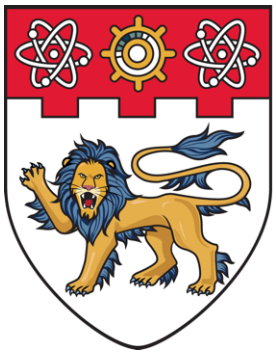


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Exploiting Item Relationships for Effective Recommendation

SUN ZHU

SCHOOL OF COMPUTER SCIENCES AND ENGINEERING

2018

Exploiting Item Relationships for Effective Recommendation

SUN ZHU

School of Computer Sciences and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirements for the degree of
Doctor of Philosophy

2018

Abstract

To alleviate the *data sparsity* and *cold start* issues in recommendation, many researchers leverage user relationships (e.g. social network) to improve the accuracy of recommender systems. However, social connections may not be available in many real systems, whereas item relationships are much easier to obtain but lack of study. Therefore, in this dissertation, we focus on exploiting item relationships for effective recommendation.

In real applications (e.g. Amazon), items are often organized by category, which is a popular way to define item relationships. Based on the assumption that users tend to have similar preferences towards items that belong to the same category, plenty of category-aware recommendation methods have been proposed. But, they mainly consider categories that are organized in a flat structure, where categories are independent and in a same level. In fact, categories can be also organized in a richer knowledge structure, i.e., category hierarchy (CH), to describe the inherent correlations among different categories, which might be more helpful to enhance the recommendation performance.

In order to take advantage of CH for better recommendation, we first propose a novel matrix factorization framework with recursive regularization – ReMF. It not only jointly models and learns the influence of hierarchically-organized categories on user-item interactions, but also provides characterization of how different categories in the hierarchy co-influence the modeling of user-item interactions. Empirical results show that ReMF consistently outperforms state-of-the-art category-aware recommendation methods.

Despite the success of ReMF, we notice that all CH based methods merely focus on the influence of vertically affiliated categories (i.e. child-parent) on user-item interactions. The relations of horizontally organized categories (i.e. siblings and cousins) in CH, however, have only been little studied. We show in real-world datasets that category relations in horizontal dimension can help explain and further model user-item interactions. To fully exploit CH, we further devise a unified matrix factorization framework –

HieVH, that seamlessly fuses both vertical and horizontal dimensions for effective recommendation. Extensive validation on real-world datasets demonstrates the superiority of HieVH against state-of-the-art algorithms. An additional benefit of HieVH is to provide better interpretations of the generated recommendations.

Recently, representation learning (RL) has proven to be more effective than matrix factorization in capturing local item relationships by modeling item co-occurrence in individual user's interaction record. We further design a unified multi-level RL based Bayesian framework – MRLR, thus benefiting from RL. By fusing item category, MRLR captures fine-grained item relationships from a multi-level item organization: items in local context (i.e., item co-occurrence relations), items affiliated to the same category, and items in user-specific ranked list. To the best of our knowledge, we are the first to investigate item category from the perspective of multi-level RL. Experimental results on multiple datasets show that MRLR consistently outperforms state-of-the-art algorithms.

Besides, with the development of semantic web, the knowledge graph (KG) has recently attracted a considerable amount of interest in recommendation, as it connects various types of features related to items (e.g., the genre, director, actor of a movie), in a unified global representation space. Utilizing such kind of heterogeneous connected information facilitates the inference of subtler item relationships from different perspectives, which are difficult to uncover with the homogeneous information (e.g., item category) only. To fully exploit the heterogeneous information encoded in KG for better recommendation, we propose a KG embedding framework – RKGE based on a novel recurrent network architecture that automatically learns semantic representations of entities and paths. In particular, RKGE learns the semantic representations of entities and paths between them via a batch of recurrent networks, and seamlessly integrates them into recommendation. Furthermore, it employs a pooling operator to discriminate the saliency of different paths in characterizing user preferences over items. Empirical study demonstrates that RKGE outperforms state-of-the-art algorithms. In addition, we show that RKGE provides meaningful explanations for the recommendation results.

To sum up, in this dissertation, we propose a series of recommendation approaches by exploiting auxiliary item relationships to deal with the *data sparsity* and *cold start* problems of recommender systems, which are natural but novel extensions of existing proposals for effective recommendation.

Acknowledgments

First of all, I would like to express my sincere gratitude towards my supervisor Prof. Jie Zhang. Without his patient guidance, encouragement, immense knowledge and advice, I would not be able to finish my Ph.D career. Under his elaborative supervision, I have not only learned the state-of-the-art data mining knowledge, but also acquired a conscientious research attitude through critical thinking. The knowledge and methodology I learnt are invaluable to my research, and even to my life. I feel very grateful to my co-supervisor Dr. Chi Xu for his continuous support, insightful discussion and patient guidance. Many thanks for sharing his extensive knowledge and experience in research. His advice on both research and my career have been invaluable.

I would like to express my deep gratitude to all my coauthors: Dr. Guibing Guo, Dr. Alessandro Bozzon, and Dr. Jie Yang. I appreciate all their efforts on the research work and I enjoy the experience of the collaborations with them all the time. Many thanks for all my friends for their constant companion and precious friendship. They include Yi Ding, Dongxia Wang, Aiping Li, Meiying Chen, Lubos Krcal, Huihuai Qiu, Hui Li, Yi Huang, Haiyan Yin, Qian Chen, Wenya Wang, Cheryl Wong, Jingting Ma and Jianjun Zhao. I would also like to appreciate all my fellow friends at the Computational Intelligence Lab (CIL) for their friendly support and help: Siwei Jiang, Yuan Liu, Zhiguang Cao, Chang Xu, Shibao Hong, Zehong Hu, Wen Song, Xiaoming Li, Shou Chen, Longkai Huang, Yu Chen. I am truly thankful for the convenient space and excellent facilities provided by CIL. Many thanks for the technique support from our administrator Kesavan Asaithambi.

Last but not least, I would like to sincerely thank my family, my mother and father, for their constant support, encouragement and love all the time. Many thanks for all the selfless sacrifices that they have made for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Scope

Given the explosive growth of information on the Web [1], recommender systems have been playing an increasingly important role by providing efficient and personalized on-line services for customers. The most well-known recommendation technique collaborative filtering (CF) [97], whereby a user's preference can be predicted by her like-minded users, inherently suffers from *data sparsity* and *cold start* problems [27, 28].

To address these issues, many researchers attempt to leverage user relationships to improve the accuracy of recommender systems. A number of trust-aware recommender systems [28, 43, 65, 116, 32] are emerging due to the advent of social networks. Significant improvements have been achieved up to date. However, the reliance on social connections may restrict the application of trust-based approaches to other scenarios where social networks are not available or supported. Furthermore, the potential noise and weaker social ties (than trust) in social networks can further hinder the generality of these approaches [32, 3]. In contrast, the side information describing item relationships is much easier to obtain and more amenable to explain the reasoning behind predictions, as users are familiar with items previously preferred by them, but do not know those allegedly like-minded users [52]. Therefore, in this dissertation, we focus on investigating how to leverage item relationships to further improve the recommendation performance.

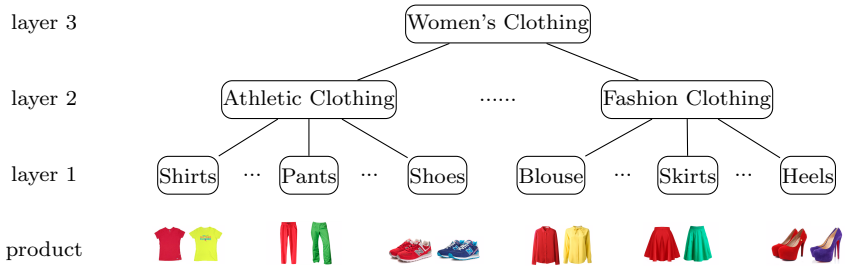


Figure 1.1: Category hierarchy for Women's Clothing in Amazon

In real-world applications, items are often organized by category. For instance, Amazon classifies the on-line products by category (e.g., clothing, electronics and sports); IMDB divides the released movies by genre (e.g., action, comedy and thriller). This enables category to become a quite popular way to define item relationships. Such kind of information, i.e., item category, can be very helpful in generating effective recommendation. Assume that Tom is into disaster movies and has already watched *Titanic*. Then, he is more likely to prefer *Poseidon Adventure*, as both of the two movies are in the category of disaster movies.

