theta$. The intensity of colour in grid $\theta^s_{s'}$ reflects the strength of migration effect from the s-th intention candidate to the s'-th intention candidate. Zoom in for a better view.

right after each epoch finishes. We illustrate this process via Figure 3.3. On these two datasets, while both losses are decreasing during training, the recommendation performance of AIR gradually rises as well. So, it can be concluded that these components are naturally combined by the loss function of AIR to simultaneously optimize the final performance.

3.3.6.2 The Impact of Category-wise User Intention

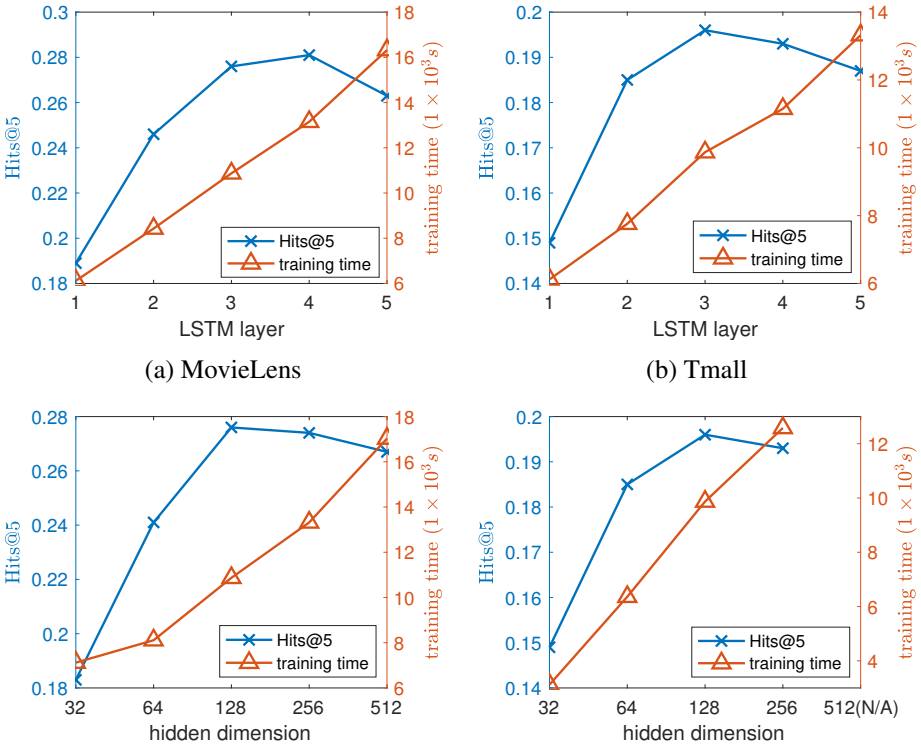
As an important component of our method, we validate the impact of category-wise user intention via AIR-IW which only models the item-wise user preference and treats all kinds of user actions homogeneously. Correspondingly, AIR-IW can only model the item-wise preference drift. The significant accuracy drop shown in Figure 3.2.(a) and 3.2.(c) verifies that our proposed user intention scheme contributes positively to the performance gain. Besides, among three of the degraded versions of AIR, AIR-IW experiences the most performance decrease compared with AIR-NA and AIR-FM. This is because AIR-IW tries to learn dynamic patterns from the sparse item-wise user performance and ignores the heterogeneity of different user action types, making the model fail to capture sufficient sequential dependencies and varied behavioral semantics. In summary, it proves that our hypothesis of collectively modelling category-wise user intention and heterogeneous user interaction behaviors instead of directly learning user preference can substantially benefit sequential top-k recommendation.

3.3.6.3 ARNN for User Intention Prediction

In Section 3.3.3, we implement AIR-NA by blocking the attention mechanism for intention migration modelling. As demonstrated in Figure 3.2, AIR-NA suffers from a noticeable drop on recommendation accuracy. In Figure 3.5, we have further visualized the intention migration matrix Θ learned by the full version of AIR model. The MovieLens dataset is used because it offers original tags of movie genre (e.g., action, comedy, etc.). From the visualization, we can directly obtain some insights by decoding the indexes of $\theta^{s}_{s'} \in \Theta$ into actual action-category pairs. For example, the intention $\theta^{12}_{88} \in \Theta$ is high, and it represents the intention migration effect from "rating a musical movie as 1/5" (corresponds to index 12) to "rating a thriller movie as 5/5" (corresponds to index 88). This indicates that if a user dislikes musical movies, there is a high chance that this user will prefer thriller movies. Also, $\theta^{89}_{65} \in \Theta$ is a strong intention migration factor, and it represents the intention migration effect from "rating a war movie as 5/5" (corresponds to index 89) to "rating a horror movie as 4/5" (corresponds to index 65). This further reflects that if a user likes war movies, it is highly possible for this user to have great interest in horror movies. Hence, the ARNN discovers the subtle yet reasonable intention migration patterns for intention prediction, thus presenting an accurate and explainable recommender system.

3.3.6.4 ITFM for Intention-Aware Recommendation

Similar to AIR-NA, in AIR-FM which replaces the ITFM unit with a traditional factorization machine (FM) [20], we also notice an obvious performance decrease on both datasets. This is due to the fact that AIR-FM simply uses the user intention vector as a kind of ad-hoc input feature instead of modelling the sequential dependencies for user-item interactions in a fine-grained manner. Besides, we also use Figure 3.4 to visualize the relationship between the predicted user intention (here the aggregated user intention $\widetilde{\boldsymbol{\gamma}}$ is used) and the top 5 recommendation results ($\widehat{y}_{ijn}$ is used). Because the categories and items are anonymized in Tmall dataset, we randomly sampled two successful recommendation cases from MovieLens only. Note that $\widetilde{\boldsymbol{\gamma}}$ is ***no longer a probability distribution*** after the category-wise



(a) MovieLens (b) Tmall

(c) MovieLens (d) Tmall

Figure 3.6: Parameter sensitivity analysis on the number of LSTM layers and the size of hidden dimension.

intention aggregation, hence $\sum_{c=1}^{N} \widetilde{\gamma}_c$ does not necessarily equal to 1. We also omit values less than 1.0×10^{-5} in the visualization for better readability. In the first case, i.e., Figure 3.4.(a)-(b), this user's intention on movie genre "Drama" is dominant, so the ITFM component naturally ranks the ground truth (marked in red) as the first item to recommend. In the second case, i.e., Figure 3.4.(c)-(d), though a relatively wide distribution of category-wise intention yields challenges for recommending the most suitable item to the user, ITFM still manages to gather information from the sequential interactions of this user, thus keeping the right item within the top selections (rank 3). Furthermore, the results of item recommendation are highly consistent with the learned user intention in terms of preferred item categories, which indicates that the proposed ITFM greatly benefits the model performance in sequential top-k recommendation.

3.3.7 Analysis on Hyperparameters (RQ3)

It is crucial for recommendation models to achieve optimal performance while preserving reasonable training efficiency. AIR contains four hyperparameters to be tuned, namely the number of LSTM layers, the size of hidden dimension and two penalty weights λ_1, λ_2. Because the impact of λ_1 and λ_2 on model training time is ignorable, they are simply determined via the grid search as mentioned in Section 3.3.4. From the perspective of performance as well as training time, we extensively investigate how sensitive AIR is to the number of LSTM layers and hidden dimension size. When training AIR on these two datasets, we study the changes in $Hits@5$ and training time simultaneously via two settings:

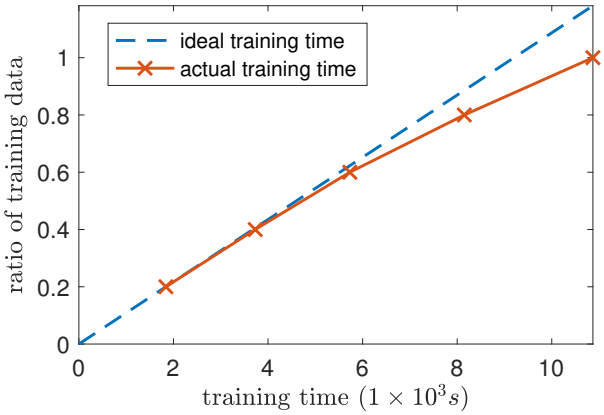


Figure 3.7: Training time for AIR w.r.t. varied data ratio.

(1) with the hidden dimension fixed to 128, we study the parameter sensitivity to the number of LSTM layers in $\{1,2,3,4,5\}$; (2) with the number of LSTM layers fixed to 3, we study the parameter sensitivity to the size of hidden dimension in $\{32,64,128,256,512\}$. Figure 3.6.(a)-(b) and Figure 3.6.(c)-(d) reports the results under the first and second setting respectively. Note that the test with 512 hidden dimension size on Tmall dataset is non-applicable as the graphic card memory overflows. As can be inferred from the figure, on both datasets, almost all the best $Hits@5$ results are achieved by a 3-layer LSTM structure with 128 as the hidden dimension size, and the training time remains in a reasonable range. However, it is worth mentioning that according to Figure 3.6.(a), though the performance produced by 4-layer LSTM is slightly better than 3-layer LSTM on MovieLens, it takes considerably more time for training (21.45% increase). Thus, we choose 3 for the number of LSTM layers and 128 for hidden dimension size as a trade-off between recommendation accuracy and model training efficiency.

3.3.8 Model Scalability (RQ4)

Due to the importance of practicality in real-life applications of recommender systems, we validate the scalability of AIR in this section. When all the parameters in the network are fixed, the training time for AIR is only associated with the number of training samples. Ideally, the training time for AIR should increase linearly as we enlarge the scale of the training data. Note that we apply the same parameter settings in Section 3.3.4 and training strategies in Section 3.2.11 for this validation.

We test the training efficiency and scalability of AIR by varying the proportions of the whole training set from MovieLens, and then report the corresponding training time for the loss function of AIR to converge (excluding I/O). The growth of training time along with the data size is shown in Fig.3.7. When the ratio of training data gradually extends from 0.2 to 1.0, the training time for AIR increases from 1.84×10^3 seconds to 10.87×10^3 seconds. It shows that the link between training time and the data scale is approximately linear. Hence, we conclude that since its linear time complexity can ensure high scalability, AIR can be efficiently trained with large-scale datasets for real-life recommendation tasks.

3.4 Summary

In this chapter, to tackle the sequential top-*k* recommendation, we present a novel attentional intention-aware recommender system (AIR) that recognizes dynamic user intention and performs accurate item recommendation. In AIR, the attentional recurrent neural network precisely captures user intention by modelling the intention migration effect, while the intention-aware temporal factorization machine effectively leverages the sequential dependencies in user transactions to perform intention-aware item recommendation. We also conduct extensive experiments to validate the effectiveness and practicality of AIR in terms of item recommendation. Thus, the specific contributions of this work are as follows:

- We point out that modelling category-wise user intention and collectively exploiting rich heterogeneous user interaction behaviors are more effective ways to address the data sparsity issue in sequential top-*k* recommendation.
- We propose AIR, a flexible yet effective model to predict user intention and conduct intention-aware recommendation on sequential data in a unified way. In AIR, we develop ARNN for user intention prediction, where our novel attention mechanism is designed for intention migration modelling. When scoring each user-item pair for top-*k* recommendation, we propose ITFM to capture the dynamic user intention patterns learned via ARNN.
- We conduct extensive experiments on two real-life datasets. The results showcase the superiority of our approach in the comparison with state-of-the-art baselines. We validate the contribution of each component in AIR via ablation tests and visualizations. Further experiments show promising practicality and scalability of AIR.

Chapter 4

Uniting Macro- and Micro-Level Sequence Modelling

The research goal in this chapter is to devise a generalizable temporal predictive analytic model that can be effectively applied to both macro- and micro-level sequence modelling in e-commerce. In this chapter, we will firstly review relevant works, then propose our model based on the identified defects of existing methods.

4.1 Literature Review: Background and Motivation

4.1.1 Predictive Analytics under Sparsity

As an important supervised learning scheme, predictive analytics play a pivotal role in various applications, ranging from recommender systems [76,91] to financial analysis [1] and online advertising [27, 92]. In a nutshell, the ultimate goal of predictive analytics is to learn an effective predictor that accurately estimates the output according to the input features, where classic predictive methods like support vector machines (SVMs) [93] and logistic regression (LR) [94] have gained extensive popularity.

When dealing with categorical features in predictive analytics, a common approach is to convert such features into one-hot encodings [21,88,95] so that standard regressors like logistic regression [94] and support vector machines [93] can be directly applied. Distinct from the continuous raw features from images and audios, features from the web-scale data are mostly discrete and categorical [69]. Due to the large number of possible category variables, the converted one-hot features are usually of high dimensionality but sparse [69], and simply using raw features rarely provides optimal results. On this occasion, the interactions among different features act as the winning formula for a wide range of data mining tasks [95–97]. The interactions among multiple raw features are usually termed as ***cross features*** [95] (a.k.a. multi-way features and combinatorial features). For example, individual variables $occupation = \{lecturer, engineer\}$ and $level = \{junior, senior\}$ can offer richer contextual

information for user profiling with cross features, such as $(junior, engineer)$ and $(senior, lecturer)$. To avoid the high cost of task-specific manual feature engineering, ***factorization machines*** (FMs) [20] are proposed to embed raw features into a latent space, and model the interactions among features via the inner product of their embedding vectors.

