Towards Effective Recommendation: Neural Networks and Adaptive Learning

Chen Ma

School of Computer Science,
McGill University, Montreal, Canada

A thesis submitted to McGill University
in partial fulfillment of the requirements of the degree of

DOCTOR of PHILOSOPHY

July 2021

2021/07/01

Abstract

With the development of Internet services, Internet users can easily access a large number of online products and applications. Although this growth provides users with more available choices, it is also difficult for users to pick up one of the most favorite items out of plenty of candidates. To reduce information overload as well as satisfy the diverse needs of users, personalized recommender systems come into being and play more and more important roles in modern society. In particular, these systems can provide personalized experiences, serve huge service demands, and bring direct benefits to both users and recommendation service providers.

However, recommendation models still face several challenges and prevent them from achieving satisfactory services. Specifically, these challenges include the difficulty of modeling user-item interactions from the sparse data, the hardship of incorporating auxiliary information from other information sources, the intricacy of capturing the user short-term interest from the item sequential dynamics, and the hardness of conducting the adaptive hyper-parameter learning according to different users or items.

In this thesis, we propose dedicated models to tackle the aforementioned challenges. *First*, to model the complex interactions between users and Point-of-Interests (POIs) in location-based recommender systems, a novel autoencoder-based model is proposed to learn the nonlinear user-POI relations, which consists of a self-attentive encoder and a neighbor-aware decoder. The self-attentive encoder adaptively distinguishes users' preferences on checked-in POIs; the neighbor-aware decoder incorporates POIs' geographical influence to make user reachability higher on unvisited neighbors of checked-in POIs. *Second*, to effectively incorporate the content auxiliary information from the item description, a gated autoencoder with the word- and neighbor-attention mechanism is proposed. The model learns items' hidden representations from ratings and contents in a gated manner. Moreover, the model also captures items' informative words and representative neighbors by word- and neighbor-attention modules, respectively. *Third*, to model the user preference transition with the temporal dynamics, a hierarchical gating network with an item-item product module is proposed for the sequential recommendation task. The model adopts a feature gating module and an instance gating module to control what item features can be passed to downstream layers, where informative latent features and representative items can be identified. *Fourth*, to conduct the adaptive hyper-parameter learning, an adaptive

margin generation scheme is proposed to generate the margins regarding different training triplets in the margin ranking loss function, where a bilevel optimization framework is adopted to alternatively update the model parameters as well as the margin values.

To demonstrate the effectiveness of the aforementioned models, a number of publicly available datasets and benchmarks with many state-of-the-art methods are utilized to validate the performance of proposed models. The evaluation results verify that our proposed models achieve better performance than that of the state-of-the-art methods. The contribution of the proposed component in each model is demonstrated in detailed ablation analysis.

Abrégé

Avec le développement des services Internet, les internautes peuvent facilement accéder à un grand nombre de produits et d'applications en ligne. Bien que cette croissance offre aux utilisateurs un plus grand nombre de choix disponibles, il est également difficile pour eux de choisir un article préféré parmi une multitude de candidats. Afin de réduire la surcharge d'informations et de satisfaire les divers besoins des utilisateurs, les systèmes de recommandation personnalisés voient le jour et jouent un rôle de plus en plus important dans la société moderne. En particulier, ces systèmes peuvent fournir des expériences personnalisées, répondre à d'énormes demandes de services et apporter des avantages directs à la fois aux utilisateurs et aux fournisseurs de services de recommandation.

Cependant, les modèles de recommandation sont toujours confrontés à plusieurs défis qui les empêchent de fournir des services satisfaisants. Plus précisément, ces défis comprennent la difficulté de modéliser les interactions entre l'utilisateur et l'article à partir de données éparses, la difficulté d'incorporer des informations auxiliaires provenant d'autres sources d'information, la complexité de capturer l'intérêt à court terme de l'utilisateur à partir de la dynamique séquentielle de l'article et la difficulté d'effectuer l'apprentissage adaptatif des hyperparamètres en fonction des différents utilisateurs ou articles.

Dans cette thèse, nous proposons des modèles spécifiques pour relever les défis susmentionnés. *Premièrement*, pour modéliser les interactions complexes entre les utilisateurs et les points d'intérêt (POI) dans les systèmes de recommandation basés sur la localisation, un nouveau modèle basé sur un encodeur automatique est proposé pour apprendre les relations non linéaires entre utilisateur-POI, qui consiste en un encodeur auto-attentif et un décodeur sensible au voisinage. Le codeur auto-attentif distingue de manière adaptative les préférences des utilisateurs sur les POI enregistrés ; le décodeur tenant compte des voisins incorpore l'influence géographique des POI pour rendre l'accessibilité de l'utilisateur plus élevée sur les voisins non visités des POI enregistrés. *Deuxièmement*, pour incorporer efficacement les informations auxiliaires du contenu de la description de l'article, nous proposons un auto-codeur à déclenchement avec le mécanisme d'attention aux mots et aux voisins. Le modèle apprend les représentations cachées des articles à partir des évaluations et du contenu d'une manière déclenchée. De plus, le modèle capture également les mots informatifs des articles et les voisins représentatifs par des modules d'attention aux mots et aux voisins, respectivement. *Troisièmement*, pour modéliser la transition des préférences

de l'utilisateur avec la dynamique temporelle, un réseau de déclenchement hiérarchique avec un module de produit article-article est proposé pour la tâche de recommandation séquentielle. Le modèle adopte un module de déclenchement de caractéristiques et un module de déclenchement d'instances pour contrôler les caractéristiques des articles qui peuvent être transmises aux couches en aval, où les caractéristiques latentes informatives et les articles représentatifs peuvent être identifiés. *Quatrièmement*, pour effectuer l'apprentissage adaptatif des hyperparamètres, un schéma de génération de marge adaptative est proposé pour générer les marges concernant différents triplets d'entraînement dans la fonction de perte de classement de marge, où un cadre d'optimisation à deux niveaux est adopté pour mettre à jour alternativement les paramètres du modèle ainsi que les valeurs de marge.

Pour démontrer l'efficacité des modèles susmentionnés, nous avons utilisé un certain nombre d'ensembles de données et de points de référence accessibles au public, avec de nombreuses méthodes de pointe pour valider la performance des modèles proposés. Les résultats de l'évaluation montrent que les modèles que nous proposons obtiennent de meilleures performances que celles des méthodes de pointe. La contribution du composant proposé dans chaque modèle est démontrée dans une analyse d'ablation détaillée.

Acknowledgements

Time flies, five years passed like a rush. I still remember the first day that I entered McGill University with the hope to make something great happen in the near future. In the last five years, I worked very hard and tried to give my best efforts on everything related to my research. The completion of this doctoral dissertation acts like a brief summary of my five years of study, research, and life. It is not possible without the help and support from my supervisors, collaborators, labmates, friends, families, etc. Their love, kindness, and encouragement switch my tough days into a rewarding experience. I would like to express my sincere gratitude.

First and foremost, I would like to express my heartfelt gratitude to Professor Xue Liu, my Ph.D. supervisor, for his unwavering support, invaluable research guidance, and constant encouragement during the research process. This feat can be achieved only because of the constant support from Prof. Liu. He is visionary and has a lot of profound thinking. He has always patiently taught me how to become a good researcher and scientist. His passion for research, teaching, and student supervision will continuously motivate me to advance my career in the future. It has been a privilege to work with him on my doctoral dissertation and benefit from his research experience.

I would like to express my gratitude to my collaborators, Professor Mark Coates, Professor Wei Qi, Professor Junyu Cao, Dr. Fernando Diaz, Dr. Bhaskar Mitra, etc. They have provided insightful comments and suggestions in this thesis as well as other research works. It is a great pleasure that I can have a chance to work with them. In particular, I am very grateful to Professor Mark Coates, who has given invaluable suggestions on my research and shared with me his enormous amount of knowledge during the last two years of my Ph.D. study. I also especially appreciate Ms. Yingxue Zhang for her generous help on matters of numerous and varied points of assistance.

I also thank my thesis committee members and examiners, Professor Xiao-Wen Chang, Professor Mark Coates, Professor Benjamin Fung, and Professor Jian Pei for being on my committee and for their constructive reviews and examinations on my academic performance.

Further, I would like to thank my present and past labmates in the Cyber-Physical System Lab at McGill University for the beneficial discussions that we have had throughout these years and great friends of mine for their kind care and support. I would not be able to

make it without their generosity and help. In particular, I would like to thank Prof. Qiao Xiang, Dr. Xi Chen, Dr. Minyuan Xia, Dr. Jing Chen, Dr. Qinglong Wang, Peng Kang, Jikun Kang, Xuan Li, Hang Li, Dan Liu, Haolun Wu, Liheng Ma, Yimeng Wu, Yifan He, Zhaodong Wang, Eddie Xu, Xi Yue, Jianing Sun. I have been very lucky to have such great friends in Montreal. I am grateful to all of them for making my life joyful.

Finally, I would like to express my greatest gratitude to my families. Their love and trust that make me endure the loneliness and hardship in a foreign country. It is their encouragement that drove me through those bad days and lighted the beacon for my future.

It is a pity that I cannot list all the names of people that provide valuable help and support during my Ph.D. journey in a limited space. I would like to thank them for being in my life and sharing their precious time with me. I wish everyone a happy, healthy, and prosperous life.

Contents

1 Introduction **1**
1.1 Challenges . . . 2
1.2 Motivations . . . 3
1.3 Contributions . . . 5
1.4 Statement of Author Contribution . . . 6

2 Background of Recommender Systems **8**
2.1 Overview . . . 8
2.2 Problem Setting . . . 9
2.2.1 Rating Prediction . . . 9
2.2.2 Item Ranking . . . 10
2.3 Classical Recommendation Methods . . . 10
2.3.1 Collaborative Filtering . . . 10
2.3.2 Content-based Filtering . . . 12
2.4 Evaluation Metrics . . . 13
2.4.1 Rating Prediction . . . 13
2.4.2 Item Ranking . . . 13

3 Related Work **15**
3.1 General Personalized Recommendation . . . 15
3.1.1 Matrix Factorization-based Methods . . . 15
3.1.2 Multi-layer Perceptron-based Methods . . . 16
3.1.3 Distance-based Methods . . . 18
3.2 Location-aware Recommendation . . . 19
3.3 Content-aware Recommendation . . . 20

3.4 Sequential Recommendation 21

4 Neural Networks for Point-of-Interest Recommendation: Capturing Complex User-Item Interactions 23

4.1 Introduction 23
4.2 Preliminaries 26
4.2.1 Definition and Notation 26
4.2.2 Autoencoders 27
4.3 Methodologies 28
4.3.1 Model Basics 29
4.3.2 Self-Attentive Encoder 30
4.3.3 Neighbor-Aware Decoder 32
4.3.4 Weighted Loss for Implicit Feedback 34
4.3.5 Network Training 35
4.4 Experiments 36
4.4.1 Datasets 36
4.4.2 Evaluation Metrics 36
4.4.3 Methods Studied 37
4.4.4 Parameter Settings 38
4.4.5 Performance Comparison 39
4.4.6 Impacts of Self-Attentive Encoder and Neighbor-Aware Decoder . . 41
4.4.7 Sensitivity of Hyper-Parameters 42
4.5 Summary 43

5 Neural Networks for Content-aware Recommendation: Incorporating Content Information 45

5.1 Introduction 45
5.2 Problem Formulation 47
5.3 Methodologies 48
5.3.1 Model Basics 48
5.3.2 Word-Attention Module 49
5.3.3 Neural Gating Layer 51
5.3.4 Neighbor-Attention Module 52

5.3.5 Weighted Loss . . . 53
5.3.6 Network Training . . . 53
5.4 Experiments . . . 53
5.4.1 Datasets . . . 53
5.4.2 Evaluation Metrics . . . 54
5.4.3 Methods Studied . . . 55
5.4.4 Experiment Settings . . . 56
5.4.5 Performance Comparison . . . 56
5.4.6 Ablation Analysis . . . 60
5.4.7 The Sensitivity of Hyper-parameters . . . 62
5.4.8 Word- and Neighbor-Attention Case Studies . . . 63
5.5 Summary . . . 64

6 Neural Networks for Sequential Recommendation: Modeling Temporal Dynamics 65
6.1 Introduction . . . 65
6.2 Problem Formulation . . . 67
6.3 Methodologies . . . 67
6.3.1 Hierarchical Gating for Group-level Influence . . . 69
6.3.2 Item-item Product . . . 72
6.3.3 Prediction Layer . . . 73
6.3.4 Network Training . . . 73
6.4 Experiments . . . 74
6.4.1 Datasets . . . 74
6.4.2 Evaluation Metrics . . . 75
6.4.3 Methods Studied . . . 75
6.4.4 Experiment Settings . . . 76
6.4.5 Performance Comparison . . . 77
6.4.6 Ablation Analysis . . . 80
6.4.7 Training Efficiency . . . 82
6.4.8 The Sensitivity of Hyper-parameters . . . 83
6.5 Summary . . . 84

7 Neural Networks for Adaptive Learning: Learning Personalized Hyper-parameters 85
7.1 Introduction . . . 85
7.2 Problem Formulation . . . 87
7.3 Methodology . . . 88
7.3.1 Wasserstein Distance for Interactions . . . 88
7.3.2 Adaptive Margin in Margin Ranking Loss . . . 90
7.3.3 User-User and Item-Item Relations . . . 93
7.3.4 Model Training . . . 94
7.4 Experiments . . . 95
7.4.1 Datasets . . . 95
7.4.2 Evaluation Metrics . . . 96
7.4.3 Methods Studied . . . 96
7.4.4 Experiment Settings . . . 97
7.4.5 Implementation Details . . . 98
7.4.6 Performance Comparison . . . 100
7.4.7 Ablation Analysis . . . 102
7.4.8 Case Study . . . 103
7.5 Summary . . . 104

8 Conclusion and Future Work 105
8.1 Conclusion . . . 105
8.2 Future Work . . . 107

A Publications 109

Bibliography 111

List of Figures

4.1 The model architecture of SAE-NAD. The yellow part is the self-attentive encoder, the green part is the neighbor-aware decoder, and the gray part is the attention network. The bright yellow rectangle is the user hidden representation. Specifically, *Att_Layer* denotes the attention layer and *Agg_Layer* denotes the aggregation layer. 29
4.2 The comparison of performance on Gowalla. 39
4.3 The comparison of performance on Foursquare. 39
4.4 The comparison of performance on Yelp. 39
4.5 The effect of d_a. 42
4.6 The effect of γ. 43

5.1 The architecture of GATE. The yellow part is the stacked AE for binary rating prediction, and the green part is the word-attention module for item content. The blue rectangle is the gating layer to fuse the hidden representations. The middle pink part is the neighbor-attention module to obtain the hidden representation of an item's neighborhood. Specifically, *Word_Att* denotes the word-attention layer, *Neighbor_Att* denotes the neighbor-attention layer, and *Agg_Layer* denotes the aggregation layer. $\odot$ is the element-wise multiplication and $\oplus$ is the element-wise addition. 48
5.2 The performance comparison on citeulike-a. 57
5.3 The performance comparison on movielens-20M. 58
5.4 The performance comparison on Amazon-Books. 58
5.5 The performance comparison on Amazon-CDs. 59
5.6 The effects of ρ and d_a. 61

6.1 An illustrative example of the feature gating, instance gating, and item-item product modules. In Figure 6.1a, the gray lines on items denote those latent features are masked off. In Figure 6.1b, the darker blue means the item is more important. In Figure 6.1c, the line linked between two items denotes the inner product, which captures the relations between the items users have accessed and the items users will access in the future. 68
6.2 The architecture of HGN. HGN consists of three major components: the embedding layer, the hierarchical gating layer, and the prediction layer. Specifically, *F Gating* denotes the feature gating module, *I Gating* denotes the instance gating module, *Aggregation* denotes the aggregation layer, and $\otimes$ denotes the element-wise multiplication. 70
6.3 The performance comparison on MovieLens-20M. 78
6.4 The performance comparison on Amazon-Books. 78
6.5 The performance comparison on Amazon-CDs. 79
6.6 The performance comparison on Children. 79
6.7 The performance comparison on Comics. 80
6.8 The dimension variations of embeddings. 83

7.1 The demonstration of the proposed model. $\mathcal{J}^{U-I}$ denotes the combined optimization regarding $\mathcal{J}^{U-I}_{inner}$ and $\mathcal{J}^{U-I}_{outer}$. $\mathcal{J}^{U-U}$ and $\mathcal{J}^{I-I}$ follow the same manner with $\mathcal{J}^{U-I}$. 88
7.3 The performance comparison on all datasets. 100

List of Tables

4.1 List of notations. 27
4.2 The statistics of datasets. 36
4.3 The performance of the self-attentive encoder and neighbor-aware decoder on Gowalla, Foursquare, and Yelp. *P@10* denotes Precision@10 and *R@10* denotes Recall@10. 41

5.1 The statistics of datasets. 54
5.2 The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method. 57
5.3 The ablation analysis on Amazon-CDs and Amazon-Books datasets in terms of Recall@10 (R@10) and NDCG@10 (N@10). 60
5.4 A case study of the word-attention. 62
5.5 A case study of the importance scores computed by the neighbor-attention module. The number inside $(\cdot)$ indicates the number of *fluctuation*'s occurrences excluding references in an article. 63

6.1 List of notations. 68
6.2 The statistics of datasets. 75

6.3 The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method. 77
6.4 The ablation analysis on GoodReads-Comics and Amazon-Books datasets. F denotes the feature gating module, I denotes the instance gating module, *avg* denotes the average pooling, and *max* denotes the max pooling. 81
6.5 The training time per epoch comparison on five datasets in terms of seconds. 82
6.6 The effect of the length $|L|$ and $|T|$. 83

7.1 The statistics of the datasets. 96
7.2 The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method. 98
7.3 The ablation analysis on the CDs and Electronics datasets in terms of Recall@10 (R@10) and NDCG@10 (N@10). *cat* denotes the concatenation operation and *add* denotes the addition operation. 102
7.4 A case study of the generated margin of sampled training triplets. The movie genre label is from the dataset. 104

Chapter 1

Introduction

With the rapid growth of Internet services and mobile devices, it has been more convenient to access amounts of online products and multimedia contents, such as movies and articles. For instance, Amazon sells more than 12 million products[1] and Yelp has 4.9 million local businesses[2]. Although this growth allows users to have multiple choices, it has also made it more difficult to select one of the user's most preferred items out of thousands of candidates. For example, users who like to watch movies may feel difficult to decide which movie to watch when there are thousands of selections; users who are gourmet eaters may feel hard to discover new restaurants tailored to their flavors.

The above needs facilitate a promising and practical service—*personalized recommender systems*. These systems are becoming increasingly essential, serving a potentially huge service demand, and bringing significant benefits to at least two parties: (i) help users easily discover products that they are interested in, and (ii) create opportunities for product providers to increase the revenue. Due to the ability to provide personalized services and bring benefits to service providers, recommender systems are extensively deployed in Internet services nowadays. For example, around 80% of watching actions in Netflix[3] and 35% of sales in Amazon[4] are brought by their recommendation engines.

To enable personalized recommendation services, the collection of user behaviors on items/products is of the essence. Generally, a recommender system includes three main

[1] `https://www.bigcommerce.com/blog/amazon-statistics/`
[2] `https://expandedramblings.com/index.php/yelp-statistics/`
[3] `https://www.wired.co.uk`
[4] `https://martechtoday.com/roi-recommendation-engines-marketing-205787`

elements: *users*, *items*, and user-item *interactions*. Users interact with items by giving ratings on items such as five stars on Netflix or performing multiple actions on items, e.g., browsing or purchasing on Amazon and check-ins on Yelp. With the user-provided feedback, user's preferences can be learned via various techniques [1–7], and items that users are interested in can be recommended.

1.1 Challenges

Although the personalized recommendation is extensively studied and investigated in the past twenty years, it still faces several challenges leading to unsatisfactory services. The causes of unsatisfactory performance are brought by the inherent sparsity of data, the hardship of modeling user-item interactions, the difficulty of incorporating auxiliary information, the intricacy of capturing the user behavior dynamics, the tedious need for hyper-parameter tuning, etc. The detailed illustrations of these challenges are summarized as follows:

- *Modeling Complex Interactions.* The user-item interaction modeling lies at the core of recommender systems. The way of how to model the user-item interaction largely affects the recommendation performance. Many recommendation models only capture the linear relationship between users and items while neglecting the intricacy and non-linearity of real-life historical interactions. For example, matrix factorization-based methods [8]—the most commonly used recommendation model—adopt the inner product between the user and item latent factors to model user-item interactions which are essentially linear models [9].

- *Incorporating Auxiliary Information.* Auxiliary information, such as genres, titles, and plots of movies, is highly useful for understanding user preference. However, the representation of auxiliary information is a non-trivial task, since most of the real-world data like texts are not directly represented by numerical values, which need to be transformed into model-executable formats. Therefore, how to represent the auxiliary information of users or items and integrate them into the user preference learning is significant for recommender systems.

- *Capturing Temporal Dynamics.* The user interest may dynamically shift from time to time. The chronological order of user-item interactions is the main feature to reflect

the transition of user interests, where the items that users will interact with may largely depend on those items that users just accessed recently. Thus, it is crucial to capture the user interest transition and take advantage of the sequential dynamics for predicting user actions in the near future.

- *Learning Adaptive Hyper-parameters.* Hyper-parameters are one of the most important factors that have an effect on the machine learning model. Typically, the hyper-parameter is manually selected and keeps fixed in the whole training process. While some recent studies show that the hyper-parameter can be learned adaptively along with other learnable parameters in the model training [10]. In the recommendation scenario, some hyper-parameters can be further cast into a personalized/adaptive manner [11, 12], where each user or item has one specific hyper-parameter instead of sharing with all other users or items. This makes the model have more flexibilities and yields performance improvement in many previous works [11, 12]. Therefore, how to adaptively learn the values of these personalized hyper-parameters is not trivial.

1.2 Motivations

To solve the above challenges, more powerful methods are required to capture the intricate characteristics and complex structures in the input data. Recently, due to the ability to represent non-linear and complex data, (deep) neural networks have been a great success in many domains, such as computer vision [13, 14], natural language processing [15, 16] and graph learning [17, 18], and bring more opportunities to reshape the conventional recommendation architectures. Compared to traditional hand-designed feature-based models, these end-to-end differentiable neural models have demonstrated promising performance in representation learning for high-dimensional and large-scale data, sequence modeling, and relational learning. In this thesis, we hope to address the aforementioned challenges by leveraging the strengths of (deep) neural networks or adopting the techniques that can improve the performance of neural networks:

- *Universal Approximation.* One of the most striking facts about neural networks is that they can approximate any function, and there is guaranteed to be a neural network no matter what the function is [19]. In the real world, it is very likely that millions of users/items dwell in the recommender system, such a large scale of data makes

the user-item interaction complex and intricate. Many conventional methods such as matrix factorization [20] and factorization machines [21] are essentially linear models. For instance, matrix factorization-based methods apply the inner product to linearly combine the latent factors of users and items as the prediction score. The linear assumption, the basis of many traditional recommendation models, is oversimplified and will greatly limit their modeling expressiveness [9, 22]. In contrast, (deep) neural networks have the ability to approximate arbitrary functions, which are a good fit to model complex user-item interactions.

- *Unstructured Data Representation.* Generally, data can be categorized into two types: structured data and unstructured data. Structured data is comprised of clearly defined data types whose pattern makes them easily searchable such as the database of student's marksheet; while unstructured data has no pre-defined format or organization, making it much more difficult to collect, process, and analyze, including formats like audio, video, and social media postings. In a recommender system, rather than solely rating items, users can also interact with items via other means like writing reviews or posting pictures. These unstructured auxiliary data can be highly helpful for inferring users' preferences. Due to the ability to effectively represent unstructured data, (deep) neural networks have advantages of learning the auxiliary information in recommender systems: (i) they do not require the explicit feature engineering, as they are capable of executing feature engineering on their own and searching for features that correlate and combining them to enable faster learning; (ii) unstructured data is easily transformed into real-value representations, and heterogeneous information such as texts and pictures can be comfortably integrated. Thanks to these two merits, the auxiliary information can be effectively modeled and incorporated by (deep) neural networks in recommender systems.

- *Sequence Learning.* Many types of data in the real world are sequential, e.g., speech, DNA, stock prices, and customer action histories. (Deep) neural networks have shown significant performance improvement on a number of sequence learning tasks such as speech recognition [23] and machine translation [24]. In particular, recurrent neural networks (RNNs) and convolutional neural networks (CNNs) are two important cornerstones that enable the powerful learning of sequences. These advances in sequence learning also facilitate the modeling of user preference shifts in recommender sys-

tems [6]. In the Internet service, users access the products or items in a chronological order, where the sequential semantics makes a difference for predicting users' future behaviors. That is, the items a user will interact with may be closely relevant to those items he/she just accessed. Thus, these (deep) neural networks tailored to sequential data can be a good means to capture the future behaviors of users.

- *Automated Hyper-parameter Learning.* The hyper-parameter is a crucial element in a machine learning or recommendation model, where the values of hyper-parameters can significantly influence the quality of models. Typically, hyper-parameters are tuned manually and it takes a long time to find reasonable ones since they waste time evaluating unpromising areas of the search space. With the advance of deep learning, hyper-parameter tuning techniques [25] are largely developed to find optimal hyper-parameters in less time. In the recommendation scenario, the hyper-parameter can be further customized into a fine-grained manner since distinct users and items dwell in the system. Different users or items may better have their own hyper-parameters to fully unleash the potential of a recommendation model. Existing recommendation models [11, 12] have demonstrated performance gain by doing so. Thus, the hyper-parameter tuning techniques need to be enhanced to learn the personalized and adaptive hyper-parameters in the recommendation model.

1.3 Contributions

To tackle the challenges mentioned in Section 1.1, we investigate and design effective neural recommendation models powered by the merits of (deep) neural networks. In particular, neural networks are utilized to model the complex and non-linear user-item interactions, integrate useful user/item auxiliary information, capture the sequential dynamics in the user behaviors, and generate adaptive hyper-parameter values. To summarize, the major contributions of this thesis are listed as follows:

- In Chapter 4, we propose an autoencoder-based model to model the complex user-POI interaction for Point-of-Interest (POI) recommendation [26], which consists of a self-attentive encoder and a neighbor-aware decoder. The self-attentive encoder adaptively discriminates users' preferences on checked-in POIs. The neighbor-aware decoder incorporates POIs' geographical influence to make user reachability higher

on unvisited neighbors of checked-in POIs.

- In Chapter 5, we propose a gated attentive-autoencoder to effectively incorporate the content auxiliary information for the content-aware recommendation [27]. The model consists of three modules: a word-attention module, a neighbor-attention module, and a neural gating layer. The word-attention module selects informative words by assigning larger attention weights. The neighbor-attention module distinguishes the neighboring items of an item in a weighted manner. The neural gating layer smoothly fuses item hidden representations from heterogeneous sources.

- In Chapter 6, we propose a hierarchical gating network to model the user interest transition for the sequential recommendation task [28]. The model is composed of three modules: a feature gating module, an instance gating module, and an item-item product module. The feature gating and instance gating modules adaptively select what item features can be passed to the downstream layers. The item-item product module explicitly captures the relations between the items that users accessed in the past and the items users will access in the near future.

- In Chapter 7, we propose an adaptive margin (a hyper-parameter in the margin ranking loss) learning module in a distance-based recommendation model [29]. A bilevel optimization scheme is proposed to alternatively update the margin-related parameters as well as the parameters of recommendation models. By incorporating the adaptive margin module, our model can generate fine-grained margins for the training triples during the training procedure. To explicitly capture the user-user/item-item relations, we adopt two additional margin ranking losses with adaptive margins to force similar user and item pairs to map closer together in the latent space.

1.4 Statement of Author Contribution

In the following, I provide a contribution statement of co-authors regarding Chapters 4, 5, 6, and 7.

Chapter 4 is based on the work [26] that is published in Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018). I proposed the idea, implemented the model, and wrote the paper draft. Yingxue Zhang

helped run two baseline models. Prof. Xue Liu and Dr. Qinglong Wang helped revise the paper. All authors participated in the discussion of the model results.

Chapter 5 is based on the work [27] that is published in Proceedings of the 12th ACM International Conference on Web Search and Data Mining (WSDM 2019). I proposed the idea, implemented the model, and wrote the paper draft. Peng Kang helped run three baseline models and draw the model design figure. Dr. Bin Wu gave constructive comments to improve the paper. Prof. Xue Liu and Dr. Qinglong Wang helped revise the paper. All authors participated in the discussion of the model results.

Chapter 6 is based on the work [28] that is published in Proceedings of the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2019). I proposed the idea, implemented the model, and wrote the paper draft. Peng Kang helped run two baseline models and draw the illustrative figures. Prof. Xue Liu helped revise the paper. All authors participated in the discussion of the model results.

Chapter 7 is based on the work [29] that is published in Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2020). I proposed the idea, implemented the model, and wrote the paper draft. Liheng Ma helped revise the formulation of the bilevel optimization and draw the model design figures. Yingxue Zhang and Dr. Ruiming Tang gave constructive suggestions to improve the paper quality. Prof. Xue Liu and Prof. Mark Coates helped revise the paper. All authors participated in the discussion of the model results.

Chapter 2

Background of Recommender Systems

In this chapter, we introduce the background of recommender systems, the problem settings in the recommendation research community, the classical recommendation methods, and the commonly-used evaluation metrics.

2.1 Overview

Thanks to the ability to provide personalized services and serve huge service demands, recommender systems have become an essential component in all kinds of Internet applications. To make accurate recommendations, effective methods have been proposed. They can be briefly divided into the following categories [30]:

- *Collaborative filtering* methods are based on the assumption that users who have similar preferences in the past would also have similar preferences in the future, and items are recommended to users based on the preferences other users have expressed for those items. This kind of method generates recommendations using only information about rating/interaction data for different users or items. By locating peer users/items with a rating history similar to the current user or item, they generate recommendations using this neighborhood.

- *Content-based filtering* methods are based on descriptions of items and profiles of users' preferences, where hand-crafted features (e.g., bag-of-words or term frequency-

inverse document frequency) shall be extracted from the item description or user profile. Then content-based filtering makes predictions regarding users' preferences or ratings, by identifying items with similar content or tailored to the user profile, on the basis of the extracted item/user features.

- *Hybrid methods* are a combination of collaborative filtering and content-based filtering, which benefit from their complementary advantages.

To learn the recommendation model, the user feedback on items is of the essence. Generally, there are two kinds of user feedback: *explicit feedback* and *implicit feedback*. Explicit feedback, such as 1 to 5 rating scales in Netflix, provides users with a mechanism to unequivocally express their preferences on items. On the other hand, implicit feedback does not directly reflect the interest of the user but it acts as a proxy for a user's interest, such as check-ins in Yelp and clicks in YouTube. Compared to explicit feedback, implicit feedback is found in abundance and easy to collect, where it does not need any extra input from the user.

2.2 Problem Setting

According to the types of user feedback that act as the supervised signal for the recommendation model, there are two main tasks in the recommendation research domain: *rating prediction* and *item ranking*. The rating prediction task always taking explicit feedback as input aims at predicting to what extent a user would like a given item. The prediction quality is typically measured by root mean square error (RMSE) or mean absolute error (MAE). On the other hand, the item ranking task, also known as top-K recommendation, directly aims at recommending the most valuable items for each user from all item candidates based on implicit feedback. The prediction quality is measured by some of the metrics in Section 2.4. Compared to the rating prediction task, the item ranking problem is more practical and challenging [31], which more accords with the real-world recommendation scenario. The detailed problem settings are introduced as follows.

2.2.1 Rating Prediction

The rating prediction task typically takes the user explicit feedback as input, i.e., users' ratings on items. This task has focused mainly on predicting the rating values for those

items that a user has deliberately chosen to rate. Formally, given the ratings in the dataset $\mathcal{D}_i = \{r_{i,j_1}, r_{i,j_2}, ..., r_{i,j_{|\mathcal{S}_i|}}\}$ for user i on items $j_1, j_2, ..., j_{|\mathcal{S}_i|}$, then the recommendation model is learned on the training set $\mathcal{S}_i$ ($\mathcal{S}_i \subset \mathcal{D}_i$ and $\mathcal{S}_i \neq \emptyset$) of each user, the recommendation model needs to make the prediction $\hat{r}_{i,j}$ for the item with a ground rating $r_{i,j} \in \mathcal{T}_i$ in the test set $\mathcal{T}_i$ ($\mathcal{S}_i \cup \mathcal{T}_i = \mathcal{D}_i$ and $\mathcal{S}_i \cap \mathcal{T}_i = \emptyset$). Then the recommendation performance is measured by the average prediction error (see Section 2.4) between $r_{i,j}$ and $\hat{r}_{i,j}$.

2.2.2 Item Ranking

The recommendation task considered in this thesis mainly works on the user implicit feedback, since it is easy to collect and is widely utilized in current recommender systems. For each user i, the user preference data is represented by a set that includes the items she has accessed, e.g., $\mathcal{D}_i = \{I_1, ..., I_j, ..., I_{|\mathcal{D}_i|}\}$, where I_j is an item index in the dataset. The item ranking (top-K recommendation) task is formulated as: given the training item set $\mathcal{S}_i$, and the non-empty test item set $\mathcal{T}_i$ (requiring that $\mathcal{S}_i \cup \mathcal{T}_i = \mathcal{D}_i$ and $\mathcal{S}_i \cap \mathcal{T}_i = \emptyset$) of user i, the model must recommend an ordered set of items $\mathcal{P}_i$ such that $|\mathcal{P}_i| \leq K$ and $\mathcal{P}_i \cap \mathcal{S}_i = \emptyset$. Then the recommendation quality is evaluated by a matching score (Section 2.4) between $\mathcal{T}_i$ and $\mathcal{P}_i$.

2.3 Classical Recommendation Methods

In the past twenty years, the research and industrial deployment of recommender systems are largely developed. More and more Internet products are embracing personalized recommendation services to better serve customers and potentially enhance revenue. During this development, a number of effective models have been proposed and some of them have built a firm foundation for the later evolution of recommender systems. Here, we briefly summarize the classical methods of aforementioned collaborative filtering and content-based filtering, which have a profound influence on later works.

2.3.1 Collaborative Filtering

Collaborative filtering (CF) is a class of methods that recommend items to users based on the preferences other users have expressed for those items. It can be further divided into *memory-based* and *model-based* categories. Memory-based methods, make preference

predictions based on the entire collection of previously rated items by the users. This approach utilizes user feedback data to compute the similarity between users or items, then the recommendation is made according to those similar users or items. Whereas model-based methods make use of the observed user feedback to learn a model, which is then used for user preference prediction.