Based upon the assumption that users tend to have similar preferences towards items that belong to the same category, plenty of category-aware methods have been proposed to date. Empirical study has demonstrated the effectiveness of item category in dealing with the concerned issues of recommendation. However, these methods mainly consider categories that are organized in a flat structure, where categories are independent and in a same level. They all ignore categories with more complicated organizations, which might be more helpful to boost the recommendation performance.

In fact, categories can be also organized in a "category scheme", i.e., a set of categories and the relations between those categories. Category hierarchy (CH) is a natural yet powerful structure to human knowledge, and it provides a machine- and human-readable description of a set of categories and their relations. Typical examples of CH include on-line products hierarchy (e.g., Amazon web store [68]), food hierarchy (e.g.,

Gowalla [62]), articles hierarchy (e.g., Wikipedia [41]), music hierarchy (e.g., Yahoo! Music [51]), and ad hierarchy (e.g., Yahoo! traffic streams [69]) and so on. Figure 1.1 shows a running example of CH for women's clothing in Amazon, where women's clothing is first divided into several generalized categories (e.g., athletic), and further classified into localized subcategories (e.g., shirts). The benefits brought by explicitly modeling category relations through CH have been investigated in a broad spectrum of disciplines, from machine learning [44, 50] to natural language processing [41], without recommender systems being exception. While most of the existing work simply blend CH into flat structure, leading to severe topological structure information loss, thus hindering further improvements of recommendation performance. Therefore, how to effectively exploit CH in recommendation is still an open research question.

Recently, representation learning (RL) has drawn much attention from various domains, with recommender systems being no exception [26, 58, 104, 18, 4]. The popularization of RL in recommendation can be mainly attributed to word embedding techniques (e.g., CBOW and Skip-gram [71, 70]) originated from the natural language processing (NLP) domain. Word embedding generally refers to the low-dimensional distributed representation of words [5], capturing syntactical and semantic relationships among words. The fast development of RL has enabled a series of methods for NLP tasks, among which the most significant are the extensions of word embedding to learn textual representations in different levels of granularity (e.g., document or paragraph RL [55]), so as to help capture richer relationships between words and paragraphs or documents.

In recommendation, RL has proven to be effective in capturing local item relationships by modeling the item co-occurrence in individual user's interaction record. Although several RL based recommendation methods [4, 18, 26, 58, 104] have been proposed to date, the potential of RL for recommendation has not been fully exploited. First, most of the RL based recommendation methods are built upon item embedding [4], which always

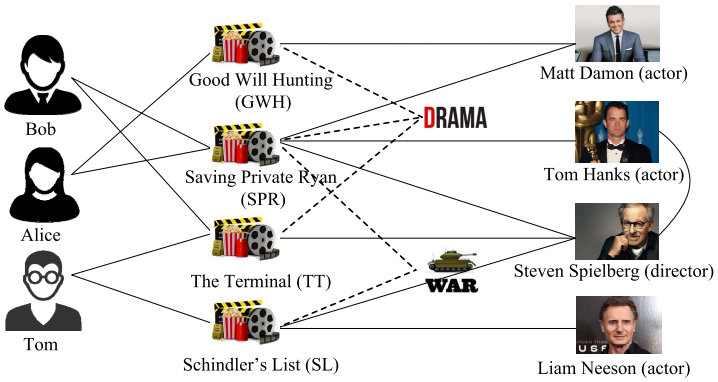


Figure 1.2: The knowledge graph in the movie domain

generate exactly same recommendation list to users who share similar interest, overlooking personalization for recommendation. Second, all existing methods ignore the possible multi-level organizations of items with the help of item category for uncovering fine-grained item relationships (similar as word-paragraph-document in NLP), which could in turn help achieve better recommendation performance.

Moreover, with the development of semantic web, the knowledge graph (KG) as auxiliary data source has recently attracted a considerable amount of interests in the community of recommender systems. Compared with CH, which is generally limited to describe categories with child-parent (i.e., *affiliatedTo*) relations, KG connects various types of features related to items (e.g., the genre, director and actor of a movie), in a unified global representation space. Leveraging the heterogeneous connected information from KG helps with the inference of subtler item relationships from different angles, which are difficult to uncover with the homogeneous information (i.e., item category) only. The recommendation accuracy can, therefore, be further boosted with the incorporation of the knowledge graph. Figure 1.2 illustrates a running example of KG in the movie domain, which contains users, movies, actors, directors and genres as entities; rating, categorizing, acting, directing and cooperating as the entity relations. Although there are several studies adopting KG to design effective recommendation algorithms [121, 122, 90, 91], most of them heavily rely on handcrafted features which requires lots of domain knowledge.

To summarize, in this dissertation, we aim to devise novel recommendation approaches by exploiting fine-grained item relationships to achieve high quality recommendations.

1.2 Research Challenges

According to the research scope illustrated in Section 1.1, three recommendation problems are defined and investigated in this dissertation, namely, category hierarchy based recommendation, representation learning based recommendation and knowledge graph based recommendation. Next we will respectively demonstrate the research challenges of these three problems.

Category Hierarchy based Recommendation. Considering a Web product recommender system, the goal is to recommend products to users. Figure 1.1 depicts a 3-layer CH of Women's Clothing, where Women's Clothing is first divide into several generalized categories (e.g., Athletic Clothing), and then further classified into localized subcategories (e.g., Shirts). Compared with flat category structure, CH contains richer knowledge, and thus can be adopted to further boost recommendation performance. Suppose a customer who favors Athletic Shirts, she may possibly like Athletic Pants and Shoes. Since Athletic Shirts, Pants and Shoes all belong to Athletic Clothing, they inherit characteristics from the athletic style. In other words, both the localized category (i.e., Shirts) and the generalized category (i.e., Athletic Clothing) may co-influence the users tastes, possibly with different degrees.

The example highlights how related categories (e.g., Shirts and Athletic Clothing, linked by the *affiliatedTo* relation) can co-influence user preferences. This observation suggests the need for category relations (e.g., the *affiliatedTo* relation) to be properly considered in recommendation methods. This co-influence could be known as a priori, but it is often best learnt from historical user-item interaction data. Existing category-aware methods, e.g., SVDFeature [16], CMF [94] and FM [82], ignore the useful information

provided by category relations, imposing a conversion step that transforms a hierarchical structure into a flat one. To better exploit CH, the main challenge is to model the co-influence of categories on user-item interactions, determined by both the category relations in the hierarchical structure and the historical user-item interaction data.

Actually, categories in the hierarchy of Figure 1.1 are organized in two dimensions: vertical dimension (e.g., Shirts and Athletic Clothing) and horizontal dimension (e.g., Shirts and Pants). The above example only considers the influence of *affiliatedTo* category relations in vertical dimension, while ignores another important dimension of CH, i.e., horizontal dimension. Sibling and cousin categories, i.e., positioned in the same layer of the hierarchy, might capture latent relations that also could be used to better characterize user-item interactions, and, consequently, to enhance recommendation accuracy.

Suppose a customer who prefers athletic style to fashion style. She may purchase more products under the category of Athletic Clothing, such as athletic shoes and pants to match each other, instead of the products under Fashion Clothing, e.g., heels or skirts. In this case, the two sibling categories Athletic Clothing and Fashion Clothing at Layer 2 are characterized by an *alternative* relation, as they are purchased by the user in a mutually exclusive fashion. The sibling categories at Layer 1, Athletic Shoes and Pants, are characterized by a *complementary* relation, as they are jointly purchased by the user. Whereas the cousin categories at this layer, e.g., Athletic Shoes and Heels are *alternative* as determined by the relation of their parent categories.