4.1.2 Evolution of Factorization Machines

To better capture the effect of feature interactions, variants of the plain FM are proposed, like field-aware FM for online advertising [92] and CoFM [84] for user behavior modelling. However, these variants are still constrained by their limited linear expressiveness [69] when modelling the subtle and complex feature interactions. Another line of research on FM-based models for predictive analytics incorporates deep neural networks (DNNs) [69, 70, 98, 99]. Recently, motivated by the capability of learning discriminative representations from raw inputs, deep neural networks (DNNs) [38] have been adopted to extend the plain FM. For instance, He *et al.* [69] bridge the cross feature scheme of FM with the non-linear form of DNN, and proposes a neural factorization machine (NFM). Instead of the straightforward inner product in FM, NFM takes the sum of all features' linear pairwise combinations into a feed-forward neural network, and generates a latent representation of high-order feature interactions. This is in a similar spirit to models like FM-supported neural network (FNN) [98], where DNNs are utilized to learn non-linear high-order feature interactions. Both NFM and FNN also use the embeddings from a pre-trained FM before applying DNNs in order to speed up training and improve the prediction accuracy. With the idea of learning high-order feature interactions with DNNs, more DNN-based FMs are devised for predictive analytics [70,71,95,100]. Qu *et al.* propose a product-based neural network (PNN) [99], which introduces a product layer between embedding layer and DNN layer, and does not rely on pre-trained FM parameters. More recently, hybrid architectures are introduced in Wide&Deep [101], DeepFM [100] and xDeepFM [70] by combining shallow components with deep ones to capture both low- and high-order feature interactions.

In short, there are two major trends of improvements over the plain FM. One is to make the model "deep" with multi-layer network structures in order to exhaustively extract useful information from feature interactions, e.g., the residual network in DeepCross [95], the pairwise product layer in PNN [99], and the compressed interaction network in xDeepFM [96]. The other is to make the model "wide" by considering multiple feature interactions in varied domains (usually coupled with "deep" structures), e.g., separately modelling user logs and texts with CoFM [84], or fusing shallow low-order output with dense high-order output via Wide&Deep [101], DeepFM [100] and xDeepFM [70]. Note that in the remainder of this chapter, to avoid ambiguity, we use the term ***FM-based models*** to imply both the plain FM and all its variants.

4.1.3 Motivation and Our Solution

However, these popular FM-based models mostly perform predictive analytics with the assumption that there is no temporal order in the data. As a result, regardless of the temporal information available

in various prediction tasks, the data will be partitioned for training/evaluation randomly rather than chronologically, such as [21, 69, 71, 102]. Considering a real-world recommendation scenario, the time-dependent order of products purchased by each user should be considered, and the recommender system can only utilize users' past purchase records to estimate their future preferences [103]. To this end, we focus on the problem of ***temporal predictive analytics*** which considers such temporal causality, and is more practical and realistic in various application scenarios.

Despite the efforts on enhancing the plain FM, all the aforementioned FM-based models still lack the consideration of the sequential dependencies within high-order feature interactions, which is proven to be critical for many temporal prediction tasks [1, 23, 37, 49]. With the rapidly increasing volume of web-scale data, temporal predictive analytics inevitably involves features that are dynamically changing over time, e.g., users' shopping transactions on e-commerce platforms. We term such features the ***dynamic features***. In contrast, we refer to features that stay fixed (e.g., user ID) as ***static features***. Let us consider a generic item recommendation task, where the goal is to predict whether a user will buy a specific item or not, as shown in Figure 4.1. Apart from the one-hot encoding of both the user and candidate item, the common way for current FM-based models to account for this user's shopping record is to derive set-category features [21, 27, 102] that mark all her/his previously bought items (see Figure 4.1). As is inferred from the user's transaction (jeans $\rightarrow$ jacket $\rightarrow$ computer $\rightarrow$ mouse), the current intent of this user is to purchase accessories for her/his new computer like keyboards, rather than other clothes. However, since traditional FM-based models view all the purchased items from a constant point of time, all these dynamic features are evenly treated when modelling feature interactions. Consequently, traditional FM-based models can hardly distinguish the likelihood of purchasing a keyboard with purchasing a belt, because there are similar items in the set-category features for both keyboards and belts, and the sequential characteristics of dynamic features cannot be properly captured. Though the recently proposed translation-based FM [104] performs recommendation by taking the sequential property of features into account, it models the influence of only the last item (i.e., the mouse), thus easily making the recommended keyboard a mismatch for the purchased computer. Moreover, for FM-based models, the deficiency of handling sequential dependencies will create a severe performance bottleneck when the diversity and amount of dynamic features grow over time.

In light of this, we aim to develop a general yet effective FM-based model to thoroughly mine the sequential information from the dynamic features for accurate temporal predictive analytics. Hence, in this thesis, we propose a ***Sequence-Aware Factorization Machine*** (SeqFM), which is the first FM-based model to systematically combine sequential dependencies with feature interactions while inheriting the non-linear expressiveness from DNNs and retaining the compactness w.r.t. the plain FM. As demonstrated in Figure 4.1, SeqFM is built upon a multi-view learning scheme. Due to different semantic meanings carried by static and dynamic features, we model different types of feature interactions in three different contexts (i.e., views): static view for static features, dynamic view for dynamic features, and cross view for both. To bypass the high demand on space and time of sequential neural models using convolutional or recurrent computations, in each specific view, we leverage the

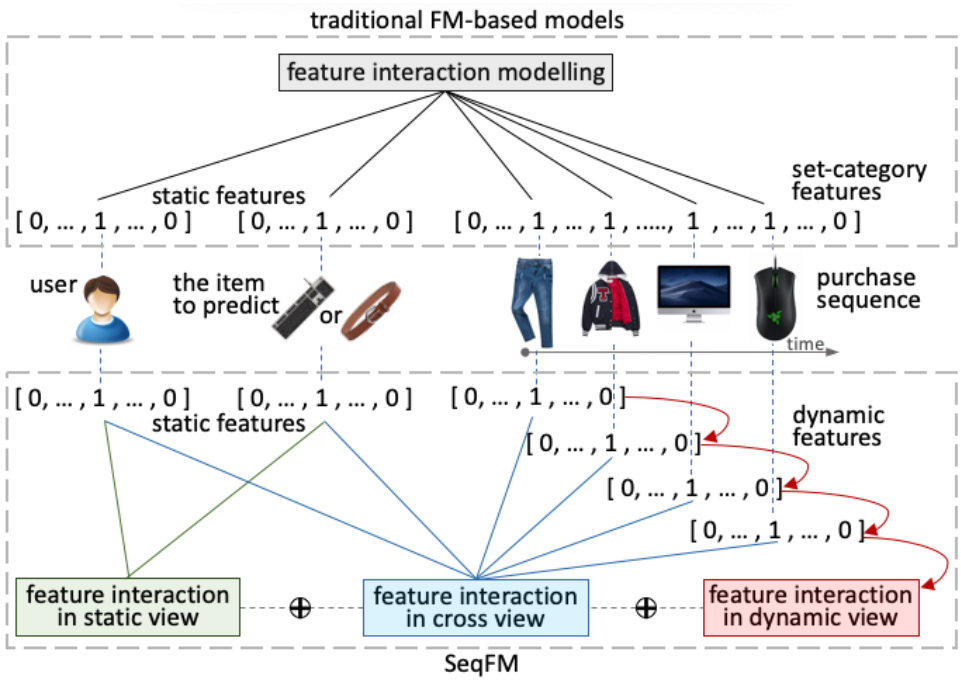


Figure 4.1: The differences in feature interaction modelling between traditional FM-based models (upper part) and our proposed SeqFM (lower part). Note that the embedding process of sparse features is omitted to be succinct.

self-attention mechanism [62], which is highly efficient and capable of uncovering sequential and semantic patterns between features. For the dynamic view and cross view, we further propose two masked self-attention units to respectively preserve the directional property of feature sequence and block irrelevant feature interactions. After encoding the high-order interactions between features via the multi-view self-attention, a shared residual feed-forward network is deployed to extract latent information from feature interactions. Intuitively, compared with "deep" or "wide" FM variants, we aim to make our model "sequence-aware", thus making full use of the contexts within dynamic features. As a flexible and versatile model, we introduce three application scenarios for SeqFM, namely ranking, classification, and regression, where corresponding experiments reveal significant improvements over existing FM-based models. Furthermore, the simple structure of SeqFM also ensures linear computational complexity and light-weight parameter size.

4.2 The Proposed Model: SeqFM

4.2.1 Preliminaries

Notations. Throughout this chapter, all vectors and matrices are respectively denoted by bold lower case and bold upper case letters, e.g., $\mathbf{g}$ and $\mathbf{G}$. All vectors are **row vectors** unless specified, e.g., $\mathbf{x} \in \mathbb{R}^{1\times n}$. To maintain simplicity, we use the superscripts $\circ$, $\triangleright$ and $\star$ to distinguish parameters in the static view, dynamic view and cross view, respectively.

Factorization Machines (FMs). FMs are originally proposed for collaborative recommendation. Specifically, for a given instance $[userID = 2, gender = male, cities\ visited = Sydney\&Shanghai]$, its

input is a high-dimensional sparse feature $\mathbf{x} \in \{0,1\}^{1\times m}$ constructed by the concatenation of multiple one-hot encodings [70, 95, 99]:

$$\mathbf{x} = \underbrace{[0,1,0,...,0]}_{\text{user ID}}\ \underbrace{[1,0]}_{\text{gender}}\ \underbrace{[0,1,0,1,0,...,0]}_{\text{cities visited}}, \tag{4.1}$$

where any real-valued feature (e.g., age) can also be directly included in $\mathbf{x}$ [21, 69], but we will focus on the sparse categorical feature in our solution. Then, FMs are linear predictors that estimate the desired output by modelling all interactions between each pair of features within $\mathbf{x}$ [20]:

$$\widehat{y} = w_0 + \sum_{i=1}^{m} w_i x_i + \sum_{i=1}^{m}\sum_{j=i+1}^{m} \langle \mathbf{v}_i, \mathbf{v}_j \rangle \cdot x_i x_j, \tag{4.2}$$

where m is the total amount of features, w_0 is the global bias, w_i is the weight assigned to the i-th feature, and $\langle .,. \rangle$ denotes the dot product of two vectors. $\mathbf{v}_i$, $\mathbf{v}_j \in \mathbb{R}^{1\times d}$ are corresponding embedding vectors for feature dimension i and j, while d is the embedding dimension. Thus, the first two terms in Eq.(4.2) can be viewed as a linear weighting scheme, while the third term models the effect of pairwise feature interactions [71].

4.2.2 Sequence-Aware Factorization Machines

In this section, we first overview our proposed Sequence-Aware Factorization Machines (SeqFM), and then detail each key component in the model.

Given a sparse feature vector $\mathbf{x} \in \{0,1\}^{1\times m}$, the output $\widehat{y}$ of SeqFM is computed via:

$$\widehat{y} = w_0 + \sum_{i=1}^{m} w_i x_i + f(\mathbf{x}), \tag{4.3}$$

where the first two terms denote the linear components similar to the ones in Eq.(4.2), and the global bias and weights of different features are modelled respectively. $f(\mathbf{x})$ denotes our proposed factorization component. Based on the construction rule of $\mathbf{x}$, it can be viewed as the additive form of the one-hot encodings for all non-zero features. Thus, $\mathbf{x} = \sum_{i=1}^{n} \mathbf{g}_i$, where $\mathbf{g}_i = [0,...,0,1,0,...,0]$ is an m-dimensional one-hot vector corresponds to one individual non-zero feature, and n denotes the total number of non-zero features.

To conduct temporal predictive analytics with sequence-awareness, we split the original sparse feature vector $\mathbf{x}$ into two views, namely the ***static view*** and ***dynamic view***. In the running example of Eq.(4.1), user ID and gender are modelled in the static view while visited cities are modelled in the dynamic view. Then, we can obtain the static feature $\mathbf{x}^{\circ} \in \{0,1\}^{1\times m^{\circ}}$ and dynamic feature $\mathbf{x}^{\triangleright} \in \{0,1\}^{1\times m^{\triangleright}}$ where $m^{\circ} + m^{\triangleright} = m$. Correspondingly, the additive form of input features naturally splits into $\mathbf{x}^{\circ} = \sum_{i=1}^{n^{\circ}} \mathbf{g}_i^{\circ}$ and $\mathbf{x}^{\triangleright} = \sum_{i=1}^{n^{\triangleright}} \mathbf{g}_i^{\triangleright}$, where n° and $n^{\triangleright}$ are the respective numbers of non-zero features in two views, and $n^{\circ} + n^{\triangleright} = n$. Here, we use feature matrices $\mathbf{G}^{\circ} \in \{0,1\}^{n^{\circ}\times m^{\circ}}$ and $\mathbf{G}^{\triangleright} \in \{0,1\}^{n^{\triangleright}\times m^{\triangleright}}$ to stack these sparse input vectors, of which each row is an individual one-hot vector.