Memory-based Method

Specifically, there are two kinds of memory-based methods, namely, user-based CF and item-based CF. Taking the user-based CF for example. Given a target user i, user-based CF first computes the similarity $w_{i,k}$ between user i and another user k by some similarity measurement such as cosine similarity. Then the predicted preference score of user i on item j (item j is not accessed by user i) is computed:

$$\hat{r}_{i,j} = \frac{\sum_k w_{i,k} \cdot r_{k,j}}{\sum_k w_{i,k}}\,, \tag{2.1}$$

where $r_{k,j}$ represents the true score made by user k on item j, and $\hat{r}_{i,j}$ represents the predicted score. Item-based CF has a similar procedure with the user-based CF except that item-based CF calculates the similarity between items. The item-based CF was successfully deployed in the early stage of Amazon's recommendation engine [32].

Model-based Method

Model-based methods discover the latent patterns that are able to reflect how user preference is generated. This kind of method often achieves better performance than memory-based methods by downgrading the effect of the data sparsity problem. Some of the most successful realizations of model-based methods are based on matrix factorization [8], where matrix factorization models the latent features underlying the interactions between users and items. Generally, matrix factorization models map both users and items to a joint latent space of dimensionality d. In the latent space, each user is represented by a latent factor vector $\mathbf{p}_i \in \mathbb{R}^d$, and the item is represented by $\mathbf{q}_j \in \mathbb{R}^d$. The elements of $\mathbf{p}_i$ measure the extent of interests the user i has regarding the latent factors, while $\mathbf{q}_j$ measures the extent to which the item j possesses these factors. The resulting inner product, $\mathbf{q}_j^\top \cdot \mathbf{p}_i$, models the interaction between user i and item j which is the overall user preference score

on the item:

$$\hat{r}_{i,j} = \mathbf{q}_j^\top \cdot \mathbf{p}_i \,. \tag{2.2}$$

Then the next question is how to learn the user and item latent factors. One typical way to is to minimize the regularized squared error on the set of known ratings [8]:

$$\min_{\mathbf{p}_i, \mathbf{q}_j} \sum_{(i,j) \in \mathcal{D}} (r_{i,j} - \mathbf{q}_j^\top \mathbf{p}_i)^2 + \lambda \left(||\mathbf{p}_i||^2 + ||\mathbf{q}_j||^2 \right), \tag{2.3}$$

where $\mathcal{D}$ is the set consisting of user-item pairs (i, j) for which $r_{i,j}$ is the observed rating or feedback in the training set, λ is the parameter that controls the extent of regularization.

2.3.2 Content-based Filtering

Content-based filtering focuses on the content information of users/items rather than neighboring users' opinions or interests. This category of methods tries to recommend items that have similar content with those the user has liked in the past. Explicit attributes or characteristics should be extracted from the content of an item, such as the textual description or other content sources like audio and image. Here, we introduce a representative example of content-based filtering.

Given a number of items with text descriptions, these descriptions of items shall be transformed into numerical features. For simplicity, here we adopt the term frequency–inverse document frequency (TF-IDF) method to organize the word importance in each item's description into a feature vector $\mathbf{x}_j \in \mathbb{R}^N$ where N is the number of all words in the corpus [33]. For each user i, we have a learnable weight vector $\mathbf{w}_i \in \mathbb{R}^N$ to measure the user taste on these words. Then the linear regression model for each user is built to predict the preference scores of user i on item j:

$$\min_{\mathbf{w}_i} \sum_{(i,j) \in \mathcal{D}} (r_{i,j} - \mathbf{w}_i^\top \mathbf{x}_j)^2 + \lambda \, ||\mathbf{w}_i||^2 \,, \tag{2.4}$$

where $\mathbf{w}_i^\top \mathbf{x}_j$ is the predicted user preference score of user i on item j.

2.4 Evaluation Metrics

The performance evaluation is an essential part of recommender systems. For different tasks, i.e., rating prediction and item ranking (top-K recommendation), different evaluation metrics are utilized.

2.4.1 Rating Prediction

Two widely used evaluation metrics for the rating prediction task is introduced as follows:

- *RMSE*, root mean square error, measures the differences between ratings predicted by a model and the value observed. RMSE tends to penalize large errors as the error term is squared.
- *MAE*, mean absolute error, is another common metric to evaluate the differences between the predicted rating and the ground-truth by measuring the absolute difference.

Formally, these metrics are defined as:

$$\begin{aligned} \text{RMSE} &= \sqrt{\frac{\sum_i \sum_{r_{i,j} \in \mathcal{T}_i} (r_{i,j} - \hat{r}_{i,j})^2}{\#\text{ of ratings}}}, \\ \text{MAE} &= \frac{\sum_i \sum_{r_{i,j} \in \mathcal{T}_i} |r_{i,j} - \hat{r}_{i,j}|}{\#\text{ of ratings}}, \end{aligned} \tag{2.5}$$

where $|\cdot|$ is the absolute value, $\mathcal{T}_i$ is the test set of user i.

2.4.2 Item Ranking

Here, we introduce four commonly used evaluation metrics to measure the item ranking performance of recommendation models. For each user in the dataset, the model recommends K items that the user has not interacted with before. These evaluation metrics compute the extent of matching between recommended items and ground-truth items. These evaluation metrics are:

- *Precision@K*, indicates what percentage of items among the top-K recommended items would be viewed by users.

- *Recall@K*, indicates what percentage of user's preferred items can emerge in the top-K recommended items.
- *MAP@K*, the mean average precision at K. Average precision is the average of precision values at all ranks (from 1 to K) where relevant items are found.
- *NDCG@K*, normalized discounted cumulative gain at K is a measure of item ranking quality. The discounted cumulative gain (DCG) is a weighted sum of the relevancy degree according to the ranked items. And the weight is a decreasing function of the rank (position) of the item, and therefore called discounted [34].

Formally, these metrics are formulated as:

$$
\begin{aligned}
\text{Precision@K} &= \frac{1}{M}\sum_{i=1}^{M}\text{pre}_i(K) = \frac{1}{M}\sum_{i=1}^{M}\frac{\mathcal{P}_i \cap \mathcal{T}_i}{K}\,, \\
\text{Recall@K} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\mathcal{P}_i \cap \mathcal{T}_i}{|\mathcal{T}_i|}\,, \\
\text{MAP@K} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{k=1}^{K}\text{pre}_i(k)\times\text{rel}_i(k)}{|\mathcal{T}_i|}, \\
\text{NDCG@K} &= \frac{1}{M}\sum_{i=1}^{M}\frac{\sum_{j\in\mathcal{T}_i}\frac{1}{\log_2\left(\text{rank}_i(j)+1\right)}}{\text{IDCG}\left(\min(K,|\mathcal{T}_i|)\right)}\,, \\
\text{IDCG}(K) &= \sum_{k=1}^{K}\frac{1}{\log_2(k+1)}\,,
\end{aligned}
\tag{2.6}
$$

where M is the number of users to evaluate, $\text{pre}_i(K)$ is the Precision@K for user i, $\text{rel}_i(k)$ is an indicator function that equals to 1 if the k-th recommended item is in the test set $\mathcal{T}_i$ otherwise equals to 0, and $\text{rank}_i(j)$ is the position of item j in the recommendation set $\mathcal{P}_i$.

Chapter 3

Related Work

In this chapter, we introduce the related work on (neural) recommender systems and the recommendation scenario with different auxiliary information.

3.1 General Personalized Recommendation

In the real-world, the user implicit feedback [35], such as users' clicking and check-in histories, widely exists in the Internet service. In many real-world recommendation scenarios, is more ubiquitous and common than explicit feedback [36] such as users' 5-star ratings. The implicit feedback setting, typically associated with the item ranking task, is also called one-class collaborative filtering (OCCF) [31]. It arises when only positive samples are available, and potentially positive samples and negative samples are mixed together. To address this general and challenging problem, many effective methods have been proposed.

3.1.1 Matrix Factorization-based Methods

Popularized by the Netflix Prize competition[1], matrix factorization (MF) based methods have become a prominent solution for the personalized recommendation [8]. In [20], Hu et al. propose a weighted regularized matrix factorization (WRMF) model to treat all the missing data as negative samples, while heuristically assigning confidence weights to positive samples. This insightful design also sheds light on further studies like [26, 27]. Rendle et al. adopt a different approach in [37], proposing a pair-wise ranking objective (Bayesian

[1] `https://www.netflixprize.com/`

Personalized Ranking) to model the pair-wise relationships between positive items and negative items for each user, where the negative samples are randomly sampled from the unobserved feedback. This ranking objective is both effective and efficient, which has been widely adopted in many of the later works like [38, 39]. Ning et al. [40] propose a sparse linear method—SLIM, which employs a sparse linear model in which the recommendation score for a new item can be calculated as an aggregation of other items. Kabbur et al. [41] present a factored item similarity-based method (FISM) for the top-K recommendation problem. FISM learns the item similarities as the product of two matrices, allowing it to generate high-quality recommendations even on sparse datasets. Furthermore, many MF methods also consider the problem from the probabilistic perspective. Johnson et al. in [42] propose a probabilistic model for matrix factorization—logistic matrix factorization, which is highly parallelizable. In [35, 43], Wang et al. and Li et al. apply the probabilistic matrix factorization [44] to learn the user preference from the implicit feedback, where the user and item latent factors are drawn from the Gaussian distribution. To allow unobserved items to have varying degrees of importance, He et al. in [45] propose to weight the missing data based on item popularity, demonstrating improved performance compared to WRMF.

3.1.2 Multi-layer Perceptron-based Methods

Due to their ability to learn more complex non-linear relationships between users and items, (deep) neural networks have been a great success in the domain of recommender systems. He et al. in [9] propose a neural network-based collaborative filtering model, where a multi-layer perceptron is used to learn the non-linear user-item interactions. This model is generic and many MF methods can be expressed and generalized under its framework. In [46], (denoising) autoencoders are employed to learn the user or item hidden representations from user implicit feedback. Autoencoder approaches can be shown to be generalizations of many of the MF methods [46]. The application of autoencoders in the recommendation model also inspires later works like [26, 27], which integrates a heuristic weighting function to measure the importance of each item. Furthermore, conventional matrix factorization and factorization machine methods benefit from the representation ability of deep neural networks for learning either the user-item relationships or the interactions with side information. In [47], Xue et al. propose an MF model with a neural network architecture. Through the neural network architecture, the model projects users and items

into low-dimensional vectors in a latent space. A customized loss function is designed to train the model, where both explicit and implicit feedback are considered. In [48, 49], Guo et al. and Lian et al. equip the factorization machine [21] with a neural network structure. As such, the model learns both high- and low-order feature interactions, as well as reduces the manual feature engineering burden. Due to the ability to naturally integrate node information and topological structure, graph neural networks (GNNs) are a great match to integrate the relationship data within the recommender system. In [50], the user-item interactions are represented as a bipartite graph, where the neighbors of a user are the items she preferred and the neighbors of an item are the users who rated it. In [39, 51], Sun et al. also model the user-user and item-item relations via GNNs along with the user-item interactions.

Recently, attention mechanism has demonstrated the effectiveness in various machine learning tasks such as image captioning [52, 53], document classification [54], and machine translation [55, 56]. Researchers also adopt the attention mechanism on recommendation tasks not only to capture the user-item interaction but also model the rich side information. In [57], Pei et al. adopt an attention scheme to learn the attention scores of user and item history in an interacting way, which is used to measure the dependencies between the user and item dynamics in shaping user-item interactions. Wang et al. [58] propose a hybrid attention model to incorporate both the model specialty factor and model timeline factor into the attention network to strategically assign attention given each specific article. This helps adaptively capture the change of editors' selection criteria. In [59], Gong et al. adopt an attention model to scan input microblogs and select trigger words. To incorporate a trigger word mechanism, they propose a novel attention-based convolutional neural network (CNN) architecture, which consists of a local attention channel and a global channel. Chen et al. [60] propose item- and component-level attention mechanisms to model the implicit feedback in the multimedia recommendation. In [61], Seo et al. propose to model user preferences and item properties using CNNs with dual local and global attention. Specifically, the local attention provides insight on a user's preferences or an item's properties, while the global attention helps CNNs focus on the semantic meaning of the whole review text. In [62], Tay et al. propose a multi-pointer attention mechanism to enhance the rating prediction accuracy, which operates with a gumbel-softmax based pointer mechanism that enables a differentiable neural architecture. This enables not only the most informative reviews to be utilized for prediction but also a deeper word-level interaction. In [26], Ma

et al. integrate the attention mechanism with the autoencoder to discriminate the user preferences on users' visited locations, which yields a fine-grained user preference learning. And in [63], Chen et al. propose an attention-based review pooling mechanism is proposed to select the important user reviews. In [64], Wang et al. propose a framework, namely KGAT, which explicitly models the high-order connectivities of the knowledge graph in an end-to-end fashion. The graph attention module [65] adaptively propagates the embeddings from a node's neighbors to update the node's representation.

3.1.3 Distance-based Methods

Due to their capacity to measure the distance between users and items, distance-based methods have been successfully applied in top-K recommendation. In [22], Hsieh et al. propose to compute the Euclidean distance between users and items. Not only users' preferences but also the user-user and item-item similarities are indirectly modeled for capturing fine-grained user preferences. In [66], Tay et al. adopt an attention-based memory-augmented neural architecture [67] that models the relationship between users and items in metric space using latent relation vectors. In [68], Zhou et al. propose a metric learning-based model that is augmented with the external memory and neural attention mechanism, where the memory network stores the embeddings of different types of user-item interactions. By doing this, it can capture the fine-grained user preference across various interaction spaces. Different from [66], Park et al. in [69] apply a translation embedding to capture more complex relations between users and items, where the translation embedding is learned from the neighborhood information of users and items. In [70], He et al. apply a distance metric to capture how the user interest shifts across sequential user-item interactions. This model has wide connections with existing methods and demonstrates its suitability for modeling third-order interactions between users, their previously consumed item, and their next item. In [71], Li et al. propose to measure the trilateral relationship from both the user-centric and item-centric perspectives and learn adaptive margins for the central user and positive item. Benefiting by learning user and item embeddings in the hyperbolic space, Tran et al. [72] explore the application of metric learning in the hyperbolic space for recommender systems, the model achieves better results than which in the Euclidean space.

3.2 Location-aware Recommendation

Recent years have witnessed the development and popularity of location-based services, such as Yelp and Foursquare. In these services, users are able to share their check-ins and opinions on Point-of-Interests (POIs), such as restaurants and shopping malls. The task of POI recommendation is to provide personalized location recommendations to different users. It plays an important role in providing better location-based services.

To make more accurate recommendations, researchers have incorporated geographical information (influence) of locations into their proposed models [73–78]. There are several ways to model the geographical influence. In particular, some researchers employ Gaussian distribution to characterize user's check-in activities. For example, Cho et al. [73] apply a two-state Gaussian mixture to model the check-ins that close to users' homes or workplaces. Cheng et al. [75] propose a multi-center discovering algorithm to detect user's check-in centers. Then Gaussian distribution is built on each center, calculating user check-in probabilities on unvisited locations together. On the other hand, researchers also adopt kernel density estimation (KDE) to estimate user's check-in activities. Ye et al. [74] discover that user's check-in behaviors are in a power-law distribution pattern. The power-law pattern reveals two locations' co-occurrence probability distribution over their distance, and this discovery is also employed in [76, 79]. Besides, Liu et al. [78] exploit geographical characteristics from location perspectives, which are modeled by two levels of neighborhoods, i.e., instance-level and region-level. Furthermore, in [80], Zhao et al. propose the geographically hierarchical pairwise ranking model, which follows an assumption: the user prefers the visited POI than the unvisited neighboring POI, and prefers the unvisited neighboring POI than the unvisited non-neighboring POI. In [81], Wang et al. model the geographical influence between two POIs using three factors: the geo-influence of POI, the geo-susceptibility of POI, and their physical distance. Geo-influence captures POI's capacity at exerting geographical influence to other POIs, and geo-susceptibility reflects POI's propensity of being geographically influenced by other POIs. In [82], Zhou et al. introduce the concepts of geographical preference and geographical influence which are specifically learned for each user and POI to depict the fine-grained geographical effect between them.

3.3 Content-aware Recommendation

Content information can be one of the most important information sources to understand user preferences. Due to the privacy issue, the user-specific data, e.g., age, gender, and occupation, is hard to access. In contrast, a plethora of item information is publicly available online, such as movie plots on Netflix, which makes collecting item metadata have less privacy concern. Therefore, making use of auxiliary item information is a feasible way to improve recommendation performance.

Researchers incorporate the content features to help alleviate the sparseness and the cold-start problem in the user-item interaction data. In some early works, McAuley et al. [83] and Wang et al. [84] apply Latent Dirichlet Allocation (LDA) [85] to learn abstract topics that occur in a collection of documents, which not only provides an interpretable latent structure for users and items, but also form recommendations about both existing and cold-start items. In recent years, deep learning models have demonstrated great power for effective text representation learning. In [35, 86], Wang et al. and Zhang et al. utilize the stacked denoising autoencoder (SDAE) on items' bag-of-words to learn the item latent representations, which allows a robust item content representation and tight coupling of deep representation learning for the content information and collaborative filtering for the ratings (feedback) matrix, respectively. Li et al. in [87] propose to combine probabilistic matrix factorization with marginalized denoising autoencoders. And in [43], Li et al. adopted a variational autoencoder to learn the latent representations from items' content, which is a Bayesian probabilistic generative model. It infers the stochastic distribution of the latent variable under the content representation through an inference network, instead of point estimates, and it leads to more robust performance. On the other hand, some studies also incorporate contextual information for a better understanding of the text. For example, in [88], the doc2vec model [89] is utilized to model the text information in user reviews for the review embedding learning. The review text embedding is then integrated with the item embedding from ratings by a neural network. And in [61, 90, 91], Kim et al., Seo et al., and Zheng et al. adopt convolution neural networks (CNNs), with max-pooling and fully connected layers, to learn the item hidden representation from item's sequence of word embeddings, where words' contextual information can be captured by the convolutional filters and the sliding window strategy. In [63], Chen et al. apply an attention model among the item content embeddings learned by the CNN to further distinguish the

importance of the user reviews. In [92], Zhou et al. present a two-headed attention fusion autoencoder model that leverages both user-generated reviews and implicit feedback to make recommendations.

3.4 Sequential Recommendation

Accurately characterizing and learning users' interests lies at the core of a personalized recommender system. In real-world Internet services, users interact with items or products chronologically and users' interests are intrinsically dynamic and evolving, which are always influenced by their historical behaviors. Typically, the user interest would be modeled separately as long-term interests and short-term interests. Thus, how to effectively model these two parts are significant in the sequential recommendation.

Some early sequential recommendation methods rely on item-item transition matrices to capture the sequential patterns in the user interaction sequence. The Markov chain [93] is a classical option to solve this problem. For example, Rendle et al. [94] propose to factorize personalized Markov chains for capturing long-term preferences and short-term transitions. He et al. [95] combines similarity-based models with high-order Markov chains to make personalized sequential recommendations. In [70], the translation-based method is proposed for sequential recommendation. Recently, benefited by the advantages of sequence learning in natural language processing, (deep) neural network-based methods are proposed to learn the sequential dynamics. For instance, Tang et al. [96] propose to apply the convolutional neural network (CNN) on item embedding sequence, where the short-term contexts can be captured by the convolutional operations. In [97–100], recurrent neural network (RNN), especially gated recurrent unit (GRU), based methods are utilized to model the sequential patterns for the session-based recommendation [97], where the hidden states of RNNs reflect the summary of the (sub)sequence. On the other hand, self-attention [56] exhibits promising performance in sequence learning and is utilized in the sequential recommendation. In [101], Kang et al. propose to leverage the self-attention for the next item prediction, which allows to capture long-term semantics, as well as makes the prediction according to a few items by using the attention model. In [102, 103], Chen et al. and Huang et al. propose to adopt memory networks [67] to memorize the important items that will play a role in predicting future user actions. And in [38], Ma et al. not only utilize the memory network to store the long-term user interest but also capture the short-term interests of users by a graph

neural network. In [104], Yu et al. propose to unify both individual- and union-level item interaction into preference inference model from multiple views. With the help of the attention mechanism, the model can obtain a unified embedding to keep the individual-level interactions with a linear combination of mapped items' features. In [105], Sun et al. propose a sequential recommendation model that employs the deep bidirectional self-attention to model user behavior sequences. This is achieved by predicting the random masked items in the sequence by jointly conditioning on their left and right context.

Chapter 4

Neural Networks for Point-of-Interest Recommendation: Capturing Complex User-Item Interactions

4.1 Introduction

The rapid growth of mobile devices and location-acquisition technologies enable the convenient access of people's real-time location information. This advancement empowers the coming of Location-based Social Networks (LBSNs), such as Yelp[1] and Foursquare[2]. In these LBSNs, users are allowed to communicate with each other, post physical positions, and share experiences associated with a location—Point-of-Interest (POI). The large number of user-POI interactions facilitates a practical and promising service—personalized POI recommendation. POI recommender systems satisfy a potentially massive need for services and offer major advantages to at least two parties: (1) helping locals or travelers to discover exciting unvisited places; (2) providing possibilities for attracting more visitors for POIs.

In the literature, effective personalized POI recommendation approaches have been proposed. These methods focus largely on collaborative filtering (CF), which can be divided into memory-based and model-based methods [106]. Memory-based methods gather a user's preference with respect to unvisited POIs according to the weighted average of ratings or preferences from similar users or POIs. For example, by considering the similarities

[1] `https://www.yelp.com/`
[2] `https://foursquare.com/`

between a user and his/her friends, [74] and [107] apply friend-based CFs to predict the user preference on unvisited POIs. On the other hand, model-based approaches make use of the user-POI history to learn a model for the recommendation. Stimulated by the Netflix Prize contest[3], some of the most common realizations of model-based approaches are based on matrix factorization (MF) [8]. The latent characteristics underlying relationships between users and POIs are discovered by MF, which predicts the user preference by the inner product of user and item latent factors. For example, a weighted regularized MF was adopted by [108], [78], and [109] to infer the user preference for unvisited POIs.

However, in large-scale data, the aforementioned techniques may not fully leverage the complex user-POI interactions. The aforementioned memory- and MF-based methods model the user preference by the weighted average of ratings or the inner product of latent factors, where the scoring functions are linear models. Furthermore, it has been shown in [9, 22] that how the inner product combines latent features linearly and limits the effectiveness and expressiveness of MF methods.

Recently, autoencoders (AEs) have been a great success in the field of recommendation research community because of the ability to represent non-linear and complex data, and provide more opportunities to reshape the conventional recommendation architectures [35, 46, 87]. Motivated by this, to cope with the complex user-POI check-in data, we propose an autoencoder-based model. The main reason why we adopt the stacked AE is that the stacked AE can effectively capture the complex relationships between users and POIs with the deep neural network structure and the non-linear activation functions. These allow richer data representations in the latent space. Our early empirical experiments also justify that equipping AE with the same weighted loss function can achieve better performance than the weighted regularized MF [20]. Furthermore, AE has strong connections with multiple MF methods [46]. As such, it is straightforward to utilize AE for modeling the user feedback data.

Nevertheless, the application of AE in the POI recommendation is a non-trivial task and several important factors should be taken into account. First, we argue that certain POIs are more representative than others in the user check-in records to reflect the preferences of users. Treating these representative POIs equally with other POIs can lead to an inaccurate understanding of the preferences of users. Thus, distinguishing the user preference degrees on checked-in POIs is important for better learning the user preference. Second, a unique

[3] https://www.netflixprize.com/

property in the check-in records is the spatial context of POIs, which is critical for grasping the user mobility patterns and improving the performance of recommendations. Therefore, how to incorporate geographical information into the neural network-based method is worthy of exploration. Third, check-in data is a kind of implicit feedback, which suggests that there are only positive samples in the data records [31]. Furthermore, users can only visit a small number of candidates from millions of POIs, which makes the user-POI check-in data extremely sparse. Thus, a challenge is how to capture the preference of users from the sparse implicit feedback.

To resolve the aforementioned challenges, a novel autoencoder-based model is proposed, namely *SAE-NAD*, with two proposed components: a self-attentive encoder (SAE) and a neighbor-aware decoder (NAD). First, unlike existing approaches that do not thoroughly investigate the implicitness of the user preference, we adopt the self-attention encoder to adaptively quantify an importance vector for each POI in a user's check-in records, which reveals the user preference from multiple perspectives. As such, it is possible to further distinguish users' preferences on checked-in POIs. The POIs with higher importance values can lead to more contribution for the hidden representation of the user, which will make the hidden representation of the user more personalized. Second, the neighbor-aware decoder is proposed to combine the geographical influence effect [74, 75], which is widespread in the action of human mobility on LBSNs. In order to quantify the effect of checked-in POIs applied to unvisited POIs, we leverage the inner product between the embedding of checked-in and unvisited POIs, along with the radial basis function (RBF) kernel (based on the pairwise distance of the corresponding POIs). By doing this, the user reachability on the adjacent and similar neighbors of checked-in POIs will be higher than the distant ones, since the user has a larger chance to visit these POIs. Third, we assign the same small weights to all unvisited POIs to model the sparse implicit feedback and assign larger weights to visited POIs according to each user's visit frequency, which allows a difference between unvisited POIs, less-visited POIs, and frequently-visited POIs. Compared with several state-of-the-art approaches and measured with validation metrics on three real-world datasets, our model is evaluated thoroughly. The experimental results demonstrate our model's advances over other state-of-the-art POI recommendation models. The highlights of this chapter are summarized as follows:

- We propose a self-attention encoder to adaptively compute an important vector for each

checked-in POI to differentiate the user preference for checked-in POIs, and to make the POI contribute to the user hidden representation according to the values of significance.

- We propose a neighbor-aware decoder to integrate the geographical influence effect, which adopts the inner product between the embedding of checked-in POIs and unvisited ones, along with the POI-POI relationships determined by the RBF kernel, to model the influence checked-in POIs exerted on unvisited ones.
- The proposed model achieves the best performance on three real-world datasets comparing to the state-of-the-art methods, exhibiting the superiority of our model.

4.2 Preliminaries

In this section, we first introduce the definitions and notations. Then we review the basic ideas of autoencoders.

4.2.1 Definition and Notation

For ease of illustration, we first summarize the definitions and notations in this chapter.

Definition 1. (**POI**) A POI is defined as a uniquely identified site (e.g., a restaurant or a shopping mall) with two attributes: an identifier and geographical coordinates (latitude and longitude).

Definition 2. (**Check-in**) A check-in is a record demonstrating a user has visited a POI at a certain time. Hence a user's check-in is represented by a 3-tuple: user ID, POI ID, and the timestamp.

Definition 3. (**POI Recommendation**) Given users' check-in records, the POI recommendation aims at recommending a list of POIs for each user that the user is interested in but never visited.

POI recommendation is commonly studied on a user-POI check-in matrix $\mathbf{R} \in \mathbb{R}^{M \times N}$, where there are M users and N locations, and each entry $r_{u,i}$ represents the frequency user u checked-in location i. We denote the binary rating matrix as $\mathbf{X} \in \mathbb{R}^{M \times N}$, where each entry $x_{u,i} \in \{0, 1\}$ indicates whether user u has visited location i. The terms POI and location are used interchangeably in this chapter. Here, following common symbolic notation, upper case bold letters denote matrices, lower case bold letters denote column

Table 4.1 List of notations.

M, N	the number of users and POIs
$\mathbf{X}$, $\hat{\mathbf{X}}$	the input data and reconstructed data
$\mathbf{R}$	the check-in frequency matrix
$\mathbf{C}$	the confidence matrix
$\mathbf{W}^*$, $\mathbf{b}^*$	the weight matrix and bias vector
a_*	the activation function
H	the dimension of the bottleneck layer
d_a	the dimension of the importance vector
γ	the parameter to control POI's correlation level
α, ϵ	the parameters of the weighting scheme
λ	the regularization term

vectors without any specification, and non-bold letters represent scalars. The notations are shown in Table 4.1.

4.2.2 Autoencoders

A single hidden-layer autoencoder (AE) is an unsupervised neural network, which is composed of two parts, i.e., an encoder and a decoder. The encoder has one activation function that maps the input data to the latent space. The decoder also has one activation function mapping the representations from the latent space to the reconstruction space. Given the input $\mathbf{x}_i$, a single hidden-layer autoencoder is shown as follows:

$$\begin{aligned} encoder &: \mathbf{z}_i = a_1(\mathbf{W}^{(1)}\mathbf{x}_i + \mathbf{b}^{(1)}), \\ decoder &: \hat{\mathbf{x}}_i = a_2(\mathbf{W}^{(2)}\mathbf{z}_i + \mathbf{b}^{(2)}), \end{aligned} \tag{4.1}$$

where $\mathbf{W}^*$, $\mathbf{b}^*$, and a_* denote the weight matrices, bias vectors, and activation functions, respectively. $\hat{\mathbf{x}}_i$ is the reconstructed version of $\mathbf{x}_i$. The output $\mathbf{z}_i$ of the encoder is the representation of $\mathbf{x}_i$ in the latent space. The goal of the autoencoder is to minimize the reconstruction error of the output and the input. The loss function is shown as follows:

$$\mathcal{L}_{AE} = \sum_{i=1}^{M} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||_2^2 . \tag{4.2}$$

Relations to word2vec. word2vec [110] is an effective and scalable method to

learn embedding representations by modeling words' contextual correlations in sentences. word2vec utilizes either of two architectures to produce distributed representations of words: continuous bag-of-words (CBOW) or continuous skip-gram. Taking continuous skip-gram for example, the input of this model is a one-hot vector to represent the current word, then the model uses the current word to predict the surrounding window of context words. This model is highly similar to AE when the input of AE is a one-hot vector. If the current word is i and target word is j, we set the activation function to identity and bias to zero, then the output of the decoder is:

$$\hat{x}_{ij} = \mathbf{W}_{j,*}^{(2)\top} \cdot \mathbf{W}_{*,i}^{(1)}, \tag{4.3}$$

where $\mathbf{W}_{*,i}^{(1)}$ and $\mathbf{W}_{j,*}^{(2)}$ are the i-th column and j-th row of $\mathbf{W}_1$ and $\mathbf{W}_2$, respectively. We further apply $softmax$ on the output of the decoder:

$$P(l_j|l_i) = \frac{\exp(\hat{x}_{ij})}{\sum_k \exp(\hat{x}_{ik})}, \tag{4.4}$$

where this probability shows how likely the word j will appear in the window of the current word i. The combination of Eqs. 4.3 and 4.4 is similar to the Eq. 2 in [110]. In our POI recommendation setting, this formula demonstrates if a user has checked-in location l_i, how likely the user would check-in location l_j. Therefore, the inner product of $\mathbf{W}_{*,i}^{(1)}$ and $\mathbf{W}_{j,*}^{(2)}$ can be used for capturing the relation between l_i and l_j in a single hidden-layer AE.

4.3 Methodologies

In this section, we introduce the proposed model for POI recommendation, which consists of two components, i.e., a self-attentive encoder and a neighbor-aware decoder, demonstrating in Figure 4.1. We first present the stacked autoencoder as our major building block. Then we illustrate the self-attentive encoder to adaptively select representative POIs that can reflect users' preferences. Next, we demonstrate the neighbor-aware decoder to model the geographical influence in POI recommendation, which is a phenomenon that users tend to check-in those unvisited POIs that close to a POI they checked-in before. Lastly, we present the loss function for implicit feedback and how to optimize the proposed model.

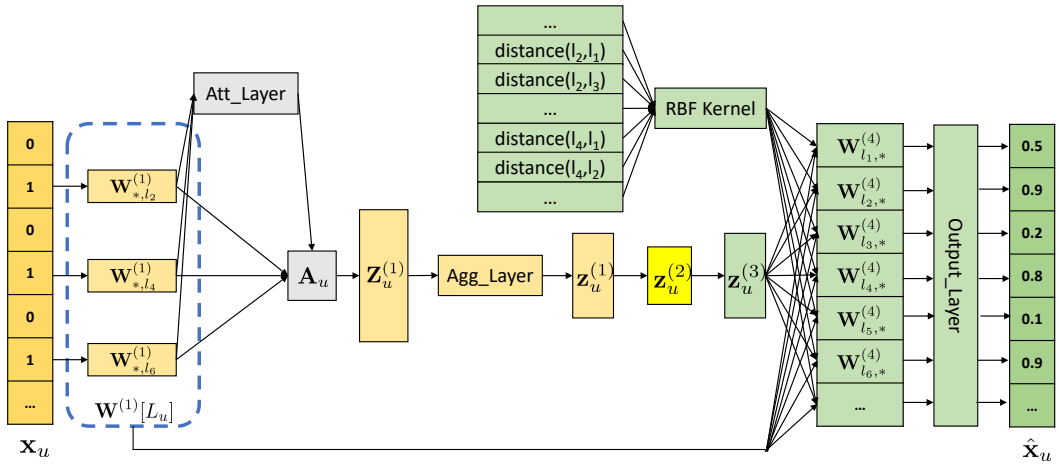


Figure 4.1 The model architecture of SAE-NAD. The yellow part is the self-attentive encoder, the green part is the neighbor-aware decoder, and the gray part is the attention network. The bright yellow rectangle is the user hidden representation. Specifically, *Att_Layer* denotes the attention layer and *Agg_Layer* denotes the aggregation layer.

4.3.1 Model Basics

To learn the user hidden representation and reconstruct user preferences on unvisited POIs, we propose to adopt a stacked autoencoder, where the deep network architecture and non-linear activation functions may capture the complex user-POI interactions [9]. Formally, the stacked autoencoder is shown as follows:

$$\text{encoder}: \begin{cases} \mathbf{z}_u^{(1)} = a_1(\mathbf{W}^{(1)}\mathbf{x}_u + \mathbf{b}^{(1)}) \\ \mathbf{z}_u^{(2)} = a_2(\mathbf{W}^{(2)}\mathbf{z}_u^{(1)} + \mathbf{b}^{(2)}) \end{cases} \quad \text{decoder}: \begin{cases} \mathbf{z}_u^{(3)} = a_3(\mathbf{W}^{(3)}\mathbf{z}_u^{(2)} + \mathbf{b}^{(3)}) \\ \hat{\mathbf{x}}_u = a_4(\mathbf{W}^{(4)}\mathbf{z}_u^{(3)} + \mathbf{b}^{(4)}) \end{cases} \tag{4.5}$$

where $\mathbf{W}^{(1)} \in \mathbb{R}^{H_1 \times N}$, $\mathbf{W}^{(2)} \in \mathbb{R}^{H \times H_1}$, $\mathbf{W}^{(3)} \in \mathbb{R}^{H_1 \times H}$, and $\mathbf{W}^{(4)} \in \mathbb{R}^{N \times H_1}$ are learnable parameter matrices of the stacked AE. H_1 is the dimension of the first hidden layer, and H is the dimension of the bottleneck layer. $\mathbf{z}_u^{(2)}$ and $\hat{\mathbf{x}}_u$ are the hidden representation and reconstructed ratings of user u, respectively.