The above example emphasizes that category relations in horizontal dimension can provide additional characterization of user-item interactions. It is, however, nontrivial to exploit such kind of relations, as the vertical affiliation of categories in different layers should also be preserved. As illustrated in the above example, users' preferences on products (e.g., Athletic Shoes and Heels) could also be affected by the relations of their vertically affiliated categories across different layers (e.g., Shoes – Athletic Clothing, Heels

– Fashion Clothing). In other words, it is often impossible to disentangle the horizontal dimension from the vertical one.

Representation Learning based Recommendation. In recommendation, representation learning is used to capture local item relationships (i.e., item co-occurrence), thus being called item embedding. Item embedding [4] learns low-dimensional item representation by modeling item co-occurrence in individual user's interaction record, thus boosting recommendation accuracy. While it helps learn better item representation, item embedding alone (e.g., Item2Vec [4], CoFactor [58], Meta-Prod2Vec [104]) does not allow for personalized recommendation. Inspired by document RL (e.g., PV-DM [55]), an important branch of work explores the potential of item embedding in personalized recommendation by learning representations for both users and items – as documents and words respectively in NLP (e.g., User2Vec [26]).

However, we argue that the potential of RL for recommendation has not been fully exploited. Existing RL based recommendation methods all ignore the possible multi-level organizations of items for uncovering fine-grained item relationships, which could in turn help achieve better performance. We are inspired by the multi-level word organizations (i.e., word-paragraph-document) in NLP, where paragraph is the intermediate level between individual words and documents. Intuitively, each paragraph conveys a key message, and all the words in the paragraph helps support such message. Analogously, we introduce item categories as the intermediate level of item organization between individual items and items rated by the same user, since items with the same category often share similar characteristics. The key point here is how to efficiently integrate item category with RL to devise a unified recommendation model from the scope of multi-level RL, as well as achieving the goal of personalized recommendation.

Knowledge Graph based Recommendation. The knowledge graph (KG) has proven to be effective in mitigating data sparsity and cold start problems in recommender

systems. It greatly helps to uncover fine-grained item relationships by providing heterogeneous information related to items, i.e., different types of features and relations. State-of-the-art approaches [64, 91, 110, 122, 132] exploit KGs by extending the matrix factorization [75] model with item similarity derived from paths connecting items. The basic intuition is that paths connecting two items in KG represent item relationships of different semantics, which facilitate the inference of user preferences from different perspectives so as to generate effective recommendations. Figure 1.2 shows the idea with a running example.

Consider a KG based movie recommender system, where Bob's preference over SPR[1] can be inferred by: 1) Bob $\xrightarrow{rate}$ TT $\xrightarrow{categorized\ by}$ Drama $\xrightarrow{categorize}$ SPR; 2) Bob $\xrightarrow{rate}$ TT $\xrightarrow{directed\ by}$ Steven Spielberg $\xrightarrow{direct}$ SPR; 3) Bob $\xrightarrow{rate}$ TT $\xrightarrow{directed\ by}$ Steven Spielberg $\xrightarrow{cooperate}$ Tom Hanks $\xrightarrow{act}$ SPR. These paths capture the semantic relations of 1) belonging to a same genre, or 2) being directed by a same director for the movies that Bob has watched. Hence, we may infer that Bob prefers either movies belonging to the genre of Drama, or those directed by Steven Spielberg based on which, we can recommend GWH (belongs to Drama) or SL (directed by Steven Spielberg) to Bob.

The example highlights that different paths connecting a same user-item pair often carry different semantic relations. Typically, they are of different importance in characterizing user preferences over items, i.e., certain paths can better describe user preferences than the others. In the example, Bob's preference over SPR can be driven more by his interest in the genre, than by his preference for the director. To fully exploit paths in KG for recommendation, it requires to capture not only the semantics of different paths but also their distinctive importance in describing user preferences.

Existing methods employing KG heavily rely on handcrafted features to represent path semantics. They define meta paths [98] (i.e., paths with specific types and fixed

[1]For all the movies, we adopt the abbreviation for short.

lengths) in advance, then discover the qualified path instances by mining the KG. The descriptive power of a meta path in characterizing entity relationships is usually defined by the number of corresponding path instances[2]. This process requires domain specific knowledge to define meaningful paths, which is time-consuming. More importantly, manually designed meta-path features are often incomplete in the coverage of possible item relationships, thus severely hindering the recommendation quality.

The popularity of representation learning (RL) recently prompted a seminal work [127] that exploits KG embedding to capture entity semantics for recommendation. In contrast to the meta-path based recommendation approaches with manually extracted paths, these methods automatically learn the embeddings of entities in the KG by leveraging the one level ego-network of entities with their properties, without dependency on the handcrafted features. These embeddings are represented in the form of low-dimensional vectors, referred to as distributed representations, which have shown to be effective in capturing entity semantics in KG [10, 60]. As a result, KG embedding based methods have achieved higher performance than meta-path based methods [127]. However, one major shortcoming of these approaches is the disregard of the semantic relations of entities that are connected by paths – i.e., those are not directly connected in KG – which has been extensively studied in meta-path based methods. Therefore, the challenge here is how to design a new data-driven method that does not rely on handcrafted features, yet can capture both semantics of entities and paths encoded in KG for recommendation.

1.3 Research Approaches

Category Hierarchy based Recommendation. To take advantages of category *affiliatedTo* relations in the vertical dimension of CH, we propose a novel approach – ReMF

[2]We use entity as a generic term to represent all the objects (e.g., user, item, item features) in a KG.

– that models the co-influence of hierarchically-organized categories on user-item interactions, and learns the strength of such co-influence from historical user-item interaction data, thus to improve recommendation performance.

Specifically, we first define the influence of an individual category as regularization [124] on item latent factors, then combine the regularization of individual categories by weighting them recursively over the hierarchy, from root to leaves, according to their organization. The regularization of the category hierarchy, named recursive regularization, is expressed as a regularization function parameterized by the weights associated to each category. We then propose a novel recommendation framework ReMF, that integrates recursive regularization into the matrix factorization model to better learn user and item latent factors. By learning the values of weights of each category from the historical user-item interaction data, ReMF characterizes the influence of different categories in a hierarchy on user-item interactions.

Based on ReMF, in order to fully exploit CH, we further consider category relations in the horizontal dimension of CH, i.e., *alternative* and *complementary*, together with category *affiliatedTo* relation in the vertical dimension, whereby a unified recommendation framework HieVH is thus proposed by seamlessly integrating both dimensions of CH, so as to help achieve better recommendation performance.

In particular, to model the vertial dimension, HieVH adapts latent factors of items by adding weighted aggregation of their affiliated categories' latent factors, to better model item latent factors. The weights are automatically learnt from data. Horizontally, category relations, i.e., *alternative* and *complementary*, are incorporated as regularizers at each layer of the hierarchy, to better model category latent factors. In so doing, through the adaption of item latent factors with category latent vectors in vertical dimension, category relations in horizontal dimension can be inherited by items. The result is a method that can seamlessly fuse vertical and horizontal dimensions of CH. While previous CH

based methods (e.g., ReMF) consider vertical dimension, we stress that it is nontrivial to extend them to integrate horizontal dimension, due to the lack of a matching mechanism in vertical dimension such as the use of category latent factors.

Representation Learning based Recommendation. To reach the full exploitation of RL for effective recommendation, we contribute a multi-level RL method – MRLR – for personalized recommendation. MRLR not only captures fine-grained item relationships by leveraging category RL as the intermediate level RL between item RL and user RL, but also achieves the goal of personalization.

As the original item embedding method only learns from item co-occurrence relationships, whereas for personalized recommendation the method has to learn from user-specific lists of rated items w.r.t. user preferences. We hence first extend the original item embedding method to a more generic Bayesian framework, under which we then fuse the likelihood function of user-specific pairwise item ranking. This unified framework can then learn user and item embedding from both item co-occurrence relationships and user-specific ranked lists of items, benefiting from user and item RL while reaching the goal of personalized recommendation.