It is worth mentioning that the dynamic feature matrix $\mathbf{G}^{\triangleright}$ is constructed in a chronological order. That is to say, $\mathbf{G}^{\triangleright}$ can be viewed as a sequence of dynamic features, so for row $i < j$, $\mathbf{g}_i^{\triangleright} \in \mathbf{G}^{\triangleright}$ is

always observed earlier than $\mathbf{g}_j^{\triangleright} \in \mathbf{G}^{\triangleright}$. As dynamic features may update frequently over time, we pose a threshold on the maximum sequence length that our model handles. To make the notations clear, we keep using $n^{\triangleright}$ to denote the maximum length for the dynamic feature sequence. If the dynamic feature sequence length is greater than the specified $n^{\triangleright}$, we consider the most recent $n^{\triangleright}$ features. If the sequence length is less than $n^{\triangleright}$, we repeatedly add a padding vector $\{0\}^{1\times m^{\triangleright}}$ to the top of $\mathbf{G}^{\triangleright}$ until the length is $n^{\triangleright}$.

So far, we can rewrite the SeqFM model in Eq.(4.3) as:

$$\widehat{y} = w_0 + [(\mathbf{G}^{\circ}\mathbf{w}^{\circ})^{\top}; (\mathbf{G}^{\triangleright}\mathbf{w}^{\triangleright})^{\top}]\mathbf{1} + f(\mathbf{G}^{\circ}, \mathbf{G}^{\triangleright}), \tag{4.4}$$

where $\mathbf{w}^{\circ} \in \mathbb{R}^{m^{\circ}\times 1}$ and $\mathbf{w}^{\triangleright} \in \mathbb{R}^{m^{\triangleright}\times 1}$ are column vectors representing weights for all features, $[\cdot;\cdot]$ denotes the horizontal concatenation of two vectors, and $\mathbf{1}$ is a $(n^{\circ}+n^{\triangleright}) \times 1$ vector consisting of 1s. In Eq.(4.4), the first two terms serve the same purpose as those in Eq.(4.3), while $f(\mathbf{G}^{\circ}, \mathbf{G}^{\triangleright})$ denotes the multi-view self-attentive factorization scheme. The work flow of SeqFM is demonstrated in Figure 4.2. In what follows, we will describe the design of $f(\mathbf{G}^{\circ}, \mathbf{G}^{\triangleright})$ in detail.

4.2.3 Embedding Layer

As demonstrated in Figure 4.2, we first convert the sparse features $\mathbf{G}^{\circ}$ and $\mathbf{G}^{\triangleright}$ into dense representations with embedding. The embedding scheme is essentially a fully connected layer that projects each one-hot feature $\mathbf{g}$ to a dense embedding vector as the following:

$$\begin{aligned} \mathbf{E}^{\circ} &= \mathbf{G}^{\circ}\mathbf{M}^{\circ}, \\ \mathbf{E}^{\triangleright} &= \mathbf{G}^{\triangleright}\mathbf{M}^{\triangleright}, \end{aligned} \tag{4.5}$$

where $\mathbf{M}^{\circ} \in \mathbb{R}^{m^{\circ}\times d}$ and $\mathbf{M}^{\triangleright} \in \mathbb{R}^{m^{\triangleright}\times d}$ are embedding matrices in the static and dynamic view, and d is the latent embedding dimension. As such, we can obtain two embedded feature matrices $\mathbf{E}^{\circ} \in \mathbb{R}^{n^{\circ}\times d}$ and $\mathbf{E}^{\triangleright} \in \mathbb{R}^{n^{\triangleright}\times d}$, where each row is a embedding vector for the original feature.

4.2.4 Static View with Self-Attention

From Eq.(4.2), it is clear that in the traditional FM, feature interactions are modelled in a vector-wise manner [70], where the dot product of two vectors is used. To better encode the subtle and fine-grained information, recent FM-based models [69, 71, 99] shift to bit-wise (a.k.a. element-wise) interactions of feature embeddings, such as element-wise product and weighted sum. In order to comprehensively capture the complex interactions among features, we propose to jointly investigate vector-wise and bit-wise feature interactions with the self-attention [62], which is a linear module that can be efficiently computed. We start with the self-attention module in the static view:

$$\mathbf{H}^{\circ} = \text{softmax}\Big(\frac{\mathbf{Q}^{\circ}\mathbf{K}^{\circ\top}}{\sqrt{d}}\Big)\mathbf{V}^{\circ}, \tag{4.6}$$

where $\mathbf{H}^{\circ} \in \mathbb{R}^{n^{\circ}\times d}$ is the latent interaction representation for all n° static features, while $\sqrt{d}$ is the scaling factor to smooth the row-wise $SoftMax$ output and avoid extremely large values of the inner

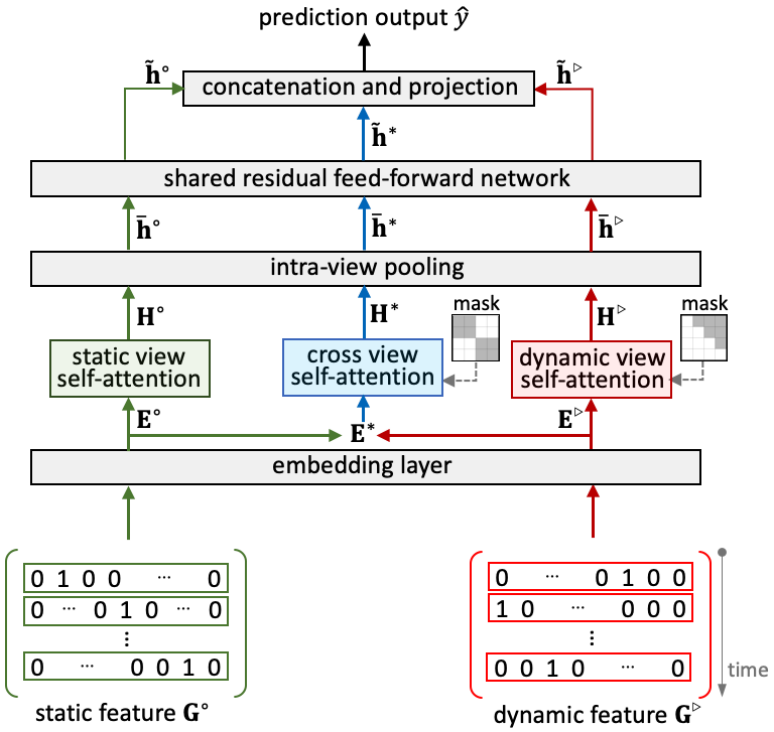


Figure 4.2: The overall architecture of SeqFM. We skip the linear term of SeqFM for better readability.

product, especially when the dimensionality is high. $\mathbf{Q}^{\circ}, \mathbf{K}^{\circ}, \mathbf{V}^{\circ} \in \mathbb{R}^{n^{\circ} \times d}$ respectively represent the queries, keys and values obtained using linear projection:

$$\begin{aligned} \mathbf{Q}^{\circ} &= \mathbf{E}^{\circ}\mathbf{W}_{Q}^{\circ}, \\ \mathbf{K}^{\circ} &= \mathbf{E}^{\circ}\mathbf{W}_{K}^{\circ}, \\ \mathbf{V}^{\circ} &= \mathbf{E}^{\circ}\mathbf{W}_{V}^{\circ}, \end{aligned} \tag{4.7}$$

and $\mathbf{W}_{Q}^{\circ}, \mathbf{W}_{K}^{\circ}, \mathbf{W}_{V}^{\circ} \in \mathbb{R}^{d \times d}$ are corresponding trainable projection weight matrices for queries, keys and values. To be concise, we reformulate the self-attention module in Eq.(4.6) and Eq.(4.7) as the following:

$$\mathbf{H}^{\circ} = \text{softmax}\Big(\frac{\mathbf{E}^{\circ}\mathbf{W}_{Q}^{\circ} \cdot (\mathbf{E}^{\circ}\mathbf{W}_{K}^{\circ})^{\top}}{\sqrt{d}}\Big) \cdot \mathbf{E}^{\circ}\mathbf{W}_{V}^{\circ}, \tag{4.8}$$

and each row $\mathbf{h}_{i}^{\circ} \in \mathbf{H}^{\circ}$ corresponds to the i-th feature. Intuitively, we have $\mathbf{h}_{i}^{\circ} = w_{i1}\mathbf{v}_{1}^{\circ} + w_{i2}\mathbf{v}_{2}^{\circ} + \cdots + w_{in^{\circ}}\mathbf{v}_{n^{\circ}}^{\circ}$ where $w_{i1}, w_{i2}, ..., w_{in^{\circ}}$ are self-attentive weights assigned to projected features $\mathbf{v}_{1}^{\circ}, \mathbf{v}_{2}^{\circ}, .., \mathbf{v}_{n^{\circ}}^{\circ} \in \mathbf{V}^{\circ}$. In fact, because the vector-wise interactions are encoded via the self-attentive weights from the dot product scheme with $SoftMax$ normalization, and the bit-wise interactions are encoded in an additive form of features, the self-attention is able to account for both bit-wise and vector-wise feature interactions between the i-th feature and all other features. Furthermore, being able to learn asymmetric interactions, the projection operation with three distinctive subspaces makes the model more flexible.

4.2.5 Dynamic View with Self-Attention

In the dynamic view, due to the nature of sequential dependencies among $n^{\triangleright}$ dynamic features, the i-th dynamic feature ($i \leq n^{\triangleright}$) will only have the interactive influence from the previous features at j ($j \leq i$).

For example, in the movie rating prediction task, we can only infer a user's rating to a new movie from her/his ratings to the movies this user has watched before. That is to say, the feature interactions in the dynamic view are ***directional***. Thus, to incorporate the directional property into the self-attention mechanism, we propose the following masked self-attention to model the feature interactions in the dynamic view:

$$\mathbf{H}^{\triangleright} = \text{softmax}\Big(\frac{\mathbf{E}^{\triangleright}\mathbf{W}_Q^{\triangleright} \cdot (\mathbf{E}^{\triangleright}\mathbf{W}_K^{\triangleright})^{\top}}{\sqrt{d}} + \mathbf{M}^{\triangleright}\Big) \cdot \mathbf{E}^{\triangleright}\mathbf{W}_V^{\triangleright}, \tag{4.9}$$

where $\mathbf{H}^{\triangleright} \in \mathbb{R}^{n^{\triangleright} \times d}$ carries the interaction contexts for all dynamic features, and $\mathbf{W}_Q^{\triangleright}, \mathbf{W}_K^{\triangleright}, \mathbf{W}_V^{\triangleright} \in \mathbb{R}^{d \times d}$. Compared with other sequential approaches like recurrent neural networks, self-attention enables vector-wise feature interactions and is more computationally efficient [23, 62]. Notably, $\mathbf{M}^{\triangleright} \in \{-\infty, 0\}^{n^{\triangleright} \times n^{\triangleright}}$ is a constant attention mask that allows each dynamic feature $\mathbf{e}_i^{\triangleright}$ to interact with $\mathbf{e}_j^{\triangleright}$ only if $j \leq i$. Specifically, in the mask $\mathbf{M}^{\triangleright}$, for its row and column indexes $i, j \leq n^{\triangleright}$, the value of each entry $m_{ij}^{\triangleright} \in \mathbf{M}^{\triangleright}$ is determined as:

$$m_{ij}^{\triangleright} = \begin{cases} 0, & \text{if } i \geq j \\ -\infty, & \text{otherwise} \end{cases}. \tag{4.10}$$

The Rationale of Attention Mask. We denote the matrix product of the query and key matrices in Eq.(4.9) as $\mathbf{A}$, i.e., $\mathbf{A} = \frac{\mathbf{E}^{\triangleright}\mathbf{W}_Q^{\triangleright} \cdot (\mathbf{E}^{\triangleright}\mathbf{W}_K^{\triangleright})^{\top}}{\sqrt{d}} \in \mathbb{R}^{n^{\triangleright} \times n^{\triangleright}}$. Similar to [62], in $\mathbf{A}$, each row $a_{i1}, a_{i2}, ..., a_{in^{\triangleright}}$ contains $n^{\triangleright}$ interaction scores between the i-th dynamic feature and all $n^{\triangleright}$ dynamic features. Then, for the i-th feature, $SoftMax$ is utilized to normalize these affinity scores to a probability distribution, i.e., $p_{i1}, p_{i2}, ..., p_{in^{\triangleright}} = \text{softmax}(a_{i1}, a_{i2}, ..., a_{in^{\triangleright}})$. By adding the attention mask $\mathbf{M}^{\triangleright}$, for the i-th feature, the interaction scores from $i+1$ become $-\infty$, while the earlier ones in the sequence remain unchanged. Consequently, with the $SoftMax$, $p_{ij} \neq 0$ for $j \leq i$ while $p_{ij} \approx 0$ for $j > i$, ensuring the interaction strength on the i-th feature only associates with historical features where $j \leq i$.