4.3.2 Self-Attentive Encoder

As presented in Section 4.3.1, we apply a stacked AE to learn users' hidden representations. In the proposed model, the input is a multi-hot user preference vector $\mathbf{x}_u \in \mathbb{R}^N$, where 1 in the vector indicates the user has been to a certain POI. Based on the input, the encoder of a vanilla stacked AE works as follows: (1) Given a user's check-in set $L_u = \{l_1, ..., l_n\}$, where l_n is the index of a POI, corresponding POI vectors (e.g., $\mathbf{W}^{(1)}_{*,l_n}$) in $\mathbf{W}^{(1)}$ are selected and summed; (2) After having the summed vector, performing the activation function to get the user hidden representation. Here, $\mathbf{W}^{(1)}$ works like a POI embedding matrix, which is similar to the word embedding matrix in the word2vec [110] model.

Since the model input is a multi-hot vector, it makes each embedding in $\mathbf{W}^{(1)}[L_u]$ equally contribute to the user hidden representation, and $[\cdot]$ is the slicing operation that selects corresponding POI vectors to form an H_1-by-n sub-matrix:

$$\mathbf{W}^{(1)}[L_u] = (\mathbf{W}^{(1)}_{*,l_1}, \mathbf{W}^{(1)}_{*,l_2}, ..., \mathbf{W}^{(1)}_{*,l_n}), \tag{4.6}$$

where $\mathbf{W}^{(1)}_{*,l_n}$ is the l_n-th column of $\mathbf{W}^{(1)}$.

However, in the user check-in history, there are some POIs that would be more representative than others that can directly reflect a user's preferences. These representative POIs should contribute more to the user hidden representation for expressing the user preference. This inspires us to apply a self-attentive mechanism, which learns a weighted sum of embeddings in $\mathbf{W}^{(1)}[L_u]$ to form the user's hidden representation.

The goal of the self-attentive encoder is to adaptively assign different importances on checked-in POIs for expressing various preference levels of users. Then the embeddings of checked-in POIs are aggregated in a weighted manner to characterize users. Given checked-in POI embeddings $\mathbf{W}^{(1)}[L_u]$ of user u, we use a single-layer network without bias to compute the importance score (attention score):

$$\mathbf{a}_u = \text{softmax}\big(\tanh(\mathbf{w}_a^\top \mathbf{W}^{(1)}[L_u])\big)\,, \tag{4.7}$$

where $\mathbf{w}_a \in \mathbb{R}^{H_1}$ is the parameter in the attention layer, the $softmax$ ensures all the computed weights sum up to 1. Then we sum up the embeddings in $\mathbf{W}^{(1)}[L_u]$ according to

the importance score provided by $\mathbf{a}_u$ to get a vector representation of the user:

$$\mathbf{z}_u^{(1)} = \sum_{l_j \in L_u} a_{u,j} \mathbf{W}_{*,l_j}^{(1)} . \tag{4.8}$$

However, the standard attention mechanism that assigning a single importance value to a POI makes the model only focus on one specific aspect of POIs [111], which is not sufficient to reflect the sophisticated human sentiment on POIs. Taking a restaurant for example. From the perspective of food flavor, a user likes this restaurant; from the perspective of the eating environment, the user may think the restaurant is not good enough. Thus, to capture the user preference from different aspects, we may need to perform multiple times of Eq. 4.7 with different sets of parameters.

Therefore, we adopt an importance score matrix to capture the effects of multiple-dimensional attention [56] on POIs. Each dimension of the importance scores represents the importance levels of checked-in POIs in a certain aspect. Suppose we want d_a aspects of attention to be extracted from the embeddings, then we can extend $\mathbf{w}_a$ to $\mathbf{W}_a \in \mathbb{R}^{d_a \times H_1}$:

$$\mathbf{A}_u = \mathrm{softmax}\big(\tanh\big(\mathbf{W}_a \mathbf{W}^{(1)}[L_u]\big)\big) , \tag{4.9}$$

where $\mathbf{A}_u \in \mathbb{R}^{d_a \times n}$ is the importance score matrix, each column of $\mathbf{A}_u$ is an importance vector of a specific POI, and each row of $\mathbf{A}_u$ depicts the importance levels of n checked-in POIs in a certain aspect. The $softmax$ is performed along the second dimension of its input. By multiplying the importance score matrix with the POI embeddings, we have:

$$\mathbf{Z}_u^{(1)} = \mathbf{A}_u \cdot (\mathbf{W}^{(1)}[L_u])^\top , \tag{4.10}$$

where $\mathbf{Z}_u^{(1)} \in \mathbb{R}^{d_a \times H_1}$ is the matrix representation of user u, which depicts the user from d_a aspects. To make the matrix representations of users fit our encoder, we have one more neural layer to aggregate users' representations from different aspects into one aspect. Then the vector representation of user u is shown:

$$\mathbf{z}_u^{(1)} = a_t(\mathbf{Z}_u^{(1)\top} \mathbf{w}_t + \mathbf{b}_t) , \tag{4.11}$$

where $\mathbf{w}_t \in \mathbb{R}^{d_a}$ is the parameter in the aggregation layer, a_t is the activation function.

4.3.3 Neighbor-Aware Decoder

In LBSNs, there is the physical distance between users and POIs, which makes the POI recommendation distinct from other recommendation tasks. In a user's check-in history, the user's occurrences are typically constrained in several certain areas. This is the well-known geographical clustering phenomenon (a.k.a geographical influence) in users' check-in activities, which has been exploited to largely improve the POI recommendation performance [74, 75, 78, 79, 108, 109]. Most of the previous studies mainly exploit geographical influence from a user's perspective: learning the geographical distribution of each user's check-ins [74, 75, 109] or by the inner product between the user latent factors and the latent factors of a certain POI's neighbors [79, 108]. Whereas the proposed neighbor-aware influence model captures the geographical influence solely from the perspective of POIs.

According to the aforementioned geographical influence, one intuition contributes to this phenomenon: users have a higher chance to check-in POIs surrounded by a POI that they visited before. From this intuition, a checked-in POI may have impacts on other unvisited POIs, and the impact level is determined by the properties and distance between the POI pairs. Inspired by the skip-gram model of word2vec, which applies the inner product to predict the context words given an input word, we also leverage similar techniques to model the influence a checked-in POI exerted on unvisited POIs (Section 4.2.2, relations to word2vec). The proposed technique can discover unvisited POIs that may be similar to the visited ones. Similarly, we treat $\mathbf{W}^{(1)}$ as the POI embedding matrix (the first weight matrix in word2vec) and $\mathbf{W}^{(4)}$ as the context POI embedding matrix (the second weight matrix in word2vec). It is also important to note that the proposed method is also similar to FISM [41], where FISM adopts two matrices of item latent factors to model the similarity between items.

Formally, given a user's check-in set $L_u = \{l_1, ..., l_n\}$, the influence checked-in POIs exerted on unvisited POIs is shown:

$$\mathbf{P}_u = \mathbf{W}^{(4)} \cdot \mathbf{W}^{(1)}[L_u]\,, \tag{4.12}$$

where $\mathbf{P}_u \in \mathbb{R}^{N \times n}$. Each column of $\mathbf{P}_u$ is the influence a certain checked-in POI applied on all other POIs (the influence on itself is set to 0).

The above inner product gives a basic indication about how related two POIs are, however, it does not explicitly take the distance between two POIs into account. According

to Tobler's First Law of Geography, everything is related to everything else, but near things are more related than distant things. To incorporate the geographical distance property, we adopt the Gaussian radial basis function kernel (RBF kernel) to further make checked-in POIs exert more influence on nearby unvisited POIs. The RBF kernel is shown as follows:

$$K(l_i, l_j) = \exp(-\gamma||\mathbf{l}_i - \mathbf{l}_j||^2)\,, \tag{4.13}$$

where $\mathbf{l}_i$ and $\mathbf{l}_j$ are the geographical coordinates of two POIs l_i and l_j. $\gamma > 0$ is a hyper-parameter to control the geographical correlation level of two given POIs, a larger value of γ will lead to a larger $K(l_i, l_j)$. The value range of RBF kernel is $K(l_i, l_j) \in [0, 1]$. For computation simplicity, if the value of $K(l_i, l_j)$ is less than 0.1, we set it to 0. We can pre-compute the pairwise RBF value of each POI pair to get a RBF value matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, where the diagonal is set to 0.

By incorporating the RBF kernel effect, our neighbor-aware influence model is shown:

$$\mathbf{P}_u = (\mathbf{W}^{(4)} \cdot \mathbf{W}^{(1)}[L_u]) \odot \mathbf{K}[L_u]\,, \tag{4.14}$$

where $\mathbf{K}[L_u] \in \mathbb{R}^{N \times n}$ is the RBF kernel value from Eq. 4.13, $\odot$ is the element-wise multiplication.

To obtain the accumulated influence from all checked-in POIs, we sum along the row of $\mathbf{P}_u \in \mathbb{R}^{N \times n}$ to get $\mathbf{p}_u \in \mathbb{R}^N$:

$$\mathbf{p}_u = \sum_{j=1}^{n} \mathbf{P}_u^{(i,j)}, i = 1, 2, ..., N, \tag{4.15}$$

where i and j are the row and column index, respectively.

To incorporate the neighbor-aware influence, the decoder of the proposed model can be rewritten as:

$$\hat{\mathbf{x}}_u = a_4(\mathbf{W}^{(4)}\mathbf{z}_u^{(3)} + \mathbf{p}_u + \mathbf{b}^{(4)}), \tag{4.16}$$

where $\mathbf{W}^{(4)}\mathbf{z}_u^{(3)}$ captures the user preference, $\mathbf{p}_u$ models the neighbor-aware geographical influence.

Discussion. As we mentioned before, the way we adopt the inner product to capture the relations between POIs is similar to FISM [41], if we treat $\mathbf{W}^{(1)}$ as $\mathbf{P}$ and $\mathbf{W}^{(4)}$ as $\mathbf{Q}$ in FISM. In FISM, the predicted rating of user u on item i is mainly estimated by

$\sum_{j \in \mathcal{R}_u^+} \mathbf{p}_j \mathbf{q}_i^\top$, where $\mathcal{R}_u^+$ is the set of items rated by user u, $\mathbf{p}_j$ and $\mathbf{q}_i$ are learned item latent factors from $\mathbf{P}$ and $\mathbf{Q}$, respectively.

4.3.4 Weighted Loss for Implicit Feedback

In the POI recommendation, check-in data is treated as implicit feedback. Since a user's check-in records only include the locations she visited, and the visit frequency may indicate the confidence level of her preference. Therefore, there are only positive examples observed in the check-in records, which makes POI recommendation a One-Class Collaborative Filtering (OCCF) problem [20, 31].

To tackle the OCCF problem and capture user preferences from check-in data, we adopt a general weighting scheme [20] to distinguish visited and unvisited POIs. Specifically, we consider all unvisited locations as negative examples and assign the weights of all negative examples to the same value, e.g., 1. As for visited locations, the weights are increased monotonically with users' check-in frequencies. With such a weighting scheme, our model not only distinguishes visited and unvisited POIs, but also discriminates the confidence levels of all visited POIs. The objective function for implicit feedback is presented as follows:

$$\mathcal{L}_{WAE} = \sum_{u=1}^{M} \sum_{i=1}^{N} ||c_{u,i} \left(x_{u,i} - \hat{x}_{u,i}\right)||_2^2 = ||\mathbf{C} \odot (\mathbf{X} - \hat{\mathbf{X}})||_F^2, \tag{4.17}$$

where $\odot$ is the element-wise multiplication of matrices. $||\cdot||_F$ is the Frobenius norm of matrices. In particular, we set the confidence matrix $\mathbf{C} \in \mathbb{R}^{M \times N}$ as follows:

$$c_{u,i} = \begin{cases} 1 + \alpha \log(1 + r_{u,i}/\epsilon) & \text{if } r_{u,i} > 0 \\ 1 & \text{otherwise} \end{cases} \tag{4.18}$$

where α and ϵ are hyper-parameters. This setting exactly encodes the observation that the frequency is a confidence of user preferences. This weighted loss with a vanilla autoencoder can be used in other recommendation tasks that take implicit feedback as input.

4.3.5 Network Training

By combining regularization terms, the objective function of the proposed model is shown as follows:

$$\mathcal{L} = ||\mathbf{C} \odot (\mathbf{X} - \hat{\mathbf{X}})||_F^2 + \lambda(||\mathbf{W}^*||_F^2 + ||\mathbf{W}_a||_F^2 + ||\mathbf{w}_t||_2^2), \tag{4.19}$$

where λ is the regularization parameter, $\mathbf{W}^*$ includes $\mathbf{W}^{(1)}$, $\mathbf{W}^{(2)}$, $\mathbf{W}^{(3)}$, and $\mathbf{W}^{(4)}$. $\mathbf{W}_a$ and $\mathbf{w}_t$ are the learned parameters in the attention layer and aggregation layer, respectively. By minimizing the objective function, the partial derivatives with respect to all the parameters can be computed by gradient descent with back-propagation. And we apply Adam [112] to automatically adapt the learning rate during the learning procedure. The mini-batch training algorithm is shown in Algorithm 1.

Algorithm 1: Training Algorithm

Input: $\mathbf{X}$, $\mathbf{C}$;
Initialize parameters $\mathbf{W}^*$, $\mathbf{W}_a$, $\mathbf{w}_t$, $\mathbf{b}^*$, $\mathbf{b}_t$;
numBatches = *M / batchSize* ;
while *iter < numIterations* **do**
    Shuffle($\mathbf{X}, \mathbf{C}$) ;
    for *batchID = 0; batchID < numBatches; batchID++* **do**
        $\mathbf{X}_{batch}, \mathbf{C}_{batch}$ = ExtractBatchData(batchID, $\mathbf{X}, \mathbf{C}$) ;
        Apply Eq. 4.6 to get $\mathbf{W}^{(1)}[L_u]$ for each user u in $\mathbf{X}_{batch}$;
        Apply Eq. 4.9, Eq. 4.10, and Eq. 4.11 to get $\mathbf{z}_u^{(1)}$;
        Apply Eq. 4.14 and Eq. 4.15 to get $\mathbf{p}_u$;
        Apply Eq. 4.5 and Eq. 4.16 to get $\hat{\mathbf{X}}_{batch}$;
        Apply Eq. 4.19 to obtain $\mathcal{L}_{batch}$ and back-propagate the error through the entire network ;
    end
end

Recommendation. At the prediction phase, the proposed model takes each user's binary rating vector $\mathbf{x}_u$ as input and obtains the reconstructed rating vector $\hat{\mathbf{x}}_u$ as output. Then the POIs that are not in the training set and have the largest prediction scores in $\hat{\mathbf{x}}_u$ are recommended to the user.

4.4 Experiments

In this section, we evaluate the proposed model with state-of-the-art methods on three real-world datasets.

4.4.1 Datasets

We evaluate the proposed model on three real-world datasets: Gowalla [73], Foursquare [113], and Yelp [113]. The Gowalla dataset was generated worldwide from February 2009 to October 2010. The Foursquare dataset comprised check-ins from April 2012 to September 2013 within the United States (except Alaska and Hawaii). The Yelp dataset was obtained from the Yelp dataset challenge round 7. Each check-in record in the above datasets includes a timestamp, a user ID, a POI ID, and the latitude and longitude of this POI.

To filter noisy data, for the Gowalla dataset, we remove users whose total check-ins are less than 20 and POIs visited less than 20 times; for the Foursquare and Yelp datasets, we eliminate those users with fewer than 10 check-in POIs, as well as those POIs with fewer than 10 visitors. The data statistics after preprocessing are shown in Table 4.2. For each user, we randomly select 20% of her visiting locations as ground truth for testing and 10% of which as the validation set. The remaining constitutes the training set. Similar data partition methods have been widely used in previous works [74, 108, 114] to validate the performance of POI recommendation. The random selection is carried out six times independently, we tune the model on one partition and report the average results on the rest five partitions.

Table 4.2 The statistics of datasets.

Dataset	#Users	#POIs	#Check-ins	Density
Gowalla	43,074	46,234	1,720,082	0.0500%
Foursquare	24,941	28,593	1,196,248	0.1006%
Yelp	30,887	18,995	860,888	0.1399%

4.4.2 Evaluation Metrics

We evaluate our model versus other models in terms of Precision@k (P@k), Recall@k (R@k), and MAP@k. For each user, Precision@k indicates what percentage of locations among the top k recommended POIs has been visited by her, while Recall@k indicates what

percentage of her visiting locations can emerge in the top k recommended POIs. MAP@k is the mean average precision at k. Average precision is the average of precision values at all ranks where relevant POIs are found.

4.4.3 Methods Studied

To demonstrate the effectiveness of our model, we compare to the following POI recommendation methods.

Traditional MF methods for implicit feedback[4]:

- **WRMF**, weighted regularized matrix factorization [20], which minimizes the square error loss by assigning both observed and unobserved check-ins with different confidential values based on matrix factorization.
- **BPRMF**, Bayesian personalized ranking [37], which optimizes the ordering of the preferences for the observed and unobserved locations.

Classical POI recommendation methods[5]:

- **MGMMF**, a multi-center Gaussian model fused with matrix factorization [75], which learns regions of activities for each user using multiple Gaussian distributions.
- **IRENMF**, instance-region neighborhood matrix factorization [78], which incorporates instance-level and region-level geographical influence into weighted matrix factorization.
- **RankGeoFM**, ranking-based geographical factorization [79], which is a ranking-based matrix factorization model that learns users' preference rankings for POIs and includes the geographical influence of neighboring POIs.

Deep learning-based methods:

- **PACE**, preference and context embedding [115], a deep neural architecture that jointly learns the embeddings of users and POIs to predict both user preferences on POIs and various contexts associated with users and POIs.

[4]The implementations are from LibRec: https://www.librec.net/

[5]In a recent study [113] that evaluated a number of POI recommendation methods, RankGeoFM and IRENMF achieve the best results on three datasets.

- **DeepAE**, a three-hidden-layer autoencoder with a weighted loss function (Section 4.3.4).

The proposed method:

- **SAE-NAD**, the proposed model with the self-attentive encoder (Section 4.3.2) and the neighbor-aware decoder (Section 4.3.3) for implicit feedback (Section 4.3.4).

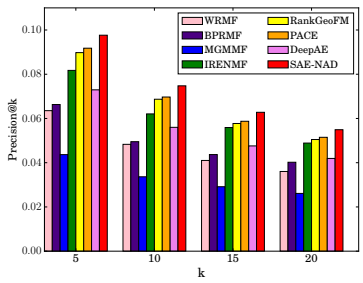
4.4.4 Parameter Settings

In the experiments, the latent dimension of all the models is set to 50. The dimension of the importance vector d_a and the geographical correlation level γ are selected by grid search, which are set to 20 and 60, respectively. The parameters α and ϵ of the weighting scheme are set to 2.0 and 1e-5, respectively. The gradient descent hyper-parameters—learning rate and regularization λ are set to 0.001 and 0.001, respectively. a_1-a_3 are set as the *tanh* function, a_4 is set to the *sigmoid* function. The batch size is set to 256. On the Gowalla dataset, we set the network architecture as $[N, 500, 50, 500, N]$; otherwise, the network architecture is set as $[N, 200, 50, 200, N]$. In addition, Dropout is used except for the first and last layer, where the Dropout probability is set to 0.5. Our model is implemented with PyTorch[6] running on GPU machines of Nvidia GeForce GTX 1080 Ti[7].

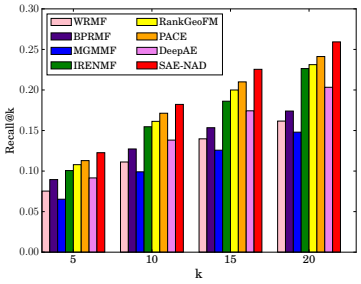
For other baseline methods, following parameter settings achieve relatively good performance. *DeepAE* adopts the same network architecture and weighted loss function with the proposed model. *PACE* uses the same network architecture (except for the hidden dimension) and hyper-parameters in the original paper. For *RankGeoFM*, the number of the nearest neighbors is set to 300, the regularization radius C is set to 1.0, the regularization balance α is set to 0.2, and the ranking margin ϵ is set to 0.3 on all datasets. As for *IRENMF*, λ_1, λ_2, and λ_3 are set to 0.015, 0.015, and 1, respectively; the instance weighting parameter α is set to 0.6; as a preprocessing step, the model uses the k-means algorithm to cluster locations into 100 groups and the number of the nearest neighbors for each location is set to 10. For *MGMMF*, the α and β of the Poisson Factor Model are set to 20 and 0.2, respectively; α, θ, and the distance threshold d of the Multi-center Gaussian Model are set to 0.2, 0.02, and 15, respectively. *WRMF* adopts the same weighting scheme as the proposed model.

[6]https://pytorch.org/

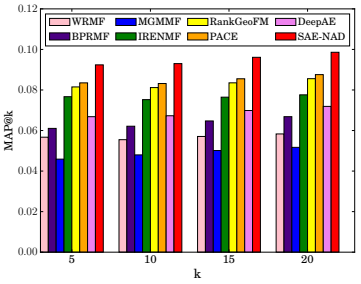
[7]Code is available at `https://github.com/allenjack/SAE-NAD`



(a) Precision@k on Gowalla

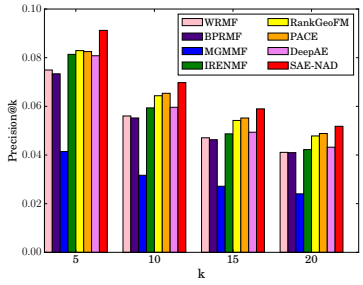


(b) Recall@k on Gowalla

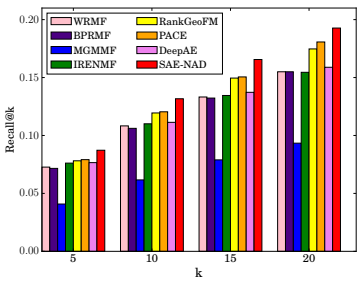


(c) MAP@k on Gowalla

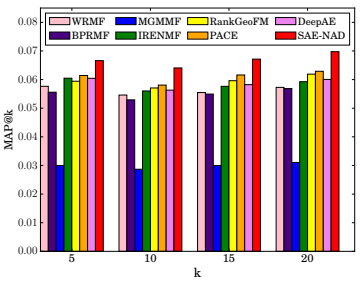
Figure 4.2 The comparison of performance on Gowalla.



(a) Precision@k on Foursquare

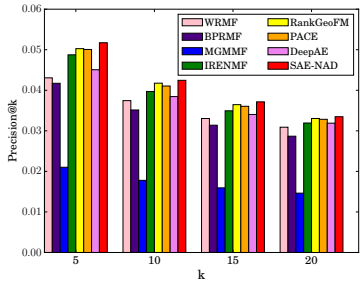


(b) Recall@k on Foursquare

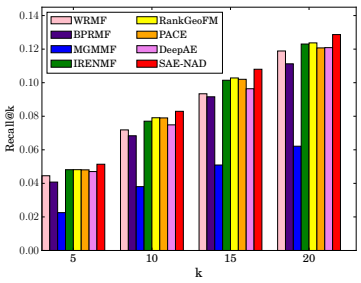


(c) MAP@k on Foursquare

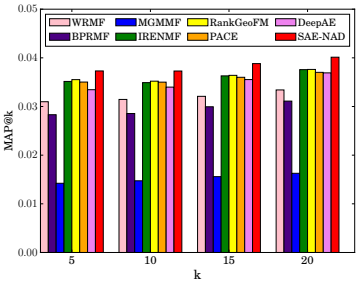
Figure 4.3 The comparison of performance on Foursquare.



(a) Precision@k on Yelp



(b) Recall@k on Yelp



(c) MAP@k on Yelp

Figure 4.4 The comparison of performance on Yelp.

4.4.5 Performance Comparison

The performance comparison of our model with other state-of-the-art methods are shown in Figures 4.2, 4.3, and 4.4.

Observations about our model:

First, our proposed model–SAE-NAD achieves the best performance on three datasets with all evaluation metrics, which illustrates the superiority of our model.

Second, SAE-NAD outperforms PACE, one possible reason is that PACE models the important geographical influence by a context graph, which does not explicitly model the user reachability to unvisited POIs. Instead, SAE-NAD directly captures the geographical influence between checked-in POIs and unvisited POIs through the neighbor-aware decoder.

Third, SAE-NAD achieves better results than DeepAE, the major reason is that DeepAE only applies a multi-layer perceptron to model the check-in data without considering other context information in the check-in records.

Fourth, SAE-NAD outperforms RankGeoFM and IRENMF. Although these two methods effectively incorporate geographical influence into a ranking model and an MF model, respectively, they still apply the inner product to predict users' preferences on POIs, which may not sufficiently capture the complex interactions between users and POIs. On the other hand, SAE-NAD adopts a deep neural structure with non-linear activation functions to model the non-trivial interactions in the user check-in data.

Fifth, although MGMMF models the geographical influence effectively, it is not good at capturing user preferences from implicit feedback. Nevertheless, SAE-NAD encodes the user's check-in frequencies into the weighting scheme, which indicates the confidence of users' preferences.

Sixth, SAE-NAD outperforms BPRMF, because BPRMF only learns the pairwise ranking of locations based on user preferences, it does not incorporate the context information such as spatial information of POIs. Besides, unlike existing methods that do not deeply explore the implicitness of users' preferences on checked-in POIs, SAE-NAD assigns an importance vector to each checked-in POI to characterize the user preference in multiple aspects.

Other observations:

First, PACE outperforms most of the baseline methods because its neural embedding part models the user-POI interactions through the implicit feedback data. In the meanwhile, the context graph incorporates the context knowledge from the unlabeled data.

Second, RankGeoFM and IRENMF both perform relatively well, which confirms the results reported in [113].

Third, although DeepAE applies a deep neural structure with the weighted loss for implicit feedback, it still does not achieve better results than RankGeoFM and IRENMF.

The reason is that DeepAE does not adopt the geographical information which is distinct for POI recommendation. But DeepAE performs better than WRMF and BPR, which may confirm that a deep network structure with non-linear activation functions can capture more sophisticated user-POI relations.

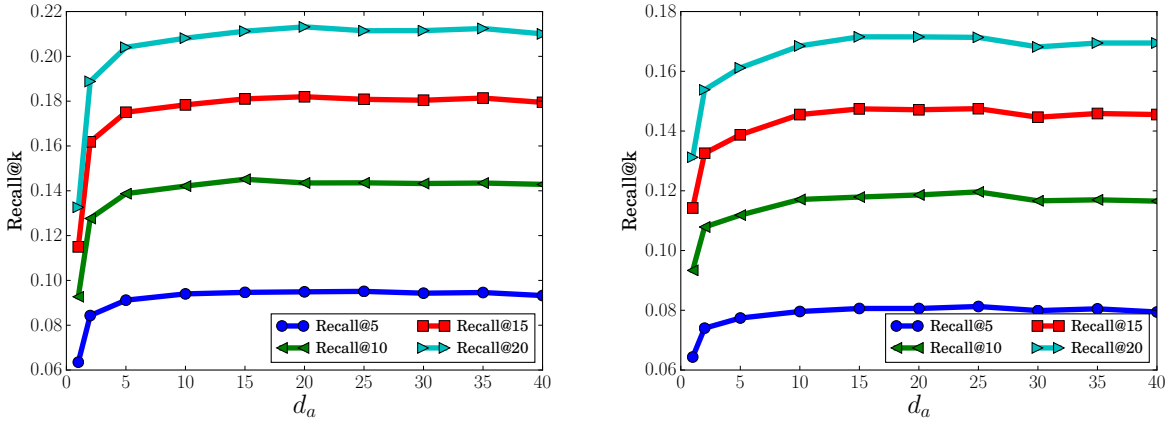
Fourth, both WRMF and BPRMF are superior to MGMMF, one possible reason is that MGMMF is based on the probabilistic factor model, which models user check-in frequencies directly, instead of modeling user preferences on POIs. On the other hand, WRMF and BPRMF are designed for implicit feedback. WRMF not only considers the observed check-ins but also gives small confidence to all unvisited locations. BPRMF leverages location pairs as training data to learn the correct ranking of location pairs.

Table 4.3 The performance of the self-attentive encoder and neighbor-aware decoder on Gowalla, Foursquare, and Yelp. *P@10* denotes Precision@10 and *R@10* denotes Recall@10.

Gowalla	P@10	R@10	MAP@10
WAE	0.05599	0.13819	0.06728
SAE-WAE	0.06039	0.14808	0.07257
NAD-WAE	0.07029	0.17915	0.08699
Foursquare	P@10	R@10	MAP@10
WAE	0.05961	0.11134	0.05632
SAE-WAE	0.06346	0.11813	0.06054
NAD-WAE	0.06598	0.12546	0.06333
Yelp	P@10	R@10	MAP@10
WAE	0.03764	0.07386	0.03198
SAE-WAE	0.03951	0.07586	0.03307
NAD-WAE	0.04115	0.08016	0.03402

4.4.6 Impacts of Self-Attentive Encoder and Neighbor-Aware Decoder

The self-attentive encoder and neighbor-aware decoder are two important components of the proposed model. To verify the performance of each component, we solely evaluate each component along with the weighted stacked autoencoder (Section 4.3.4). Here, we denote the stacked autoencoder with the weighted loss as *WAE* (equals to DeepAE), the self-attentive encoder (SAE) with WAE as *SAE-WAE*, and the neighbor-aware decoder (NAD) with WAE as *NAD-WAE*. The performance is shown in Table 4.3.



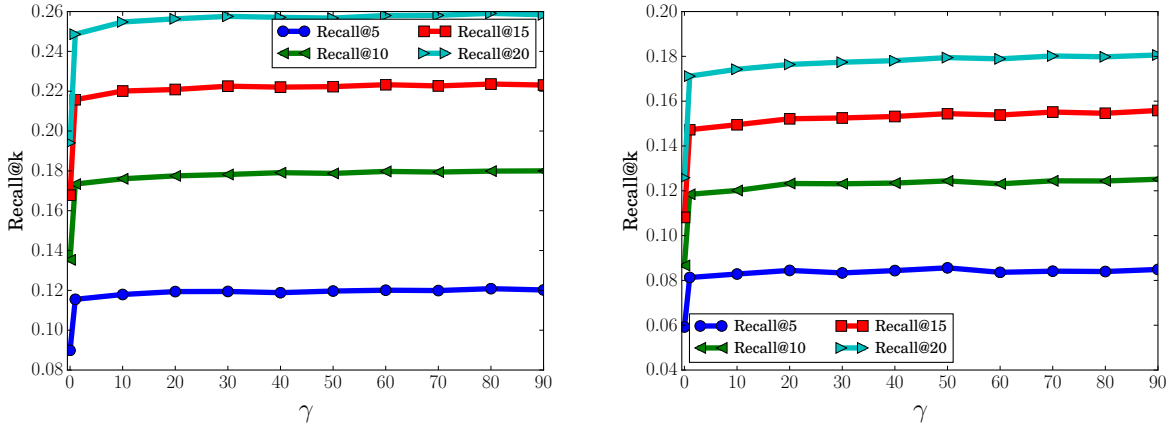
(a) d_a on Gowalla. (b) d_a on Foursquare.

Figure 4.5 The effect of d_a.

The results in Table 4.3 exhibit the effectiveness of the individual component of the proposed model. There are several observations: (1) The autoencoder with the weighted loss (WAE) achieves a reasonably good result, which even better than some baseline methods that incorporating the geographical influence. This illustrates that the frequency of the implicit feedback is a significant factor to reveal user preferences. (2) By adopting the self-attention mechanism, SAE-WAE outperforms WAE on three datasets. The reason is that the self-attentive encoder attends the POIs that are more representative to reflect user preferences, leading to more personalized and effective user hidden representations. (3) NAD-WAE achieves better performance than SAE-WAE and WAE on three datasets. The reason why NAD-WAE performs better is that NAD-WAE captures the correlations between checked-in POIs and unvisited POIs, and applies these effects to the last layer of the decoder which directly determines the model output. The results further confirm that modeling geographical influence is essential for POI recommendation.

4.4.7 Sensitivity of Hyper-Parameters

In the proposed model, two hyper-parameters are critical for performance improvements: the number of attention aspects d_a in the self-attentive encoder (Section 4.3.2) and the geographical correlation level γ in the neighbor-aware decoder (Section 4.3.3). The effects of these two parameters are shown in Figures 4.5 and 4.6. Due to the space limit, we



(a) γ on Gowalla.

(b) γ on Foursquare.

Figure 4.6 The effect of γ.

only present the effects on Gowalla and Foursquare datasets, the parameter effects on Yelp dataset have similar trends.

The variation of d_a is shown in Figure 4.5. We can observe that a single importance value from the attention layer is not sufficient to express the complex human sentiment on checked-in POIs. By assigning an importance vector to each checked-in POI, the user preference on those visited POIs can be captured from different aspects. With the increase of d_a, the model performance largely improves and becomes steady.

The variation of γ is shown in Figure 4.6. From the figure, we can observe that when $\gamma = 0$ the model does not consider the distance between POIs, leading to unsatisfactory results. This also verifies the significance of geographical influence in POI recommendation. The larger value of γ strengthens the correlated level between two certain POIs, which makes geographical neighbors of checked-in POIs play a significant role in inferring users' preferences.

4.5 Summary

In this chapter, we propose an autoencoder-based model for POI recommendation, which consists of a self-attentive encoder and a neighbor-aware decoder. In particular, the self-attentive encoder is used to adaptively discriminate the degree of user preference on each checked-in POI, by assigning an importance score vector. The neighbor-aware decoder is

adopted to model the geographical influence checked-in POIs exerted on unvisited POIs, which differentiates the user reachability on unvisited POIs. Experimental results on three real-world datasets clearly validate the improvements of our model over many state-of-the-art baseline methods.