Next, we further extend the personalized recommendation framework to multi-level RL by considering multi-level granularity of item organizations, so as to help capture fine-grained item relationships. Specifically, we introduce item category as the intermediate level between items in the same user-specific ranked list and individual items. By leveraging category RL to adapt item embeddings, item category is seamlessly integrated into the recommendation framework. The rationale behind is that items in a same category generally share similar characteristics. For instance, online products are often described by categories as meta-data such as clothing, books, electronics, and so on.

The unified Bayesian framework therefore facilitates multi-level RL by combining RL in all the three levels (i.e., individual item, item category, and user). Although item

category has recently been intensively studied [33, 117], we are the first to investigate it from the perspective of multi-level RL, which enables our framework to capture the fine-grained relationships of items in local context (i.e., item co-occurrence relationships), in the same category, and in user-specific ranked item list.

Knowledge Graph based Recommendation. Inspired by the two lines of work that adopts KG for better recommendation, i.e., meta-path based methods and KG embedding based ones, illustrated in Section 1.2, we seek for a new data-driven method that does not rely on handcrafted features, yet can capture both semantics of entities and paths encoded in KG for recommendation.

To this end, we consider to use recurrent neural networks (RNN) [17, 114, 126] to learn the semantic relations between entities encoded by paths to improve recommendation. An important advantage of RNN is that it can model sequences with varying lengths, making it particularly suitable for modeling paths – i.e., sequences of different numbers of entities in KG. Most importantly, RNN can not only model the semantics of entities (with an embedding layer [119]), but also the semantics of paths that connect different entities by encoding the entire path, thus providing a unified approach for learning representations of both entities and relations to fully exploit KG semantics for recommendation. However, the application of RNN to model KG for recommendation is not trivial, given the different descriptive power of paths in characterizing user preferences.

We therefore proposes a unified recurrent knowledge graph embedding framework RKGE, which is able to not only learn semantic representations of entities and paths in a fully automatic way, but also to automatically discriminate the importance of different paths for recommendation. In order to learn the relations between a pair of entities, RKGE first mines all the paths linking paired entities which carry different semantics, without predefining the specific types of the path. It then encodes all paths between the entity pair through a batch of RNNs, with each path modeled by a single RNN. RKGE

is thus flexible in capturing different numbers of paths with various lengths that connect entity pairs. The different effects of paths are then learned through a pooling operation, which further discriminates the importance of different paths and aggregates their effects for learning user preferences.

1.4 Research Contributions

Our major contributions in this dissertation are summarized as follows:

- We define and investigate three recommendation problems, which aim at utilizing auxiliary item relationships to help achieve high recommendation performance from different perspectives. They are respectively category hierarchy, representation learning, and knowledge graph based recommendation.

- To leverage category *affiliatedTo* relation in vertical dimension of CH, we propose a novel regularization method named Recursive Regularization for modeling the co-influence of categories in the hierarchy on user-item interactions. Based on this, a new recommendation framework ReMF is then proposed to learn hierarchical category influence from historical user-item interaction data. Different from other existing CH based methods that simply convert CH into flat structure, ReMF models the topological structure of CH, and automatically learns the co-influence of categories in different layers of CH. Experimental results show that ReMF can largely outperform state-of-the-art methods.

- To fully exploit category relations in both vertical and horizontal dimensions of CH, we propose a unified recommendation framework HieVH that seamlessly integrates both dimensions for effective recommendation. Besides category *affiliatedTo* relation in vertical dimension of HieVH further considers two types of semantically rich

category relations in horizontal dimension, i.e., *complementary* and *alternative* relations. Therefore, HieVH advances existing CH based recommendation methods, which only consider vertical dimension of CH. Extensive validation demonstrates the superiority of HieVH against the state-of-the-art. An additional benefit of HieVH is to provide better interpretations of the generated recommendations.

- To take advantage of RL in capturing local item relationships, we propose a multi-level RL framework for personalized recommendation – MRLR. Different from other existing RL based recommendation approaches which ignore either personalization or multi-level item organization, MRLR learns both user and item embeddings from a multi-level item organization for better recommendation. Therefore, it benefits from RL as well as achieves the goal of personalized recommendation. Empirical validation on real-world datasets shows that MRLR achieves better recommendation performance than the state-of-the-art algorithms.

- To employ the heterogeneous connected information encoded in KG, we propose a KG embedding recommendation framework – RKGE – based on a novel recurrent network architecture for high quality recommendation. Different from previous KG based recommendation methods, which either heavily reply on handcrafted features and domain knowledge, or fail to capture the semantics of entities and entity relations, RKGE can not only learn the semantic representation of different types of entities, but also automatically capture entity relations encoded in the KG. Extensive validation on two real-world datasets demonstrates the superiority of RKGE against other counterparts.

Figure 1.3 depicts the overall structure for relations of the research problems considered in this dissertation, and summarize the contributions. To address the *data sparsity*

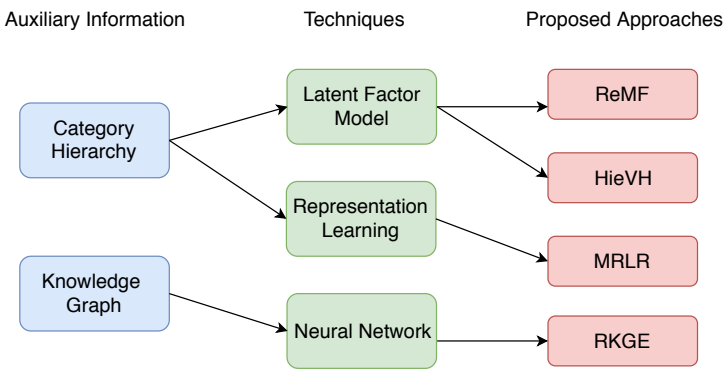


Figure 1.3: The overall structure to illustrate relations of the research problems considered in this dissertation.

and *cold start* problems of recommender systems, we propose a series of novel recommendation approaches from two different angles, i.e., auxiliary information and techniques. First, we leverage the vertical *affiliatedTo* relation of category hierarchy as auxiliary information to help infer item relationships, and then incorporated them into latent factor model to propose ReMF; Second, by considering both vertical and horizontal dimensions of category hierarchy, another latent factor model based approach HieVH is devised. Third, as representation learning technique has proven to be more effective to capture item relationships than latent factor model, we integrate item category information with representation learning to design MRLR; Lastly, we consider more complicated auxiliary information, i.e., the knowledge graph, which is capable of inferring more fine-grained item relationships from different angles. Meanwhile, from the perspective of techniques, as neural network is more efficient to capture complex interaction patterns between entities, e.g., nonlinear relations between entities, we propose a neural network based approach RKGE to capture semantics of both entities and entity relations encoded in the knowledge graph, thus to further enhance recommendation performance.

1.5 Dissertation Organization

The rest of the dissertation is organized as follows. In Chapter 2, we provide an overview of previous studies related to our research problems. Chapter 3 presents our approach to category hierarchy based recommendation from the angle of leveraging the *affiliatedTo* category relation in vertical dimension of CH. From the point of utilizing both vertical and horizontal dimensions of CH, Chapter 4 introduces our unified framework w.r.t. category hierarchy based recommendation to further enhance the recommendation performance. In Chapter 5, we demonstrate the proposed model for representation learning based recommendation. Chapter 6 illustrates the proposed approach for knowledge graph based recommendation. Finally, Chapter 7 concludes this dissertation and points out several promising directions for future work.

Chapter 2

Literature Review

In this chapter, we provide an overview of state-of-the-art recommendation algorithms that are related to our study. Specifically, we first make a brief introduction about one of the most successful recommendation techniques, i.e., collaborative filtering (CF), and then focus on the recommender systems that exploit item relationships to help resolve the inherent issues of CF, i.e., *data sparsity* and *cold start*. Particularly, we survey the literature related to the three recommendation problems that have been defined in Chapter 1, i.e., category hierarchy based recommendation, representation learning based recommendation and knowledge graph based recommendation, and point out their strength and limitations in the light of recommendation performance.