4.2.6 Cross View with Self-Attention

Because static and dynamic features possess varied semantics, in the cross view, we deploy the third attention head to model how static features interact with dynamic features. Similarly, we define another masked self-attention unit below:

$$\mathbf{H}^{\star} = \text{softmax}\Big(\frac{\mathbf{E}^{\star}\mathbf{W}_Q^{\star} \cdot (\mathbf{E}^{\star}\mathbf{W}_K^{\star})^{\top}}{\sqrt{d}} + \mathbf{M}^{\star}\Big) \cdot \mathbf{E}^{\star}\mathbf{W}_V^{\star}, \tag{4.11}$$

where $\mathbf{E}^{\star} \in \mathbb{R}^{(n^{\circ}+n^{\triangleright}) \times d}$ represents the cross view feature matrix constructed by vertically concatenating feature matrices from both static and dynamic views along the first dimension:

$$\mathbf{E}^{\star} = \begin{bmatrix} \mathbf{E}^{\circ} \\ \mathbf{E}^{\triangleright} \end{bmatrix}. \tag{4.12}$$

In Eq.(4.11), $\mathbf{H}^{\star} \in \mathbb{R}^{(n^{\circ}+n^{\triangleright}) \times d}$ stacks the interaction contexts for all $n^{\circ} + n^{\triangleright}$ features, and there are corresponding query, key and value projection matrices $\mathbf{W}_Q^{\star}, \mathbf{W}_K^{\star}, \mathbf{W}_V^{\star} \in \mathbb{R}^{d \times d}$. $\mathbf{M}^{\star} \in \{-\infty, 0\}^{(n^{\circ}+n^{\triangleright}) \times (n^{\circ}+n^{\triangleright})}$

is the attention mask devised for the cross view. Each entry $m^{\star}_{ij} \in \mathbf{M}^{\star}$ is formulated via:

$$m^{\star}_{ij} = \begin{cases} 0, & \text{if } i \leq n^{\circ} < j \text{ or } j \leq n^{\circ} < i \\ -\infty, & \text{otherwise} \end{cases}. \tag{4.13}$$

Following the explanation of the attention mask in Section 4.2.5, our cross view attention mask blocks possible feature interactions within the same category, and only allows cross-category feature interactions (i.e., interactions between static features and dynamic features). Intuitively, with this dedicated view, the model further extracts information from the mutual interactions between static properties and dynamic properties of features in a fine-grained manner.

4.2.7 Intra-View Pooling Operation

After calculating the representations for feature interactions in all three views, we feed these latent feature matrices into our proposed intra-view pooling layer, which compresses all latent vectors from each feature matrix into a unified vector representation. Specifically, for $\mathbf{h}^{\circ}_i \in \mathbf{H}^{\circ}$, $\mathbf{h}^{\triangleright}_i \in \mathbf{H}^{\triangleright}$ and $\mathbf{h}^{\star}_i \in \mathbf{H}^{\star}$, the intra-view pooling operation is defined as:

$$\overline{\mathbf{h}}^{view} = \frac{1}{n^{view}} \sum_{i=1}^{n^{view}} \mathbf{h}^{view}_i, \tag{4.14}$$

where $(\overline{\mathbf{h}}^{view}, \mathbf{h}^{view}_i, n^{view}) \in \{(\overline{\mathbf{h}}^{\circ}, \mathbf{h}^{\circ}_i, n^{\circ}), (\overline{\mathbf{h}}^{\triangleright}, \mathbf{h}^{\triangleright}_i, n^{\triangleright}), (\overline{\mathbf{h}}^{\star}, \mathbf{h}^{\star}_i, n^{\circ} + n^{\triangleright})\}$, and we use $\overline{\mathbf{h}}^{\circ}$, $\overline{\mathbf{h}}^{\triangleright}$ and $\overline{\mathbf{h}}^{\star}$ to denote the final vector representations after the pooling operation for static view, dynamic view and cross view, respectively. Compared with the standard self-attention encoder structure in [62], the intra-view pooling operation does not introduce additional model parameters. Moreover, the intra-view pooling operation compactly encodes the information of pairwise feature interactions in the static, dynamic and cross views.

4.2.8 Shared Residual Feed-Forward Network

With the multi-view self-attention and the intra-view pooling, all feature interactions are aggregated with adaptive weights. However, it is still a linear computation process. To further model the complex, non-linear interactions between different latent dimensions, we stack a shared l-layer residual feed-forward network upon the intra-view pooling layer:

$$\begin{aligned} \widetilde{\mathbf{h}}^{view}_{(1)} &= \overline{\mathbf{h}}^{view} + \text{ReLU}(\text{LN}(\overline{\mathbf{h}}^{view})\mathbf{W}_1 + \mathbf{b}_1), \\ \widetilde{\mathbf{h}}^{view}_{(2)} &= \widetilde{\mathbf{h}}^{view}_{(1)} + \text{ReLU}(\text{LN}(\widetilde{\mathbf{h}}^{view}_{(1)})\mathbf{W}_2 + \mathbf{b}_2), \\ &\cdots \\ \widetilde{\mathbf{h}}^{view}_{(l)} &= \widetilde{\mathbf{h}}^{view}_{(l-1)} + \text{ReLU}(\text{LN}(\widetilde{\mathbf{h}}^{view}_{(l-1)})\mathbf{W}_l + \mathbf{b}_l), \end{aligned} \tag{4.15}$$

where $(\overline{\mathbf{h}}^{view}, \widetilde{\mathbf{h}}^{view}) \in \{(\overline{\mathbf{h}}^{\circ}, \widetilde{\mathbf{h}}^{\circ}), (\overline{\mathbf{h}}^{\triangleright}, \widetilde{\mathbf{h}}^{\triangleright}), (\overline{\mathbf{h}}^{\star}, \widetilde{\mathbf{h}}^{\star})\}$, ReLU is the rectified linear unit for non-linear activation, while $\mathbf{W} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^{1 \times d}$ are weight and bias in each layer. Note that though the

network parameters are different from layer to layer, the three views share the same feed-forward network, as shown in Figure 4.2. In the following, we introduce the three key components in the shared residual feed-forward network.

Residual Connections. The core idea behind residual networks is to propagate low-layer features to higher layers by residual connection [105]. By combining low-layer interaction features with the high-layer representations computed by the feed-forward network, the residual connections essentially allow the model to easily propagate low-layer features to the final layer, which can help the model enhance its expressive capability using different information learned hierarchically. Intuitively, in our shared residual feed-forward network, to generate a comprehensive representation for feature interactions in each view, the l-th layer iteratively fine-tunes the representation learned by the $(l-1)$-th layer (i.e., $\widetilde{\mathbf{h}}^{view}_{(l-1)}$) by adding a learned residual, which corresponds to the second term in Eq.(4.15).

Layer Normalization. In Eq.(4.15), $\mathrm{LN}(\cdot)$ denotes the layer normalization function [106], which is beneficial for stabilizing and accelerating neural network training process by normalizing the layer inputs across features. Unlike batch normalization [107], in layer normalization, each sample from a batch uses independent statistics [23], and the computation at training and test times follows the same process. Specifically, for an arbitrary layer input $\widetilde{\mathbf{h}}^{view}_{(l')}$, $\mathrm{LN}(\widetilde{\mathbf{h}}^{view}_{(l')})$ is calculated as:

$$\mathrm{LN}(\widetilde{\mathbf{h}}^{view}_{(l')}) = \mathbf{s} \odot \frac{\widetilde{\mathbf{h}}^{view}_{(l')} - \mu}{\varepsilon} + \mathbf{b}, \tag{4.16}$$

where $l' \leq l$ and $\widetilde{\mathbf{h}}^{view}_{(l')} \in \{\widetilde{\mathbf{h}}^{\circ}_{(l')}, \widetilde{\mathbf{h}}^{\triangleright}_{(l')}, \widetilde{\mathbf{h}}^{\star}_{(l')}\}$. Also, $\odot$ is the element-wise product, μ and ε are respectively the mean and variance of all elements in $\widetilde{\mathbf{h}}^{view}_{(l')}$. Note that a small bias term will be added to ε in case $\varepsilon = 0$. The scaling weight $\mathbf{s} \in \mathbb{R}^{1\times d}$ and the bias term $\mathbf{b} \in \mathbb{R}^{1\times d}$ are parameters to be learned which help restore the representation power of the network.

Layer Dropout. To prevent SeqFM from overfitting the training data, we adopt dropout [108] on all the layers of our shared residual feed-forward network as a regularization strategy. In short, we randomly drop the neurons with the ratio of $\rho \in (0,1)$ during training. Hence, dropout can be viewed as a form of ensemble learning which includes numerous models that share parameters [109]. It is worth mentioning that all the neurons are used when testing, which can be seen as a model averaging operation [108] in ensemble learning.

4.2.9 View-Wise Aggregation

With the $\widetilde{\mathbf{h}}^{\circ}_{(l)}$, $\widetilde{\mathbf{h}}^{\triangleright}_{(l)}$ and $\widetilde{\mathbf{h}}^{\star}_{(l)}$ calculated by the l-layer shared residual feed-forward network, we perform view-wise aggregation to combine all the information from different types of feature interactions. The final representation is generated by horizontally concatenating the latent representations from three views:

$$\mathbf{h}^{agg} = [\widetilde{\mathbf{h}}^{\circ}_{(l)}; \widetilde{\mathbf{h}}^{\triangleright}_{(l)}; \widetilde{\mathbf{h}}^{\star}_{(l)}], \tag{4.17}$$

where $\mathbf{h}^{agg} \in \mathbb{R}^{1\times 3d}$ denotes the aggregated representation of non-linear, high-order feature interactions within SeqFM. Since the representations learned by the shared residual feed-forward network are

sufficiently expressive with an appropriate network depth l, we do not apply extra learnable weights to the view-wise aggregation scheme.

4.2.10 Output Layer

After the aggregation of the latent representations from the static, dynamic and cross views, the final vector representation $\mathbf{h}^{agg}$ is utilized to compute the scalar output for the multi-view self-attentive factorization component via vector dot product:

$$f(\mathbf{G}^{\circ}, \mathbf{G}^{\triangleright}) = \langle \mathbf{p}, \mathbf{h}^{agg} \rangle, \tag{4.18}$$

where $\mathbf{p} \in \mathbb{R}^{1 \times 3d}$ is the projection weight vector. At last, we summarize the entire prediction result of SeqFM as:

$$\widehat{y} = w_0 + [(\mathbf{G}^{\circ}\mathbf{w}^{\circ})^{\top}; (\mathbf{G}^{\triangleright}\mathbf{w}^{\triangleright})^{\top}]\mathbf{1} + \langle \mathbf{p}, \mathbf{h}^{agg} \rangle. \tag{4.19}$$

As the scopes of both the input and output are not restricted, SeqFM is a flexible and versatile model which can be adopted for different tasks. In Section 4.3, we will introduce how SeqFM is applied to ranking, classification, and regression tasks as well as the optimization strategy of SeqFM.

4.2.11 Time Complexity Analysis

Excluding the embedding operation that is standard in all FM-based models, the computational cost of our model is mainly exerted by the self-attention units and the feed-forward network. As the three self-attention units are deployed in parallel, we only consider the cross view attention head that takes the most time to compute. Hence, for each training sample, the overall time complexity of these two components is $O((n^{\circ} + n^{\triangleright})^2 d) + O(ld^2) = O((n^{\circ} + n^{\triangleright})^2 d + ld^2)$. Because l is typically small, the dominating part is $O((n^{\circ} + n^{\triangleright})^2 d)$. As n° is constant in the static view and $n^{\triangleright}$ is fixed with a threshold, SeqFM has linear time complexity w.r.t. the scale of the data.

4.3 Applications and Optimization of SeqFM

We hereby apply SeqFM to three different temporal predictive analytic settings, involving ranking, classification, and regression tasks. We also describe our optimization strategy.