Chapter 5

Neural Networks for Content-aware Recommendation: Incorporating Content Information

5.1 Introduction

To construct personalized recommender systems, two sorts of information are for the most part accessible and utilized: user feedback and item descriptions, e.g., users' clicking histories on movies and movies' plots. Approaches based on item text modeling such as latent dirichlet allocation (LDA) [116], stacked denoising autoencoder (SDAE) [117], and variational autoencoder (VAE) [118] have been proposed to additionally utilize items' descriptions [35,43,84], e.g., reviews, abstracts, or synopses, to enhance the top-K recommendation performance. Collaborative deep learning (CDL) [35] and collaborative variational autoencoder (CVAE) [43] are two representative methods, which explicitly connect the recommendation task with the learning of item content. In particular, CVAE and CDL apply a VAE and an SDAE, respectively, to learn hidden representations from items' bag-of-words, which are integrated with the probabilistic matrix factorization (PMF) [44]. Then learned item hidden representations are regularized with PMF's item latent factors.

Although current approaches have proposed successful models and obtained satisfactory outcomes, there are still many considerations that need to be considered in order to improve the performance. First, previous studies [35, 43] learn the hidden representations of item

content from items' (normalized) bag-of-words vectors, which does not take into account the importance of different words when describing a certain item. Equally treating the informative words along with other words may lead to an incomplete understanding of the item content. Second, previous works [43, 90] apply a weighted regularization term to fuse the hidden representations from heterogeneous information sources, such as items' ratings and descriptions. This may not fully exploit the data from heterogeneous sources and cause cumbersome hyper-parameter tuning, as various data sources are characterized by distinct statistical properties and different orders of magnitude, which is usually the case for heterogeneous data. Third, it is also important to note that the relationship between items, e.g. movies of the same director and citations between research articles, has been ignored in previous works. It is highly possible that closely related items can share the same themes or have similar attributes. As such, discovering the preferences of users on neighbors of an item often gains extra benefits from inferring the preferences of users for this item.

To address the problems mentioned above, we propose a novel recommendation model, gated attentive-autoencoder (GATE), for the content-aware recommendation. GATE is developed on a stacked autoencoder (AE) model with several effective modules: a word-attention module, a neighbor-attention module, and a neural gating structure. The encoder of the stacked AE encodes the user's implicit feedback on a certain item into the item's hidden representation. Then, from the item's sequence of words, the word-attention module learns the item embedding, where the informative words can be adaptively chosen without using complex recurrent or convolutional neural networks. To smoothly fuse the representations of items' ratings and descriptions, we propose a neural gating layer to extract and combine the salient parts of these two hidden representations, which is motivated by the long short-term memory (LSTM) [119]. Moreover, item-item relationships provide valuable auxiliary knowledge to forecast the user preference, since closely related items can have the same themes or characteristics. As a result, we apply a neighbor-attention module to learn the hidden representation of an item's neighborhood. By modeling users' preferences on the item's neighborhood, the users' preferences on this item can be indirectly reflected. On four real-world datasets, we evaluate our model extensively with several state-of-the-art approaches and multiple validation metrics. The experimental results not only demonstrate the improvements of our model over other baselines but also show the effectiveness of the gating layer and attention modules.

To summarize, the major contributions of this chapter are listed as follows:

- To learn the hidden representations from items' content information, we apply a word-attention module to adaptively distinguish informative words. As such, the item content can be better comprehended. Our word-attention module can achieve the same performance with complex recurrent or convolutional neural networks yet with fewer parameters.
- To effectively fuse the hidden representations from items' contents and ratings, we propose a neural gating layer to extract and combine the salient parts of them.
- On the basis of item-item relations, a neighbor-attention module is utilized to learn the hidden representation of an item's neighborhood. By modeling user preferences on the neighborhood of an item, it provides a significant supplement for inferring user preferences on this item.
- The two proposed attention modules are capable of interpreting and visualizing the important words and neighbors of items, respectively. Experiments on four real-world datasets indicate that the proposed GATE model substantially outperforms the state-of-the-art content-aware recommendation approaches.

5.2 Problem Formulation

The recommendation task considered in this paper takes implicit feedback [20] as the training and test data. The user preferences are presented by an m-by-n binary matrix $\mathbf{R}$. The entire collection of n items is represented by a list of documents $\mathcal{D}$, where each document in $\mathcal{D}$ is represented by a sequence of words. The item relations are presented by a binary adjacent matrix $\mathbf{N} \in \mathbb{R}^{n \times n}$, where $N_{ij} = 1$ if item i and j are related or connected. Given the item descriptions $\mathcal{D}$, the item relations $\mathbf{N}$, and part of the observed preference in $\mathbf{R}$, the problem is to predict the rest of preference scores in $\mathbf{R}$.

Here, following common symbolic notation, upper case bold letters denote matrices, lower case bold letters denote column vectors without any specification, and non-bold letters represent scalars.

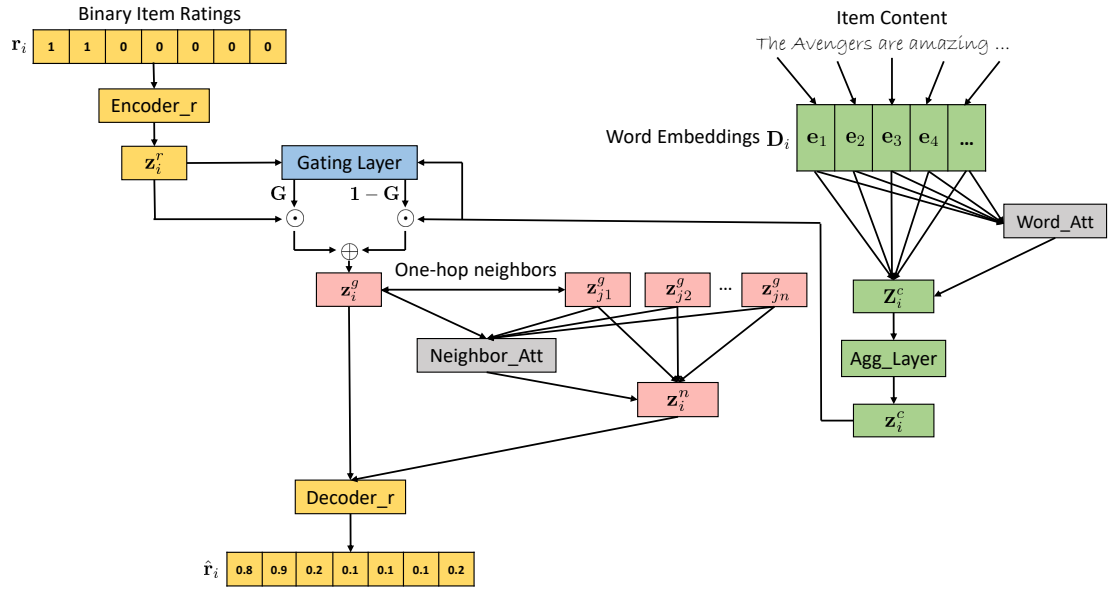


Figure 5.1 The architecture of GATE. The yellow part is the stacked AE for binary rating prediction, and the green part is the word-attention module for item content. The blue rectangle is the gating layer to fuse the hidden representations. The middle pink part is the neighbor-attention module to obtain the hidden representation of an item's neighborhood. Specifically, *Word_Att* denotes the word-attention layer, *Neighbor_Att* denotes the neighbor-attention layer, and *Agg_Layer* denotes the aggregation layer. $\odot$ is the element-wise multiplication and $\oplus$ is the element-wise addition.

5.3 Methodologies

In this section, we introduce the proposed model, which is shown in Figure 5.1. We first illustrate the basic model to learn item representations from users' binary ratings. We then introduce the multi-dimensional attention for learning item representations from word sequences. Next, we present the neural gating layer to combine the item representations from ratings and contents. We then demonstrate how to learn the hidden representation of an item's neighborhood and utilize it to assist in inferring user preferences. Lastly, we go through the loss function and training process of the proposed model.

5.3.1 Model Basics

The substantial increase of users and items makes the user-item interactions more complex and hard to model. Classical matrix factorization (MF) methods apply the inner product to

predict user preferences on items, which linearly combines users' and items' latent factors. However, it has been shown in [9, 22] how the linear combination of the inner product can limit the expressiveness of MF. Inspired by the recent works using autoencoders (AEs) to model explicit feedback [120] and implicit feedback [46], we also adopt AE as our base building block due to its ability to learn richer representations and the close relationship to MF [46].

To capture users' preferences on an item, we apply a stacked AE to encode users' binary ratings $\mathbf{r}_i \in \mathbb{R}^m$ on a certain item i into the item's rating hidden representation $\mathbf{z}_i^r$ (the superscript r indicates the hidden representation is learned from items' binary ratings):

$$\text{encoder}: \begin{cases} \mathbf{z}_i^{(1)} = a_1(\mathbf{W}_1\mathbf{r}_i + \mathbf{b}_1) \\ \mathbf{z}_i^r = a_2(\mathbf{W}_2\mathbf{z}_i^{(1)} + \mathbf{b}_2) \end{cases} \quad \text{decoder}: \begin{cases} \mathbf{z}_i^{(3)} = a_3(\mathbf{W}_3\mathbf{z}_i^r + \mathbf{b}_3) \\ \hat{\mathbf{r}}_i = a_4(\mathbf{W}_4\mathbf{z}_i^{(3)} + \mathbf{b}_4) \end{cases} \tag{5.1}$$

where $\mathbf{W}_1 \in \mathbb{R}^{h_1 \times m}$, $\mathbf{W}_2 \in \mathbb{R}^{h \times h_1}$, $\mathbf{W}_3 \in \mathbb{R}^{h_1 \times h}$, and $\mathbf{W}_4 \in \mathbb{R}^{m \times h_1}$ are the weight matrices. m is the number of users, h_1 is the dimension of the first hidden layer, and h is the dimension of the bottleneck layer. $\mathbf{r}_i$ is a multi-hot vector, where $r_{u,i} = 1$ indicates that the user u prefers the item i.

5.3.2 Word-Attention Module

Unlike previous works [22, 35, 43] learning item embeddings from bag-of-words and neglecting the importances of different words, we propose a word-attention module based on items' word sequences. Compared to learning from items' bag-of-words, the attention weights learned by our module adaptively select the informative words with different importances, and make the informative words contribute more to depict items.

Embedding Layer. In the proposed module, the input of item i is a sequence of l_i words from its text description, where each word is represented as a one-hot vector. At the embedding layer, the one-hot encoded vector is converted into a low-dimensional real-valued dense vector representation by a word embedding matrix $\mathbf{E} \in \mathbb{R}^{h \times v}$, where h is the dimension of the word embedding and v is the size of the vocabulary. After converted by

the embedding layer, the item text is represented as:

$$\mathbf{D}_i = \begin{bmatrix} & | & | & | & \\ ... & \mathbf{e}_{j-1} & \mathbf{e}_j & \mathbf{e}_{j+1} & ... \\ & | & | & | & \end{bmatrix},$$

where $\mathbf{D}_i \in \mathbb{R}^{h \times l_i}$ and $\mathbf{e}_j \in \mathbb{R}^h$.

Multi-dimensional Attention. Inspired by the Transformer [56] solely relying on attention mechanisms for machine translation, we apply a multi-dimensional attention mechanism on word sequences to learn items' hidden representations without using complex recurrent or convolutional neural networks. The reason is that, in the real-world scenario, users may care more about the topics or motifs of items that can be illustrated in a few words, rather than the word-word relations in the sequence.

The goal of the word-attention is to assign different importances on words, then aggregate word embeddings in a weighted manner to characterize the item. Given word embeddings of an item $\mathbf{D}_i$, a vanilla attention mechanism to compute the attention weights is represented by a two-layer neural network:

$$\mathbf{a}_i = \text{softmax}(\mathbf{w}_{a_1}^\top \tanh(\mathbf{W}_{a_2}\mathbf{D}_i + \mathbf{b}_{a_2} \otimes_{\text{outer}} \mathbf{1}_{|l_i|})), \tag{5.2}$$

where $\mathbf{w}_{a_1} \in \mathbb{R}^h$, $\mathbf{W}_{a_2} \in \mathbb{R}^{h \times h}$, and $\mathbf{b}_{a_2} \in \mathbb{R}^h$ are the parameters to be learned, the $softmax(\cdot)$ ensures all the computed weights sum up to 1. Then we sum up the embeddings in $\mathbf{D}_i$ according to the weights provided by $\mathbf{a}_i$ to get the vector representation of the item (the superscript c indicates the hidden representation is learned from items' contents):

$$\mathbf{z}_i^c = \sum_{\mathbf{e}_j \in D_i} a_{i,j}\mathbf{e}_j . \tag{5.3}$$

However, assigning a single importance value to a word embedding usually makes the model focus on a specific aspect of an item content [111]. It can be multiple aspects in the item content that together characterize this item, especially when the number of words is large.

Thus, we need multiple $\mathbf{a}_i$ to focus on different parts of the item content. Based on this inspiration, we adopt a matrix instead of $\mathbf{a}_i$ to capture the multi-dimensional attention and

assign an attention weight vector to each word embedding. Each dimension of the attention weight vector represents an aspect of relations among all embeddings in $\mathbf{D}_i$. Suppose we want d_a aspects of attention to be extracted from the embeddings, then we extend $\mathbf{w}_a$ to $\mathbf{W}_{a_1} \in \mathbb{R}^{d_a \times h}$, which behaves like a high level representation of a fixed query "what are the informative words" over other words in the text:

$$\mathbf{A}_i = \text{softmax}\big(\mathbf{W}_{a_1}\tanh(\mathbf{W}_{a_2}\mathbf{D}_i + \mathbf{b}_{a_2} \otimes_{\text{outer}} \mathbf{1}_{|l_i|}) + \mathbf{b}_{a_1} \otimes_{\text{outer}} \mathbf{1}_{|d_a|}\big)\,, \tag{5.4}$$

where $\mathbf{A}_i \in \mathbb{R}^{d_a \times l_i}$ is the attention weight matrix, $\mathbf{b}_{a_1} \in \mathbb{R}^{d_a}$ is the bias term, and the $softmax$ is performed along the second dimension of its input. By multiplying the attention weight matrix with word embeddings, we have the matrix representation of an item:

$$\mathbf{Z}_i^c = \mathbf{A}_i\mathbf{D}_i^\top\,, \tag{5.5}$$

where $\mathbf{Z}_i^c \in \mathbb{R}^{d_a \times h}$ is the matrix representation of the item. Then we have another neural layer to aggregate the item matrix representation into a vector representation. The hidden representation of the item is revised as:

$$\mathbf{z}_i^c = a_t(\mathbf{Z}_i^{c\top}\mathbf{w}_t)\,, \tag{5.6}$$

where $\mathbf{w}_t \in \mathbb{R}^{d_a}$ is the parameter in the aggregation layer, $a_t(\cdot)$ is the activation function.

5.3.3 Neural Gating Layer

We have obtained the item hidden representations from two heterogeneous data sources, i.e., the binary ratings and the content descriptions of items. The next aim is to combine these two kinds of hidden representations to facilitate the user preference prediction on unrated items. Unlike previous works [35, 43] regularizing these two kinds of hidden representations, we propose a neural gating layer to adaptively merge them. This is inspired by the gates in long short-term memory (LSTM) [119]. The gate $\mathbf{G}$ and the fused item hidden representation $\mathbf{z}_i^g$ are computed by:

$$\begin{aligned}\mathbf{G} &= \sigma(\mathbf{W}_{g_1}\mathbf{z}_i^r + \mathbf{W}_{g_2}\mathbf{z}_i^c + \mathbf{b}_g)\,,\\ \mathbf{z}_i^g &= \mathbf{G}\odot\mathbf{z}_i^r + (\mathbf{1}-\mathbf{G})\odot\mathbf{z}_i^c\,,\end{aligned} \tag{5.7}$$

where $\mathbf{W}_{g_1} \in \mathbb{R}^{h \times h}$, $\mathbf{W}_{g_2} \in \mathbb{R}^{h \times h}$, and $\mathbf{b}_g \in \mathbb{R}^h$ are the parameters in the gating layer, $\sigma(\cdot)$ is the sigmoid function. By using a gating layer, the salient parts from these two hidden representations can be extracted and smoothly combined.

5.3.4 Neighbor-Attention Module

Some items have an inherent relationship between each other, e.g., paper citations. Those closely related items may form a local neighborhood that shares the same topic or has the same attributes. Therefore, for a certain item, if a user is interested in its neighborhood, the user may also be interested in this item. For example, if a user likes superhero movies, she would also be willing to watch the movie produced by Marvel. Besides, in an item's local neighborhood, some items may be more representative, which should play an important role in describing the neighborhood. Inspired by this intuition, we propose a neighbor-attention module to learn the neighborhood hidden representation of a certain item. This attention mechanism is similar to which in the machine translation [55].

Formally, we define the neighbor set of item i as $\mathcal{N}_i$, which can be obtained from the item adjacency matrix[1] $\mathbf{N}$. The neighborhood hidden representation $\mathbf{z}_i^n$ of item i is computed by:

$$\begin{aligned} s_{i,j} &= \tanh(\mathbf{z}_i^{g\top} \mathbf{W}_n \mathbf{z}_j^g), \forall j \in \mathcal{N}_i \,, \\ \mathbf{a}_i &= \text{softmax}(\mathbf{s}_i) \,, \\ \mathbf{z}_i^n &= \sum_{j \in \mathcal{N}_i} a_{i,j} \mathbf{z}_j^g \,, \end{aligned} \tag{5.8}$$

where $\mathbf{W}_n \in \mathbb{R}^{h \times h}$ is the parameters to be learned in the neighbor-attention layer.

To simultaneously capture users' preferences on a certain item and its neighborhood, the decoder in Eq. 5.1 is rewritten as:

$$\begin{aligned} \mathbf{z}_i^{(3,g)} &= a_3(\mathbf{W}_3 \mathbf{z}_i^g + \mathbf{b}_3) \,, \\ \mathbf{z}_i^{(3,n)} &= a_3(\mathbf{W}_3 \mathbf{z}_i^n + \mathbf{b}_3) \,, \\ \hat{\mathbf{r}}_i &= a_4(\mathbf{W}_4 \mathbf{z}_i^{(3,g)} + \mathbf{W}_4 \mathbf{z}_i^{(3,n)} + \mathbf{b}_4) \,. \end{aligned} \tag{5.9}$$

[1] For items that do not inherently have item-item relations, we can compute the item-item similarity from the binary rating matrix $\mathbf{R}$ and set a threshold to select neighbors.

5.3.5 Weighted Loss

To model the user preference from implicit feedback, we follow a similar manner in [20] to plug in a confidence matrix in the square loss function:

$$\mathcal{L}_{AE} = \sum_{i=1}^{n} \sum_{u=1}^{m} ||C_{u,i}(R_{u,i} - \hat{R}_{u,i})||_2^2 = ||\mathbf{C}^\top \odot (\mathbf{R}^\top - \hat{\mathbf{R}}^\top)||_F^2 \,, \tag{5.10}$$

where $\odot$ is the element-wise multiplication of matrices. $||\cdot||_F$ is the Frobenius norm of matrices. In particular, we set the confidence matrix $\mathbf{C} \in \mathbb{R}^{m \times n}$ as follows,

$$C_{u,i} = \begin{cases} \rho & \text{if } R_{u,i} = 1 \\ 1 & \text{otherwise} \end{cases} \tag{5.11}$$

where the hyper-parameter $\rho > 1$ is a constant.

5.3.6 Network Training

By combining with regularization terms, the objective function of the proposed model is shown as follows:

$$\mathcal{L} = \mathcal{L}_{AE} + \lambda(||\mathbf{W}_*||_F^2 + ||\mathbf{w}_t||_2^2) \,, \tag{5.12}$$

where λ is the regularization parameter. By minimizing the objective function, the partial derivatives with respect to all the parameters can be computed by gradient descent with back-propagation. We apply Adam [112] to automatically adapt the learning rate during the learning procedure.

5.4 Experiments

In this section, we evaluate the proposed model with state-of-the-art methods on four real-world datasets.

5.4.1 Datasets

The proposed models are evaluated on four real-world datasets from various domains with different sparsities: *citeulike-a* [84], *movielens-20M* [121], *Amazon-Books* and *Amazon-*

CDs [122]. The *citeulike-a* dataset provides user preferences on articles as well as article titles, abstracts, and citations. The *movielens-20M* is a user-movie dataset where the movie description is crawled from TMDB[2]. The *Amazon-Books* and *Amazon-CDs* datasets are adopted from the Amazon review dataset[3], which covers a large amount of user-item interaction data, e.g., review, rating, helpfulness rating of the review. We select the user review with the highest helpfulness rating as the item's description. In order to be consistent with the implicit feedback setting, we keep those with ratings no less than four (out of five) as positive feedback and treat all other ratings as missing entries on the last three datasets. Since items in the latter three datasets do not inherently have the item-item relations, we compute the item-item similarity from binary rating matrix $\mathbf{R}$ and set the threshold as 0.2 to select neighbors of items. To filter noisy data, we only keep the users with at least ten ratings and the items at least with five ratings. The data statistics after preprocessing are shown in Table 5.1. For each user, we randomly select 20% and 10% of her rated items for testing and validation, respectively. The remaining constitutes the training set. The random selection is carried out five times independently, and we report the average results.

Table 5.1 The statistics of datasets.

Dataset	#Users	#Items	#Ratings	#Words	Density
citeulike-a	5,551	16,980	204,986	8,000	0.217%
ML20M	138,493	18,307	19,977,049	12,397	0.788%
Books	65,476	41,264	1,947,765	27,584	0.072%
CDs	24,934	24,634	478,048	24,341	0.078%

5.4.2 Evaluation Metrics

We evaluate our model versus other methods in terms of *Recall@k* and *NDCG@k*. For each user, Recall@k (R@k) indicates what percentage of her rated items can emerge in the top k recommended items. NDCG@k (N@k) is the normalized discounted cumulative gain at k, which takes the position of correctly recommended items into account.

[2]https://www.themoviedb.org/
[3]http://jmcauley.ucsd.edu/data/amazon/

5.4.3 Methods Studied

To demonstrate the effectiveness of our model, we compare to the following recommendation methods.

Classical methods for implicit feedback:

- **WRMF**, weighted regularized matrix factorization [20], which minimizes the squared error loss by assigning user rated and unrated items with different confidential values.
- **CDAE**, collaborative denoising autoencoder [46], which utilizes the denoising autoencoder to learn the user hidden representation from implicit feedback.

Methods learning from bag-of-words:

- **CDL**, collaborative deep learning [35], is a probabilistic feedforward model for joint learning of stacked denoising autoencoder (SDAE) and collaborative filtering.
- **CVAE**, collaborative variational autoencoder [43], is a generative latent variable model that jointly models the generation of content and rating and uses variational Bayes with inference network for variational inference.
- **CML+F**, collaborative metric learning with item features [22], which learns a metric space to encode not only users' preferences but also the user-user and item-item similarities.

Methods learning from word sequences:

- **ConvMF**, convolutional matrix factorization [90], which applies the convolutional neural network (CNN) to capture contextual information of documents and integrates CNN into the probabilistic matrix factorization (PMF).
- **JRL**, joint representation learning [88], is a framework that learns joint representations from different information sources for top-K recommendation.

The proposed method:

- **GATE**, the proposed model, fuses hidden representations from items' ratings and contents by a gating layer, moreover, the word-attention and neighbor-attention are adopted for selecting informative words and learning hidden representations of items' neighborhoods, respectively.

Given our extensive comparisons against the state-of-the-art methods, we omit comparisons with methods such as HFT [83], CTR [84], SVDFeature [123], and DeepMusic [124] since they have been outperformed by the recently proposed CDL, CVAE, and JRL.

5.4.4 Experiment Settings

In the experiments, the latent dimension of all the models is set to 50. WRMF adopts the same heuristic weighting function with the proposed model. For CDAE, we follow the settings in the original paper. For CDL, we set $a = 1$, $b = 0.01$, and find that when $\lambda_u = 1$, $\lambda_v = 10$, $\lambda_n = 100$, and $\lambda_w = 0.0001$ can achieve good performance. For CVAE, the parameters $a = 1$, $b = 0.01$ are the same. When $\lambda_u = 0.1$, $\lambda_v = 10$, $\lambda_r = 0.01$, CVAE can achieve good performance. For CML+F, we follow the author's code to set the margin $m = 2.0$, $\lambda_f = 0.1$, and $\lambda_c = 1$, respectively. The item features are learned by a multi-layer perceptron with a 512-dimensional hidden layer and 0.3 dropout. For ConvMF, we set the CNN configuration the same as the original paper and find it can achieve a good result when $a = 1$, $b = 0.01$, $\lambda_u = 0.1$, and $\lambda_v = 10$. For JRL, we follow the original paper setting to set batch size as 64, the number of negative samples $t = 5$, and $\lambda_1 = 1$. The network architectures of above methods are also set the same with the original papers.

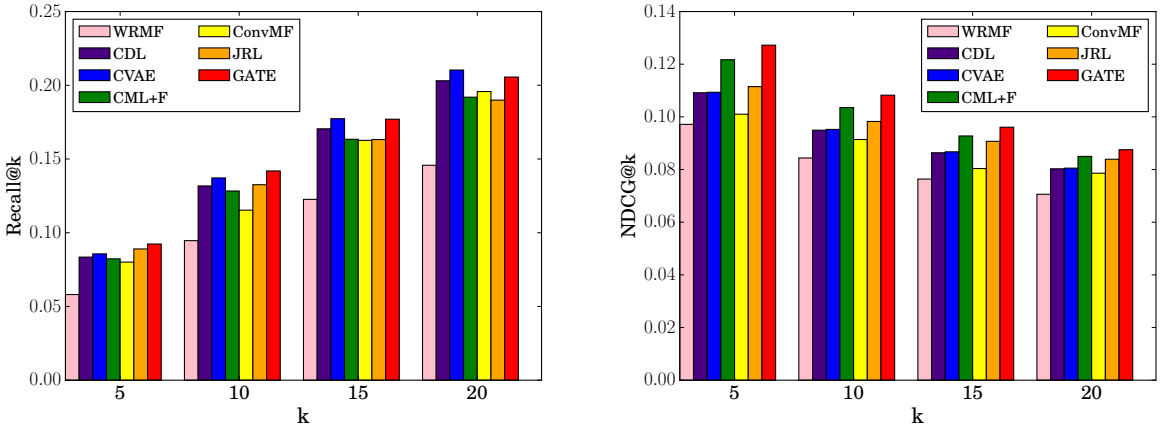
For GATE, the ratings of an item are a binary rating vector from all users; the content of an item is the word sequence from its description. We set the maximum length of the word sequence to 300, and the same setting is also adopted in ConvMF and JRL. Hyper-parameters are set by grid search. The network architecture is set to $[m, 100, 50, 100, m]$ on all datasets. ρ is set to 5 on citeulike-a, 20 on movielens-20M, 15 on Amazon-Books, and 20 on Amazon-CDs, respectively. d_a is set to 20, where its effect is shown in Section 5.4.6. The learning rate and λ are set to 0.01 and 0.001, respectively. The activation function is set to *tanh*. And the batch size is set to 1024. Our experiments are conducted with PyTorch[4] running on GPU machines of Nvidia GeForce GTX 1080 Ti[5].

5.4.5 Performance Comparison

The performance comparison results are shown in Figures 5.2, 5.3, 5.4 and 5.5, and Table 5.2. Since CDAE is not as good as other state-of-the-art methods when the dataset becomes

[4]https://pytorch.org/

[5]The code is available on Github: `https://github.com/allenjack/GATE`



(a) Recall@k on citeulike-a (b) NDCG@k on citeulike-a

Figure 5.2 The performance comparison on citeulike-a.

sparse, we do not present the results of CDAE in the aforementioned figures.

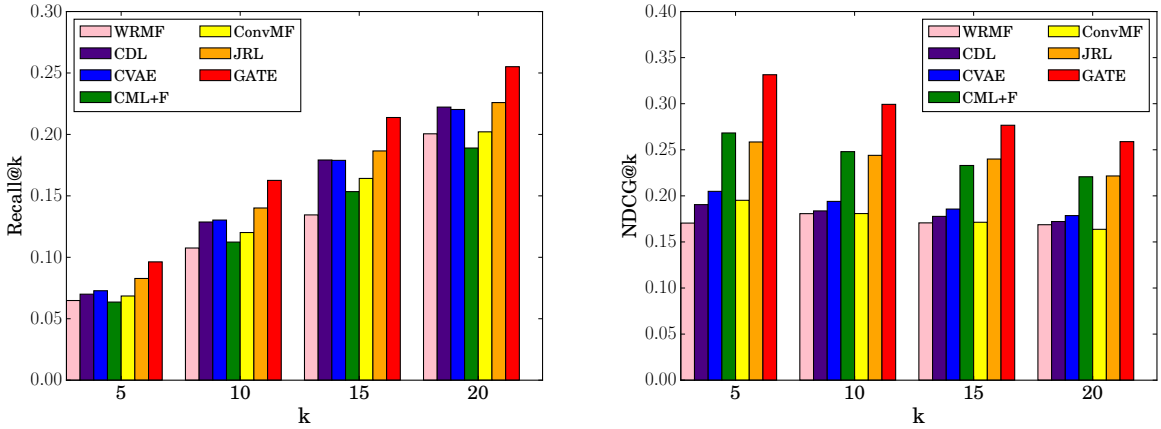
Table 5.2 The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method.

	WRMF	**CDAE**	**CDL**	**CVAE**	**CML+F**	**ConvMF**	**JRL**	**GATE**	**Improv.**
Recall@10									
citeulike-a	0.0946	0.0888	0.1317	<u>0.1371</u>	0.1283	0.1153	0.1325	**0.1419**	3.50%
movielens-20M	0.1075	0.0751	0.1287	0.1303	0.1123	0.1201	<u>0.1401</u>	**0.1625****	15.99%
Amazon-Books	0.0553	0.0132	0.0648	0.0632	0.0756	0.0524	<u>0.0924</u>	**0.1133***	22.62%
Amazon-CDs	0.0779	0.0191	<u>0.0827</u>	0.0811	0.0824	0.0753	0.0816	**0.1057*****	27.81%
NDCG@10									
citeulike-a	0.0843	0.0736	0.0949	0.0952	<u>0.1035</u>	0.0914	0.0982	**0.1082**	4.54%
movielens-20M	0.1806	0.1774	0.1836	0.1939	<u>0.2479</u>	0.1807	0.2439	**0.2992****	20.69%
Amazon-Books	0.0377	0.0105	0.0393	0.0384	0.0456	0.0324	<u>0.0592</u>	**0.0708*****	19.59%
Amazon-CDs	0.0357	0.0105	0.0356	0.0349	0.0364	0.0323	<u>0.0386</u>	**0.0477*****	23.58%

Observations about our model:

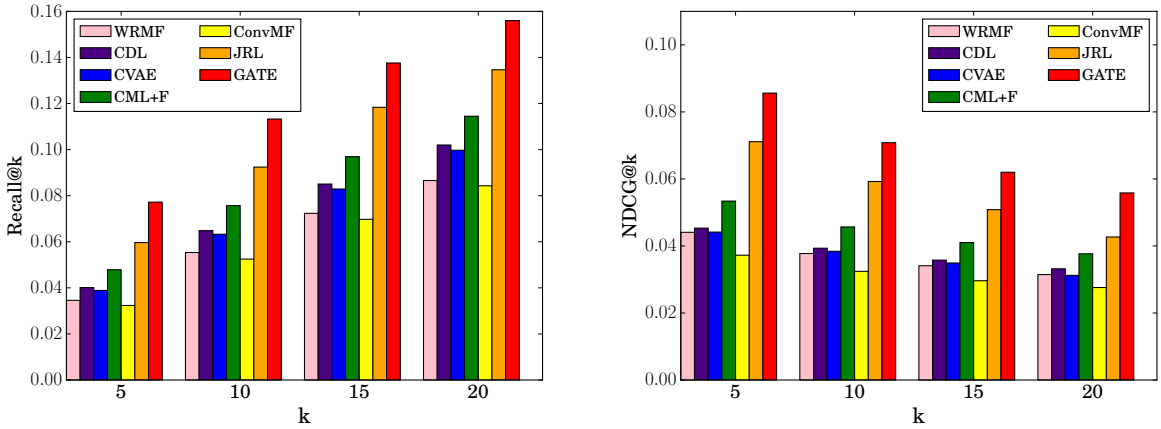
First, the proposed model—GATE, achieves the best performance on three datasets with all evaluation metrics, except for the Recall@15 and Recall@20 on citeulike-a, which illustrates the superiority of our model.

Second, GATE obtains better results than JRL and ConvMF. Although JRL and ConvMF capture the contextual information in item descriptions by the *doc2vec* model [89] and



(a) Recall@k on movielens-20M (b) NDCG@k on movielens-20M

Figure 5.3 The performance comparison on movielens-20M.

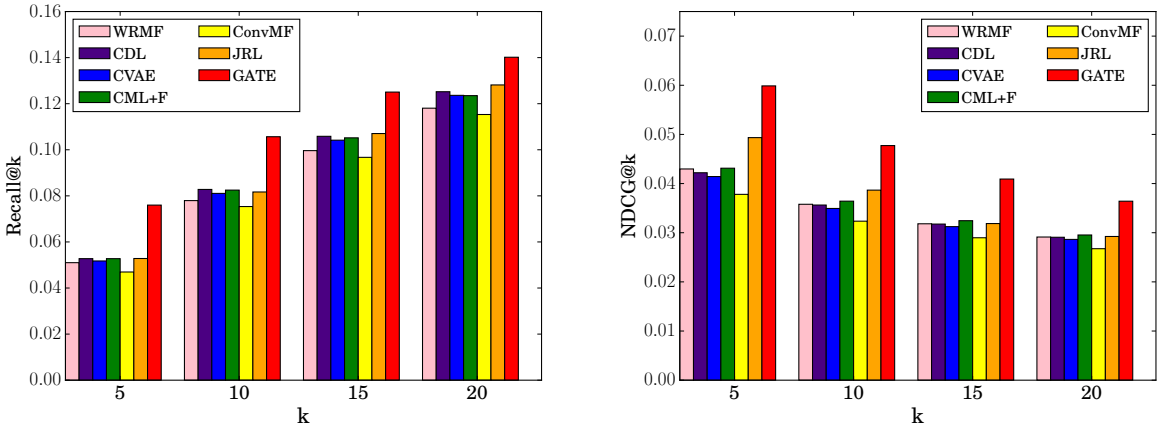


(a) Recall@k on Amazon-Books (b) NDCG@k on Amazon-Books

Figure 5.4 The performance comparison on Amazon-Books.

the convolutional neural network, respectively, they equally treat each word of items, which does not consider the effects of informative words, leading to the incomplete understanding of item content information.