2.1 Collaborative Filtering

Recommender systems have become an important part of modern applications to help deal with the *information overload* problem, where the goal is to actively recommend relevant items to users by modeling user preference based upon the historical user-item interaction records (e.g., user-item rating matrix). One of the most successful and prevalent recommendation techniques is collaborative filtering (CF) [93, 97], which is built upon the assumption that a user's preference can be inferred by aggregating the tastes of her similar users.

2.1.1 Memory- and Model-based Approaches

Generally, two types of CF approaches are widely investigated, namely, memory-based methods and model-based methods, which will be elaborated in the following.

2.1.1.1 Memory-based Approaches

Memory-based approaches [19] exploit user-user or item-item similarity derived from the user-item rating matrix to make recommendations. User-oriented methods [108] and item-oriented methods [86] are the two kinds of typical memory-based approaches.

Specifically, user-oriented approaches identify like-minded users who can complement each other's ratings. In other words, the ratings of target users are predicted based on the ratings of similar users found in the system. Whereas, item-oriented approaches evaluate a user's preference for an item based on the ratings of neighboring items rated by the same user. Memory-based methods are thus also called neighborhood-based methods, as the key point is to find a number of reliable neighbors for the target users or items, when generating recommendations.

Various types of similarity measurements have been proposed, so as to help efficiently uncover the nearest neighbors of users or items in the system. The widely utilized approaches include cosine similarity (COS) [84], Pearson correlation coefficient (PCC) [11], Jaccard Similarity (JS) [67], Bayesian Similarity (BS) [30], etc. The similarity computation thus has direct and significant influence on the performance of memory-based recommendation methods. Although memory-based approaches are adopted in some real applications such as CiteULike[1], Youtube[2] and Last.fm[3], they have been recognized to be ineffective to large-scale data sets due to the time-consumption searching in user or item space.

[1] http://www.citeulike.org/
[2] http://www.youtube.com/
[3] http://www.last.fm/

2.1.1.2 Model-based Approaches

In contrast, model-based CF approaches aim to build models by adopting data mining or machine learning techniques on user-item rating matrix to uncover the complex user behavior patterns (offline). The learned models are then used to predict ratings of unknown items (online). Therefore, they can better adapt and scale up to large-scale data sets. Besides, model-based approaches usually achieve better recommendation performance than memory-based ones [45]. Typical examples include Bayesian based models [72, 73], clustering based models [87, 15, 85, 115, 31], regression based models [78, 106, 57], topic model based approaches [7, 38, 54], latent factor based models [93, 52, 82, 83, 75], representation learning based models [58, 102, 26, 104, 4], and neural network based models [107, 129, 77], etc. Among them, latent factor based models, representation learning based models and neural network based models are the most prevalent approaches, which are also the fundamental models of our proposed approaches in this dissertation.

Latent Factor Models. Due to the high efficiency, state-of-the-art recommendation methods are mainly dominated by the latent factor model (LFM), which decomposes the high-dimensional user-item rating matrix into low-dimensional user and item latent matrices. The basic idea behind is that both users and items can be characterized by a number of latent features, and thus the prediction can be computed as the inner product of user-feature and item-feature vectors. Many effective approaches fall into this category, such as matrix factorization (MF) [75], non-negative matrix factorization (NMF) [128], tensor factorization (TF) [48], factorization machine (FM) [82], SVD++ [52], collective matrix factorization (CMF) [94] and SVDFeature [16]. LFM based approaches learn and model users' rating patterns by employing global statistical information of user-item interaction data.

Representation Learning based Models. In contrast to LFM, representation learning (RL) based methods have proven to be effective in capturing local item relationships.

It is thus called item embedding in recommendation. These methods are mostly inspired by word embedding techniques, which can be traced back to the classical neural network language model [5], and the recent breakthrough of Word2Vec techniques, including CBOW and Skip-gram [70, 71]. Item embedding learns low-dimensional item representation by modeling item co-occurrence in individual user's interaction record [26, 4, 58], thus boosting recommendation accuracy.

Neural Network based Models. Stemming from the success in related domains (e.g., computer vision, speech recognition, and natural language processing), neural network (NN) based methods have recently attracted major research interests from the recommendation community. In contrast to LFM and RL based methods, NN based models (e.g., AutoRec [88], NCF [34]) can learn nonlinear latent representations through various types of activation functions (e.g., sigmoid, ReLU [76]). Recently, recurrent neural network (RNN) based approaches [36, 39, 46, 114] have gained significant enhancement in recommendation thanks to the ability of preserving historical information over time or dealing with the sequence information for recommendation. To sum up, NN based methods possess essential advantages and have shown to be more effective to enhance recommendation performance. While the model complexity of NN based models is generally higher than latent factor based model and representation learning based one.

2.1.2 Incorporation of Auxiliary Information

Although traditional CF based methods achieve significant success on recommender systems, it inherently suffers from *data sparsity* and *cold start* problems [29]. The former refers to that most users only rate a small portion of items, while the latter indicates some users only rate a small number of items.

To resolve these issues, a notable research field is the trust-aware recommender systems which take into account additional user relationships. Many approaches [32, 116,

65, 27, 28, 43] are emerging with the advent of social networks. The intuition is that social friends share similar preferences and influence each other by recommending items. It has been shown that such additional side information among users is useful to deal with the concerned issues of traditional CF based methods and thus to boost recommendation accuracy. However, these approaches suffer from several issues. First, social information may be unavailable for some real applications. Second, some users may only have few friends, or even not be active in the social networks, i.e., cold start is also a problem of social networks. Third, the potential noise and weaker social ties (than trust) in social networks may also produce negative effects on the generality of these approaches [32, 20].

Another related line of research focuses on exploiting the side information of items, given its effectiveness in boosting recommendation performance [52]. The rationale behind is that users tend to have similar preferences towards a set of *correlated* items. Plenty of approaches [62, 51, 92, 40, 49, 99] have been proposed by utilizing a wide range of item relationships such as category, genre, location, etc. Compared with user relationships (e.g., social network), item relationships (e.g., item category) are more amenable to help explain the underlying reason behind recommendations, as users are familiar with items previously preferred by them, but do not know those allegedly like-minded users [52]. Furthermore, the item side information is generally available in the real systems, and thus is much easier to be obtained. Therefore, in this dissertation, we focus on exploiting item relationships to alleviate the concerned issues of CF.

In the following, we will provide a detailed literature review about the three recommendation problems defined in Chapter 1 which aim to exploit item relationships to tackle the inherent issues of recommender systems, thus achieving high recommendation performance. They are respectively category hierarchy based recommendation, representation learning based recommendation, and knowledge graph based recommendation.

2.2 Category Hierarchy based Recommendation

This section reviews studies related to category hierarchy based recommendation. Specifically, we start with the generic category-aware recommendation models that exploit flat category structure for better recommendation, then focus on the approaches that take advantages of category hierarchy, followed by the methods that adopt implicit hierarchy.

2.2.1 Modeling Flat Structure

Many category-aware recommendation methods consider only categories with a flat structure. Early works simply employ item category in data pre-filtering, which causes the data sparsity problem to be even more severe. For instance, Sharma et al. [89] propose a memory-based CF approach to compute users' category-specific neighbors by dividing users' ratings into different sub-groups according to product category. The underlying assumption is that users may rate products similarly in certain categories but differently in the others. Hwang et al. [42] refer to category experts as those who have high expertise in specific categories, and contend that users have the tendency to seek advice from category experts rather than strange users of similar preferences. They propose a memory-based method by aggregating the ratings of category experts instead of those of similar users. Yang et al. [118] develop a model-based method by inferring category-specific social trust circles from the rating matrix and social networks. The authors argue that a user may trust different subsets of friends dependent on item categories. Liu et al. [62] propose a novel category-aware POI recommendation model by leveraging the transition patterns of users' preference over location category.