4.3.1 SeqFM for Ranking

We deploy SeqFM for next-POI (point-of-interest) recommendation, which is commonly formulated as a ranking task [91, 110]. For each user, next-POI recommendation aims to predict a personalized ranking on a set of POIs and return the top-K POIs according to the predicted ranking. This is accomplished by estimating a ranking score for each given user-POI pair $(user, POI)$. For this ranking

task, the input of SeqFM is formulated as follows:

$$\mathbf{G}^{\circ} = \begin{bmatrix} \mathbf{g}_1^{\circ} \\ \mathbf{g}_2^{\circ} \\ \vdots \\ \mathbf{g}_{n^{\circ}}^{\circ} \end{bmatrix} \begin{matrix} \rightarrow \text{user one-hot} \\ \rightarrow \text{candidate POI one-hot} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{other static features} \end{matrix}, \mathbf{G}^{\triangleright} = \begin{bmatrix} \mathbf{g}_1^{\triangleright} \\ \mathbf{g}_2^{\triangleright} \\ \vdots \\ \mathbf{g}_{n^{\triangleright}}^{\triangleright} \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{one-hot sequence of visited POIs}. \tag{4.20}$$

Note that other static features include the user/POI's side information (e.g., occupation, gender, etc.) and are optional subject to availability. We denote the $(user, POI)$ pair as (u, v) to be concise. For each user u, we denote an observed user-POI interaction as a positive pair (u, v^+). Correspondingly, a corrupted user-POI pair (u, v^-) can be constructed, where v^- is a POI that user u has never visited. Thus, a training sample is defined as a triple $(u_i, v_j^+, v_k^-) \in \mathscr{S}$, and $\mathscr{S}$ denotes the set of all training samples. Following [19], we leverage the Bayesian Personalized Ranking (BPR) loss to optimize SeqFM for the ranking task:

$$\begin{aligned} \mathscr{L} &= -\log \prod_{(u_i, v_j^+, v_k^-) \in \mathscr{S}} \sigma(\widehat{y}_{ij} - \widehat{y}_{ik}) \\ &= -\sum_{(u_i, v_j^+, v_k^-) \in \mathscr{S}} \log \Big(\sigma(\widehat{y}_{ij} - \widehat{y}_{ik}) \Big), \end{aligned} \tag{4.21}$$

where $\sigma(\cdot)$ is the *Sigmoid* function. We omit the regularization term for model parameters as the layer dropout scheme is already capable of preventing our model from overfitting. For each user u_i, $\widehat{y}_{ij}$ and $\widehat{y}_{ik}$ respectively denote the ranking score for item v_j^+ and item v_k^-. The rationale of the BPR loss is that, the ranking score for a POI visited by the user should always be higher than the ranking score for an unvisited one.

4.3.2 SeqFM for Classification

For classification task, we conduct click-through rate (CTR) prediction, which is also one of the most popular applications for FM-based models [70, 92, 95, 100, 101]. Given an arbitrary user and her/his previously visited links (e.g., web pages or advertisements), the target of CTR prediction is to predict whether this user will click through a given link or not. We formulate the input of SeqFM for this classification task as:

$$\mathbf{G}^{\circ} = \begin{bmatrix} \mathbf{g}_1^{\circ} \\ \mathbf{g}_2^{\circ} \\ \vdots \\ \mathbf{g}_{n^{\circ}}^{\circ} \end{bmatrix} \begin{matrix} \rightarrow \text{user one-hot} \\ \rightarrow \text{candidate link one-hot} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{other static features} \end{matrix}, \mathbf{G}^{\triangleright} = \begin{bmatrix} \mathbf{g}_1^{\triangleright} \\ \mathbf{g}_2^{\triangleright} \\ \vdots \\ \mathbf{g}_{n^{\triangleright}}^{\triangleright} \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \text{one-hot sequence of clicked links}. \tag{4.22}$$

To enable the capability of classification, a *Sigmoid* operation is added to the output layer. To keep the notations clear, we re-formulate the $\widehat{y}$ in Eq.(4.19) as:

$$\widehat{y} = \sigma(w_0 + [(\mathbf{G}^{\circ}\mathbf{w}^{\circ})^{\top}; (\mathbf{G}^{\triangleright}\mathbf{w}^{\triangleright})^{\top}]\mathbf{1} + \langle \mathbf{p}, \mathbf{h}^{agg} \rangle), \tag{4.23}$$

where $\sigma(\cdot)$ denotes the *Sigmoid* function. Here, the $\widehat{y} \in (0,1)$ can be viewed as the possibility of observing a $(user, link)$ instance. By replacing $(user, link)$ with the notion (u,v), we quantify the prediction error with log loss, which is a special case of the cross-entropy:

$$\begin{aligned}\mathscr{L} &= -\sum_{(u_i,v_j^+)\in\mathscr{S}^+} \log \widehat{y}_{ij} - \sum_{(u_i,v_j^-)\in\mathscr{S}^-} \log(1-\widehat{y}_{ij}) \\ &= -\sum_{(u_i,v_j)\in\mathscr{S}} \Big(y_{ij} \log \widehat{y}_{ij} + (1-y_{ij}) \log(1-\widehat{y}_{ij}) \Big),\end{aligned} \tag{4.24}$$

where $\mathscr{S} = \mathscr{S}^+ \cup \mathscr{S}^-$ is the set of labeled (u,v) pairs. Since we only have positive labels of observed interactions denoted by $(u,v^+) \in \mathscr{S}^+$, we uniformly sample negative labels $(u,v^-) \in \mathscr{S}^-$ from the unobserved interactions during training and control the number of negative samples w.r.t. the size of the positive ones.

4.3.3 SeqFM for Regression

Finally, we apply SeqFM to a regression task, namely rating prediction which is useful for mining users' preferences and personalities [21,76]. We use the same problem setting as [20,21], that is, given a user and her/his rated items, we estimate this user's rating to a new target item. SeqFM takes the following as its input:

$$\mathbf{G}^{\circ} = \begin{bmatrix} \mathbf{g}_1^{\circ} \\ \mathbf{g}_2^{\circ} \\ \vdots \\ \mathbf{g}_{n^{\circ}}^{\circ} \end{bmatrix} \begin{matrix} \rightarrow \text{user one-hot} \\ \rightarrow \text{target item one-hot} \\ \left.\begin{matrix} \\ \\ \end{matrix}\right\} \text{other static features} \end{matrix}, \mathbf{G}^{\triangleright} = \begin{bmatrix} \mathbf{g}_1^{\triangleright} \\ \mathbf{g}_2^{\triangleright} \\ \vdots \\ \mathbf{g}_{n^{\triangleright}}^{\triangleright} \end{bmatrix} \left.\begin{matrix} \\ \\ \\ \\ \end{matrix}\right\} \begin{matrix} \text{one-hot sequence} \\ \text{of rated items} \end{matrix}. \tag{4.25}$$

We denote each $(user, item)$ pair as (u,v). For each (u_i, v_j), the emitted output $\widehat{y}_{ij}$ is a continuous variable that tries to match up with the ground truth rating y_{ij}. Thus, we can directly apply the squared error loss below:

$$\mathscr{L} = \sum_{(u_i,v_j)\in\mathscr{S}} (\widehat{y}_{ij} - y_{ij})^2, \tag{4.26}$$

where S denotes the training set. Note that sampling negative training cases is unnecessary in the conventional rating prediction task.

4.3.4 Optimization Strategy

As SeqFM is built upon the deep neural network structure, we can efficiently apply Stochastic Gradient Decent (SGD) algorithms to learn the model parameters by minimizing each task-specific loss $\mathscr{L}$. Hence, we leverage a mini-batch SGD-based algorithm, namely Adam [63] optimizer. For different tasks, we tune the hyperparameters using grid search. Specifically, the latent dimension (i.e., factorization factor) d is searched in $\{8,16,32,64,128\}$; the depth of the shared residual feed-forward network l is searched in $\{1,2,3,4,5\}$; the maximum sequence length $n^{\triangleright}$ is searched in $\{10,20,30,40,50\}$; and the dropout ratio ρ is searched in $\{0.5,0.6,0.7,0.8,0.9\}$. We will further

discuss the impact of these key hyperparameters on the prediction performance of SeqFM in Section 4.4. For ranking and classification tasks, we draw 5 negative samples for each positive label during training. In addition, we set the batch size to 512 according to device capacity and the learning rate to 1×10^{-4}. We iterate the whole training process until $\mathscr{L}$ converges.

4.4 Experiments

In this section, we outline the evaluation protocols for our proposed SeqFM, and then perform experiments to evaluate SeqFM in various temporal prediction tasks. In particular, we aim to answer the following research questions (RQs) via experiments:

RQ1: How effectively can SeqFM perform temporal predictive analytics compared with state-of-the-art FM-based models?

RQ2: How do the hyperparameters affect the performance of SeqFM in different prediction tasks?

RQ3: How SeqFM benefits from each component of the proposed model structure?

RQ4: How is the training efficiency and scalability of SeqFM when handling large-scale data?

4.4.1 Datasets

To validate the performance of SeqFM in terms of ranking, classification, and regression, for each task we consider two real-world datasets, whose properties are introduced below.

- **Gowalla (Ranking):** This is a global POI check-in dataset[1] collected from February 2009 to October 2010. Each user's visited POIs are recorded with a timestamp.
- **Foursquare (Ranking):** This POI check-in dataset[2] is generated world-wide from April 2012 to September 2013, containing users' visited POIs at different times.
- **Trivago (Classification):** This dataset is obtained from the ACM RecSys Challenge[3] in 2019. It is a web search dataset consisting of users' visiting (e.g., clicking) logs on different webpage links.
- **Taobao (Classification):** It is a subset of user shopping log data released by Alibaba[4]. We extract and sort users' clicking behavior on product links chronologically.
- **Beauty (Regression):** A series of users' product ratings[5] are crawled from Amazon from May 1996 to July 2014, and different product categories are treated as separate datasets. Beauty is one of the largest categories.

[1] https://snap.stanford.edu/data/loc-gowalla.html
[2] https://sites.google.com/site/yangdingqi/home/foursquare-dataset
[3] http://www.recsyschallenge.com/2019/
[4] https://tianchi.aliyun.com/
[5] http://snap.stanford.edu/data/amazon/productGraph/

Table 4.1: Statistics of datasets in use.

Task	Dataset	#Instance	#User	#Object	#Feature (Sparse)
Ranking	Gowalla	1,865,119	34,796	57,445	149,686
	Foursquare	1,196,248	24,941	28,593	82,127
Classification	Trivago	2,810,584	12,790	45,195	103,180
	Taobao	1,970,133	37,398	65,474	168,346
Regression	Beauty	198,503	22,363	12,101	46,565
	Toys	167,597	19,412	11,924	50,748

- **Toys (Regression):** This is another Amazon user rating dataset on toys and games.

All datasets used in our experiment are in large scale and publicly available. The primary statistics are shown in Table 4.1, where we use the word "object" to denote the POI, link, and item in different applications. Following [19, 72, 91, 111], we filter out inactive users with less than 10 interacted objects and unpopular objects visited by less than 10 users. Note that for Beauty and Toys, we directly use the provided versions without further preprocessing.

4.4.2 Baseline Methods

We briefly introduce the baseline methods for comparison below. First of all, we choose the latest and popular FM-based models as the common baselines for all ranking, classification, and regression tasks. Then, for each task, we further select two state-of-the-art methods originally proposed for the specific task scenario as an additional competitor.

- **FM:** This is the original Factorization Machine [20] with proven effectiveness in many prediction tasks [21, 102].
- **Wide&Deep:** Proposed by the Google team [101], the Wide&Deep model uses a DNN to learn latent representations of concatenated features.
- **DeepCross:** It stacks multiple residual network blocks upon the concatenation layer for feature embeddings in order to learn deep cross features [95].
- **NFM:** The Neural Factorization Machine [69] encodes all feature interactions via multi-layer DNNs coupled with a bit-wise bi-interaction pooling layer.
- **AFM:** The Attentional Factorization Machine [71] introduces an attention network to distinguish the importance of different pairwise feature interactions.
- **SASRec (Ranking):** This is the Self-Attention-based Sequential Recommendation Model [23] with long-term and short-term context modelling.
- **TFM (Ranking):** The Translation-based Factorization Machine [104] learns an embedding and translation space for each feature dimension, and adopts Euclidean distance to quantify the strength of pairwise feature interactions.

- **DIN (Classification):** The Deep Interest Network [27] can represent users' diverse interests with an attentive activation mechanism for CTR prediction.

- **xDeepFM (Classification):** It stands for the Extreme Deep Factorization Machine [70], which has a compressed interaction network to model vector-wise feature interactions to perform CTR prediction.

- **RRN (Regression):** The Recurrent Recommender Network [76] is a deep autoregressive model for temporal rating prediction.

- **HOFM (Regression):** This is the Higher-Order Factorization Machine described in [112]. HOFM improves [20] with space-saving and time-efficient kernels to allow shared parameters for prediction tasks.

4.4.3 Evaluation Metrics

To fit the scenario of temporal predictive analytics, we adopt the ***leave-one-out*** evaluation protocol which is widely used in the literature [19, 27, 88, 104]. Specifically, within each user's transaction, we hold out her/his last record as the ground truth for test and the second last record for validation. All the rest records are used to train the models. Set-category features are used as input for all FM-based baseline models.