Third, GATE outperforms CML+F, CVAE, and CDL. The reasons are two-fold: (1) these three methods learn the item content representation through bag-of-words, which neglects the effect that important words can describe the topics or synopses of items; (2) these three methods link the hidden representations from different data sources by a regu-



(a) Recall@k on Amazon-CDs (b) NDCG@k on Amazon-CDs

Figure 5.5 The performance comparison on Amazon-CDs.

larization term, which may not smoothly balance the effects of various data representations and incur tedious hyper-parameter tuning.

Fourth, GATE achieves better results than WRMF and CDAE. The reason is that these two methods do not incorporate the content information, which is crucial when the user-item interaction data is sparse.

Fifth, it is important to note that all the compared methods do not consider the user preference on an item's neighborhood, which is captured by the neighbor-attention module of GATE.

Sixth, GATE does not significantly improve the performance over other methods on the citeulike-a dataset. One possible reason is that the citeulike-a dataset is relatively small, which makes GATE overfit the data.

Other observations:

First, all the results reported on citeulike-a and movielens-20M are better than the results on Amazon-Books and Amazon-CDs, the major reason is that the latter two datasets are more sparse and the data sparsity declines the recommendation performance.

Second, JRL and CML+F perform better than other state-of-the-art methods on more sparse datasets. The reason may be that JRL models the contextual information in the item descriptions, which captures the word-word relations in the text. On the other hand, CML+F encodes user-item relationships and user-user/item-item similarities in a joint metric space, which are helpful to find users' preferred items when the data is sparse.

Third, although ConvMF models the contextual information from items' descriptions, it still does not perform better than JRL, CML+F, CVAE, and CDL. One possible reason is that the regularization term in ConvMF does not effectively pick up the latent features learned from text to benefit the item latent factors learned from matrix factorization.

Fourth, CVAE and CDL achieve similar results on all datasets. One reason is that they have a similar Bayesian probabilistic framework.

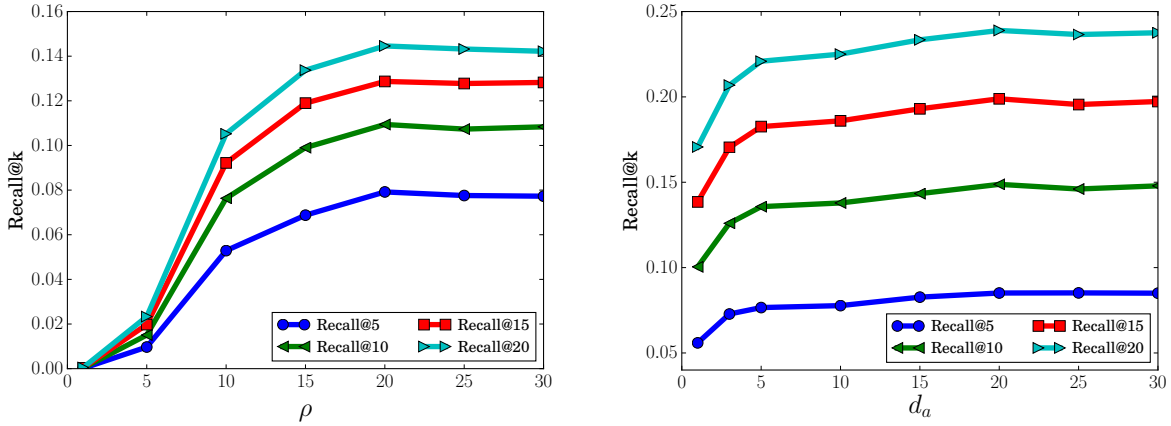
Fifth, WRMF and CDAE only adopt implicit feedback as input and do not model the auxiliary information, that is why their performance drops when the dataset becomes sparse. In addition, WRMF has similar results with CDL and CVAE on some metrics, which may illustrate that CDL and CVAE may not fully take advantage of the heterogeneous data.

Table 5.3 The ablation analysis on Amazon-CDs and Amazon-Books datasets in terms of Recall@10 (R@10) and NDCG@10 (N@10).

Architecture	*CDs*		*Books*	
	R@10	N@10	R@10	N@10
(1) stacked AE	0.0672	0.0315	0.0745	0.0484
(2) reg: AE + W_Att	0.0676	0.0318	0.0304	0.0265
(3) gating: AE + W_Att	0.0816	0.0353	0.0793	0.0515
(4) gating: AE + GRU	0.0818	0.0352	0.0789	0.0512
(5) gating: AE + CNN	0.0777	0.0335	0.0791	0.0495
(6) GATE	**0.1057**	**0.0477**	**0.1133**	**0.0708**

5.4.6 Ablation Analysis

To verify the effectiveness of the proposed word-attention, gating layer, and neighbor-attention modules, we conduct an ablation analysis in Table 5.3 to demonstrate the performance each module contributes to the GATE model. In (1), we utilize the weighted stacked AE without any other components. In (2), we regularize $\mathbf{z}_i^r$ and $\mathbf{z}_i^c$ the by $L2$ norm on the top of (1), following the same manner in [35,43]. We tried the regularization parameters $\{0.01, 0.1, 0.5, 1, 10\}$, where 0.1 gives the best results. In (3), we plug the gating layer to connect $\mathbf{z}_i^r$ and $\mathbf{z}_i^c$ on the top of (1). In (4), we adopt the a recurrent neural network structure–gated recurrent units (GRUs) [125] to learn $\mathbf{z}_i^c$, which is also linked to $\mathbf{z}_i^r$ by the proposed gating layer. In (5), we replace the GRUs in (4) with a convolutional neural network (CNN), where the structure and hyper-parameters are set the same in [90]. In



(a) ρ on Amazon-CDs (b) d_a on movielens-20M

Figure 5.6 The effects of ρ and d_a.

(6), we present the overall GATE model to show the significance of the neighbor-attention module.

From the results shown in Table 5.3, we have some observations. First, from (2) and (3), the gating layer achieves better results than regularization. One possible reason is that the neural gate can extract representative parts and mask off insignificant parts from the input hidden representations. Second, from (3), (4), and (5), we observe that our word-attention module has similar performance with GRUs and CNNs but with *fewer* parameters[6] (if we set the word embedding size to 50 ($h = 50$), then the number of learned parameters of our word-attention module is 3,590, the number of parameters of the one-recurrent-layer GRU is 15,300, the number of parameters of the CNN in [90] is 75,350). This result demonstrates that the proposed word-attention module can effectively learn the item hidden representation from items' descriptions. Third, from (1), (3), and (6), we observe that our neighbor-attention may play a critical role in the overall model. The results demonstrate that modeling users' preferences on an item's neighborhood is an effective supplement for inferring their preferences on this item.

[6]We verified the number of parameters of all three models by the *named_parameters()* function provided by PyTorch.

5.4.7 The Sensitivity of Hyper-parameters

The effects of ρ and d_a are shown in Figure 5.6, which have similar trends on other datasets. We can observe that with the increase of ρ, the performance improves and becomes stable. The reason is that the larger value of ρ makes the model concentrate more on the items that users interacted with before, where users' preferences are more accurately captured. For the variation of d_a, we verify that utilizing a vector to measure the importance of a word is more effective than just using a single value in our scenario because the score vector describes the relations between each word from different aspects. Note that we do not include the neighbor-attention module when testing the effect of d_a.

Table 5.4 A case study of the word-attention.

The Summary of Article 16797 in *citeulike-a*
We present the first parallel implementation of the T-Coffee consistency-based multiple aligner. We benchmark it on the Amazon Elastic Cloud (EC2) and show that the parallelization procedure is reasonably effective. We also conclude that for a web server with moderate usage (10K hits/month) the cloud provides a cost-effective alternative to in-house deployment.
The Summary of Article 120 in *citeulike-a*
We identify a metaphor for the design activity : we view design as bricolage. We start from describing bricolage, and we proceed to the relationship of design to art . We obtain a characterisation of design that enables us to show that both traditional and contemporary design are forms of bricolage. We examine the consequences of ' design as bricolage' for the relationship between design and science and for the extent of the design activity .

5.4.8 Word- and Neighbor-Attention Case Studies

Table 5.5 A case study of the importance scores computed by the neighbor-attention module. The number inside $(\cdot)$ indicates the number of *fluctuation*'s occurrences excluding references in an article.

Target	Neighbor	Score
Fluctuations in network dynamics	Genomic analysis of regulatory network dynamics reveals large topological changes (0)	0.07172
	Frequency of occurrence of numbers in the World Wide Web (10)	0.22090
	Complex networks: Structure and dynamics (16)	0.26835
	Noise in protein expression scales with natural protein abundance (**36**)	**0.43903**

To visualize the word-attention effects, we conduct a case study on the citeulike-a dataset. We sum along the first dimension of $\mathbf{A}_i \in \mathbb{R}^{d_a \times l_i}$ (Eq. 5.4) to get $\mathbf{a}_i \in \mathbb{R}^{l_i}$, which can be treated as the accumulated attention weights of each word. For the ease of visualization, we normalize $\mathbf{a}_i$ following the same procedure in [111] and words with lower scores are not colored. Two examples of word-attention visualization are shown in Table 5.4. From the first example, we can observe that the words *aligner* and *cloud* have the highest importance scores, which may reflect the topic and platform of this paper. On the other hand, the words *present*, *show*, and *conclude* are widely used in all the papers, which are less attractive. In the second example, the situation is the same. The most important word that selected by the word-attention is *metaphor*, which may reveal the motif of the article.

The neighbor-attention case study is shown in Table 5.5. The neighbors of the target article are provided by the citation graph of the citeulike-a dataset. From this case, we observe that the neighbor attention score can identify an item's important neighbors. In the example, the target article finds a scaling rule in network dynamics, and the fourth neighbor of the target also observes the same scaling behavior for all groups of genes. If we treat *fluctuation* as the key topic of the target article, the number of *fluctuation*'s occurrences in the target's neighbors may reveal how related the target with its neighbors. We also list the count of *fluctuation* within the article body. The counts of *fluctuation* further verify the importance scores computed by our neighbor-attention module.

5.5 Summary

In this chapter, we propose a gated autoencoder with the word- and neighbor-attention. The model learns items' hidden representations from ratings and contents in a gated manner. Moreover, the model also captures items' informative words and representative neighbors by word- and neighbor-attention modules, respectively. Experimental results on four real-world datasets clearly validate the performance of our model over many state-of-the-art methods and show the effectiveness of the gating and attention modules.

Chapter 6

Neural Networks for Sequential Recommendation: Modeling Temporal Dynamics

6.1 Introduction

In all sorts of Internet services and applications, users browse items or products in a sequential order, where the items the user will browse may be closely related to those items that the user just accessed. This property enables a non-trivial recommendation task—sequential recommendation that considers the history of user actions as an activity series organized by the operating timestamp. For one key reason, it is difficult to conduct this task: the hardship of inferring the short-term interests and intentions of users. Indeed, the actions of users on products are decided together by both the long-term and short-term interests of users. With the large amount of accumulated data, the long-term user interests can be effectively modeled. However, in a short-term time context, how to make good use of temporal dynamics to forecast the user behavior in the near future is non-trivial.

To capture the sequential dynamics in the user action history, effective models are proposed to learn the short-term user preference in the sequential user interactions, such as Markov Chains (MCs), convolutional neural networks (CNNs), and recurrent neural networks (RNNs). MC-based methods [94, 95] apply a K-order Markov chain to make recommendations based on the K previous actions. CNN-based methods [96] utilize convo-

lutional filters and sliding window strategies to capture the short-term contexts for future prediction. RNN-based methods [97, 98, 100] adopt gated recurrent (GRU) or long short-term memory (LSTM) units to learn the user-item sequence, where the short-term user interests are captured by the hidden states of RNNs.

While existing approaches have proposed successful models and obtained acceptable performance, we argue that there are still several avenues for further improving the model performance. First, previous studies [96–98, 100] learn the user action sequence by CNN or RNN structures, which does not pay special attention to the specific parts of features of different items. Neglecting the representative characteristics of items in a short-term period can fail to catch the true interests of users. Second, these approaches based on CNN or RNN often do not distinguish the importance of items according to the tastes of users. It can result in an inadequate interpretation of user intentions if considering these informative items along with other items equally. Third, it is also important to note that the relations between items are disregarded in previous works [96, 100, 101]. It has a larger chance that users will interact with closely relevant items one after the other. As such, the explicit capture of the item-item relationship can significantly benefit anticipating the user engagement of potential items.

To solve the problems described above, we propose a novel recommendation model, the hierarchical gating network (HGN), for the sequential recommendation without the use of complex recurrent or convolutional neural networks. HGN is integrated with the matrix factorization model and optimized by the Bayesian Personalized Ranking (BPR) objective [37], which consists of a feature gating module, an instance gating module, and an item-item product module. In particular, the feature gating module allows the adaptive selections of attractive latent features within a certain item based on the user preference. Then the selected user-specific features of items will be passed to the instance gating module. At the instance gating module, important items that can reflect the short-term user interests will be distinguished and selected to predict what items the user will interact with in the future. Thus, the feature gating and instance gating modules form a hierarchical gating network to control what informative features or items can be passed to the downstream layers. Item-item relations, on the other hand, provide valuable auxiliary knowledge to predict the sequential actions of users, as users may access closely relevant items one after the other. Thus, we apply an item-item product module to explicitly capture the relations between the items users have interacted with and those items users will interact

with in the future. We extensively evaluate our model with many state-of-the-art methods and different validation metrics on five real-world datasets. The experimental results not only demonstrate the improvements of our model over other baselines but also show the effectiveness of the gating and item-item product modules.

To summarize, the major contributions of this chapter are listed as follows:

- To infer the user interests in a short-term context, we propose a hierarchical gating network to control what item latent features and which relevant item can be passed to the downstream layers. Our hierarchical gating network achieves better performance compared with complex recurrent or convolutional neural networks yet with fewer parameters and faster training speed.
- To explicitly capture the item-item relations, we utilize an item-item product module to learn the relationships between the items users have interacted with and those items the user will interact with in the near future.
- Experiments on five real-world datasets show that the proposed HGN model significantly outperforms the state-of-the-art methods for the sequential recommendation task.

6.2 Problem Formulation

The recommendation task considered in this paper takes sequential implicit feedback as training data. The user preference is presented by a user-item sequence in the chronological order $\mathcal{S}^i = (\mathcal{S}^i_1, \mathcal{S}^i_2, ..., \mathcal{S}^i_{|\mathcal{S}^i|})$, where $\mathcal{S}^i_j$ is an item index that user i has interacted with. Given the earlier subsequence $\mathcal{S}^i_{1:t}(t < |\mathcal{S}^i|)$ of M users, the problem is to recommend a list of items from N items for each user and evaluate whether the items in $\mathcal{S}^i_{t:|\mathcal{S}^i|}$ will appear in the recommended list.

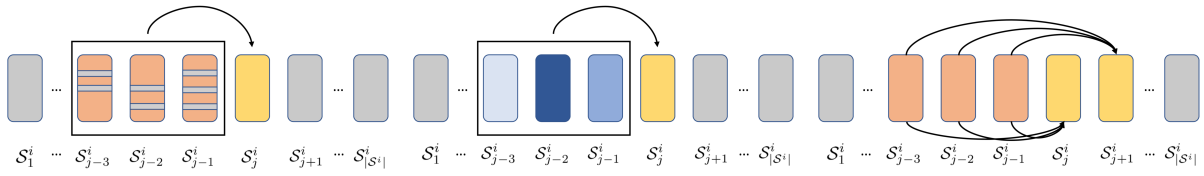
Here, following common symbolic notation, upper case bold letters denote matrices, lower case bold letters denote column vectors without any specification, and non-bold letters represent scalars. The major symbols are listed in Table 6.1.

6.3 Methodologies

To model the sequential recommendation task, for each user i, we extract every $|L|$, i.e. $L = (\mathcal{S}^i_j, \mathcal{S}^i_{j+1}, ..., \mathcal{S}^i_{j+|L|-1})$, successive items as input and their next $|T|$ items as the targets

Table 6.1 List of notations.

M, N	the number of users and items
$\mathcal{S}^i$	the item sequence of user i
$\mathbf{S}_{i,l}$	the embeddings of the l-th subsequence of user i
$\mathbf{W}_{g_*}, \mathbf{w}_{g_*}$	the learnable parameters in the gating layers
$\mathbf{U}$	the user embedding matrix
$\mathbf{E}$	the input item embedding matrix
$\mathbf{Q}$	the output item embedding matrix
d	the dimension of the embeddings
$\hat{r}_{i,j}$	the prediction score of user i on item j
λ	the regularization term



(a) Feature gating (b) Instance gating (c) Item-item product

Figure 6.1 An illustrative example of the feature gating, instance gating, and item-item product modules. In Figure 6.1a, the gray lines on items denote those latent features are masked off. In Figure 6.1b, the darker blue means the item is more important. In Figure 6.1c, the line linked between two items denotes the inner product, which captures the relations between the items users have accessed and the items users will access in the future.

to be predicted. The problem can be formulated as: in the user-item interaction sequence $\mathcal{S}^i$, given the $|L|$ successive items, how likely other items will be interacted subsequently.

In the sequential recommendation problem, the prediction of users' preferences on items can be modeled in two perspectives: long-term interests and short-term interests. The long-term user preference modeling has been widely investigated in the conventional top-K recommendation methods, such as matrix factorization [20, 37]. On the other hand, how to capture the short-term user interests from the sequential data is the key point for performance improvement.

For the short-term interest modeling, we argue that there are two kinds of relationships existing between items users have interacted with and items users will access in the future: group-level and instance-level relations. The group-level influence illustrates a phenomenon that several items in L together have an impact on the items user may interact in the future. For example, if a user has bought a bed frame and a mattress, a pillow is probably a more

suitable recommendation than a brand-new burger. On the other hand, the instance-level influence depicts the strong relation between a single item in L and a single item in T. For example, if a user bought a mobile phone, she may also need to buy a screen protector or a case. Thus, these two kinds of relations together determine users' short-term interests.

In this section, we introduce the proposed model to capture both the long-term interests and short-term interests of users for the sequential recommendation, which is shown in Figure 6.1 and Figure 6.2. We first illustrate the hierarchical gating network for learning users' group-level preferences. Next, we present the inner product of item embeddings to model the item-item relations. Then we introduce the prediction layer for aggregating the long-term and short-term interests of users. Lastly, we go through the loss function and training process of the proposed model.

6.3.1 Hierarchical Gating for Group-level Influence

In the sequential recommendation, taking advantage of the properties of sequential data to learn the (sub)sequence representation is a critical point, where an item may be closely related to its previous or subsequent items, or a group of previous items will have an impact on future items. In previous works, researchers have utilized various methods to model the group-level sequential interactions, e.g., convolutional neural networks [96], recurrent neural networks [57, 97, 98, 100], and the self-attention model [101]. Different from previous works, we propose a hierarchical gating network for modeling group-level user-item interactions, which consists of two components: a feature gating module and an instance gating module. These two modules allow the selection of effective latent features and relevant items, respectively, for predicting the subsequent items. Our proposed gating network is both effective and efficient (Section 6.4).

Feature Gating

Unlike previous works [96, 100, 101] that only operate on the item-level, we provide a learnable feature gating module to select salient latent features of items from the feature-level. For a certain item, some parts of the latent features are more relevant to predict the subsequent items. For example, for a big fan of Robert Downey Jr., after watching Iron Man I and Iron Man II, it is better to recommend Iron Man III rather than Aquaman, although Aquaman is also a superhero movie. Thus, to capture the representative item features

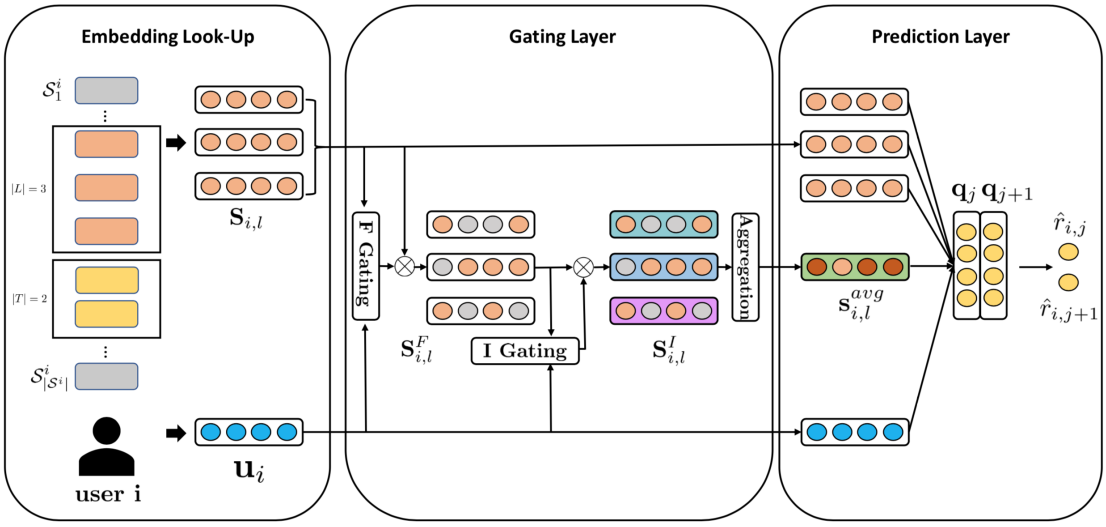


Figure 6.2 The architecture of HGN. HGN consists of three major components: the embedding layer, the hierarchical gating layer, and the prediction layer. Specifically, *F Gating* denotes the feature gating module, *I Gating* denotes the instance gating module, *Aggregation* denotes the aggregation layer, and $\otimes$ denotes the element-wise multiplication.

based on users' preferences is a necessary point to capture.

Embedding Layer. In the proposed module, the input is a sequence of $|L|$ items, where each item is represented by a unique index. At the embedding layer, the item index is converted into a low-dimensional real-valued dense vector representation by an item embedding matrix $\mathbf{E} \in \mathbb{R}^{d \times N}$, where d is the dimension of the item embedding and N is the number of items. After converted by the embedding layer, the item subsequence embeddings are represented as:

$$\mathbf{S}_{i,l} = \begin{bmatrix} & | & | & | & \\ \dots & \mathbf{e}_{j-1} & \mathbf{e}_j & \mathbf{e}_{j+1} & \dots \\ & | & | & | & \end{bmatrix},$$

where $\mathbf{S}_{i,l} \in \mathbb{R}^{d \times |L|}$ indicates the embeddings of the l-th subsequence of user i, $\mathbf{e}_j \in \mathbb{R}^d$ is the j-th column of the embedding matrix $\mathbf{E}$.

Gated Linear Unit. Inspired by the gated linear unit (GLU) proposed by Dauphin et al. in [126], which is utilized to control what information should be propagated for predicting the next word in the language modeling task, we also adopt a similar model to

select what features are relevant to predict future items. The GLU in the original paper is shown:

$$(\mathbf{X} * \mathbf{W} + \mathbf{b}) \otimes \sigma(\mathbf{X} * \mathbf{V} + \mathbf{c}),$$

where $\mathbf{X}$ is the input embeddings, $\mathbf{W}$, $\mathbf{V}$, $\mathbf{b}$, $\mathbf{c}$ are learnable parameters, σ is the sigmoid function, $*$ is the convolution operation, and $\otimes$ is the element-wise product between matrices.

Personalized Feature Gating. However, directly applying the GLU to select item features does not explicitly consider the user preference on items. For a certain item, a user may just focus a specific part of the item and neglect other unattractive parts. For example, a user may only care about whether the starring role is Tom Cruise rather than the movie content.

Therefore, to capture the item features that tailored to users' preferences, we need to modify the GLU to be user-specific. To reduce the number of learnable parameters, we apply the inner product instead of the convolution operation in the original GLU (the superscript F indicates the item sequence embeddings are learned from the feature gating module):

$$\mathbf{S}^F_{i,l} = \mathbf{S}_{i,l} \otimes \sigma\big(\mathbf{W}_{g_1} \cdot \mathbf{S}_{i,l} + (\mathbf{W}_{g_2} \cdot \mathbf{u}_i) \otimes_{\text{outer}} \mathbf{1}_{|L|} + \mathbf{b}_g \otimes_{\text{outer}} \mathbf{1}_{|L|}\big), \tag{6.1}$$

where $\mathbf{u}_i \in \mathbb{R}^d$ is the embedding of user i, $\mathbf{W}_{g_1}, \mathbf{W}_{g_2} \in \mathbb{R}^{d \times d}$ and $\mathbf{b}_g \in \mathbb{R}^d$ are learnable parameters, $\otimes$ is the element-wise product between matrices, and $\otimes_{\text{outer}}$ is the outer product between two vectors. By doing this, user-specific features of items can be passed to downstream layers.

Instance Gating

Personalized Instance Gating. Since our formulated problem is: given $|L|$ successive items, how likely other items will appear after L in the near feature, we argue that there are some items are more relevant in L to predict the items users will interact. However, existing works either do not consider the representative items in L [96, 100] or apply attention models to capture the representative items [99,101]. Unlike previous works benefiting from attention models, we adopt an instance-level gating module to select the informative items

that are helpful to predict items in the near future according to users' preferences:

$$\mathbf{S}_{i,l}^{I} = \mathbf{S}_{i,l}^{F} \otimes \sigma\big((\mathbf{w}_{g_3}^{\top} \cdot \mathbf{S}_{i,l}^{F}) \otimes_{\text{outer}} \mathbf{1}_{|L|} + (\mathbf{u}_i^{\top} \cdot \mathbf{W}_{g_4}) \otimes_{\text{outer}} \mathbf{1}_{|L|}\big)\,, \tag{6.2}$$

where $\mathbf{S}_{i,l}^{I} \in \mathbb{R}^{d \times |L|}$ is the sequence embedding after the instance gating, $\mathbf{w}_{g_3} \in \mathbb{R}^{d}$, $\mathbf{W}_{g_4} \in \mathbb{R}^{d \times |L|}$ are learnable parameters. By applying the instance gating, the representative items will contribute more to make predictions about the future items and irrelevant items will be largely neglected.

Aggregation Layer. To make the item embeddings $\mathbf{S}_{i,l}^{I}$ into one group-level latent representation, we can either apply average pooling or max pooling on $\mathbf{S}_{i,l}^{I}$:

$$\mathbf{s}_{i,l}^{avg} = \text{avg\_pooling}\left(\mathbf{S}_{i,l}^{I}\right), \tag{6.3}$$

$$\mathbf{s}_{i,l}^{max} = \text{max\_pooling}\left(\mathbf{S}_{i,l}^{I}\right), \tag{6.4}$$

where $\mathbf{s}_{i,l}^{avg}, \mathbf{s}_{i,l}^{max} \in \mathbb{R}^{d}$. Since the item embeddings have manipulated by the feature-level and instance-level gating modules, the informative features and items have been selected and irrelevant ones have been eliminated. Thus, the average pooling will accumulate the informative parts in these embeddings. On the other hand, max-pooling directly selects the most representative features from each embedding to form the group-level representation.

6.3.2 Item-item Product

The relation between two single items is an important factor to model in the recommendation task and has been widely studied in many years [41,127], e.g., item-based collaborative filtering methods utilizing the rating vectors of two items to calculate the similarity. However, most of the recent works [96, 100, 101] only consider the sequential recommendation from the group-level, but do not explicitly capture the item-item relations between the items in L and the items user will interact in the future. Since strongly related item pairs will appear in L and T simultaneously. Unlike previous works, we apply the inner product between the input item embeddings and the output item embeddings to capture the item relations between L and T:

$$\sum_{\mathbf{e}_j \in \mathbf{S}_{i,l}} \mathbf{e}_j^{\top} \cdot \mathbf{Q}\,,$$

where $\mathbf{Q} \in \mathbb{R}^{d \times N}$ is the output item embeddings, the sum of multiplication results captures the accumulated item-item relation scores from each item in L to all other items.

6.3.3 Prediction Layer

After applying the hierarchical gating network to capture the short-term interests of users and item-item product to capture the relevant item pairs, we adopt the classical matrix factorization term to capture the global and long-time interests of users. Given the l-th subsequence to predict, the prediction score of user i on item j is:

$$\hat{r}_{i,j} = \mathbf{u}_i^\top \cdot \mathbf{q}_j + \mathbf{s}_{i,l}^{avg\top} \cdot \mathbf{q}_j + \frac{1}{|L|} \sum_{\mathbf{e}_k \in \mathbf{S}_{i,l}} \mathbf{e}_k^\top \cdot \mathbf{q}_j \,, \tag{6.5}$$

where $\mathbf{q}_j \in \mathbb{R}^d$ is the j-th column of the output item embedding $\mathbf{Q}$. In the prediction layer, the first term captures the user long-term interests, the second term models the user short-term interests, and the third term reflects the relations between item pairs.

6.3.4 Network Training

As the training data is from the user implicit feedback, we optimize the proposed model by the Bayesian Personalized Ranking objective [37]: optimizing the pairwise ranking between the positive and non-observing items:

$$\arg\min_{\mathbf{U},\mathbf{Q},\mathbf{E},\mathbf{\Theta}} \sum_{(i,L_i,j,k)\in\mathcal{D}} -\log \sigma(\hat{r}_{i,j} - \hat{r}_{i,k}) + \lambda(||\mathbf{U}||^2 + ||\mathbf{Q}||^2 + ||\mathbf{E}||^2 + ||\mathbf{\Theta}||^2) \,, \tag{6.6}$$

where L_i denotes one of the $|L|$ successive items of user i, j denotes the item that in T_i, and k denotes the randomly sampled negative item, $\mathbf{\Theta}$ is the parameters in the gating network, λ is the regularization parameter. By minimizing the objective function, the partial derivatives with respect to all the parameters can be computed by gradient descent with back-propagation. We apply Adam [112] to automatically adapt the learning rate during the learning procedure.

Time complexity. The computational complexity of our model for each L is mainly due to the feature gating layer and item-item product module, which is $O(|L|d^2 + |L|Nd)$ ($|L|$ is the length of L, d is the dimension of embeddings, and N is the number of items). This computational complexity makes our model scalable on large datasets. We empirically

test the training speed with other state-of-the-art methods and find that our model is faster than other methods (Section 6.4.7).

6.4 Experiments

In this section, we evaluate the proposed model with state-of-the-art methods on five real-world datasets[1].

6.4.1 Datasets

The proposed model is evaluated on five real-world datasets from various domains with different sparsities: *MovieLens-20M* [121], *Amazon-Books* and *Amazon-CDs* [122], *Goodreads-Children* and *Goodreads-Comics* [128]. *MovieLens-20M* is a user-movie dataset collected from the *MovieLens* website, where this dataset has 20 million user-movie interactions. The *Amazon-Books* and *Amazon-CDs* datasets are adopted from the Amazon review dataset[2] with different categories, i.e., CDs and Books, which cover a large amount of user-item interaction data, e.g., user ratings and reviews. *Goodreads-Children* and *Goodreads-Comics* datasets[3] are collected in late 2017 from *goodreads* website with different genres, and we use the genres of Children and Comics. In order to be consistent with the implicit feedback setting, we keep those with ratings no less than four (out of five) as positive feedback and treat all other ratings as missing entries on all datasets. To filter noisy data, we only keep the users with at least ten ratings and the items at least with five ratings. The data statistics after preprocessing are shown in Table 6.2.

For each user, we hold the 70% of interactions in the user sequence as the training set and use the next 10% of interactions as the validation set for hyper-parameter tuning. The remaining 20% constitutes the test set for reporting model performance. Note that during the testing procedure, the input sequences include the interactions in both the training set and validation set. The execution of all the models is carried out five times independently, and we report the average results.

[1]The code is available on Github: `https://github.com/allenjack/HGN`
[2]http://jmcauley.ucsd.edu/data/amazon/
[3]https://sites.google.com/eng.ucsd.edu/ucsdbookgraph/home

Table 6.2 The statistics of datasets.

Dataset	#Users	#Items	#Interactions	Density
ML20M	129,797	13,649	9,921,393	0.560%
Books	52,406	41,264	1,856,747	0.086%
CDs	17,052	35,118	472,265	0.079%
Children	48,296	32,871	2,784,423	0.175%
Comics	34,445	33,121	2,411,314	0.211%

6.4.2 Evaluation Metrics

We evaluate our model versus other methods in terms of *Recall@k* and *NDCG@k*. For each user, Recall@k (R@k) indicates what percentage of her rated items can emerge in the top k recommended items. NDCG@k (N@k) is the normalized discounted cumulative gain at k, which takes the position of correctly recommended items into account.

6.4.3 Methods Studied

To demonstrate the effectiveness of our model, we compare to the following recommendation methods.

Classical methods for implicit feedback:

- **BPRMF**, the Bayesian Personalized Ranking based matrix factorization [37], which is a classic method for learning pairwise personalized rankings from user implicit feedback. Specifically, we use BPR-MF for model learning.

State-of-the-art session-based recommendation methods:

- **GRU4Rec**, gated recurrent unit for recommendation [97], which uses recurrent neural networks to model user-item interaction sequences for session-based recommendation. Each user sequence is treated as a session.

- **GRU4Rec+**, an improved version of GRU4Rec [100], which adopts a more effective loss function and sampling strategy, and shows significant performance gains on top-K recommendation compared with GRU4Rec.

- **NextItNet**, the next item recommendation net [129], applies dilated convolutional neural networks to increase the receptive fields without relying on the pooling operation.

State-of-the-art sequential recommendation methods:

- **Caser**, convolutional sequence embedding model [96], which captures high-order Markov chains by applying convolution operations on the embeddings of the $|L|$ recent items.
- **SASRec**, self-attention based sequential model [101], which uses an attention mechanism to identify relevant items for predicting the next item.

The proposed method:

- **HGN**, the proposed model, applies a hierarchical gating network to learn the group-level representations of a sequence of items and adopts the item-item product to explicitly capture the item-item relations.

Given our extensive comparisons against the state-of-the-art methods, we omit comparisons with methods such as FMC and FPMC [94], Fossil [95], since they have been outperformed by the recently proposed Caser and SASRec.

6.4.4 Experiment Settings

In the experiments, the latent dimension of all the models is set to 50. For those session-based methods, we treat each user sequence as one session. For GRU4Rec and GRU4Rec+, we find that when the learning rate is 0.001, and batch size is 50 can achieve good performance. These two methods adopt Top1 loss and BPR-max loss, respectively. For NextItNet, we following the original settings in the paper to set the learning rate to 0.001, the kernel size to 3, the dilated levels to 1 and 2, the batch size to 32. For Caser, we follow the settings in the author-provided code to set $|L| = 5$, $|T| = 3$, the number of horizontal filters to 16, the number of vertical filters to 4, where Caser can achieve good results. For SASRec, we set the number of self-attention blocks to 2, the batch size to 128, and the maximum sequence length to 50. The network architectures of the above methods are also set the same as the original papers. The hyper-parameters are tuned using the validation set.