Later, some methods attempt to integrate item category into a learning model without information loss caused by pre-filtering. Ji et al. [45] devise a two-phase layered learning model. It first calculates user's average rating to each category, and then learns accurate estimates of users' rating for individual item by adopting item keywords. Recently, Hu et

al. [40] develop a geographical neighborhood based business recommendation approach by taking into account the influence of business category. They argue that a business's latent feature vector can be influenced by its corresponding category latent feature vectors. Therefore, they adopt the weighted linear combination of a business vector and its corresponding category vectors as the final business vector.

Furthermore, several techniques originally designed for other scenarios can be potentially adapted and applied to the category-aware recommendation, including:

- The presence of user-item rating matrix and item-category affiliation matrix inspires us to adopt the technique of collective matrix factorization (CMF) [61, 94], which takes advantage of correlations among different datasets and simultaneously factorizes coupled matrices.

- A straightforward technique to handle auxiliary information is known as Multiverse Recommendation (MR) [48], which is based on Tensor Factorization (TF). TF is a generalization of matrix factorization in multiple dimensions. By assuming that user, item and category are located at independent dimensions respectively, a 3-D tensor is composed, which is then factorized into low-dimensional user, item and category latent factors for better recommendation.

- Another relevant technique is SVDFeature, which is a machine learning tookit, and devised by Chen et al. [16]. The basic idea behind is that an item's latent factor is influenced by those of its corresponding categories.

- Rendle et al. [82] design a classic feature based recommendation model, i.e., factorization machines (FM), that combines the advantages of support vector machines (SVM) with the factorization model. FM is a general predictor working with any real valued feature vector, and it usually performs faster and provides better recommendation performance than the other three methods mentioned above.

However, all of the above methods merely consider item category with a flat structure, ignoring the relationships among different categories. They all fail to handle the situation where categories are hierarchically organized. Therefore, some researchers attempt to make use of category hierarchy to further enhance the recommendation accuracy, which will be introduced in the next subsection.

2.2.2 Modeling Category Hierarchy

Some studies on taxonomy-aware recommendation incorporate hierarchy into recommendation. For example, Ziegler et al. [133] propose to model a user's taxonomy preferences as a flat vector, where each element corresponds to the user's preference over a taxonomy feature. The user's preference is modeled as the frequency that the user rates items characterized by the feature (i.e., category). Based on the assumption that users who share similar item preferences may also share similar taxonomic preferences, Weng et al. [112] further propose a novel recommendation method that combines the users' preferences towards items and the additional taxonomic preferences together to generate better quality recommendations. In contrast, the previous model in [133] only considers users' taxonomic preferences when making recommendations. After that, Albadvi et al. [2] also propose a similar approach, however it models each taxonomic feature as a preference vector, where the elements are feature attributes (e.g., price, brand). All of these methods, however, overlook the relations among different features (i.e., categories).

Later, several approaches are proposed based upon SVDFeature [16]. For instance, Dror et al. [51] design a new matrix factorization model for Yahoo! Music competition that incorporates the taxonomy hierarchy of track album and artist. They predict user preferences by fusing item (e.g., track) latent factors with category (e.g., album, artist) latent vectors. Based on the same intuition, Minh et al. [74] later propose a matrix factorization based method to the Track 2 task of KDD Cup 2011[4]. They utilize the

[4]http://www.kdd.org/kdd2011/kddcup.shtml

item relationships information from the provided taxonomy to constrain item latent vectors and adopt the pairwise ranking route [83] at the meanwhile, resulting in improved predictive performance. Kanagal et al. [47] also devise a taxonomy-aware matrix factorization ranking model, which combines taxonomies and latent factors using additive models. They develop efficient algorithms to train the proposed model, which scales to a large number of users or items and develop scalable inference algorithms by exploiting the structure of the taxonomy. In addition, they extend the model to account for the temporal dynamics of user interests using high-order Markov chains. After that, Lu et al. [63] propose a music recommendation model with taxonomy hierarchy. Different from the approaches devised in [51, 74, 47], they assume that user latent factors are influenced by the linked taxonomies in the hierarchy with equal weights.

Although category relations are considered, all the methods above cannot fully exploit the information encoded in category hierarchy as they simply add relevant category latent factors to the corresponding item or user latent factors, without taking into account the dependent influence of hierarchically-organized categories on user-item interactions. To sum up, blending category hierarchy into all the above models requires converting the hierarchy into a flat structure, thus losing the structural information encoded in the hierarchy. This severely hinders further improvements of recommendation performance.

There are still some works endeavoring to exploit the structural information of category hierarchy. Menon et al. [69] propose an ad-click prediction method for online advertising via considering ad hierarchy. They devise regularizers for the node (i.e., ads and their affiliated categories in the hierarchy) and its direct parent node to constrain the distance of their corresponding latent factors. However, the model assumes that given its parent, a node is conditionally independent of all higher level nodes in the hierarchy. Recently, based on the intuition that an item's property is influenced by its visual appearance, He et al. [33] propose an efficient sparse hierarchical embedding method for

visually-aware recommendation, called Sherlock. It is scalable and allows simultaneously learning both general and subtle visual dimensions of items, captured by different layers on the category hierarchy with different degrees. However, it manually defines the different effects of categories in the hierarchy on item latent factors.

In summary, existing methods are incapable to model the co-influence of hierarchically-organized categories on user-item interactions, thus restricting their applications in recommendation. Most of them simply blend category hierarchy into flat structure, leading to server topological information loss, thus hindering further recommendation performance improvements. In contrast, our proposed framework ReMF can better exploit the auxiliary category hierarchy through the automatic learning of hierarchical category influence by a parameterized regularization traversing from root to leaf categories. Furthermore, we argue that the potential of category hierarchy has not been fully employed, i.e., the influence of hierarchy should not be limited only to the category *affiliatedTo* relation in the vertical dimension, but also relations in horizontal one. Therefore, our unified approach HieVH seamlessly models both dimensions of category hierarchy for effective recommendation by further considering semantically rich category relations, i.e., *alternative* and *complementary* in the horizontal dimension of CH.

2.2.3 Modeling Implicit Hierarchy

There are a few methods [109, 130] leveraging implicit hierarchy for better recommendation. Based on the assumption that user preferences and item properties in real-world recommender systems exhibit certain hierarchical structure, unfortunately explicit hierarchy is not always available on the web, Zhang et al. [130] propose a novel recommendation framework HSR. It enables to capture the influence of implicit hierarchical structures of both users and items on the user-item interactions simultaneously. To alleviate the data sparsity problem of recommender systems, Wang et al. [109] design a novel taxonomy-aware matrix factorization approach that automatically discovers the taxonomies of items

from online shopping data and jointly learns a taxonomy-based recommendation system. The underlying assumption is also similar with SVDFeature [16], i.e., an item's latent factor is influenced by those of its affiliated categories in the taxonomy.

The implicit hierarchical structure, however, is merely learnt from historical data [109, 130], thus cannot truly uncover the inherent relations among different categories, leading to poor recommendation performance. On the contrary, explicit hierarchy is generally expert-induced, injecting prior knowledge and encoding category relations in the semantical level. In this dissertation, we therefore focus on how to take advantage of explicit category hierarchy for effective recommendation.

2.3 Representation Learning based Recommendation

Representation learning based methods have recently drawn much attention in the community of recommender systems. In this section, we review related state-of-the-art RL based recommendation methods, which are mainly classified into two types, i.e., non-personalized RL approaches and personalized ones.

2.3.1 Non-personalized RL Approaches

In contrast to LFM based approaches (e.g., matrix factorization), RL based approaches have shown to be highly effective in capturing local item relationships by modeling item co-occurrence in individual user's interaction record. Several RL based methods have been proposed to date. For instance, inspired by Word2Vec, Barkan and Koenigstein [4] propose a neural item embedding method (Item2Vec) for item-based collaborative filtering that produces embedding for items in a latent space. The method is capable of inferring item-item relationships even when user information is not available. Later, Vasile et al. extend Item2Vec to a more generic approach named Meta-Prod2Vec [104], which is a novel method to compute item similarities for recommendation that leverages

existing item meta-data. The proposed method leverages past user interactions with items and their attributes (i.e., categories) to compute low-dimensional embeddings of items. Specifically, the item metadata is injected into the model as side information to regularize the item embeddings.