Evaluating Ranking Performance. To evaluate the ranking performance, we adopt the well-established Hits Ratio at Rank K (HR@K) and Normalized Discounted Cumulative Gain at Rank K (NDCG@K) which are commonly used in information retrieval and recommender systems [91, 110]. Specifically, for each positive test instance $(user, POI) \in \mathscr{S}^{test}$, we mix the POI with J random POIs that are never visited by the user. Afterwards, we rank all these $J+1$ POIs for the user. Then, we use HR@K to measure the ratio that the ground truth item has a hit (i.e., is present) on the top-K list, and use NDCG@K to further evaluate whether if the model can rank the ground truth as highly as possible:

$$\text{HR@}K = \frac{\#hit@K}{|\mathscr{S}^{test}|}, \text{NDCG@}K = \frac{\sum_{s \in \mathscr{S}^{test}} \sum_{r=1}^{K} \frac{rel_{s,r}}{\log_2(r+1)}}{|\mathscr{S}^{test}|}, \tag{4.27}$$

where $\#hit@K$ is the number of hits in the test set. For each test case $s \in \mathscr{S}^{test}$, $rel_{s,r} = 1$ if the item ranked at r is the ground truth, otherwise $rel_{s,r} = 0$. We set $J = 1,000$ to balance the running time and task difficulty. For K, we adopt the popular setting of $5, 10, 20$ for presentation.

Evaluating Classification Performance. We adopt two evaluation metrics for the classification task, namely Area under the ROC Curve (AUC) [95,96] and Root Mean Squared Error (RMSE) [69,71]. For each positive test instance $(user, link) \in \mathscr{S}^{test}$, we draw a random negative link that the user has never clicked, and predict the interaction possibility for both links. AUC measures the probability that a positive instance will be ranked higher than the negative one. It only takes into account the order of predicted instances and is insensitive to class imbalance problem. In contrast, RMSE evaluates the distance between the predicted possibility and the true label for each instance.

Table 4.2: Ranking task (next-POI recommendation) results. Numbers in bold face are the best results for corresponding metrics. For both HR@K and NDCG@K, the higher the better.

Method	Gowalla						Foursquare					
	HR@K			NDCG@K			HR@K			NDCG@K		
	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20	K=5	K=10	K=20
FM [20]	0.232	0.318	0.419	0.158	0.187	0.211	0.241	0.303	0.433	0.169	0.201	0.217
Wide&Deep [101]	0.288	0.401	0.532	0.199	0.238	0.267	0.233	0.317	0.422	0.165	0.192	0.218
DeepCross [95]	0.273	0.379	0.505	0.182	0.204	0.241	0.282	0.355	0.492	0.198	0.210	0.229
NFM [69]	0.286	0.395	0.525	0.199	0.236	0.264	0.239	0.325	0.435	0.170	0.198	0.225
AFM [71]	0.295	0.407	0.534	0.204	0.242	0.270	0.279	0.379	0.504	0.199	0.212	0.233
SASRec [23]	0.310	0.424	0.559	0.209	0.253	0.285	0.266	0.350	0.467	0.175	0.204	0.216
TFM [104]	0.307	0.430	0.556	0.216	0.256	0.283	0.283	0.390	0.512	0.203	0.223	0.248
SeqFM	**0.345**	**0.467**	**0.603**	**0.243**	**0.283**	**0.316**	**0.324**	**0.431**	**0.554**	**0.227**	**0.262**	**0.293**

Evaluating Regression Performance. We evaluate the regression performance with Mean Absolute Error (MAE) and Root Relative Squared Error (RRSE), which are popular among relevant research communities [1, 37]. Mathematically, they are defined as follows:

$$\text{MAE} = \frac{\sum_{y \in \mathscr{S}^{test}} |\widehat{y} - y|}{|\mathscr{S}^{test}|}, \text{RRSE} = \frac{\sqrt{\frac{\sum_{y \in |\mathscr{S}^{test}|} (\widehat{y} - y)^2}{|\mathscr{S}^{test}|}}}{VAR_{\mathscr{S}^{test}}}, \tag{4.28}$$

where $\widehat{y}$ and y denote the predicted and real value respectively, and $VAR_{\mathscr{S}^{test}}$ is the variance of all ground truth values. MAE directly reveals the gap between prediction and ground truth, while RRSE is the normalized root mean square error and is independent of the data scale and distribution.

4.4.4 Parameter Settings

To be consistent, we report the overall performance of SeqFM on all tasks with a unified parameter set $\{d = 64, l = 1, n^{\triangleright} = 20, \rho = 0.6\}$. Detailed discussions on the effects of different parameter settings will be shown in Section 4.4.6. For all baseline methods, since all tasks are conducted on standard and generic datasets with common evaluation metrics, we adopt the optimal parameters in their original works.

4.4.5 Prediction Performance (RQ1)

We summarize the performance of all models in terms of ranking, classification, and regression with Table 4.2, 4.3, and 4.4 respectively. We discuss our findings regarding the effectiveness results as follows.

Ranking Performance. The results of the ranking task (next-POI recommendation) are reported in Table 4.2. Note that higher HR@K and NDCG@K values imply better prediction performance. Obviously, on both Gowalla and Foursquare, SeqFM significantly and consistently outperforms all existing FM-based models with $K \in \{5, 10, 20\}$. In particular, the advantages of SeqFM in terms of HR@5 and NDCG@5 imply that our model can accurately rank the ground truth POI in the top-5

Table 4.3: Classification task (CTR prediction) results. Numbers in bold face are the best results for corresponding metrics. For AUC, the higher the better; while for RMSE, the lower the better.

Method	Trivago		Taobao	
	AUC	RMSE	AUC	RMSE
FM [20]	0.729	0.564	0.602	0.597
Wide&Deep [101]	0.782	0.529	0.629	0.590
DeepCross [95]	0.845	0.433	0.735	0.391
NFM [69]	0.767	0.537	0.616	0.583
AFM [71]	0.811	0.465	0.656	0.544
DIN [27]	0.923	0.338	0.781	0.375
xDeepFM [70]	0.913	0.350	0.804	0.363
SeqFM	**0.957**	**0.319**	**0.826**	**0.335**

positions, which can better suit each user's intent and boost the recommendation success rate. Though SASRec shows promising effectiveness on Gowalla, it underperforms when facing higher data sparsity on Foursquare. Another observation is that all FM-based models with deep neural networks (i.e., Wide&Deep, DeepCross, NFM and AFM) outperform the plain FM. As a model specifically designed for sequential recommendation, TFM naturally performs better than the common baselines on both POI check-in datasets. However, SeqFM still achieves higher ranking effectiveness. This is because TFM is designed to only consider the most recently visited object (POI) in the dynamic feature sequence, while SeqFM utilizes the self-attention mechanism to extract richer information from the entire sequence.

Classification Performance. We list all the results of the classification task (CTR prediction) in Table 4.3. A better result corresponds to a higher AUC score and a lower RMSE value. At the first glance, it is clear that our SeqFM achieves the highest classification accuracy on both Trivago and Taobao. Similar to the observations from the ranking task, existing variants of the plain FM show the benefit of adopting deep neural networks. As for the task-specific models for CTR prediction, the attentive activation unit in DIN can selectively determine the weights of different features based on a given link, while xDeepFM is able to thoroughly model the high-order interactions among different features with its dedicated interaction network. However, there is a noticeable performance gap between both additional baselines and our proposed SeqFM. This proves the insight of our work, which points out that instead of simply treating all dynamic features as flat set-category features in existing FM-based models, the sequence-aware interaction scheme for dynamic features in SeqFM is more helpful for temporal predictive analytics.

Regression Performance. Table 4.4 reveals all models' performance achieved in the regression task (rating prediction) on Beauty and Toys. For both MAE and RRSE metrics, the lower the better. As demonstrated by the results, despite the intense competition in the regression task, SeqFM yields significant improvements on the regression accuracy over all the baselines. Furthermore, though showing competitive regression results, the additional baseline HOFM is still limited by its linear mathematical form, so approaches based on deep neural networks like RRN, NFM and AFM perform slightly better owing to their non-linear expressiveness. Apart from that, we notice that compared with the performance achieved by the plain FM, other FM-based approaches only shows marginal

Table 4.4: Regression task (rating prediction) results. Numbers in bold face are the best results for corresponding metrics. For both MAE and RRSE, the lower the better.

Method	Beauty		Toys	
	MAE	RRSE	MAE	RRSE
FM [20]	1.067	1.125	0.778	1.023
Wide&Deep [101]	0.965	1.090	0.753	0.989
DeepCross [95]	0.949	1.003	0.761	1.010
NFM [69]	0.931	0.986	0.735	0.981
AFM [71]	0.945	0.994	0.741	0.997
RRN [76]	0.943	0.989	0.739	0.983
HOFM [112]	0.952	1.054	0.748	1.001
SeqFM	**0.890**	**0.975**	**0.704**	**0.956**

advantages against it in the regression task. In contrast, with 13% and 7% relative improvements on RRSE over the plain FM, our proposed SeqFM highlights the importance of fully utilizing the sequential dependencies for predictive analytics.

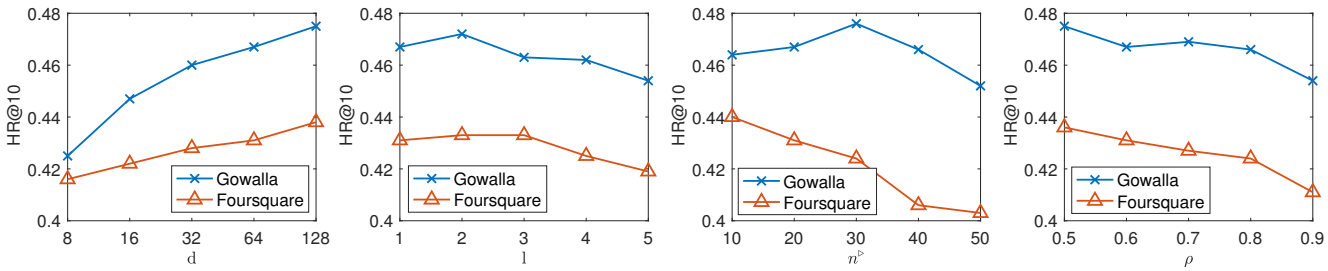
To summarize, the promising effectiveness of SeqFM is thoroughly demonstrated in ranking, classification, and regression tasks. In the comparison with state-of-the-art baselines on a wide range of datasets, the considerable improvements from our model further imply that SeqFM is a general and versatile model that suits different types of temporal prediction tasks.

4.4.6 Impact of Hyperparameters (RQ2)

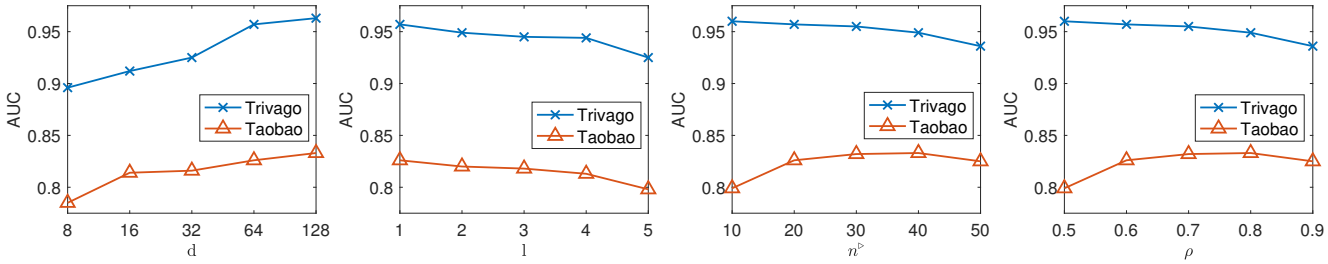
We answer the second research question by investigating the performance fluctuations of SeqFM with varied hyperparameters. Particularly, as mentioned in Section 4.3.4, we study our model's sensitivity to the latent dimension d, the depth of residual feed-forward network l, the maximum sequence length $n^{\triangleright}$, as well as the dropout ratio ρ. For each test, based on the standard setting $\{d = 64, l = 1, n^{\triangleright} = 20, \rho = 0.6\}$, we vary the value of one hyperparameter while keeping the others unchanged, and record the new prediction result achieved. To show the performance differences, we demonstrate HR@10 for ranking, AUC for classification, and MAE for regression. Figure 4.3 lays out the results with different parameter settings.

Impact of d. The value of the latent dimension d is examined in $\{8, 16, 32, 64, 128\}$. As an important hyperparameter in deep neural networks, the latent dimension is apparently associated with the model's expressiveness. In general, SeqFM benefits from a relatively larger d for all types of tasks, but the performance improvement tends to become less significant when d reaches a certain scale (32 and 64 in our case). It is worth mentioning that with $d = 16$, SeqFM still outperforms nearly all the baselines in the temporal prediction tasks, which further proves the effectiveness of our proposed model.