For HGN, we follow the same setting in Caser to set $|L| = 5$ and $|T| = 3$, where the length effects are shown in the Section 6.4.8. Hyper-parameters are tuned by grid search on the validation set. The network embedding size d is also set to 50. The learning learning

rate and λ are set to 0.001 and 0.001, respectively. The batch size is set to 4096. Our experiments are conducted with PyTorch[4] running on GPU machines (Nvidia GeForce GTX 1080 Ti).

6.4.5 Performance Comparison

The performance comparison results are shown in Figures 6.3, 6.4, 6.5, 6.6, 6.7, and Table 6.3.

Table 6.3 The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method.

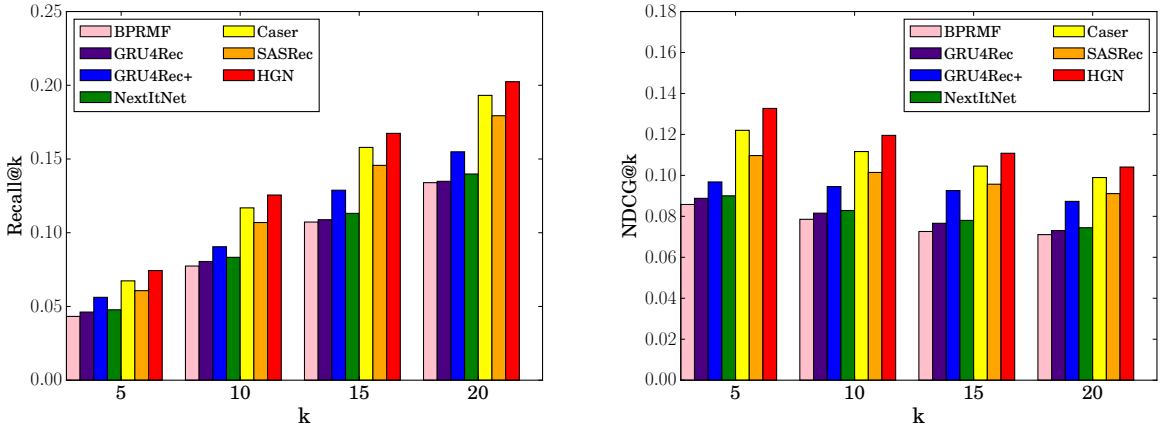
	BPRMF	**GRU4Rec**	**GRU4Rec+**	**NextItRec**	**Caser**	**SASRec**	**HGN**	**Improv.**
Recall@10								
MovieLens-20M	0.0774	0.0804	0.0904	0.0833	<u>0.1169</u>	0.1069	**0.1255***	7.36%
Amazon-Books	0.0260	0.0266	0.0301	0.0303	0.0297	<u>0.0358</u>	**0.0429*****	19.83%
Amazon-CDs	0.0269	0.0302	<u>0.0356</u>	0.0310	0.0297	0.0341	**0.0426****	19.66%
GoodReads-Children	0.0814	0.0857	0.0978	0.0879	0.1060	<u>0.1165</u>	**0.1263***	8.41%
GooReads-Comics	0.0788	0.0958	0.1288	0.1078	0.1473	<u>0.1494</u>	**0.1743*****	16.67%
NDCG@10								
MovieLens-20M	0.0785	0.0815	0.0946	0.0828	<u>0.1116</u>	0.1014	**0.1195***	7.07%
Amazon-Books	0.0151	0.0157	0.0173	0.0174	0.0216	<u>0.0240</u>	**0.0298*****	24.17%
Amazon-CDs	0.0145	0.0154	0.0171	0.0155	0.0163	<u>0.0193</u>	**0.0233****	20.73%
GoodReads-Children	0.0664	0.0715	0.0821	0.0720	0.0943	<u>0.1007</u>	**0.1130***	12.21%
GoodReads-Comics	0.0713	0.0912	0.1328	0.1171	<u>0.1629</u>	0.1592	**0.1927*****	18.29%

Observations about our model:

First, the proposed model—HGN, achieves the best performance on five datasets with all evaluation metrics, which illustrates the superiority of our model.

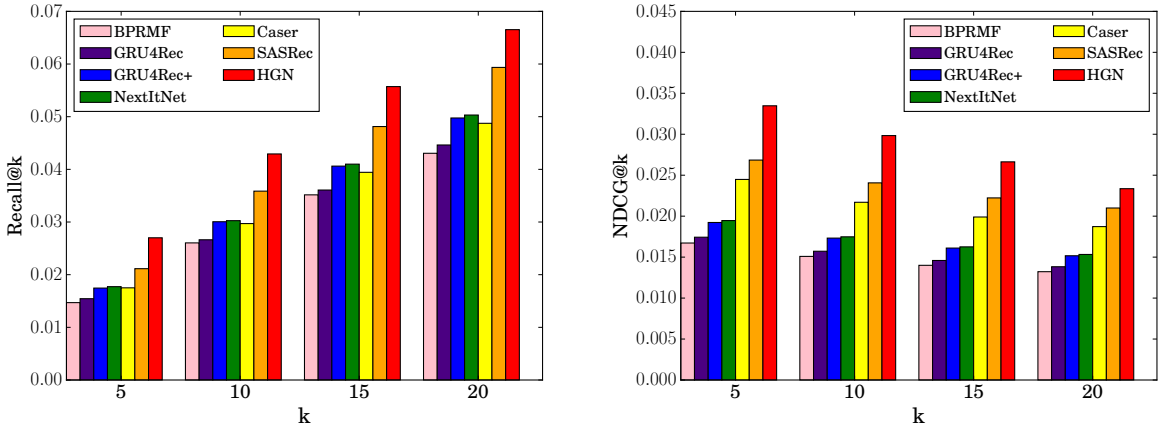
Second, HGN achieves better performance than SASRec. The reasons are three-fold: (1) SASRec only applies the instance-level selection but neglecting the feature-level one, which plays an important role in learning short-term user interests (Section 6.4.6); (2) SASRec adopts a hyper-parameter—the maximum sequence length to reduce the computation burden, where only using part of the user data may lead to the insufficient understanding of long-term user interests; (3) SASRec does not explicitly model the item-item relations between two closely relevant items, which is captured by our item-item product module.

[4]https://pytorch.org/



(a) Recall@k on MovieLens-20M (b) NDCG@k on MovieLens-20M

Figure 6.3 The performance comparison on MovieLens-20M.

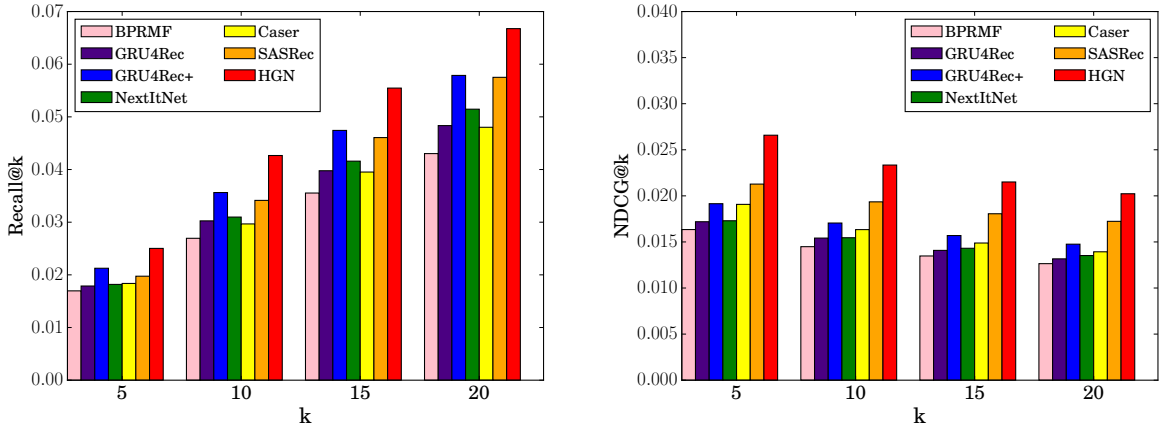


(a) Recall@k on Amazon-Books (b) NDCG@k on Amazon-Books

Figure 6.4 The performance comparison on Amazon-Books.

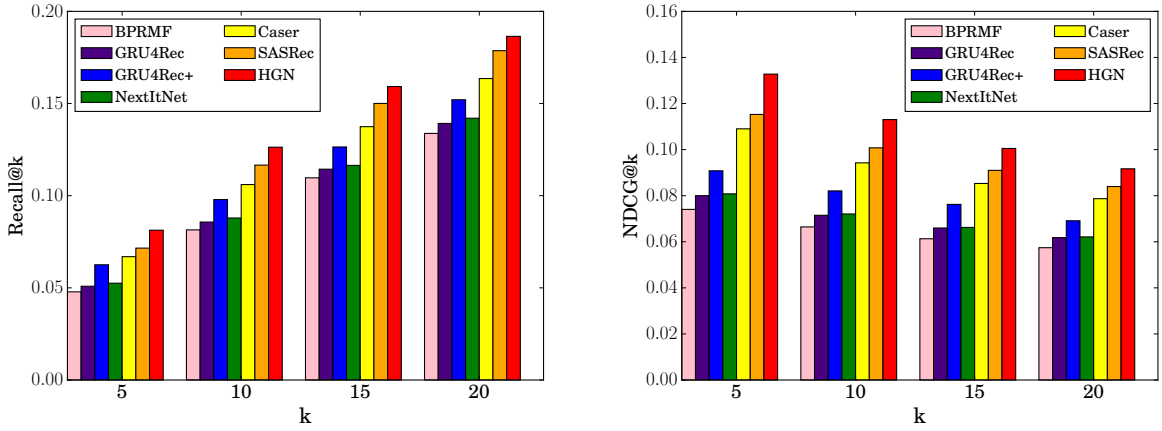
Third, HGN outperforms Caser, one major reason is that Caser only applies CNNs to learn the group-level representation of several successive items without considering the item importance for different users.

Fourth, HGN obtains better results than GRU4Rec, GRU4Rec+, and NextItNet. Two possible reasons are: (1) these models are session-based methods without explicitly modeling the long-term user interests; (2) these methods equally treat all the items in a short context, which may fail to capture the short-term user intentions.



(a) Recall@k on Amazon-CDs (b) NDCG@k on Amazon-CDs

Figure 6.5 The performance comparison on Amazon-CDs.



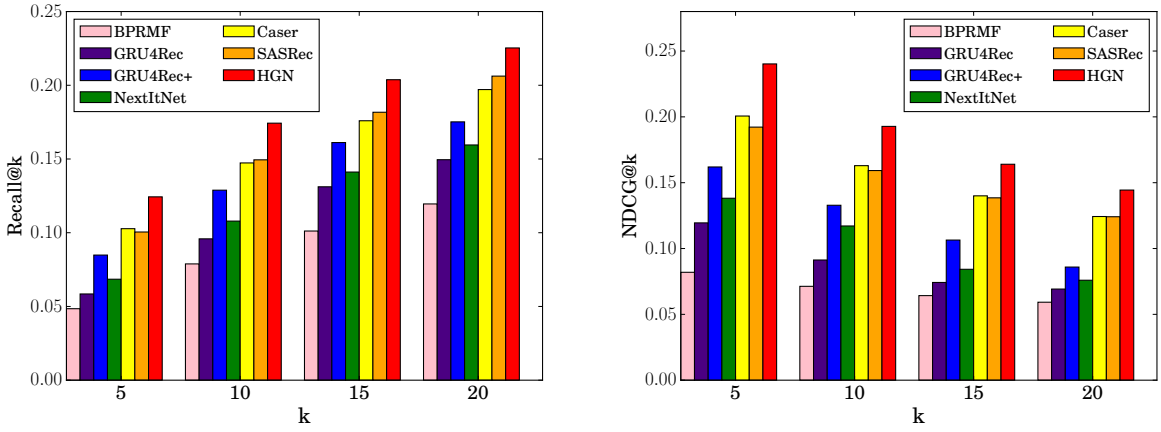
(a) Recall@k on Children (b) NDCG@k on Children

Figure 6.6 The performance comparison on Children.

Fifth, HGN outperforms BPRMF. Since BPRMF only captures the long-term interests of users, which does not incorporate the sequential patterns of user-item interactions. On the top of BPRMF, HGN adopts a hierarchical gating network to capture the sequential dynamics in the user actions and an item-item product module to explicitly capture the item-item relations, which leads to better performance.

Other observations:

First, all the results reported on MovieLens-20M, GoodReads-Children and GoodReads-



(a) Recall@k on Comics (b) NDCG@k on Comics

Figure 6.7 The performance comparison on Comics.

Comics are better than the results on other datasets, the major reason is that other datasets are more sparse and the data sparsity declines the recommendation performance.

Second, SASRec outperforms Caser on most of the datasets. The main reason is that SASRec adaptively attends items that would reflect the short-term user interests.

Third, SASRec and Caser achieve better performance than GRU4Rec, GRU4Rec+, and NextItNet in most cases. One possible reason is that SASRec and Caser both explicitly plug the user embeddings in their models, which allows the long-term user interests modeling.

Fourth, GRU4Rec+ performs better than other methods on one dataset. The reason is that GRU4Rec+ not only captures the sequential patterns in the user-item sequence but also has a promising object function—*BPR-max*.

Fifth, all the methods perform better than BPR. This illustrates that only effectively modeling the long-term user interests is not sufficient to capture the user sequential behaviors.

6.4.6 Ablation Analysis

To verify the effectiveness of the proposed feature gating, instance gating, and item-item product modules, we conduct an ablation analysis in Table 6.4 to demonstrate the importance each module contributes to the HGN model. In (1), we utilize only the BPR matrix factorization without any other components. In (2), we only incorporate the feature gat-

Table 6.4 The ablation analysis on GoodReads-Comics and Amazon-Books datasets. F denotes the feature gating module, I denotes the instance gating module, *avg* denotes the average pooling, and *max* denotes the max pooling.

Architecture	*Comics*		*Books*	
	R@10	N@10	R@10	N@10
(1) BPR	0.0911	0.0802	0.0310	0.0177
(2) BPR+F+avg	0.1555	0.1624	0.0361	0.0266
(3) BPR+F+max	0.1456	0.1550	0.0355	0.0240
(4) BPR+I+avg	0.1538	0.1591	0.0351	0.0254
(5) BPR+I+max	0.1489	0.1585	0.0329	0.0241
(6) BPR+GRU	0.1456	0.1581	0.0289	0.0216
(7) BPR+CNN	0.1305	0.1387	0.0278	0.0207
(8) BPR+F+I+avg	0.1635	0.1791	0.0391	0.0250
(9) BPR+F+I+max	0.1569	0.1658	0.0355	0.0234
(10) HGN	**0.1743**	**0.1927**	**0.0429**	**0.0298**

ing and apply the average pooling on the embeddings after the feature gating, on the top of (1). In (3), we replace the average pooling in (2) with max pooling. In (4), we only include the instance gating and apply the average pooling on the top of (1). In (5), we replace the average pooling in (4) with max-pooling. In (6), we adopt a recurrent neural network structure—gated recurrent unit (GRU) [125] to learn the group-level representations of items. In (7), we replace the GRU in (6) with a convolutional neural network (CNN), where the structure and hyper-parameters are set the same in Caser [96]. In (8), we both apply the feature and instance gating with average pooling. In (9), we replace the average pooling with max pooling. In (10), we present the overall HGN model to show the significance of the item-item product module.

From the results shown in Table 6.4, we have some observations. First, from (1) and all others, we can observe that the conventional BPR matrix factorization to capture the long-term user interests cannot effectively model the short-term user interests. Second, from (2), (3), (4) and (5), the feature gating seems to achieve slightly better results than the instance gating. And the average pooling is slightly better than the max pooling, one possible reason is that the average pooling makes the representative item features accumulated, which results in a more effective representation of a group of $|L|$ successive items. Third, from (6), (7), and (8), we observe that our hierarchical gating network

achieves better performance than GRU and CNN but with *fewer* learnable parameters[5] (if we set the item embedding size to 50 ($d = 50$), then the number of learnable parameters of our hierarchical gating network is 5,350, the number of parameters of the one-recurrent-layer GRU is 15,300, the number of parameters of the CNN in [96] is 26,154). This result demonstrates that the proposed hierarchical gating network can effectively capture the sequential patterns in the user-item interaction sequence. Lastly, from (1), (8), and (9), we observe that by incorporating the item-item product, the performance further improves. The results demonstrate that explicitly capturing the relations between the items users accessed and those items users may interact with in the future can provide a significant supplement to model the user sequential dynamics.

6.4.7 Training Efficiency

In this section, we evaluate the training efficiency with other state-of-the-art methods in terms of the training speed (time taken for one epoch of training). Since GRU4Rec+ has been compared with SASRec in [101], we omit the training time comparison with GRU4Rec+. To make a fair comparison, we set the max sequence length of SASRec as 300 to cover more than 95% of the sequence. All the experiments are conducted on a single GPU of Nvidia GeForce GTX 1080 Ti. All the compared methods are executed 20 epochs and we report the average computation time, which is shown in Table 6.5. Note that the time reported only includes the training time of models without including the negative sampling time.

Table 6.5 The training time per epoch comparison on five datasets in terms of seconds.

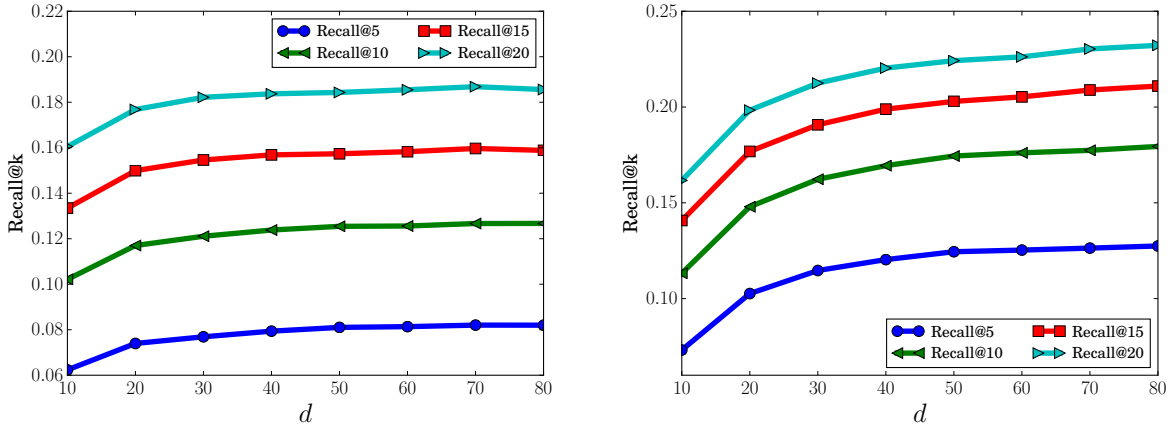
	CDs	Books	ML20M	Children	Comics
HGN	**0.957s**	**2.086s**	**28.304s**	**3.496s**	**2.228s**
SASRec	2.242s	16.154s	39.937s	14.913s	10.468s
Caser	5.063s	17.577s	63.702s	28.593s	25.657s

From the results in Table 6.5, we can observe that HGN yields the fastest training speed on all datasets. As we have discussed in section 6.3.4, our model has less item complexity than SASRec, which is $O(N^2d + Nd^2)$. Thus, our proposed model has better training efficiency both theoretically and practically.

[5]We verified the number of parameters of all three models by the *named_parameters()* function provided by PyTorch.

Table 6.6 The effect of the length $|L|$ and $|T|$.

Settings	*CDs*		*Comics*	
	R@5	R@10	R@5	R@10
$\|L\|$=3, $\|T\|$=1	0.0260	0.0415	0.1202	0.1684
$\|L\|$=3, $\|T\|$=2	**0.0291**	0.0448	0.1275	0.1758
$\|L\|$=3, $\|T\|$=3	0.0289	0.0450	**0.1296**	**0.1793**
$\|L\|$=5, $\|T\|$=1	0.0254	0.0417	0.1155	0.1645
$\|L\|$=5, $\|T\|$=2	0.0261	0.0432	0.1215	0.1711
$\|L\|$=5, $\|T\|$=3	0.0290	**0.0456**	0.1238	0.1738
$\|L\|$=8, $\|T\|$=1	0.0220	0.0372	0.1083	0.1566
$\|L\|$=8, $\|T\|$=2	0.0248	0.0401	0.1142	0.1636
$\|L\|$=8, $\|T\|$=3	0.0260	0.0413	0.1160	0.1658



(a) d on Children (b) d on Comics

Figure 6.8 The dimension variations of embeddings.

6.4.8 The Sensitivity of Hyper-parameters

We present the effect of two hyper-parameters: the dimension of the item embeddings d and the length of successive items $|L|$ and $|T|$. The effects of these two parameters are shown in Figure 6.8 and Table 6.6. Due to the space limit, we only present the effects on two datasets, the parameter effects on other datasets have similar trends.

The variation of d is shown in Figure 6.8. We can observe that a small dimension of item embeddings is not sufficient to express the latent features of items. By increasing the dimension of item embeddings, the model has more capacity to model the complex features

of items. With the increase of d, the model performance largely improves and becomes steady.

The variation of $|L|$ and $|T|$ is shown in Table 6.6. We observe that when $|L|$ is fixed, a larger value of $|T|$, i.e. 3, can achieve better performance. This may illustrate that a group of $|L|$ items may determine several items that the user will interact with in the near future. We also observe that smaller $|L|$ has better results than larger ones. One possible reason is that larger $|L|$ may include too many irrelevant items for predicting future items.

6.5 Summary

In this chapter, we propose a hierarchical gating network with an item-item product module for the sequential recommendation. The model adopts a feature gating module and an instance gating module to control what item features can be passed to downstream layers, where informative latent features and items can be selected. Moreover, we apply an item-item product module to capture the relations between closely relevant items. Experimental results on five real-world datasets validate the performance of our model over many baselines and demonstrate the effectiveness of the gating and item-item product modules.

Chapter 7

Neural Networks for Adaptive Learning: Learning Personalized Hyper-parameters

7.1 Introduction

In the personalized recommender system, modeling the user-item interaction lies at the core. There are two common ways adopted in recent recommendation models to infer user preference: matrix factorization (MF) and multi-layer perceptrons (MLPs). MF-based methods (e.g., [20, 37]) apply the inner product between latent factors of users and items to predict the user preferences for different items. The latent factors strive to depict the user preference and item properties, respectively, in the latent space. In contrast, MLP-based methods (e.g., [9, 130]) adopt (deep) neural networks to learn non-linear user-item relationships. Compared to MF approaches that are limited by the inner product calculation, MLPs can generate better latent feature combinations between the embeddings of users and items [9].

However, both MF-based and MLP-based methods violate the triangle inequality [131], and as a result may fail to capture the fine-grained preference information [22]. As a concrete example in [69], if a user accessed two items, MF or MLP-based methods will put both items close to the user, but will not necessarily put these two items close to each other, even if they share similar properties.

To address the limitations of MF and MLP-based methods, the metric (distance) learning approaches have been utilized in the recommendation model [22, 66, 69, 71], since the distance naturally satisfies the triangle inequality. These approaches project users and items into a low-dimensional metric space, where the user preference is measured by the distance to items. Specifically, CML [22] and LRML [66] are two representative models. CML minimizes the Euclidean distance between users and accessed items of users, which benefits fine-grained user-user/item-item similarity learning. LRML incorporates a memory network [67] to introduce additional capacity to learn relations between users and items in the metric space.

In spite of the fact that previous distance-based methods have achieved satisfactory performance, we argue that several avenues are existing for further improving performance. First, previous distance-based methods [22, 66, 69, 71] learn the user and item embeddings in a deterministic manner without considering the uncertainty. Completely relying on learned deterministic embeddings can lead to an incorrect interpretation of user preferences. Second, most of the existing methods [22, 66, 69] adopt the margin ranking loss (hinge loss) with a fixed margin, where the margin is a hyper-parameter. We argue that the margin value should be adaptive and relevant to corresponding training samples. Furthermore, different training phases may need different magnitudes of margin values. Setting a fixed value may not be an optimal choice. Third, previous distance-based methods [22, 66, 69, 71] do not explicitly model user-user and item-item relationships. Closely-related users are more likely to have the same preferences, and it is likely that a user would favor them if two items have similar attributes. When inferring the preference of a user, we should explicitly take into account the user-user and item-item similarities.

To address the aforementioned shortcomings, we propose a Probabilistic Metric Learning model with the Adaptive Margin (PMLAM) for the top-K recommendation. PMLAM consists of three major components: (1) a user-item interaction module, (2) an adaptive margin generation module, and (3) a user-user/item-item relation modeling module. To catch the uncertainties of the learned user and item embeddings, each user or item is parametrized with a Gaussian distribution, where the associated distribution parameters are learned from our model. In the user-item interaction module, we take the Wasserstein distance to calculate the distance between users and items, taking into account not only the means but also the uncertainties. In the adaptive margin generation module, we model the learning of adaptive margins as a bilevel (inner and outer) optimization problem [132],

where we build a proxy function to explicitly link the learning of margin related parameters with the outer objective function. Specifically, we apply a two-layer MLP to generate the adaptive margins according to different training triplets. In the user-user and item-item relation modeling module, we incorporate two margin ranking losses with adaptive margins for similar user-pairs and item-pairs, respectively, to explicitly encourage similar users or items to be mapped closer to one another in the latent space. We extensively evaluate our model by comparing with many state-of-the-art methods, using two performance metrics on five real-world datasets. The experimental results not only demonstrate the improvements of our model over other baselines but also show the effectiveness of the proposed modules.

To summarize, the major highlights of this chapter are:

- To model the uncertainties in the learned user/item embeddings, we represent each user and item as a Gaussian distribution. The Wasserstein distance is leveraged to measure the user preference on items while simultaneously handling the uncertainty.
- To generate an adaptive margin, we treat the margin generation process as a bilevel optimization problem, where we build a proxy function to explicitly update the margin generation-related parameters.
- To explicitly model the user-user and item-item relationships, we apply two margin ranking losses with adaptive margins to make similar users and items map closer to one another in the latent space, respectively.
- Experiments on five real-world datasets show that the proposed PMLAM model significantly outperforms the state-of-the-art methods for the top-K recommendation task.

7.2 Problem Formulation

The recommendation task considered in this chapter takes as input the user implicit feedback. For each user i, the user preference data is represented by a set that includes the items she preferred, e.g., $\mathcal{D}_i = \{I_1, ..., I_j, ..., I_{|\mathcal{D}_i|}\}$, where I_j is an item index in the dataset. The top-K recommendation task in this chapter is formulated as: given the training item set $\mathcal{S}_i$, and the non-empty test item set $\mathcal{T}_i$ (requiring that $\mathcal{S}_i \cup \mathcal{T}_i = \mathcal{D}_i$ and $\mathcal{S}_i \cap \mathcal{T}_i = \emptyset$) of user i, the model must recommend an ordered set of items $\mathcal{X}_i$ such that $|\mathcal{X}_i| \leq K$ and

$\mathcal{X}_i \cap \mathcal{S}_i = \emptyset$. Then the recommendation quality is evaluated by a matching score between $\mathcal{T}_i$ and $\mathcal{X}_i$, such as Recall@K.

7.3 Methodology

In this section, we present the proposed model shown in Figure 7.1. We first introduce the user-item interaction module, which captures the user-item interactions by calculating the Wasserstein distance between users' and items' distributions. Then we describe the adaptive margin generation module, which generates adaptive margins during the training process. Next, we present the user-user and item-item relation modeling modules. Lastly, we specify the objective function and explain the training process that is used to optimize the proposed model.

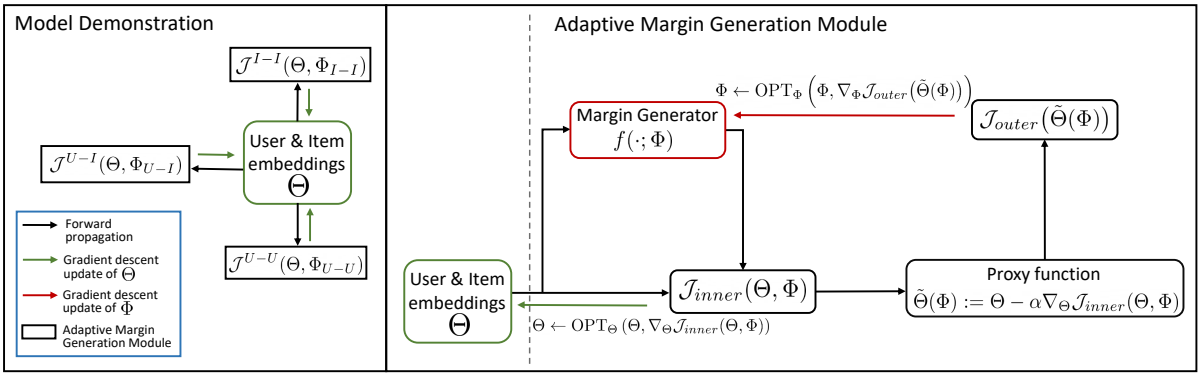


Figure 7.1 The demonstration of the proposed model. $\mathcal{J}^{U-I}$ denotes the combined optimization regarding $\mathcal{J}^{U-I}_{inner}$ and $\mathcal{J}^{U-I}_{outer}$. $\mathcal{J}^{U-U}$ and $\mathcal{J}^{I-I}$ follow the same manner with $\mathcal{J}^{U-I}$.

7.3.1 Wasserstein Distance for Interactions

Previous works [22,66] use the user and item embeddigns in a deterministic manner and do not measure or learn the uncertainties of user preferences and item properties. Motivated by probabilistic matrix factorization (PMF) [44], we represent each user or item as a single Gaussian distribution. In contrast to PMF, which applies Gaussian priors on user and item embeddings, users and items in our model are parameterized by Gaussian distributions, where the means μ and covariances Σ are directly learned. Specifically, the latent factors

of user i and item j are represented as:

$$\begin{aligned} \mathbf{u}_i &\sim \mathcal{N}(\mu_i^{(U)}, \mathbf{\Sigma}_i^{(U)}) \,, \\ \mathbf{v}_j &\sim \mathcal{N}(\mu_j^{(I)}, \mathbf{\Sigma}_j^{(I)}) \,. \end{aligned} \tag{7.1}$$

Here $\mu_i^{(U)}$ and $\mathbf{\Sigma}_i^{(U)}$ are the learned mean vector and covariance matrix of user i, respectively; $\mu_j^{(I)}$ and $\mathbf{\Sigma}_j^{(I)}$ are the learned mean vector and covariance matrix of item j. To limit the complexity of the model and reduce the computational overhead, we assume that the embedding dimensions are uncorrelated. Thus, $\mathbf{\Sigma}$ is a diagonal covariance matrix that can be represented as a vector. Specifically, $\mu \in \mathbb{R}^h$ and $\mathbf{\Sigma} \in \mathbb{R}^h$, where h is the dimension of the latent space.

Since the users and items are represented by probabilistic distributions, the widely used distance metrics for deterministic embeddings like Euclidean distance may not properly measure the distance between distributions. Thus, a distance measure between distributions is needed. Among the commonly used distance metric between distributions, we adopt the Wasserstein distance to measure the user preference on items. The reasons are twofold: (i) the Wasserstein distance satisfies all the properties a distance should have; and (ii) the Wasserstein distance has a simple form when calculating the distance between Gaussian distributions [133]. Formally, the p-th Wasserstein distance between two probability measures μ and ν on a Polish metric space [134] $(\mathcal{X}, d)$ is defined as [135, 136]:

$$\mathcal{W}_p(\mu, \nu) := \Big(\inf_{J \in \mathcal{J}(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{X}} d(x,y)^p dJ(x,y) \Big)^{\frac{1}{p}} ,$$

where $d(\cdot,\cdot)^p$ is an arbitrary distance with p^{th} moment [137] for a deterministic variable, $p \in [1, +\infty)$; and $\mathcal{J}(\mu, \nu)$ denotes the set of all measures J on $\mathcal{X} \times \mathcal{X}$ which admit μ and ν as marginals. When $p \geq 1$, the p-th Wasserstein distance preserves all properties of a metric, including both symmetry and the triangle inequality.

The calculation of the general Wasserstein distance is computation-intensive [138]. To reduce the computational cost, we use Gaussian distributions for the latent representations of users and items. Then when $p = 2$, the 2-nd Wasserstein distance (abbreviated as $\mathcal{W}_2$) has a closed form solution, thus making the calculation process much faster. Specifically, we

have the following formula to calculate the $\mathcal{W}_2$ distance between user i and item j [136,139]:

$$\begin{aligned}&\mathcal{W}_2\left(\mathcal{N}(\mu_i^{(U)}, \mathbf{\Sigma}_i^{(U)}), \mathcal{N}(\mu_j^{(I)}, \mathbf{\Sigma}_j^{(I)})\right)^2 \\ &= ||\mu_i^{(U)} - \mu_j^{(I)}||_2^2 + \text{trace}\left(\mathbf{\Sigma}_i^{(U)} + \mathbf{\Sigma}_j^{(I)} - 2\left((\mathbf{\Sigma}_i^{(U)})^{\frac{1}{2}}\mathbf{\Sigma}_j^{(I)}(\mathbf{\Sigma}_i^{(U)})^{\frac{1}{2}}\right)^{\frac{1}{2}}\right).\end{aligned} \tag{7.2}$$

In our setting, we focus on diagonal covariance matrices, thus $\mathbf{\Sigma}_i^{(U)}\mathbf{\Sigma}_j^{(I)} = \mathbf{\Sigma}_j^{(I)}\mathbf{\Sigma}_i^{(U)}$. For simplicity, we use $\mathcal{W}_2(i,j)^2$ to denote the left hand side of Eq. 7.2. Then Eq. 7.2 can be simplified as:

$$\mathcal{W}_2(i,j)^2 = ||\mu_i^{(U)} - \mu_j^{(I)}||_2^2 + ||(\mathbf{\Sigma}_i^{(U)})^{\frac{1}{2}} - (\mathbf{\Sigma}_j^{(I)})^{\frac{1}{2}}||_2^2. \tag{7.3}$$

According to the above equation, the time complexity of calculating $\mathcal{W}_2$ distance between the latent representations of users and items is linear with the embedding dimension.