Experimental results have demonstrated that Meta-Prod2Vec consistently outperforms Item2Vec on recommendation tasks both globally and in the cold-start regime, which suggests the effectiveness of the incorporation of item category for better recommendation. However, they all fail to provide personalized recommendation, as item embedding techniques are only utilized to learn better item representation, ignoring the user representation. Therefore, an important branch of work explores the potential of item embedding in personalized recommendation by learning representations for both users and items.

2.3.2 Personalized RL Approaches

Item embedding alone (e.g., Item2Vec [4], Meta-Prod2Vec [104]) does not allow for personalized recommendation. Inspired by document RL (e.g., PV-DM [55]), several researchers endeavor to extend RL for personalization by learning representations for both users and items – as documents and words respectively in NLP.

Grbovic et al. [26] first introduce the User2Vec recommendation framework, which simultaneously learns representations of items and users by considering the user as a global context, motivated by the paragraph2vec algorithm [55]. One of the main advantages of the User2Vec approach is that the product recommendations are specifically tailored for that user based on his purchase history. Later, Liang et al. [58] propose the CoFactor method, which jointly decomposes the user-item interaction matrix and the item-item co-occurrence matrix with shared item latent factors. For each pair of items, the co-occurrence matrix encodes the number of users who have consumed both

items. CoFactor is also inspired by the recent success of word embedding models (e.g., Word2Vec) which can be interpreted as factorizing the word co-occurrence matrix. It is thus equivalent to item embedding.

However, we argue that the potential of RL for recommendation has not been fully exploited, as these methods all ignore the possible multi-level organizations of items for uncovering fine-grained item relationships in recommendation (similar as word-paragraph-document in NLP), which could in turn help achieve better personalized recommendation performance. Therefore, we contribute a multi-level RL method for personalized recommendation (MRLR). Specifically, our method is inspired by paragraphs in NLP as the intermediate level of word organization between individual words and documents. Analogously, we introduce item categories as the intermediate level of item organization between individual items and items rated by the same user, since items with the same category often share similar characteristics. By leveraging category RL as the intermediate level RL between item RL and user RL, MRLR is able to capture fine-grained item relationships, as well as achieve personalized recommendation, so as to further enhance recommendation performance.

2.4 Knowledge Graph based Recommendation

The knowledge graph (KG) has proven to be effective in mitigating data sparsity and cold start problems in recommendation. It greatly helps to discover subtler item relationships by providing heterogeneous information related to items, i.e., different types of features and relations, thus facilitating to infer user preference towards items from different angles. This section provides an overview of the state-of-the-art methods that exploit KG for better recommendation. They are generally classified into three types, namely graph based methods, meta path based methods, and embedding based ones.

2.4.1 Graph based Approaches

A line of research focuses on making use of KG by designing graph based methods. Early work [25] proposes a method by applying the spreading activation technique [81] on KG to provide lower rating estimation error and higher coverage for recommendation compared to those collaborative filtering methods only using user-item interactions. This proves the usefulness of KG for better recommendation. Later, Pham et al. [79, 80] propose HeteRS to solve recommendation problems in event-based social networks. They transform the recommendation problem into a node proximity calculation problem and employ Markov chain to solve it. By automatically learning the transition parameters, HeteRS not only achieves superior performance, but also helps understand the roles of different types of entities in recommendation. After that, Catherine et al. [12] investigate a recommendation approach that employs a general purpose probabilistic logic system called ProPPR, standing for Programming with Personalized PageRank, to infer user preferences through logic reasoning based on KGs. Recently, Chaudhair et al. [14] introduce a recommender system – RERA – that adopts a novel normalized version of Personalized Page Rank to rank candidate items for recommendation.

Nevertheless, these graph based methods are mainly based on the random walk process [21], which can be easily biased to the popular and centered entities in KG. More importantly, they only make use of the topological structure of KG without considering to model the semantics of entities and entity relations encoded in the KG, thus deteriorating the recommendation accuracy.

2.4.2 Meta Path based Approaches

Two entity types can be connected via different paths in KG. These paths may contain different entities and relations in different orders and have various lengths because of the multiplicity of KG. To clearly describe the path types, meta path [98, 120] is introduced

to help predefine the specific format and length of the paths, as well as capturing different semantics carried by KG. To measure the proximity of the entities that are connected by meta path, a series of metrics have been proposed – to name a few – Personalized PageRank score [13], Random Walk score [59], and PathSim Score [98]. State-of-the-art recommendation methods utilizing KG have been dominated by meta path, which generally leverage meta path to build feature space and then manually extract features from KG for better recommendation.

In order to take advantage of KG, Yu et al. [121] devise HeteMF – a matrix factorization [75] recommendation framework with meta path based entity similarity. It decomposes the user-item rating matrix, meanwhile adopts graph regularization [95] to constrain the distance of latent vectors of similar items that are connected by meta paths. Later, they propose HeteRec [123] to learn user preference diffusion to the unrated items that are connected with their rated items via different meta paths in KG. This model is designed for implicit feedback and estimated by the Bayesian ranking optimization technique [83]. It is further extended by the same authors of [123] to incorporate personalization via clustering users based on their interests. Note that all the above methods employ meta path in the scope of item-item relationships.

There are also some related works from the perspectives of either user-user or user-item relationships. For example, Luo et al. [64] investigate a social network based recommendation algorithm on KG named HeteCF to model the relationships of user-item, user-user and item-item by meta-path based similarity. In order to accurately capture semantic relationships among users, Shi et al. [91] propose the SemRec model and introduce the concept of weighted meta path, which aims to depict the path semantics by distinguishing subtle differences among link attribute values. Later, the same authors design a matrix factorization based dual regularization framework SimMF [90], whereby they design regularization terms for both users and items with the help of meta path

based similarity. Similarly, Wang et al. [110] and Zheng et al. [132] also devise matrix factorization approaches by regularizing user-user relationships with the computed meta path based similarity.

Despite of their success for recommendation, all existing meta path based methods suffer from an essential limitation: they heavily depend on the handcrafted features. That is to say, the specific types and the lengths of the paths need to be predefined in a subtle manner, which makes it difficult to generate a comprehensive feature space as well as time-consuming, thus leading to a limited recommendation quality.

2.4.3 Embedding based Approaches

The most recently proposed algorithm is collaborative knowledge graph embedding (CKE) [127], which jointly learns the item latent representations in collaborative filtering as well as from the KG. To capture the structured information of entities and their rich relations, CKE embeds KG into a continuous vector space via TransR [60], which is the state-of-the-art embedding approach for heterogeneous network. By modeling the relations between any two directly connected entities, CKE automatically extracts item representations from KGs structural content. The empirical study demonstrates the superiority of CKE against graph and meta path based methods.

However, TransR cannot explicitly learn the relations of paired entities that are linked by a path, thus failing to capture the full semantics carried by KG. In contrast, our proposed framework – RKGE – is designed to learn both semantics of the entities and the relations among entities. It first automatically mines all the linked paths with different semantics between entity pairs, then models all the paths by a batch of RNNs, meanwhile learns the respective path saliency on users preference towards items. By doing so, RKGE not only takes advantage of representation learning for better KG embedding, but also achieves a full exploitation of entity relations.

2.5 Summary

This section provides a systematic summary of the current research related to our study. We first briefly introduce one of the most successful techniques for recommender systems, i.e., collaborative filtering, which includes memory-based methods and model-based methods. Then, we make a comprehensive survey about the state-of-the-art studies that corresponds to the three recommendation problems defined in Chapter 1, namely, category hierarchy based recommendation, representation learning based recommendation, and knowledge graph based recommendation.