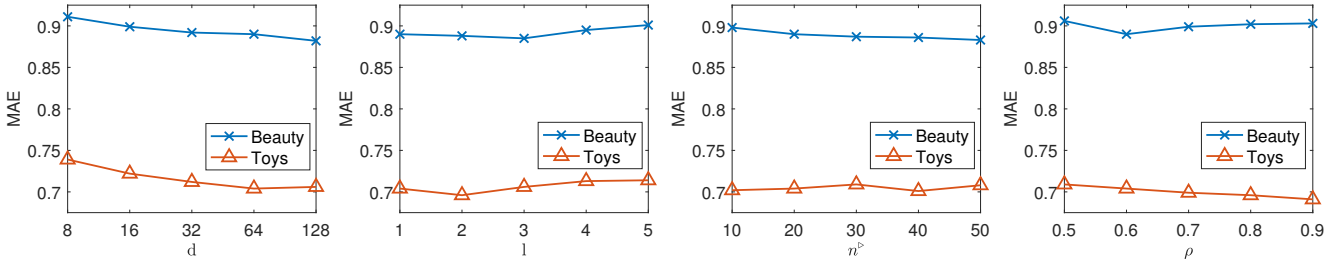
Impact of l. We study the impact of the depth of our shared residual feed-forward network with $l \in \{1, 2, 3, 4, 5\}$. For regression task, there is a slight performance growth for SeqFM as l in creases. Though stacking more deep layers in the neural network may help the model yield better performance



(a) Above: ranking performance w.r.t. d, l, $n^{\triangleright}$ and ρ.



(b) Above: classification performance w.r.t. d, l, $n^{\triangleright}$ and ρ.



(c) Above: regression performance w.r.t. d, l, $n^{\triangleright}$ and ρ.

Figure 4.3: Parameter sensitivity analysis.

in some specific applications, for both ranking and classification tasks, SeqFM generally achieves higher prediction results with a smaller l. This is because deeper networks bring excessive parameters that can lead to overfitting, and the information learned by deeper layers may introduce noise to the model.

Impact of $n^{\triangleright}$. As can be concluded from Figure 4.3, SeqFM behaves differently on varied datasets when the maximum sequence length $n^{\triangleright}$ is adjusted in $\{10, 20, 30, 40, 50\}$. This is due to the characteristics of sequential dependencies in different datasets. For instance, in Gowalla and Foursquare, users tend to choose the next POI close to their current check-in location (i.e., the previous POI), thus forming sequential dependencies in short lengths. As a result, a larger $n^{\triangleright}$ will take more irrelevant POIs as the input, and eventually causes the performance decrease. In contrast, in Taobao, users' clicking behavior is usually motivated by their intrinsic long-term preferences, so a relatively larger $n^{\triangleright}$ can help the model achieve better results in such scenarios.

Impact of ρ. The impact of different dropout ratios is investigated via $\rho \in \{0.5, 0.6, 0.7, 0.8, 0.9\}$. Overall, the best prediction performance of SeqFM is reached when ρ is between 0.6 and 0.8. From Figure 4.3 we can draw the observation that a lower dropout ratio is normally useful for preserving the model's ability to generalize to unseen test data (e.g., Foursquare and Trivago). However, on some datasets, a smaller ρ comes with lower performance (e.g., Taobao and Beauty) because too many blocked neurons may result in underfitting during training.

Table 4.5: Ablation test with different model architectures. Numbers in bold face are the best results for corresponding metrics, and "↓" marks a severe (over 5%) performance drop.

Architecture	HR@10		AUC		MAE	
	Gowalla	Foursquare	Trivago	Taobao	Beauty	Toys
Default	**0.467**	**0.431**	**0.957**	**0.826**	**0.890**	**0.704**
Remove SV	0.455	0.420	0.892↓	0.765↓	0.959↓	0.762↓
Remove DV	0.424↓	0.396↓	0.862↓	0.731↓	0.972↓	0.772↓
Remove CV	0.430↓	0.404↓	0.963	0.754↓	0.935↓	0.763↓
Remove RC	0.457	**0.431**	0.898↓	0.761↓	0.918	0.719
Remove LN	0.461	0.423	0.933	0.798	0.922	0.720

4.4.7 Importance of Key Components (RQ3)

To better understand the performance gain from the major components proposed in SeqFM, we conduct ablation test on different degraded versions of SeqFM. Each variant removes one key component from the model, and the corresponding results on three tasks are reported. Table 4.5 summarizes prediction outcomes in different tasks. Similar to Section 4.4.6, HR@10, AUC and MAE are used. In what follows, we introduce the variants and analyze their effect respectively.

Remove Static View (Remove SV). The attention head in the static view models the interactions among all the static features. After removing it, a noticeable performance drop has been observed, especially on classification and regression tasks. In our application of SeqFM, the static view directly models interaction between the user and the target object (i.e., POI, link, and item), which is rather important especially when the task relies on mining users' personal preferences (e.g., the rating prediction task).

Remove Dynamic View (Remove DV). The modelling of the sequential interactions among dynamic features is crucial to the model's performance in temporal predictive analytics. Hence, a significant (over 5%) performance decrease has appeared in all three tasks. The results verify that the sequence-awareness plays a pivotal role when prediction tasks involve dynamic features. Specifically, the most severe performance drop is exerted in the classification task, including a 10% decrease on Trivago and 12% decrease on Taobao. As these two datasets record users' clicking behaviors on the product links provided, the entire dynamic feature sequence carries the long-term preference of each user. So, considering the dynamic dependencies can actually help our model accurately capture the rich information from the dynamic features, and eventually yield competitive prediction effectiveness.

Remove Cross View (Remove CV). Similar to the effect of discarding the dynamic view, SeqFM suffers from the obviously inferior performance (over 5% drop) regarding 5 datasets after the cross view with self-attention head is removed. Apparently, in this degraded version of SeqFM, the interactions between static features and dynamic features are discarded, leading to a significant loss of information. This verifies the contribution of the self-attention head in the cross view to our model's final performance in all three tasks.

Remove Residual Connections (Remove RC). Without residual connections, we find that the performance of SeqFM gets worse, especially on Trivago and Taobao datasets. Presumably this is

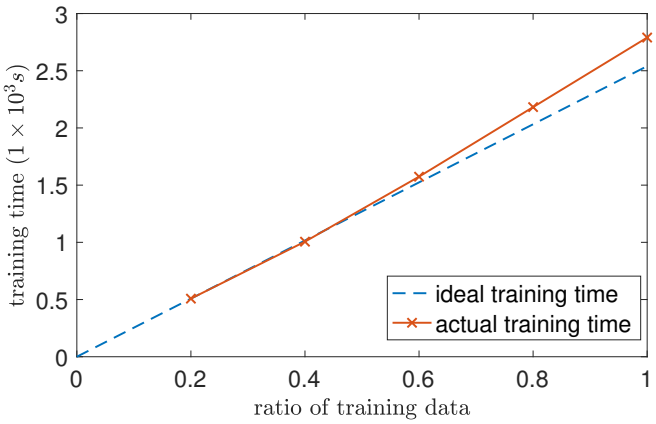


Figure 4.4: Training time of SeqFM w.r.t varied data proportions.

because information in lower layers (i.e., the output generated by the attention head) cannot be easily propagated to the final layer, and such information is highly useful for making predictions, especially on datasets with a large amount of sparse features.

Remove Layer Normalization (Remove LN). The layer normalization operation is introduced mainly for the purpose of stabilizing the training process by scaling the input with varied data scales for deep layers. Removing the layer normalization also shows a negative impact on the prediction performance, especially in the regression task where the properly normalized features can usually generate better results.

4.4.8 Training Efficiency and Scalability (RQ4)

We test the training efficiency and scalability of SeqFM by varying the proportions of the training data in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, and then report the corresponding time cost for the model training. It is worth noting that the Trivago dataset is used for scalability test since it contains the most instances. The growth of training time along with the data size is shown in Figure 4.4. When the ratio of training data gradually extends from 0.2 to 1.0, the training time for SeqFM increases from 0.51×10^3 seconds to 2.79×10^3 seconds. It shows that the dependency of training time on the data scale is approximately linear. Hence, we conclude that SeqFM is scalable to even larger datasets.

4.5 Summary

In this chapter, we propose SeqFM, a sequence-aware factorization machine for temporal predictive analytics. For the first time, we incorporate sequential dependencies into FM-based models by proposing a novel multi-view self-attention scheme to model the interactions between different features. SeqFM is then successfully applied to three different temporal prediction tasks including ranking, classification and regression. The experimental results showcase that SeqFM is a powerful yet general model that can yield superior performance in a wide range of real-world applications. We summarize the specific contributions of this work as follows:

- We point out that mining features' sequential dependencies can greatly benefit the modelling of feature interactions in real-world FM-based models. We introduce, to the best of our knowledge, the first study to endow FM-based models with full sequence-awareness for temporal predictive analytics.
- We propose SeqFM, a novel sequence-aware factorization machine. SeqFM utilizes an innovative multi-view self-attention scheme to model the high-order feature interactions in a sequence-aware manner.
- We conduct extensive experiments on a wide range of benchmark datasets to showcase the superiority of SeqFM in different temporal predictive analytic tasks, validate the importance of sequence-awareness in SeqFM, and reveal promising practicality and scalability of SeqFM.

Chapter 5

Conclusion

In this thesis, we have systematically investigated the problem of sequence modelling in the e-commerce scenario. Specifically, we characterize sequence modelling for e-commerce into two main fields, i.e., macro-level sequence modelling and micro-level sequence modelling that have different focuses and applications. While e-commerce applications of macro-level sequence modelling such as sales prediction [113], stock price forecasting [114] and trend discovery [115] are largely beneficial to decision-makers like sales managers, traders and marketers, micro-level sequence modelling lays more emphasis on enhancing the interactive engagement and maximizing commercial values of individual end-users, such as sequential recommendation [72], targeted advertising [116] and personalized route planning [117]. Hence, this set the tone for the first two research objectives, where we have tackled macro- and micro-level sequence modelling with two concrete tasks, i.e., sales prediction and sequential recommendation. Both tasks have seen wide applications, and are representative research directions in modelling sequential e-commerce data at both macro- and micro-levels. For sales prediction, we have proposed a novel model named TADA, which is based on an encoder-decoder multi-task LSTM architecture. TADA utilizes two attention mechanisms to selectively obtain contextual information in the decoding (i.e., prediction) stage and gather knowledge from similar historical sales trends, thus achieving the state-of-the-art prediction accuracy. For sequential recommendation, our proposed model AIR innovatively uses an attentional LSTM to model the transition patterns of category-wise user intention instead of the commonly used item-wise user preference, which greatly alleviates the sparse nature of user-item interactions in the recommendation data. Furthermore, an intention-aware factorization machine is proposed in AIR to fully leverage the learned user intention to produce fine-grained and personalized item recommendations under the sequential setting.

On top of addressing sequence modelling for e-commerce from each perspective individually, we have extended our investigation by devising a generic supervised machine learning model that can perform both macro- and micro-level sequence modelling and suit a wide variety of their applications. In this regard, we have made a successful attempt by enhancing the long-standing factorization machine (FM) model to the temporal predictive analysis setting. The new model we have developed is called SeqFM, in which the novel multi-view directional self-attention units can support feature

interaction modelling with sequential dependencies, leading to significant performance gain over both FM-based prediction models and state-of-the-art deep neural models in multiple sequence modelling tasks including ranking, classification, and regression.

Along this line of research, there are two promising future directions worth investigating. One is fully explainable sequence modelling techniques, which aim to generate intuitive interpretations to either convince the end-users of the model output [4] (e.g., recommendation results) or provide decision-makers with data-driven insights to help humans derive new knowledge in e-commerce business analysis. The other is to develop memory-efficient and computationally efficient deep methods for sequence modelling, which will enable immense potential in applications [8] on resource-constrained IoT and mobile devices.

Bibliography

[1] T. Chen, H. Yin, H. Chen, L. Wu, H. Wang, X. Zhou, and X. Li. Tada: trend alignment with dual-attention multi-task recurrent neural networks for sales prediction. In *ICDM*, pages 49–58, 2018.

[2] T. Chen, H. Yin, H. Chen, R. Yan, Q. V. H. Nguyen, and X. Li. Air: Attentional intention-aware recommender systems. In *ICDE*, pages 304–315, 2019.

[3] T. Chen, H. Yin, Q. V. H. Nguyen, W.-C. Peng, X. Li, and X. Zhou. Sequence-aware factorization machines for temporal predictive analytics. *ICDE*, pages 1405–1417.

[4] T. Chen, H. Yin, G. Ye, Z. Huang, Y. Wang, and M. Wang. Try this instead: Personalized and interpretable substitute recommendation. *SIGIR*, 2020.

[5] T. Chen, H. Yin, H. Chen, H. Wang, X. Zhou, and X. Li. Online sales prediction via trend alignment-based multitask recurrent neural networks. *Knowledge and Information Systems*, pages 1–29, 2019.

[6] H. Chen, H. Yin, T. Chen, W. Wang, X. Li, and X. Hu. Social boosted recommendation with folded bipartite network embedding. *TKDE*, 2020.

[7] X. Ren, H. Yin, T. Chen, H. Wang, Q. V. H. Nguyen, , Z. Huang, and X. Zhang. Crsal: Conversational recommender systems with adversarial learning. *TOIS*, 2020.