7.3.2 Adaptive Margin in Margin Ranking Loss

To learn the distance-based model, most of the existing works [22, 66] apply the margin ranking loss to measure the user preference difference between positive items and negative items. Specifically, the margin ranking loss makes sure the distance between a user and a positive item is less than the distance between the user and a negative item by a fixed margin $m > 0$. The loss function is:

$$\mathcal{L}_{Fix}(i,j,k;\Theta) = [d(i,j;\Theta)^2 - d(i,k;\Theta)^2 + m]_+, \tag{7.4}$$

where $j \in \mathcal{S}_i$ is an item that user i has accessed, and $k \notin \mathcal{S}_i$ is a randomly sampled item treated as the negative example, and $[z]_+ = \max(z, 0)$. Thus, (i,j,k) represents a training triplet.

The safe margin m in the margin ranking loss is a crucial hyper-parameter that has a major impact on the model performance. A fixed margin value may not achieve satisfactory performance. First, using a fixed value does not allow for adaptation to distinguish the properties of the training triplets. For example, some users have broad interests, so the margins for these users should not be so large as to make potential preferred items too far from the user. Other users have very focused interests, and it is desirable to have a larger margin to avoid recommending items that are not directly within the focus. Second, in different training phases, the model may need different magnitudes of margins. For

instance, in the early stages of training, the model is not reliable enough to make strong predictions on user preferences, and thus imposing a large margin risks pushing potentially positive items too far from a user. Third, to achieve satisfactory performance, the selection of a fixed margin involves tedious hyper-parameter tuning. Based on these considerations, we conclude that setting a fixed margin value for all training triplets may limit the model expressiveness.

To address the problems outlined above, we propose an adaptive margin generation scheme which generates margins according to the training triplets. Formally, we formulate the margin ranking loss with an adaptive margin as:

$$\mathcal{L}_{Ada}(i,j,k;\Theta,\Phi) = [d(i,j;\Theta)^2 - d(i,k;\Theta)^2 + f(i,j,k;\Phi)]_+ . \tag{7.5}$$

Here $f(i,j,k;\Phi)$ is a function that generates the specific margin based on the corresponding user and item embeddings and Φ is the learnable set of parameters associated with $f(\cdot)$. Then we could consider optimizing Θ and Φ simultaneously:

$$\Theta^* = \underset{\Theta,\Phi}{\operatorname{argmin}} \sum_i \sum_{j\in\mathcal{S}_i} \sum_{k\notin\mathcal{S}_i} \mathcal{L}_{Ada}(i,j,k;\Theta,\Phi) . \tag{7.6}$$

Unfortunately, directly minimizing the objective function as in Eq. 7.6 does not achieve the desired purpose of generating suitable adaptive margins. Since the margin-related term explicitly appears in the loss function, constantly decreasing the value of the generated margin is the straightforward way to reduce the loss. As a result all generated margins have very small values or are set to zero, leading to unsatisfactory results.

Bilevel Optimization

We model the learning of recommendation models and the generation of adaptive margins as a bilevel optimization problem [140]:

$$\begin{aligned} &\min_{\Phi} \mathcal{J}_{outer}\left(\Theta^*(\Phi)\right) := \sum_i \sum_{j\in\mathcal{S}_i} \sum_{k\notin\mathcal{S}_i} \mathcal{L}_{Fix}\left(i,j,k;\Theta^*(\Phi)\right) \\ &\text{s.t. } \Theta^*(\Phi) = \underset{\Theta}{\operatorname{argmin}}\, \mathcal{J}_{inner}(\Theta,\Phi) := \sum_i \sum_{j\in\mathcal{S}_i} \sum_{k\notin\mathcal{S}_i} \mathcal{L}_{Ada}(i,j,k;\Theta,\Phi) . \end{aligned} \tag{7.7}$$

Here Θ contains the model parameters μ and $\mathbf{\Sigma}$. The objective function $\mathcal{J}_{inner}$ attempts to minimize $\mathcal{L}_{Ada}$ with respect to Θ while the objective function $\mathcal{J}_{outer}$ optimizes $\mathcal{L}_{Fix}$ with respect to Φ through $\Theta^*(\Phi)$. For simplicity, the m of $\mathcal{L}_{Fix}$ in $\mathcal{J}_{outer}$ is set to 1 for guiding the learning of $f(\cdot;\Phi)$. Thus, we can have an alternating optimization to learn Θ and Φ:

- Θ *update phase* (Inner Optimization): Fix Φ and optimize Θ.
- Φ *update phase* (Outer Optimization): Fix Θ and optimize Φ.

Approximate Gradient Optimization

As most existing models utilize gradient-based methods for optimization, a simple approximation strategy with less computation is introduced as follows:

$$\nabla_{\Phi}\mathcal{J}_{outer}\left(\Theta^*(\Phi)\right) \approx \nabla_{\Phi}\mathcal{J}_{outer}\left(\Theta - \alpha\nabla_{\Theta}\mathcal{J}_{inner}(\Theta,\Phi)\right) . \tag{7.8}$$

In this expression, Θ denotes the current parameters including μ and $\mathbf{\Sigma}$, and α is the learning rate for one step of inner optimization. Related approximations have been validated in [11, 141]. Thus, we can define a proxy function to link Φ with the outer optimization:

$$\tilde{\Theta}(\Phi) := \Theta - \alpha\nabla_{\Theta}\mathcal{J}_{inner}(\Theta,\Phi) . \tag{7.9}$$

For simplicity, we use two optimizers OPT_{Θ} and OPT_{Φ} to update Θ and Φ, respectively. The iterative procedure is shown in Alg. 2.

Algorithm 2: Iterative Optimization Procedure

Initialize optimizers OPT_{Θ} and OPT_{Φ} ;
while *not converged* **do**
  Θ Update (fix Φ^t):
    $\Theta^{t+1} \longleftarrow \mathrm{OPT}_{\Theta}\left(\Theta^t, \nabla_{\Theta^t}\mathcal{J}_{inner}(\Theta^t,\Phi^t)\right)$;
  Proxy:
    $\tilde{\Theta}^{t+1}(\Phi^t) := \Theta^t - \alpha\nabla_{\Theta^t}\mathcal{J}_{inner}(\Theta^t,\Phi^t)$;
  Φ Update (fix Θ^t):
    $\Phi^{t+1} \longleftarrow \mathrm{OPT}_{\Phi}\left(\Phi^t, \nabla_{\Phi^t}\mathcal{J}_{outer}\left(\tilde{\Theta}^{t+1}(\Phi^t)\right)\right)$;
end

The design of $f(\cdot;\Phi)$

We parameterize $f(i,j,k;\Phi)$ with a neural network to generate the margin based on (i,j,k):

$$\begin{aligned} \mathbf{z}_{ijk} &= \tanh(\mathbf{W}_1 \cdot \mathbf{s}_{ijk} + \mathbf{b}_1)\,, \\ m_{ijk} &= \text{softplus}(\mathbf{W}_2 \cdot \mathbf{z}_{ijk} + \mathbf{b}_2)\,, \end{aligned} \tag{7.10}$$

where $\mathbf{W}_*$ and $\mathbf{b}_*$ are learnable parameters in $f(\cdot;\Phi)$, $\mathbf{s}_{ijk}$ is the input to generate the margin, and $m_{ijk} \in \mathbb{R}$ is the generated margin of (i,j,k). The activation function *softplus* guarantees $m_{ijk} > 0$. To make $\mathbf{s}_{ijk}$ be discriminative to reflect the relation between (i,j,k) and m_{ijk}, we find the following form of $\mathbf{s}_{ijk}$ can be a fine-grained indicator:

$$\begin{aligned} \chi(\mathbf{u}_i, \mathbf{v}_j) &= [(u_{i,1} - v_{j,1})^2, (u_{i,2} - v_{j,2})^2, ..., (u_{i,h} - v_{j,h})^2]^\top \\ \mathbf{s}_{ijk} &= [\chi(\mathbf{u}_i, \mathbf{v}_j);\ \chi(\mathbf{u}_i, \mathbf{v}_k);\ \chi(\mathbf{u}_i, \mathbf{v}_k) \ominus \chi(\mathbf{u}_i, \mathbf{v}_j)]\,. \end{aligned} \tag{7.11}$$

Here $\chi(\mathbf{u}_i, \mathbf{v}_j) \in \mathbb{R}^h$ is introduced to mimic the calculation of Euclidean distance without summing over all dimensions. $\ominus$ denotes element-wise subtraction and $[...;...]$ denotes the concatenation operation. To improve the robustness of $f(\cdot;\Phi)$, we take as inputs the sampled embeddings $\mathbf{u}_i$ and $\mathbf{v}_j$. In order to perform backpropagation from $\mathbf{u}_i$ and $\mathbf{v}_j$, we adopt the reparameterization trick [118] for Eq. 7.1:

$$\begin{aligned} \mathbf{u}_i &= \mu_i^{(U)} + (\mathbf{\Sigma}_i^{(U)})^{\frac{1}{2}} \odot \epsilon_1\,, \\ \mathbf{v}_j &= \mu_j^{(I)} + (\mathbf{\Sigma}_j^{(I)})^{\frac{1}{2}} \odot \epsilon_2\,, \end{aligned} \tag{7.12}$$

where $\epsilon_1, \epsilon_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$ and $\odot$ is the element-wise muliplication.

Discussion. Note that our adaptive margin generation scheme is a general module, which can be plugged into many applications with margin ranking loss, such as computer vision [142, 143] and knowledge graphs [144, 145].

7.3.3 User-User and Item-Item Relations

It is important to model the relationships between pairs of users or pairs of items when developing recommender systems and strategies for doing so effectively have been studied for many years [41, 127, 146]. For example, item-based collaborative filtering methods use item rating vectors to calculate the similarities between the items. Closely-related users

or items may share the same interests or have similar attributes. For a certain user, items similar to the user's preferred items are potential recommendation candidates.

Despite this intuition, previous distance-based recommendation methods [22,66] do not explicitly take the user-user or item-item relationships into consideration. As a result of relying primarily on user-item information, the systems may fail to generate appropriate user-user or item-item distances. To model the relationships between similar users or items, we employ two ranking margin losses with adaptive margins to encourage similar users or items to be mapped closer together in the latent space. Formally, the similarities between users or items are calculated from the user implicit feedback, which can be represented by a binary user-item interaction matrix. We set a threshold on the calculated similarities to identify the similar users and items for a specific user i and item j, respectively, denoted as $\mathcal{N}_i^U$ and $\mathcal{N}_j^I$. We adopt the following losses for user pairs and item pairs, respectively:

$$\begin{cases} \mathcal{J}_{outer}^{U-U} & := \sum_i \sum_{p \in \mathcal{N}_i^U} \sum_{q \notin \mathcal{N}_i^U} \mathcal{L}_{Fix}(i,p,q;\tilde{\Theta}_{U-U}^{t+1}) \,, \\ \mathcal{J}_{inner}^{U-U} & := \sum_i \sum_{p \in \mathcal{N}_i^U} \sum_{q \notin \mathcal{N}_i^U} \mathcal{L}_{Ada}(i,p,q;\Theta^t,\Phi_{U-U}^t) \,, \end{cases} \tag{7.13}$$

$$\begin{cases} \mathcal{J}_{outer}^{I-I} & := \sum_j \sum_{p \in \mathcal{N}_j^I} \sum_{q \notin \mathcal{N}_j^I} \mathcal{L}_{Fix}(j,p,q;\tilde{\Theta}_{I-I}^{t+1}) \,, \\ \mathcal{J}_{inner}^{I-I} & := \sum_j \sum_{p \in \mathcal{N}_j^I} \sum_{q \notin \mathcal{N}_j^I} \mathcal{L}_{Ada}(j,p,q;\Theta^t,\Phi_{I-I}^t) \,, \end{cases} \tag{7.14}$$

where q is a randomly sampled user in Eq. 7.13 and a randomly sampled item in Eq. 7.14. $U-U$ denotes the user-user relation and $I-I$ denotes the item-item relation. We use Φ_{U-U}^t and Φ_{I-I}^t to update Θ_{U-U}^{t+1} and Θ_{I-I}^{t+1}, respectively, which are same as in Alg. 2. We denote the indicator in Eq. 7.11 as $\mathbf{s}_{ijk}^{U-I}$, then we generate $\mathbf{s}_{ijq}^{U-U}$ and $\mathbf{s}_{ijq}^{I-I}$ following the procedure described by Eq. 7.11.

7.3.4 Model Training

Let us denote the losses $\mathcal{J}_{inner}^{U-I}$ and $\mathcal{J}_{outer}^{U-I}$ to capture the interactions between users and items. Then we combine the loss functions presented in Section 7.3.3 to optimize the proposed model:

$$\begin{aligned} \mathcal{J}_{inner} &= \mathcal{J}_{inner}^{U-I} + \mathcal{J}_{inner}^{U-U} + \mathcal{J}_{inner}^{I-I} \,, \\ \mathcal{J}_{outer} &= \mathcal{J}_{outer}^{U-I} + \mathcal{J}_{outer}^{U-U} + \mathcal{J}_{outer}^{I-I} + \lambda ||\Phi||_F^2 \,, \end{aligned} \tag{7.15}$$

where λ is a regularization parameter. We follow the same training scheme of Section 7.3.2 to train Eq. 7.15. To mitigate the curse of dimensionality issue [147] and prevent overfit-

ting, we bound all the user/item embeddings within a unit sphere after each mini-batch training: $||\mu|| \leqslant 1$ and $||\mathbf{\Sigma}|| \leqslant 1$. When minimizing the objective function, the partial derivatives with respect to all the parameters can be computed by gradient descent with back-propagation. We apply the Adam [112] algorithm to automatically adapt the learning rate during the learning procedure.

Recommendation Phase. In the testing phase, for a certain user i, we compute the distance $\mathcal{W}_2(i,j)^2$ between user i and each item j in the dataset. Then the items that are not in the training set and have the shortest distances are recommended to user i.

7.4 Experiments

In this section, we evaluate the proposed model, comparing it with the state-of-the-art methods on five real-world datasets.

7.4.1 Datasets

The proposed model is evaluated on five real-world datasets from various domains with different sparsities: *Books*, *Electronics* and *CDs* [122], *Comics* [128] and *Gowalla* [73]. The *Books*, *Electronics* and *CDs* datasets are adopted from the Amazon review dataset with different categories, i.e., books, electronics and CDs. These datasets include a significant amount of user-item interaction data, e.g., user ratings and reviews. The *Comics* dataset was collected in late 2017 from the *GoodReads* website with different genres, and we use the genres of comics. The *Gowalla* dataset was collected worldwide from the Gowalla website (a location-based social networking website) over the period from February 2009 to October 2010. In order to be consistent with the implicit feedback setting, we retain any ratings no less than four (out of five) as positive feedback and treat all other ratings as missing entries for all datasets. To filter noisy data, we only include users with at least ten ratings and items with at least five ratings. Table 7.1 shows the data statistics.

We employ five-fold cross-validation to evaluate the proposed model. For each user, the items she accessed are randomly split into five folds. We pick one fold each time as the ground truth for testing, and the remaining four folds constitute the training set. The average results over the five folds are reported.

Table 7.1 The statistics of the datasets.

Dataset	#Users	#Items	#Interactions	Density
Books	77,754	66,963	2,517,343	0.048%
Electronics	40,358	28,147	524,906	0.046%
CDs	24,934	24,634	478,048	0.079%
Comics	37,633	39,623	2,504,498	0.168%
Gowalla	64,404	72,871	1,237,869	0.034%

7.4.2 Evaluation Metrics

We evaluate all models in terms of *Recall@k* and *NDCG@k*. For each user, Recall@k (R@k) indicates the percentage of her rated items that appear in the top k recommended items. NDCG@k (N@k) is the normalized discounted cumulative gain at k, which takes the position of correctly recommended items into account.

7.4.3 Methods Studied

To demonstrate the effectiveness of our model, we compare to the following recommendation methods.
Classical methods for implicit feedback:

- **BPRMF**, Bayesian Personalized Ranking-based Matrix Factorization [37], which is a classic method for learning pairwise personalized rankings from user implicit feedback.

Classical neural-based recommendation methods:

- **NCF**, Neural Collaborative Filtering [9], which combines the matrix factorization (MF) model with a multi-layer perceptron (MLP) to learn the user-item interaction function.
- **DeepAE**, the deep autoencoder [26], which utilizes a three-hidden-layer autoencoder with a weighted loss function to capture the user implicit feedback.

State-of-the-art distance-based recommendation methods:

- **CML**, Collaborative Metric Learning [22], which learns a metric space to encode the user-item interactions and to implicitly capture the user-user and item-item similarities.
- **LRML**, Latent Relational Metric Learning [66], which exploits an attention-based memory-augmented neural architecture to model the relationships between users and items.

- **TransCF**, Collaborative Translational Metric Learning [69], which employs the neighborhood information of users and items to construct translation vectors capturing the intensity and the heterogeneity of user–item relationships.
- **SML**, Symmetric Metric Learning with adaptive margin [71], which measures the trilateral relationship from both the user-centric and item-centric perspectives and learns adaptive margins for the target user and positive item.

The proposed method:

- **PMLAM**, the proposed model, which represents each user and item as Gaussian distributions to capture the uncertainties in user preferences and item properties, and incorporates an adaptive margin generation mechanism to generate the margins based on the sampled user-item triplets.

7.4.4 Experiment Settings

In the experiments, the latent dimension of all the models is set to 50 for a fair comparison. All the models adopt the same negative sampling strategy as the proposed model, unless otherwise specified. For BPRMF, the Adam [112] algorithm is used to update the model parameters rather than conventional stochastic gradient descent; the learning rate is set to 0.001 and the regularization parameter is set to 0.001. With these parameters, the model can achieve good results. For NCF, we follow the same model structure as in the original paper [9]. The learning rate is set to 0.001 and the batch size is set to 1024. For DeepAE, we adopt the same model structure employed in the author-provided code and set the batch size to 512. The weight of the positive items is selected from $\{5, 10, 15, 20\}$ by a grid search and the weights of all other items are set to 1 as recommended in [20]. For CML, we use the authors' implementation to set the margin to $m = 1$ and the regularization parameter to $\lambda_c = 1$. For LRML, the learning rate is set to 0.001, and the number of memories is selected from $\{5, 10, 20, 25, 50, 100\}$ by a grid search. For TransCF, we follow the settings in the original paper to select λ, λ_{nbr}, $\lambda_{dist} \in \{0, 0.001, 0.01, 0.1\}$ and set the margin to 1 and batch size to 1000, respectively. For SML, we follow the author's code to set the user and item margin bound l to 1.0, λ to 0.01 and γ to 10, respectively.

For our model, the learning rate and λ are set to 0.001 and 0.001, respectively. We randomly sample 10 unobserved users or items as negative samples for each user and positive

item on *Electronics* and *CDs* datasets, while this number is set to 2 for the other datasets to speed up the training process. The batch size is set to 5000 on all datasets. The dimension h is set to 50. The user and item embeddings are initialized by drawing each vector element independently from a zero-mean Gaussian distribution with a standard deviation of 0.01. Our experiments are conducted with PyTorch running on GPU machines (Nvidia Tesla P100).

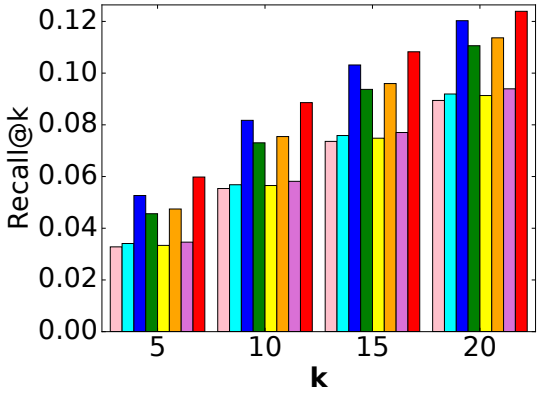
Table 7.2 The performance comparison of all methods in terms of *Recall@10* and *NDCG@10*. The best performing method is boldfaced. The underlined number is the second best performing method. $*$, $**$, $***$ indicate the statistical significance for $p <= 0.05$, $p <= 0.01$, and $p <= 0.001$, respectively, compared to the best baseline method based on the paired t-test. *Improv.* denotes the improvement of our model over the best baseline method.

	BPRMF	**NCF**	**DeepAE**	**CML**	**LRML**	**TransCF**	**SML**	**PMLAM**	**Improv.**
Recall@10									
Books	0.0553	0.0568	<u>0.0817</u>	0.0730	0.0565	0.0754	0.0581	**0.0885****	8.32%
Electronics	0.0243	0.0277	0.0253	<u>0.0395</u>	0.0299	0.0353	0.0279	**0.0469*****	18.73%
CDs	0.0730	0.0759	0.0736	<u>0.0922</u>	0.0822	0.0851	0.0793	**0.1129*****	22.45%
Comics	0.1966	0.2092	<u>0.2324</u>	0.1934	0.1795	0.1967	0.1713	**0.2417**	4.00%
Gowalla	0.0888	0.0895	<u>0.1113</u>	0.0840	0.0935	0.0824	0.0894	**0.1331*****	19.58%
NDCG@10									
Books	0.0391	0.0404	<u>0.0590</u>	0.0519	0.0383	0.0542	0.0415	**0.0671****	13.72%
Electronics	0.0111	0.0125	0.0134	<u>0.0178</u>	0.0117	0.0148	0.0105	**0.0234*****	31.46%
CDs	0.0383	0.0402	0.0411	<u>0.0502</u>	0.0420	0.0461	0.0423	**0.0619*****	23.30%
Comics	0.2247	0.2395	<u>0.2595</u>	0.2239	0.1922	0.2341	0.1834	**0.2753***	6.08%
Gowalla	0.0806	0.0822	<u>0.0944</u>	0.0611	0.0670	0.0611	0.0823	**0.0984***	4.23%

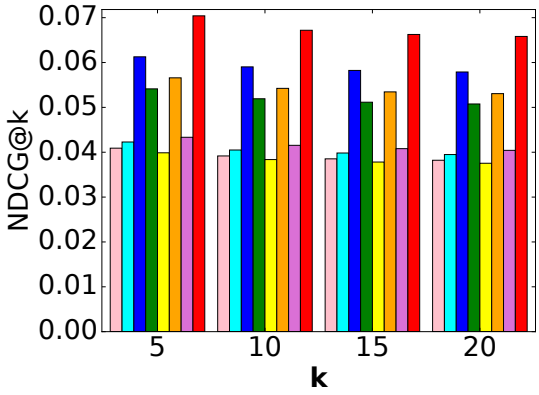
7.4.5 Implementation Details

To speed up the training process, we implement a two-phase sampling strategy. We sample a number of candidates, e.g., 500, of negative samples for each user every 20 epochs to form a candidate set. During the next 20 epochs, the negative samples of each user are sampled from her candidate set. This strategy can be implemented using multiple processes to further reduce the training time.

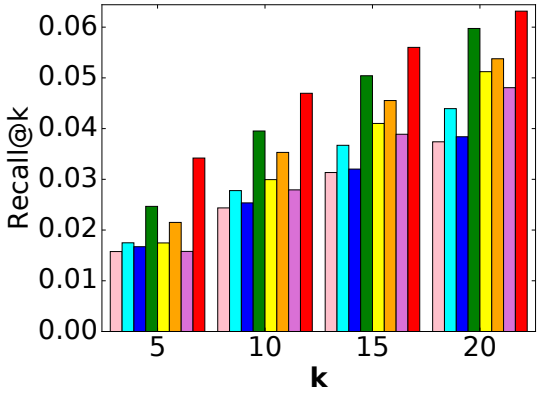
Since none of the processed datasets has inherent user-user/item-item information, we treat the user-item interaction as a user-item matrix and compute the cosine similarity for the user and item pairs, respectively [146]. We set a threshold, e.g., 0.2 on Amazon and Gowalla datasets and 0.4 on the Comics dataset, to select the neighbors. These thresholds are chosen to ensure a reasonable degree of connectivity in the constructed graphs.



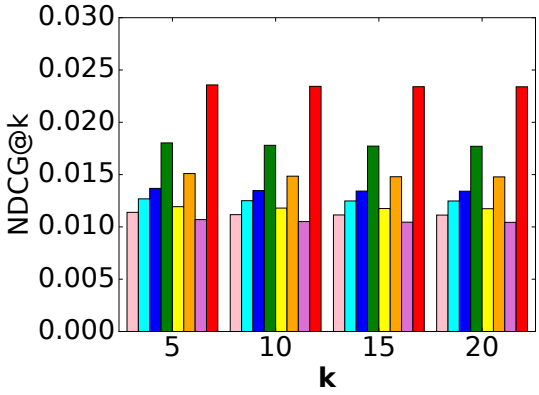
(a) Recall@k on Books



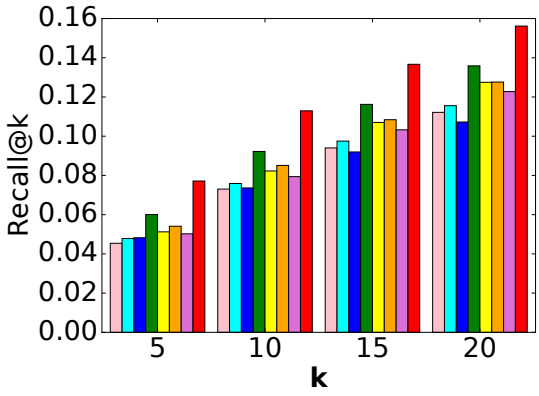
(b) NDCG@k on Books



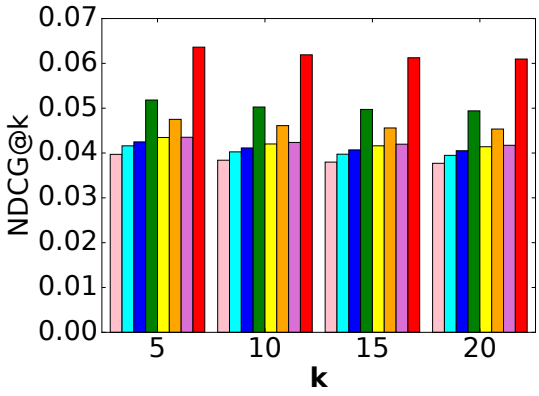
(c) Recall@k on Electronics



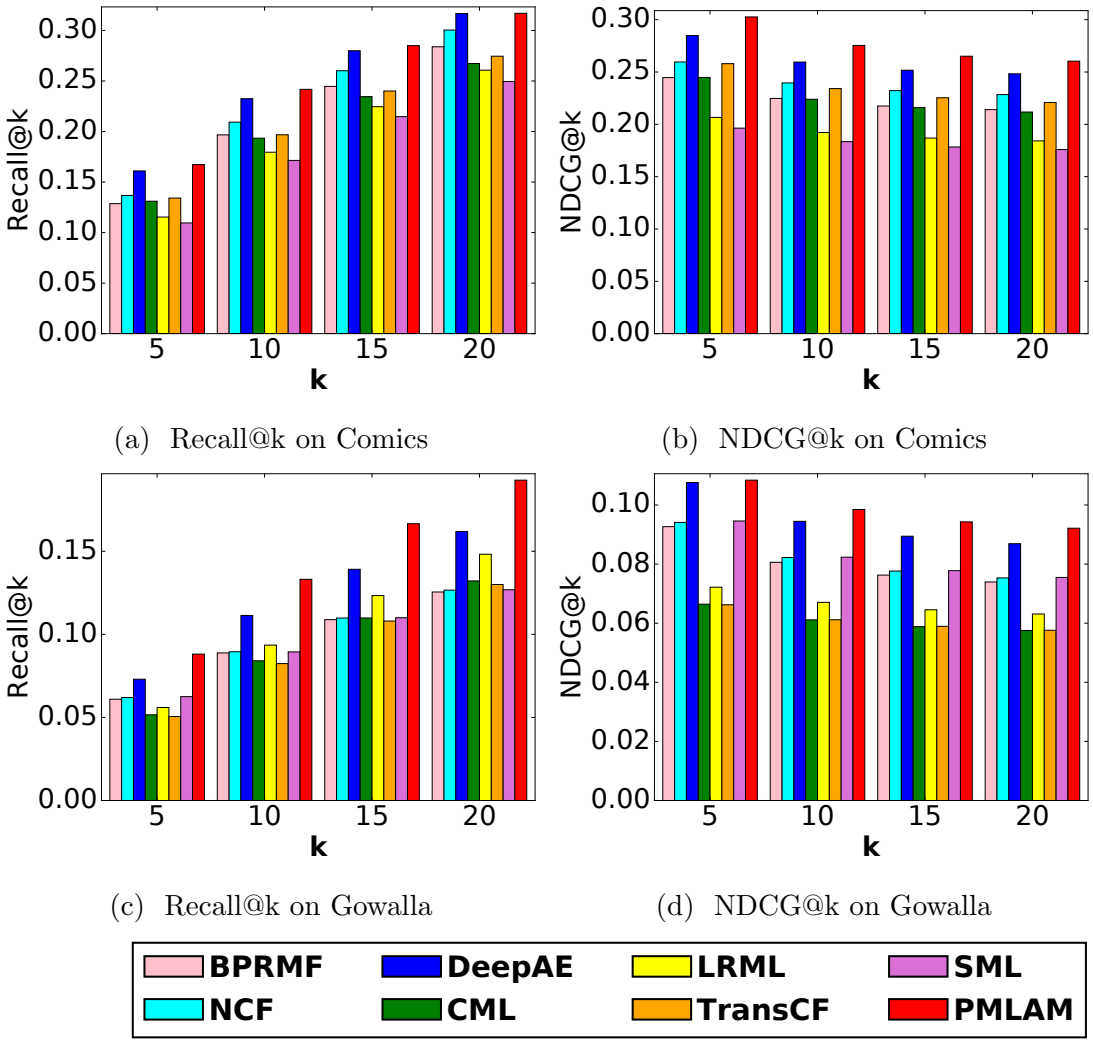
(d) NDCG@k on Electronics



(e) Recall@k on CDs



(f) NDCG@k on CDs



(a) Recall@k on Comics (b) NDCG@k on Comics

(c) Recall@k on Gowalla (d) NDCG@k on Gowalla

Figure 7.3 The performance comparison on all datasets.

7.4.6 Performance Comparison

The performance comparison is shown in Figure 7.3 and Table 7.2. Based on these results, we have several observations.

Observations about our model:

First, the proposed model, PMLAM, achieves the best performance on all five datasets with both evaluation metrics, which illustrates the superiority of our model.

Second, PMLAM outperforms SML. Although SML has an adaptive margin mechanism, it is achieved by having a learnable scalar margin for each user and item and adding

a regularization term to prevent the learned margins from being too small. It can be challenging to identify an appropriate regularization weight via hyperparameter tuning. By contrast, PMLAM formulates the adaptive margin generation as a bilevel optimization task, avoiding the need for additional regularization. PMLAM employs a neural network to generate the adaptive margin, so the number of parameters related to margin generation does not increase with the number of users or items.

Third, PMLAM achieves better performance than TransCF. One major reason is that TransCF only considers the items rated by a user and the users who rated an item as the neighbors of the user and item, respectively, which neglects the user-user/item-item relations. PMLAM models the user-user/item-item relations by two margin ranking losses with adaptive margins.

Fourth, PMLAM makes better recommendations than CML and LRML. These methods apply a fixed margin for all user-item triplets and do not measure or model the uncertainty of learned user/item embeddings. PMLAM represents each user and item as a Gaussian distribution, where the uncertainties of learned user preferences and item properties are captured by the covariance matrices.

Fifth, PMLAM outperforms NCF and DeepAE. These are MLP-based recommendation methods with the ability to capture non-linear user-item relationships, but they violate the triangle inequality when modeling user-item interaction. As a result, they can struggle to capture the fine-grained user preference for particular items [22].

Other observations:

First, all of the results reported for the Comics dataset are considerably better than those for the other datasets. The other four datasets are sparser and data sparsity negatively impacts recommendation performance.

Second, CML, LRML and TransCF perform better than SML on most of the datasets. The adaptive margin regularization term in SML struggles to adequately counterbalance SML's tendency to reduce the loss by imposing small margins. Although it is reported that SML outperforms CML, LRML and TransCF in [71], the experiments are conducted on three relatively small-scale datasets with only several thousands of users and items. We experiment with much larger datasets; identifying a successful regularization setting appears to be more difficult as the number of users increases.

Third, TransCF outperforms LRML on most of the datasets. One possible reason is that TransCF has a more effective translation embedding learning mechanism, which incorpo-

rates the neighborhood information of users and items. TransCF also has a regularization term to further pull positive items closer to the anchor user.

Fourth, CML achieves better performance than LRML on most of the datasets. CML integrates the weighted approximate-rank pairwise (WARP) weighting scheme [148] in the loss function to penalize lower-ranked positive items. The comparison between CML and LRML in [66] removes this component of CML. The WARP scheme appears to play an important role in improving CML's performance.

Fifth, DeepAE outperforms NCF. The heuristic weighting function of DeepAE can impose useful penalties to errors that occur during training when positive items are assigned lower prediction scores.

Table 7.3 The ablation analysis on the CDs and Electronics datasets in terms of Recall@10 (R@10) and NDCG@10 (N@10). *cat* denotes the concatenation operation and *add* denotes the addition operation.

Architecture	*CDs*		*Electronics*	
	R@10	N@10	R@10	N@10
(1) Fix^{U-I} + Deter_Emb	0.0721	0.0371	0.0241	0.0090
(2) Fix^{U-I} + Gauss_Emb	0.0815	0.0434	0.0296	0.0110
(3) Ada^{U-I} + Deter_Emb	0.0777	0.0415	0.0338	0.0125
(4) Ada^{U-I}-*cat* + Deter_Emb	0.0408	0.0204	0.0139	0.0055
(5) Ada^{U-I}-*add* + Deter_Emb	0.0311	0.0158	0.0050	0.0018
(6) Ada^{U-I} + Gauss_Emb	0.0856	0.0454	0.0365	0.0155
(7) Ada^{U-I} + Fix^{U-U} + Fix^{I-I}	0.0966	0.0526	0.0429	0.0189
(8) PMLAM	**0.1129**	**0.0619**	**0.0469**	**0.0234**

7.4.7 Ablation Analysis

To verify and assess the relative effectiveness of the proposed user-item interaction module, the adaptive margin generation module, and the user-user/item-item relation module, we conduct an ablation study. Table 7.3 reports the performance improvement achieved by each module of the proposed model. Note that we compute the Euclidean distance between deterministic embeddings. In (1), which serves as a baseline, we use the hinge loss with a fixed margin (Eq. 7.4) on deterministic embeddings of users and items to capture the user-item interaction (m is set to 1 which is commonly used in [22, 66, 69]). In (2), as an alternative baseline, we apply the same hinge loss as in (1), but replace the deterministic

embeddings with parameterized Gaussian distributions (Section 7.3.1). In (3), we use the adaptive margin generation module (Section 7.3.2) to generate the margins for deterministic embeddings. In (4), we concatenate the deterministic embeddings of (i, j, k) to generate $\mathbf{s}_{ijk}$ instead of using Eq. 7.11. In (5), we sum the deterministic embeddings of (i, j, k) to generate $\mathbf{s}_{ijk}$ instead of using Eq. 7.11. In (6), we combine (2) and (3) to generate the adaptive margins for Gaussian embeddings. In (7), we augment (6) with user-user/item-item modeling but with a fixed margin, where the margin is also set to 1. In (8), we add the user-user/item-item modeling with adaptive margins (Section 7.3.3) to replace the fixed margins in the configuration of (7).