[8] Q. Wang, H. Yin, T. Chen, Z. Huang, H. Wang, Y. Zhao, and Q. V. H. Nguyen. Next point-of-interest recommendation on resource-constrained mobile devices. In *WWW*, 2020.

[9] S. Zhang, H. Yin, T. Chen, Q. V. H. Nguyen, Z. Huang, and L. Cui. Gcn-based user representation learning for unifying robust recommendation and fraudster identification. *SIGIR*, 2020.

[10] R. Qiu, H. Yin, Z. Huang, and T. Chen. Gag: Global attributed graph neural network for streaming session-based recommendation. *SIGIR*, 2020.

[11] K. Sun, T. Qian, T. Chen, Y. Liang, Q. V. H. Nguyen, and H. Yin. Where to go next: Modelling long- and short-term user preferences for point-of-interest recommendation. In *AAAI*, 2020.

[12] H. Chen, H. Yin, T. Chen, Q. V. H. Nguyen, W.-C. Peng, and X. Li. Exploiting centrality information with graph convolutions for network representation learning. In *ICDE*, pages 590–601, 2019.

[13] L. Guo, H. Yin, Q. Wang, T. Chen, A. Zhou, and N. Quoc Viet Hung. Streaming session-based recommendation. In *SIGKDD*, pages 1569–1577, 2019.

[14] S. Zhang, H. Yin, Q. Wang, T. Chen, H. Chen, and Q. V. H. Nguyen. Inferring substitutable products with deep network embedding. *IJCAI*, pages 4306–4312, 2019.

[15] K. Sun, T. Qian, H. Yin, T. Chen, Y. Chen, and L. Chen. What can history tell us? identifying relevant sessions for next-item recommendation. In *CIKM*, 2019.

[16] R. Carbonneau, K. Laframboise, and R. Vahidov. Application of machine learning techniques for supply chain demand forecasting. *European Journal of Operational Research*, 2008.

[17] A. Kochak and S. Sharma. Demand forecasting using neural network for supply chain management. *International journal of mechanical engineering and robotics research*, 2015.

[18] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[19] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461, 2009.

[20] S. Rendle. Factorization machines. In *ICDM*, pages 995–1000, 2010.

[21] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. Fast context-aware recommendations with factorization machines. In *SIGIR*, pages 635–644, 2011.

[22] J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, 2018.

[23] W.-C. Kang and J. McAuley. Self-attentive sequential recommendation. *ICDM*, pages 197–206, 2018.

[24] W. Wang, H. Yin, Z. Huang, Q. Wang, X. Du, and Q. V. H. Nguyen. Streaming ranking based recommender systems. In *SIGIR*, 2018.

[25] X. Ding, Y. Zhang, T. Liu, and J. Duan. Deep learning for event-driven stock prediction. In *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[26] Z. Hu, W. Liu, J. Bian, X. Liu, and T.-Y. Liu. Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 261–269, 2018.

[27] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai. Deep interest network for click-through rate prediction. In *SIGKDD*, pages 1059–1068, 2018.

[28] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, and K. Gai. Deep interest evolution network for click-through rate prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5941–5948, 2019.

[29] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[30] G. Ristanoski, W. Liu, and J. Bailey. Time series forecasting using distribution enhanced linear regression. In *PAKDD*, pages 484–495. Springer, 2013.

[31] P. J. Rousseeuw and A. M. Leroy. *Robust regression and outlier detection*, volume 589. John wiley & sons, 2005.

[32] A. Wilson and R. Adams. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, pages 1067–1075, 2013.

[33] W. Yan, H. Qiu, and Y. Xue. Gaussian process for long-term time-series forecasting. In *IJCNN*, pages 3420–3427. IEEE, 2009.

[34] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[35] T. Chen and C. Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD*, pages 785–794. ACM, 2016.

[36] J. Zhou and A. K. Tung. Smiler: A semi-lazy time series prediction system for sensors. In *SIGMOD*, pages 1871–1886. ACM, 2015.

[37] G. Lai, W. Chang, Y. Yang, and H. Liu. Modeling long-and short-term temporal patterns with deep neural networks. *SIGIR*, pages 95–104, 2018.

[38] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[39] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[40] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *SSST-8 ACL*, 2014.

[41] D. Yao, C. Zhang, J. Huang, and J. Bi. Serm: A recurrent model for next location prediction in semantic trajectories. In *CIKM*, pages 2411–2414, 2017.

[42] X. Zheng, J. Han, and A. Sun. A survey of location prediction on twitter. *TKDE*, 30(9):1652–1671, 2018.

[43] R. Mehrotra, A. H. Awadallah, M. Shokouhi, E. Yilmaz, I. Zitouni, A. El Kholy, and M. Khabsa. Deep sequential models for task satisfaction prediction. In *CIKM*, pages 737–746. ACM, 2017.

[44] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014.

[45] W. Bao, J. Yue, and Y. Rao. A deep learning framework for financial time series using stacked autoencoders and long-short term memory. *PlOS ONE*, 12(7), 2017.

[46] A. Sordoni, Y. Bengio, H. Vahabi, C. Lioma, J. Grue Simonsen, and J.-Y. Nie. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM*, 2015.

[47] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*, pages 1171–1179, 2015.

[48] Y. Wu, J. M. Hernández-Lobato, and Z. Ghahramani. Dynamic covariance models for multivariate financial time series. *ICML*, 2013.

[49] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. *IJCAI*, pages 2627–2633, 2017.

[50] K. L. Caballero Barajas and R. Akella. Dynamically modeling patient's health state from electronic medical records: A time series approach. In *SIGKDD*, pages 69–78, 2015.

[51] W. Chen, S. Wang, x. Zhang, L. Yao, L. Yue, B. Qian, and X. Li. Eeg-based motion intention recognition via multi-task rnns. SDM, 2018.

[52] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB*, pages 697–708, 2005.

[53] N. Q. V. Hung, C. T. Duong, N. T. Tam, M. Weidlich, K. Aberer, H. Yin, and X. Zhou. Argument discovery via crowdsourcing. *VLDBJ*, 2017.

[54] J. D. Hamilton. *Time series analysis*, volume 2. Princeton university press Princeton, 1994.

[55] H.-F. Yu, N. Rao, and I. S. Dhillon. Temporal regularized matrix factorization for high-dimensional time series prediction. In *NIPS*, pages 847–855, 2016.

[56] M. Shokouhi. Detecting seasonal queries by time-series analysis. In *SIGIR*, pages 1171–1172. ACM, 2011.

[57] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, pages 1764–1772, 2014.

[58] T. Idé and S. Kato. Travel-time prediction using gaussian process regression: A trajectory-based approach. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 1185–1196. SDM, 2009.

[59] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

[60] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, pages 2048–2057, 2015.

[61] H. Yin, H. Chen, X. Sun, H. Wang, Y. Wang, and Q. V. H. Nguyen. Sptf: a scalable probabilistic tensor factorization model for semantic-aware behavior prediction. In *ICDM*, 2017.

[62] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017.

[63] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015.

[64] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. *EMNLP*, 2016.

[65] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

[66] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li. Pme: projected metric embedding on heterogeneous networks for link prediction. In *SIGKDD*, 2018.

[67] M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-aware recommender systems. *ACM Computing Surveys*, 2018.

[68] Y. Sun and Y. Zhang. Conversational recommender system. *SIGKDD*, 2018.

[69] X. He and T.-S. Chua. Neural factorization machines for sparse predictive analytics. In *SIGIR*, pages 355–364, 2017.

[70] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun. xdeepfm: Combining explicit and implicit feature interactions for recommender systems. *SIGKDD*, pages 1754–1763, 2018.

[71] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, and T.-S. Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. *IJCAI*, pages 3119–3125, 2017.

[72] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[73] J. Ye, Z. Zhu, and H. Cheng. What's your next move: User activity prediction in location-based social networks. In *SDM*, 2013.

[74] Q. He, D. Jiang, Z. Liao, S. C. Hoi, K. Chang, E.-P. Lim, and H. Li. Web query recommendation via sequential query prediction. In *ICDE*, 2009.

[75] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. *ICLR*, 2015.

[76] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing. Recurrent recommender networks. In *WSDM*, pages 495–503, 2017.

[77] Y. Song, A. M. Elkahky, and X. He. Multi-rate deep learning for temporal recommendation. In *SIGIR*, 2016.

[78] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T.-Y. Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI*, 2014.

[79] B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*, 2018.

[80] D. T. LE, H. W. LAUW, and Y. Fang. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. IJCAI, 2018.

[81] S. Zhang, Y. Tay, L. Yao, and A. Sun. Next item recommendation with self-attention. *arXiv preprint*, 2018.

[82] B. Hu, C. Shi, W. X. Zhao, and P. S. Yu. Leveraging meta-path based context for top-n recommendation with a neural co-attention model. In *SIGKDD*, 2018.

[83] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *SIGIR*, 2017.

[84] L. Hong, A. S. Doumith, and B. D. Davison. Co-factorization machines: modeling user interests and predicting individual decisions in twitter. In *WSDM*, pages 557–566, 2013.

[85] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, pages 39–46, 2010.

[86] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM TIIS*, 2016.

[87] Q. Wang, H. Yin, Z. Hu, D. Lian, H. Wang, and Z. Huang. Neural memory streaming recommender networks with adversarial training. In *SIGKDD*, 2018.

[88] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[89] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *CIKM*, 2017.

[90] J. Wang, P. Huang, H. Zhao, Z. Zhang, B. Zhao, and D. L. Lee. Billion-scale commodity embedding for e-commerce recommendation in alibaba. *SIGKDD*, 2018.

[91] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han. Bridging collaborative filtering and semi-supervised learning: A neural approach for poi recommendation. In *SIGKDD*, pages 1245–1254, 2017.

[92] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin. Field-aware factorization machines for ctr prediction. In *RecSys*, pages 43–50, 2016.

[93] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. *TIST*, page 27, 2011.

[94] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant. *Applied logistic regression*, volume 398. John Wiley & Sons, 2013.

[95] Y. Shan, T. R. Hoens, J. Jiao, H. Wang, D. Yu, and J. Mao. Deep crossing: Web-scale modeling without manually crafted combinatorial features. In *SIGKDD*, pages 255–262, 2016.

[96] J. Lian, F. Zhang, M. Hou, H. Wang, X. Xie, and G. Sun. Practical lessons for job recommendations in the cold-start scenario. In *RecSys Challenge*, page 4, 2017.

[97] G. Liu, T. T. Nguyen, G. Zhao, W. Zha, J. Yang, J. Cao, M. Wu, P. Zhao, and W. Chen. Repeat buyer prediction for e-commerce. In *SIGKDD*, pages 155–164, 2016.

[98] W. Zhang, T. Du, and J. Wang. Deep learning over multi-field categorical data. In *ECIR*, pages 45–57, 2016.

[99] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang. Product-based neural networks for user response prediction. In *ICDM*, pages 1149–1154, 2016.

[100] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.

[101] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, et al. Wide & deep learning for recommender systems. In *DLRS*, pages 7–10, 2016.

[102] S. Rendle. Factorization machines with libfm. *TIST*, page 57, 2012.

[103] P. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, pages 67–119, 2014.

[104] R. Pasricha and J. McAuley. Translation-based factorization machines for sequential recommendation. In *RecSys*, pages 63–71, 2018.

[105] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[106] J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[107] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015.

[108] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, pages 1929–1958, 2014.

[109] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio. An empirical analysis of dropout in piecewise linear networks. *arXiv preprint arXiv:1312.6197*, 2013.

[110] H. Yin, X. Zhou, B. Cui, H. Wang, K. Zheng, and Q. V. H. Nguyen. Adapting to user interest drift for poi recommendation. *TKDE*, pages 2566–2581, 2016.

[111] H. Li, Y. Ge, R. Hong, and H. Zhu. Point-of-interest recommendations: Learning potential check-ins from friends. In *SIGKDD*, pages 975–984, 2016.

[112] M. Blondel, A. Fujino, N. Ueda, and M. Ishihata. Higher-order factorization machines. In *NIPS*, pages 3351–3359, 2016.

[113] X. Yu, Y. Liu, X. Huang, and A. An. A quality-aware model for sales prediction using reviews. In *Proceedings of the 19th international conference on World wide web*, pages 1217–1218, 2010.

[114] L. Zhang, C. Aggarwal, and G.-J. Qi. Stock price prediction via discovering multi-frequency trading patterns. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 2141–2149, 2017.

[115] G. Xun, Y. Li, J. Gao, and A. Zhang. Collaboratively improving topic discovery and word embeddings by coordinating global and local contexts. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 535–543, 2017.

[116] C. Pei, X. Yang, Q. Cui, X. Lin, F. Sun, P. Jiang, W. Ou, and Y. Zhang. Value-aware recommendation based on reinforcement profit maximization. In *The World Wide Web Conference*, pages 3123–3129, 2019.

[117] J. Dai, B. Yang, C. Guo, and Z. Ding. Personalized route recommendation using big trajectory data. In *2015 IEEE 31st international conference on data engineering*, pages 543–554. IEEE, 2015.