From the results in Table 7.3, we have several observations. *First*, from (1) and (2), we observe that by representing the user and item as Gaussian distributions and computing the distance between Gaussian distributions, the performance improves. This suggests that measuring the uncertainties of learned embeddings is significant. *Second*, from (1) and (3) along with (2) and (6), we observe that incorporating the adaptive margin generation module improves performance, irrespective of whether deterministic or Gaussian embeddings are used. These results demonstrate the effectiveness of the proposed margin generation module. *Third*, from (3), (4) and (5), we observe that our designed inputs (Eq. 7.11) for margin generation facilitate the production of appropriate margins compared to commonly used embedding concatenation or summation operations. *Fourth*, from (2), (3) and (6), we observe that (6) achieves better results than either (2) or (3), demonstrating that Gaussian embeddings and adaptive margin generation are compatible and can be combined to improve the model performance. *Fifth*, compared to (6), we observe that the inclusion of the user-user and item-item terms in the objective function (7) leads to a large improvement in recommendation performance. This demonstrates that explicit user-user/item-item modeling is essential and can be an effective supplement to infer user preferences. *Sixth*, from (7) and (8), we observe that adaptive margins also improve the modeling of the user-user/item-item relations.

7.4.8 Case Study

In this section, we conduct case studies to confirm whether the adaptive margin generation can produce appropriate margins. To achieve this purpose, we train our model on the MovieLens-100K dataset. This dataset provides richer side information about movies (e.g.,

movie genres), making it easier for us to illustrate the results. Since we only focus on the adaptive margin generation, we use deterministic embeddings of users and items to avoid the interference of other modules. We randomly sample users from the dataset. For each user, we sample one item that the user has accessed as the positive item and two items the user did not access as negative items, where one item has a similar genre with the positive item and the other does not. The case study results are shown in Table 7.4.

As shown in Table 7.4, our adaptive margin generation module tends to generate a smaller margin value when the negative movie has a similar genre with the positive movie, while generating larger margins when they are distinct. These help put the user potentially preferred items not too far from the user.

Table 7.4 A case study of the generated margin of sampled training triplets. The movie genre label is from the dataset.

User	Positive	Sampled Movie	Margin
405	*Scream* (Thriller)	*Four Rooms* (Thriller)	**1.2752**
		Toy Story (Animation)	12.8004
	French Kiss (Comedy)	*Addicted to Love* (Comedy)	**2.6448**
		Batman (Action)	12.4607
66	*Air Force One* (Action)	*GoldenEye* (Action)	**0.3216**
		Crumb (Documentary)	5.0010
	The Godfather (Crime)	*The Godfather II* (Crime)	**0.0067**
		Terminator (Sci-Fi)	3.6335

7.5 Summary

In this chapter, we propose a distance-based recommendation model for the top-K recommendation. Each user and item in our model are represented by Gaussian distributions with learnable parameters to handle the uncertainties. By incorporating an adaptive margin scheme, our model can generate fine-grained margins for the training triples during the training procedure. To explicitly capture the user-user/item-item relations, we adopt two margin ranking losses with adaptive margins to force similar user and item pairs to map closer together in the latent space. Experimental results on five real-world datasets validate the performance of our model, demonstrating improved performance compared to many state-of-the-art methods and highlighting the effectiveness of the Gaussian embeddings and the adaptive margin generation scheme.

Chapter 8

Conclusion and Future Work

This chapter concludes the thesis. Section 8.1 summarizes contributions of our work and Section 8.2 outlines existing challenges and several directions for future work.

8.1 Conclusion

Due to the information/choice overload, recommender systems are playing more and more important roles in modern society. These systems help users not only easily find the items that they are interested in but also increase the revenue of the recommendation service provider. However, the personalized recommender system is still facing several challenges: the difficulty of modeling complex user-item interactions, the hardship of incorporating side information, the intricacy of capturing the user behavior dynamics, and the hardness of conducting adaptive learning. To tackle these challenges, dedicated models are proposed in this thesis.

To model the complex interactions between users and point-of-interests (POIs), we propose a novel autoencoder-based model to learn the nonlinear user-POI relations, namely SAE-NAD, which consists of a self-attentive encoder (SAE) and a neighbor-aware decoder (NAD). In particular, the self-attentive encoder adaptively differentiates the user preference degrees in multiple aspects, by adopting a multidimensional attention mechanism. To incorporate the geographical context information, a neighbor-aware decoder is proposed to make users' reachability higher on the nearby neighbors of checked-in POIs, which is achieved by the inner product of POI embeddings together with the radial basis function (RBF) kernel. Our model outperforms five state-of-the-art methods on the POI recom-

mendation task. We also demonstrate that incorporating the geographical influence can improve the recommendation performance by 10-20%.

To effectively incorporate the content auxiliary information, we propose a gated autoencoder with the word- and neighbor-attention. The model learns items' hidden representations from ratings and contents in a gated manner. Moreover, the model also captures items' informative words and representative neighbors by word- and neighbor-attention modules, respectively. The experimental results not only demonstrate the performance of our model over other state-of-the-art methods but also provide interpretable results attributed to the attention modules. Furthermore, the proposed attention mechanism yields fewer learnable parameters compared with classical convolutional or recurrent neural networks for learning the content embedding.

To model the temporal dynamics, we propose a hierarchical gating network with an item-item product module for the sequential recommendation. The model adopts a feature gating module and an instance gating module to control what item features can be passed to downstream layers, where informative latent features and important items can be identified. Moreover, we apply an item-item product module to capture the relations between closely relevant items. Experimental results on five real-world datasets validate the performance of our model over many state-of-the-art methods and demonstrate the effectiveness of the gating and item-item product modules. Further, our proposed model achieves faster training speed with fewer parameters compared with other state-of-the-art methods.

To conduct the adaptive/personalized hyper-parameter learning, we develop a distance-based recommendation model with two novel aspects: (1) each user and item are parameterized by Gaussian distributions to capture the learning uncertainties; (2) an adaptive margin generation scheme is proposed to generate the margins regarding different training triplets. In the adaptive margin generation module, we model the learning of adaptive margins as a bilevel (inner and outer) optimization problem, where we build a proxy function to explicitly link the learning of margin related parameters with the outer objective function. Via a comparison using five real-world datasets with state-of-the-art methods, the proposed model outperforms the best existing models by 4-22% in terms of recall@K on the top-K recommendation.

8.2 Future Work

My future research will continue the line of important topics discussed above and explore new directions to tackle practical problems of recommender systems. As recommender systems involved in a large number of applications and services, there is a wide range of improvement for various research topics:

FATE Problem in Recommender Systems. FATE is an abbreviation of Fairness, Accountability, Transparency and Ethics. Recommender systems are shown to be highly biased and lack accountability or transparency. Thus, solely focusing on the satisfaction and personalization of customers may affect the business. That is, only considering the benefits of customers may hurt the benefits of other participants. For example, the user feedback in recommender systems is usually long-tailed and highly imbalanced that a few popular items have larger occurrences than others. Training on the imbalanced data may make popular items have high exposures as the recommendation candidates, which will reduce the exposure proportion of other items, resulting in unfairness. To tackle this, our research seeks to build effective recommendation models to answer three questions: (1) How can recommendation models assist users and offer enhanced insights, while avoiding exposing them to discrimination in health, housing, law enforcement, and employment? (2) How can the model balance the need for efficiency and exploration with fairness and sensitivity to users? (3) How can the model be trusted by individuals and communities?

Recommendation with Multiple Stakeholders. Traditional recommender systems only focus on optimizing the utility of the end-users who are the receiver of the recommendations. By contrast, the multi-stakeholder recommendation attempts to generate recommendations that satisfy the needs of the end-users and other parties or stakeholders. However, there has not been a comprehensive treatment of the integration of multiple stakeholders into recommendation calculations. To simultaneously take the utilities of all stakeholders into consideration, I plan to explore the potential idea of connecting the Pareto-efficient condition with the multiple-objective optimization. This is promising that the model can generate the Pareto Frontier for each end-user. Then one specific recommendation that is satisfied by all stakeholders can be selected.

Transferable Recommender Systems. The single-domain recommender system only focuses on one domain and ignores the user interests in other domains, which greatly exacerbates the sparsity and cold start problems. A tangible solution for these problems

is to apply domain adaptation techniques, where a model is assisted with the knowledge learned from other domains. In this research project, I plan to explore the use of transfer learning in the recommendation scenario, which can improve learning tasks in one domain by using knowledge transferred from others. The combination of different domains into a single model helps improve the recommendation quality by having a more compact and semantically richer user latent representation.

Online Recommender Simulator. Current recommendation research is typically evaluated in a one-time fashion, where the data is statically split without varying with time. This offline evaluation strategy largely mismatches the real-world online recommendation scenario. One main reason for not conducting the online evaluation is the lack of such an online evaluation environment. To tackle this challenge, I plan to build a simulator with reinforcement learning techniques that are trained from the real-world data, where each customer is represented as an agent and has its decision-making policy from the recommendation model. When a customer enters a query, the simulator returns a list of items according to the query based on the recommendation model. Moreover, the simulator can incrementally update the recommendation model accordingly. As such, the recommendation model can be evaluated in an online fashion.

Natural Language Understanding for Recommendation. Except for giving ratings to items or products, nowadays, users are also supposed to write down their reviews regarding the items they have rated or purchased. These user reviews provide a significant supplement for understanding user preference. With the rapid growth of Natural Language Processing (NLP), the techniques to understand the human language have been largely developed. One representative example is Transformer [56] which has been successfully used in many NLP tasks, such as BERT [149]. One interesting future work can how we can better make use of these pre-trained language models to better understand the user preference.

Appendix A

Publications

Published or Accepted

1. **Chen Ma**, Liheng Ma, Yingxue Zhang, Haolun Wu, Xue Liu and Mark Coates, "Knowledge-Enhanced Top-K Recommendation in Poincaré Ball", in the 35th AAAI Conference on Artificial Intelligence (AAAI 2021) [150].

2. Jiapeng Wu, Yishi Xu, Yingxue Zhang, **Chen Ma**, Mark Coates and Jackie Chi Kit Cheung, "TIE: A Framework for Embedding-based Incremental Temporal Knowledge Graph Completion", in the 44rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021) [151].

3. Junyu Cao, Xue Liu, **Chen Ma** and Wei Qi, "Stall Economy: The Value of Mobility and Precision Deployment of Retail on Wheels", in the 31st Annual Conference of Production and Operations Management Society (POMS 2021) [152].

4. Peng Kang, Jianping Zhang, **Chen Ma** and Guiling Sun, "ATM: Attentional Text Matting", in the 2021 IEEE Winter Conference on Applications of Computer Vision (WACV 2021) [153].

5. **Chen Ma**, Liheng Ma, Yingxue Zhang, Ruiming Tang, Xue Liu and Mark Coates, "Probabilistic Metric Learning with Adaptive Margin for Top-K Recommendation", in the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2020 Research Track) [29].

6. Jianing Sun, Yingxue Zhang, Wei Guo, Huifeng Guo, Ruiming Tang, Xiuqiang He, **Chen Ma** and Mark Coates, "Neighbor Interaction Aware Graph Convolution Networks for Recommendation", in the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2020) [39].

7. Hang Li, **Chen Ma**, Wei Xu and Xue Liu, "Feature Statistics Guided Efficient Filter Pruning", in the 29th International Joint Conference on Artificial Intelligence (IJCAI 2020) [154].

8. **Chen Ma**, Liheng Ma, Yingxue Zhang, Jianing Sun, Xue Liu and Mark Coates, "Memory Augmented Graph Neural Networks for Sequential Recommendation", in the 34th AAAI Conference on Artificial Intelligence (AAAI 2020) [38].

9. Jianing Sun, Yingxue Zhang, **Chen Ma**, Mark Coates, Huifeng Guo, Ruiming Tang and Xiuqiang He, "Multi-Graph Convolution Collaborative Filtering", in the 19th IEEE International Conference on Data Mining (IEEE ICDM 2019) [51].

10. **Chen Ma**, Peng Kang and Xue Liu, "Hierarchical Gating Networks for Sequential Recommendation", in the 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2019 Research Track) [28].

11. **Chen Ma**, Peng Kang, Bin Wu, Qinglong Wang and Xue Liu, "Gated Attentive-Autoencoder for Content-Aware Recommendation", in the 12th ACM International Conference on Web Search and Data Mining (WSDM 2019) [27].

12. **Chen Ma**, Yingxue Zhang, Qinglong Wang and Xue Liu, "Point-of-Interest Recommendation: Exploiting Self-Attentive Autoencoders with Neighbor-Aware Influence", in the 27th ACM International Conference on Information and Knowledge Management (CIKM 2018) [26].

13. Landu Jiang, Yu Hua, **Chen Ma** and Xue Liu, "SunChase: Energy-Efficient Route Planning for Solar-Powered EVs", in the 37th IEEE International Conference on Distributed Computing Systems (ICDCS 2017) [155].

14. Xi Chen, **Chen Ma**, Michel Allegue and Xue Liu, "Taming the Inconsistency of Wi-Fi Fingerprints for Device-Free Passive Indoor Localization", in the 36th IEEE International Conference on Computer Communications (INFOCOM 2017) [156].

Bibliography

[1] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, no. 1, pp. 5:1–5:38, 2019.

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

[3] S. Wu, W. Zhang, F. Sun, and B. Cui, "Graph neural networks in recommender systems: A survey," *CoRR*, vol. abs/2011.02260, 2020.

[4] J. Tang, X. Hu, and H. Liu, "Social recommendation: a review," *Soc. Netw. Anal. Min.*, vol. 3, no. 4, pp. 1113–1133, 2013.

[5] J. Bao, Y. Zheng, D. Wilkie, and M. F. Mokbel, "Recommendations in location-based social networks: a survey," *GeoInformatica*, vol. 19, no. 3, pp. 525–565, 2015.

[6] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *ACM Comput. Surv.*, vol. 51, no. 4, pp. 66:1–66:36, 2018.

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

[8] Y. Koren, R. M. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *WWW*. ACM, 2017, pp. 173–182.

[10] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil, "Bilevel programming for hyperparameter optimization and meta-learning," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 80. PMLR, 2018, pp. 1563–1572.

[11] S. Rendle, "Learning recommender systems with adaptive regularization," in *Proc. ACM Int. Conf. Web Search and Data Mining*, 2012.

[12] Y. Chen, B. Chen, X. He, C. Gao, Y. Li, J. Lou, and Y. Wang, "λopt: Learn to regularize recommender models in finer levels," in *KDD*. ACM, 2019, pp. 978–986.

[13] A. Voulodimos, N. Doulamis, A. D. Doulamis, and E. Protopapadakis, "Deep learning for computer vision: A brief review," *Comput. Intell. Neurosci.*, vol. 2018, pp. 7 068 349:1–7 068 349:13, 2018.

[14] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li, "Imagenet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[15] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing [review article]," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, 2018.

[16] Y. Kim, "Convolutional neural networks for sentence classification," in *EMNLP*. ACL, 2014, pp. 1746–1751.

[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR (Poster)*, 2017.

[19] K. Hornik, M. B. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[20] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *ICDM*. IEEE Computer Society, 2008, pp. 263–272.

[21] S. Rendle, "Factorization machines," in *ICDM*. IEEE Computer Society, 2010, pp. 995–1000.

[22] C. Hsieh, L. Yang, Y. Cui, T. Lin, S. J. Belongie, and D. Estrin, "Collaborative metric learning," in *WWW*. ACM, 2017, pp. 193–201.

[23] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *ICASSP*. IEEE, 2013, pp. 6645–6649.

[24] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *CoRR*, vol. abs/1409.0473, 2014.

[25] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowl. Based Syst.*, vol. 212, p. 106622, 2021.

[26] C. Ma, Y. Zhang, Q. Wang, and X. Liu, "Point-of-interest recommendation: Exploiting self-attentive autoencoders with neighbor-aware influence," in *CIKM*. ACM, 2018, pp. 697–706.

[27] C. Ma, P. Kang, B. Wu, Q. Wang, and X. Liu, "Gated attentive-autoencoder for content-aware recommendation," in *WSDM*. ACM, 2019, pp. 519–527.

[28] C. Ma, P. Kang, and X. Liu, "Hierarchical gating networks for sequential recommendation," in *KDD*. ACM, 2019, pp. 825–833.

[29] C. Ma, L. Ma, Y. Zhang, R. Tang, X. Liu, and M. Coates, "Probabilistic metric learning with adaptive margin for top-k recommendation," in *KDD*. ACM, 2020, pp. 1036–1044.

[30] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

[31] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. M. Lukose, M. Scholz, and Q. Yang, "One-class collaborative filtering," in *ICDM*. IEEE Computer Society, 2008, pp. 502–511.

[32] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, 2003.

[33] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-paper recommender systems: a literature survey," *Int. J. Digit. Libr.*, vol. 17, no. 4, pp. 305–338, 2016.

[34] Y. Wang, L. Wang, Y. Li, D. He, and T. Liu, "A theoretical analysis of NDCG type ranking measures," in *COLT*, ser. JMLR Workshop and Conference Proceedings, vol. 30. JMLR.org, 2013, pp. 25–54.

[35] H. Wang, N. Wang, and D. Yeung, "Collaborative deep learning for recommender systems," in *KDD*. ACM, 2015, pp. 1235–1244.

[36] R. Salakhutdinov, A. Mnih, and G. E. Hinton, "Restricted boltzmann machines for collaborative filtering," in *ICML*, ser. ACM International Conference Proceeding Series, vol. 227. ACM, 2007, pp. 791–798.

[37] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "BPR: bayesian personalized ranking from implicit feedback," in *UAI*. AUAI Press, 2009, pp. 452–461.

[38] C. Ma, L. Ma, Y. Zhang, J. Sun, X. Liu, and M. Coates, "Memory augmented graph neural networks for sequential recommendation," in *AAAI*. AAAI Press, 2020, pp. 5045–5052.

[39] J. Sun, Y. Zhang, W. Guo, H. Guo, R. Tang, X. He, C. Ma, and M. Coates, "Neighbor interaction aware graph convolution networks for recommendation," in *SIGIR*. ACM, 2020, pp. 1289–1298.

[40] X. Ning and G. Karypis, "SLIM: sparse linear methods for top-n recommender systems," in *ICDM*. IEEE Computer Society, 2011, pp. 497–506.

[41] S. Kabbur, X. Ning, and G. Karypis, "FISM: factored item similarity models for top-n recommender systems," in *KDD*. ACM, 2013, pp. 659–667.

[42] C. C. Johnson, "Logistic matrix factorization for implicit feedback data," *Advances in Neural Information Processing Systems*, vol. 27, p. 78, 2014.

[43] X. Li and J. She, "Collaborative variational autoencoder for recommender systems," in *KDD*. ACM, 2017, pp. 305–314.

[44] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*. Curran Associates, Inc., 2007, pp. 1257–1264.

[45] X. He, H. Zhang, M. Kan, and T. Chua, "Fast matrix factorization for online recommendation with implicit feedback," in *SIGIR*. ACM, 2016, pp. 549–558.

[46] Y. Wu, C. DuBois, A. X. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-n recommender systems," in *WSDM*. ACM, 2016, pp. 153–162.

[47] H. Xue, X. Dai, J. Zhang, S. Huang, and J. Chen, "Deep matrix factorization models for recommender systems," in *IJCAI*. ijcai.org, 2017, pp. 3203–3209.

[48] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "Deepfm: A factorization-machine based neural network for CTR prediction," in *IJCAI*. ijcai.org, 2017, pp. 1725–1731.

[49] J. Lian, X. Zhou, F. Zhang, Z. Chen, X. Xie, and G. Sun, "xdeepfm: Combining explicit and implicit feature interactions for recommender systems," in *KDD*. ACM, 2018, pp. 1754–1763.

[50] X. Wang, X. He, M. Wang, F. Feng, and T. Chua, "Neural graph collaborative filtering," in *Proc. ACM Int. Conf. Research and Development in Information Retrieval*, 2019.

[51] J. Sun, Y. Zhang, C. Ma, M. Coates, H. Guo, R. Tang, and X. He, "Multi-graph convolution collaborative filtering," in *Proc. IEEE Int. Conf. Data Mining*, 2019.

[52] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 37. JMLR.org, 2015, pp. 2048–2057.

[53] Q. You, H. Jin, Z. Wang, C. Fang, and J. Luo, "Image captioning with semantic attention," in *CVPR*. IEEE Computer Society, 2016, pp. 4651–4659.

[54] Z. Yang, D. Yang, C. Dyer, X. He, A. J. Smola, and E. H. Hovy, "Hierarchical attention networks for document classification," in *HLT-NAACL*. The Association for Computational Linguistics, 2016, pp. 1480–1489.

[55] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *EMNLP*. The Association for Computational Linguistics, 2015, pp. 1412–1421.

[56] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 6000–6010.

[57] W. Pei, J. Yang, Z. Sun, J. Zhang, A. Bozzon, and D. M. J. Tax, "Interacting attention-gated recurrent networks for recommendation," in *CIKM*. ACM, 2017, pp. 1459–1468.

[58] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, "Dynamic attention deep model for article recommendation by learning human editors' demonstration," in *KDD*. ACM, 2017, pp. 2051–2059.

[59] Y. Gong and Q. Zhang, "Hashtag recommendation using attention-based convolutional neural network," in *IJCAI*. IJCAI/AAAI Press, 2016, pp. 2782–2788.

[60] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T. Chua, "Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention," in *SIGIR*. ACM, 2017, pp. 335–344.

[61] S. Seo, J. Huang, H. Yang, and Y. Liu, "Interpretable convolutional neural networks with dual local and global attention for review rating prediction," in *RecSys*. ACM, 2017, pp. 297–305.

[62] Y. Tay, A. T. Luu, and S. C. Hui, "Multi-pointer co-attention networks for recommendation," in *KDD*. ACM, 2018, pp. 2309–2318.

[63] C. Chen, M. Zhang, Y. Liu, and S. Ma, "Neural attentional rating regression with review-level explanations," in *WWW*. ACM, 2018, pp. 1583–1592.

[64] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua, "KGAT: knowledge graph attention network for recommendation," in *KDD*. ACM, 2019, pp. 950–958.

[65] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *ICLR (Poster)*. OpenReview.net, 2018.

[66] Y. Tay, L. A. Tuan, and S. C. Hui, "Latent relational metric learning via memory-based attention for collaborative ranking," in *Proc. Int. Conf. World Wide Web*, 2018.

[67] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," in *NIPS*, 2015, pp. 2440–2448.

[68] X. Zhou, D. Liu, J. Lian, and X. Xie, "Collaborative metric learning with memory network for multi-relational recommender systems," in *IJCAI*. ijcai.org, 2019, pp. 4454–4460.

[69] C. Park, D. Kim, X. Xie, and H. Yu, "Collaborative translational metric learning," in *Proc. IEEE Int. Conf. Data Mining*, 2018.

[70] R. He, W. Kang, and J. McAuley, "Translation-based recommendation," in *RecSys*. ACM, 2017, pp. 161–169.

[71] M. Li, S. Zhang, F. Zhu, W. Qian, L. Zang, J. Han, and S. Hu, "Symmetric metric learning with adaptive margin for recommendation," in *AAAI*. AAAI Press, 2020, pp. 4634–4641.

[72] L. V. Tran, Y. Tay, S. Zhang, G. Cong, and X. Li, "Hyperml: A boosting metric learning approach in hyperbolic space for recommender systems," in *WSDM*. ACM, 2020, pp. 609–617.

[73] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *KDD*. ACM, 2011, pp. 1082–1090.

[74] M. Ye, P. Yin, W. Lee, and D. L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *SIGIR*. ACM, 2011, pp. 325–334.

[75] C. Cheng, H. Yang, I. King, and M. R. Lyu, "Fused matrix factorization with geographical and social influence in location-based social networks," in *AAAI*. AAAI Press, 2012.

[76] B. Liu, Y. Fu, Z. Yao, and H. Xiong, "Learning geographical preferences for point-of-interest recommendation," in *KDD*. ACM, 2013, pp. 1043–1051.

[77] Q. Yuan, G. Cong, and A. Sun, "Graph-based point-of-interest recommendation with geographical and temporal influences," in *CIKM*. ACM, 2014, pp. 659–668.

[78] Y. Liu, W. Wei, A. Sun, and C. Miao, "Exploiting geographical neighborhood characteristics for location recommendation," in *CIKM*. ACM, 2014, pp. 739–748.

[79] X. Li, G. Cong, X. Li, T. N. Pham, and S. Krishnaswamy, "Rank-geofm: A ranking based geographical factorization method for point of interest recommendation," in *SIGIR*. ACM, 2015, pp. 433–442.

[80] S. Zhao, T. Zhao, I. King, and M. R. Lyu, "Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation," in *WWW (Companion Volume)*. ACM, 2017, pp. 153–162.

[81] H. Wang, H. Shen, W. Ouyang, and X. Cheng, "Exploiting poi-specific geographical influence for point-of-interest recommendation," in *IJCAI*. ijcai.org, 2018, pp. 3877–3883.

[82] X. Zhou, C. Mascolo, and Z. Zhao, "Topic-enhanced memory networks for personalised point-of-interest recommendation," in *KDD*. ACM, 2019, pp. 3018–3028.

[83] J. J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *RecSys*. ACM, 2013, pp. 165–172.

[84] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *KDD*. ACM, 2011, pp. 448–456.

[85] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[86] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W. Ma, "Collaborative knowledge base embedding for recommender systems," in *KDD*. ACM, 2016, pp. 353–362.

[87] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *CIKM*. ACM, 2015, pp. 811–820.

[88] Y. Zhang, Q. Ai, X. Chen, and W. B. Croft, "Joint representation learning for top-n recommendation with heterogeneous information sources," in *CIKM*. ACM, 2017, pp. 1449–1458.

[89] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," in *ICML*, ser. JMLR Workshop and Conference Proceedings, vol. 32. JMLR.org, 2014, pp. 1188–1196.

[90] D. H. Kim, C. Park, J. Oh, S. Lee, and H. Yu, "Convolutional matrix factorization for document context-aware recommendation," in *RecSys*. ACM, 2016, pp. 233–240.

[91] L. Zheng, V. Noroozi, and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in *WSDM*. ACM, 2017, pp. 425–434.

[92] J. P. Zhou, Z. Cheng, F. Pérez, and M. Volkovs, "TAFA: two-headed attention fused autoencoder for context-aware recommendations," in *RecSys*. ACM, 2020, pp. 338–347.

[93] C. Cheng, H. Yang, M. R. Lyu, and I. King, "Where you like to go next: Successive point-of-interest recommendation," in *IJCAI*. IJCAI/AAAI, 2013, pp. 2605–2611.

[94] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*. ACM, 2010, pp. 811–820.

[95] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*. IEEE, 2016, pp. 191–200.

[96] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*. ACM, 2018, pp. 565–573.

[97] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," *CoRR*, vol. abs/1511.06939, 2015.

[98] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *RecSys*. ACM, 2017, pp. 130–137.

[99] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *CIKM*. ACM, 2017, pp. 1419–1428.

[100] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *CIKM*. ACM, 2018, pp. 843–852.

[101] W. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*. IEEE Computer Society, 2018, pp. 197–206.

[102] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *WSDM*. ACM, 2018, pp. 108–116.

[103] J. Huang, W. X. Zhao, H. Dou, J. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *SIGIR*. ACM, 2018, pp. 505–514.

[104] L. Yu, C. Zhang, S. Liang, and X. Zhang, "Multi-order attentive ranking model for sequential recommendation," in *AAAI*. AAAI Press, 2019, pp. 5709–5716.

[105] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," in *CIKM*. ACM, 2019, pp. 1441–1450.

[106] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, "Recommender systems survey," *Knowl.-Based Syst.*, vol. 46, pp. 109–132, 2013.

[107] J. Zhang and C. Chow, "igslr: personalized geo-social location recommendation: a kernel density estimation approach," in *SIGSPATIAL/GIS*. ACM, 2013, pp. 324–333.

[108] D. Lian, C. Zhao, X. Xie, G. Sun, E. Chen, and Y. Rui, "Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation," in *KDD*. ACM, 2014, pp. 831–840.

[109] H. Li, Y. Ge, R. Hong, and H. Zhu, "Point-of-interest recommendations: Learning potential check-ins from friends," in *KDD*. ACM, 2016, pp. 975–984.

[110] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.

[111] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *CoRR*, vol. abs/1703.03130, 2017.

[112] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[113] Y. Liu, T. Pham, G. Cong, and Q. Yuan, "An experimental evaluation of point-of-interest recommendation in location-based social networks," *PVLDB*, vol. 10, no. 10, pp. 1010–1021, 2017.

[114] H. Gao, J. Tang, X. Hu, and H. Liu, "Content-aware point of interest recommendation on location-based social networks," in *AAAI*. AAAI Press, 2015, pp. 1721–1727.

[115] 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 *KDD*. ACM, 2017, pp. 1245–1254.

[116] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, 2003.

[117] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, 2010.

[118] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2014.

[119] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[120] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie, "Autorec: Autoencoders meet collaborative filtering," in *WWW (Companion Volume)*. ACM, 2015, pp. 111–112.

[121] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *TiiS*, vol. 5, no. 4, pp. 19:1–19:19, 2016.

[122] R. He and J. McAuley, "Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering," in *WWW*. ACM, 2016, pp. 507–517.

[123] T. Chen, W. Zhang, Q. Lu, K. Chen, Z. Zheng, and Y. Yu, "Svdfeature: a toolkit for feature-based collaborative filtering," *Journal of Machine Learning Research*, vol. 13, pp. 3619–3622, 2012.

[124] A. van den Oord, S. Dieleman, and B. Schrauwen, "Deep content-based music recommendation," in *NIPS*, 2013, pp. 2643–2651.

[125] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *EMNLP*. ACL, 2014, pp. 1724–1734.

[126] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language modeling with gated convolutional networks," in *ICML*, ser. Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 933–941.

[127] X. Ning, C. Desrosiers, and G. Karypis, "A comprehensive survey of neighborhood-based recommendation methods," in *Recommender Systems Handbook*. Springer, 2015, pp. 37–76.

[128] M. Wan and J. McAuley, "Item recommendation on monotonic behavior chains," in *RecSys*. ACM, 2018, pp. 86–94.

[129] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," in *WSDM*. ACM, 2019, pp. 582–590.

[130] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *Proc. ACM Conf. Recommender Systems*, 2016.

[131] A. Shrivastava and P. Li, "Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS)," in *Advances in Neural Information Processing Systems*, 2014.

[132] A. Sinha, P. Malo, and K. Deb, "A review on bilevel optimization: From classical to evolutionary approaches and applications," *IEEE Trans. Evolutionary Computation*, 2018.

[133] A. Mallasto and A. Feragen, "Learning from uncertain curves: The 2-wasserstein metric for gaussian processes," in *Advances in Neural Information Processing Systems*, 2017.

[134] S. M. Srivastava, *A course on Borel sets*. Springer Science & Business Media, 2008.

[135] C. Villani, *Optimal transport: old and new*. Springer Science & Business Media, 2008.

[136] C. R. Givens and R. M. Shortt, "A class of wasserstein metrics for probability distributions." *The Michigan Mathematical Journal*, 1984.

[137] G. Casella and R. L. Berger, *Statistical inference*. Duxbury Pacific Grove, CA, 2002.

[138] Y. Xie, X. Wang, R. Wang, and H. Zha, "A fast proximal point method for computing exact wasserstein distance," in *Proc. Conf. Uncertainty in Artificial Intelligence*, 2019.

[139] M. Gelbrich, "On a formula for the l2 wasserstein metric between measures on euclidean and hilbert spaces," *Mathematische Nachrichten*, 1990.

[140] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Annals OR*, 2007.

[141] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *Proc. Int. Conf. Learning Representations*, 2019.

[142] M. Engilberge, L. Chevallier, P. Pérez, and M. Cord, "Sodeep: A sorting deep net to learn ranking loss surrogates," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2019.

[143] H. Doughty, D. Damen, and W. W. Mayol-Cuevas, "Who's better? who's best? pairwise deep ranking for skill determination," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2018.

[144] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *AAAI*, 2016.

[145] G. Ji, K. Liu, S. He, and J. Zhao, "Knowledge graph completion with adaptive sparse transfer matrix," in *AAAI*, 2016.

[146] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *WWW*. ACM, 2001, pp. 285–295.

[147] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, 2013.

[148] J. Weston, S. Bengio, and N. Usunier, "Large scale image annotation: learning to rank with joint word-image embeddings," *Machine Learning*, 2010.

[149] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT (1)*. Association for Computational Linguistics, 2019, pp. 4171–4186.

[150] C. Ma, L. Ma, Y. Zhang, H. Wu, X. Liu, and M. Coates, "Knowledge-enhanced top-k recommendation in poincaré ball," in *AAAI*. AAAI Press, 2021, pp. 4285–4293.

[151] J. Wu, Y. Xu, Y. Zhang, C. Ma, M. Coates, and J. C. K. Cheung, "TIE: A framework for embedding-based incremental temporal knowledge graph completion," 2021.

[152] J. Cao, C. Ma, and W. Qi, "Stall economy: The value of mobility and precision deployment of retail on wheels," *Available at SSRN 3711896*, 2020.

[153] P. Kang, J. Zhang, C. Ma, and G. Sun, "ATM: attentional text matting," in *WACV*. IEEE, 2021, pp. 3901–3910.

[154] H. Li, C. Ma, W. Xu, and X. Liu, "Feature statistics guided efficient filter pruning," in *IJCAI*. ijcai.org, 2020, pp. 2619–2625.

[155] L. Jiang, Y. Hua, C. Ma, and X. Liu, "Sunchase: Energy-efficient route planning for solar-powered evs," in *ICDCS*. IEEE Computer Society, 2017, pp. 383–393.

[156] X. Chen, C. Ma, M. Allegue, and X. Liu, "Taming the inconsistency of wi-fi fingerprints for device-free passive indoor localization," in *INFOCOM*. IEEE, 2017, pp. 1–9.