By analyzing strength and weakness of existing works w.r.t. the three recommendation problems in depth, we propose a series of recommendation framework to alleviate their respective limitations:

(1) Most of existing CH based recommendation approaches adopt CH for better recommendation by directly blending the hierarchical structure into flat structure, thus leading to server information loss and limited performance improvements. To tackle with this issue, we propose a novel matrix factorization framework with recursive regularization – ReMF – to better leverage the topological structure of CH, i.e., vertical category *affiliatedTo* relation of CH in Chapter 3;

(2) Most of existing CH based recommendation methods only focus on the vertical *affiliatedTo* category relations, while ignore another importance dimension of CH. Therefore, to make a fully exploration of CH, we design a unified framework HieVH to seamlessly incorporate category relations in both vertical and horizontal dimensions of CH for better recommendation in Chapter 4;

(3) Most of existing RL based recommendation methods ignore either personalization or multi-level item organization for better recommendation. To resolve this issue, a

unified Bayesian framework MRLR is then devised to make full use of representation learning for better recommendation in Chapter 5;

(4) Most of existing KG based recommendation approaches either heavily rely on hand-crafted features and domain knowledge, or fail to capture the semantics of entities and entity relations encoded in the KG. Therefore, in Chapter 6, we propose a KG embedding recommendation approach based on a novel recurrent network architecture, so as to fully exploit the heterogeneous information encoded in KG, thus achieving high quality recommendation.

Chapter 3

ReMF: Recommendation with Recursive Regularization

As emphasized in Chapter 1, the categories with *affiliatedTo* relation in category hierarchy (CH) can co-influence user preferences, possibly with different degrees. This suggests the need for category relation (i.e., *affiliatedTo*) to be properly considered in recommendation methods. This co-influence could be known as a priori, but it is often best learnt from historical user-item interaction data. Existing category-aware methods, e.g., SVDFeature [16], CMF [94] and FM [82], ignore the useful information provided by category relation, imposing a conversion step that transforms a hierarchical structure into a flat one.

This chapter proposes a novel approach that models the co-influence of hierarchically-organized categories on user-item interactions, and learns the strength of such co-influence from historical user-item interaction data, to improve recommendation performance. We first define the influence of an individual category as regularization on latent factors, then combine the regularization of individual categories by weighting them recursively over the hierarchy, from root to leaves, according to their organization. The regularization of the CH, named recursive regularization, is expressed as a regularization function parameterized by the weights associated to each category. We then propose a novel recommendation framework ReMF [117], that integrates recursive regularization into the matrix factorization model to better learn latent factors. By learning the values of weights of each

category from the historical user-item interaction data, ReMF characterizes the influence of different categories in the hierarchy on user-item interactions.

This chapter is organized as follows: we first introduce the recursive regularization method for modeling the co-influence of category with *affiliatedTo* relation in Section 3.1; based on this, we then propose the recommendation framework ReMF that incorporates recursive regularization to achieve high quality recommendation in section 3.2, followed by the empirical evaluation in section 3.3. Finally, Section 3.4 concludes this chapter.

3.1 Recursive Regularization

We adopt the regularization technique [124] to model the influence of auxiliary categories. To do so, we have to consider category relations, and further allow for the learning of category influence from historical user-item interaction data. For this we introduce a novel regularization method, named recursive regularization, that models the co-influence of categories by recursively weighting each category influence, traversing from root to leaves in the category hierarchy.

3.1.1 Preliminaries

We first introduce the notations utilized in this chapter. Let $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ be the set of m users, and $\mathcal{I} = \{v_1, v_2, \ldots, v_n\}$ be the set of n items. Given a user-item interaction matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$, $\mathbf{R}_{ij}$ is a positive number denoting the rating given by u_i to v_j. $\mathbf{O} \in \mathbb{R}^{m \times n}$ denotes the indicator matrix, where $\mathbf{O}_{ij} = 1$ indicates that u_i rates v_j, and $\mathbf{O}_{ij} = 0$ otherwise. $\mathcal{C} = \{C_1, C_2, \ldots, C_t\}$ is the set of categories, each of which describes at least one item in $\mathcal{I}$.

The categories are organized hierarchically in a tree structure, where each node represents a category in $\mathcal{C}$. The edge between a parent node $C_p \in \mathcal{C}$ and a child node $C_c \in \mathcal{C}$ represents a directed affiliation relationship, i.e., C_c belongs to C_p. Figure 3.1(a) shows

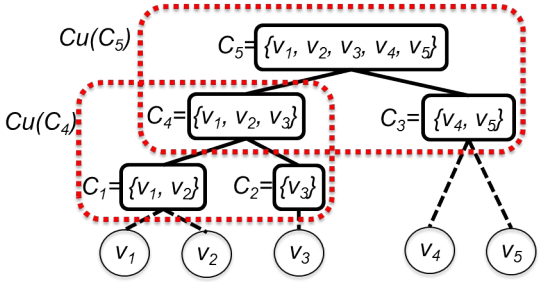
Table 3.1: Mathematical Notations for ReMF

Notation	Description
$\mathcal{U}$,$\mathcal{I}$	user, item set
u_i, v_j/v_k	the i^{th} user in $\mathcal{U}$, and j^{th}/k^{th} item in $\mathcal{I}$
$\mathbf{R}_{ij}$	rating given by user u_i to item v_j
$\hat{\mathbf{R}}_{ij}$	estimated rating for user u_i to item v_j
$\mathbf{O}$	indicator matrix indicating missing entries in $\mathbf{R}$
$\mathbf{U}_i$, $\mathbf{V}_j$	latent factors of user u_i and item v_j
$\mathcal{C}$	hierarchically-organized category set
C	category in the hierarchy
$Dis(c)$	regularization induced by *isolated* category C
$Cu(c)$	category unit with parent node C
$\mathbf{I}'(C)$	regularization by *isolated* category unit $Cu(C)$
g, s	weighting parameters in propagating category influence
$\mathbf{I}(C)$	regularization by category unit $Cu(C)$ in hierarchy
$\mathbf{I}(\mathcal{C})$	regularization by category hierarchy $\mathcal{C}$
$\mathbf{C}_{jk}$	regularization coefficient between $\mathbf{V}_j$ and $\mathbf{V}_k$
α	impact of recursive regularization
λ	regularizaton coefficient to avoid over-fitting
$\mathcal{J}$	objective function of ReMF framework

an example containing three leaf categories C_1, C_2, C_3, i.e., categories with no children. C_1, C_2 are children of the internal category C_4. C_3 and C_4 are children of the root category C_5. For simplicity, we assume that each item is explicitly associated with at most one leaf category in $\mathcal{C}$. Table 3.1 summarizes all the notations throughout this approach.

Our method is built on one of the most successful latent factor model (LFM) – matrix factorization (MF) [75], which assumes the existence of latent structures in the user-item interaction matrix. By uncovering latent factors of users and items, it approximates the observed ratings and estimates the unobserved ratings. MF solves an optimization problem shown as follows:

$$\min_{\mathbf{U},\mathbf{V}} \frac{1}{2}\sum_{i,j}\mathbf{O}_{ij}(\mathbf{R}_{ij}-\mathbf{U}_i\mathbf{V}_j^T)^2+\frac{\lambda}{2}(\|\mathbf{U}\|_F^2+\|\mathbf{V}\|_F^2), \qquad \text{(Eq. 3.1)}$$



3.1.a: Category hierarchy

	v_1	v_2	v_3	v_4	v_5
v_1	-	1	$g_5+s_5g_4$	g_5	g_5
v_2	1	-	$g_5+s_5g_4$	g_5	g_5
v_3	$g_5+s_5g_4$	$g_5+s_5g_4$	-	g_5	g_5
v_4	g_5	g_5	g_5	-	1
v_5	g_5	g_5	g_5	1	-

3.1.b: Regularizer coefficients

Figure 3.1: Category hierarchy and its corresponding regularization coefficients. (a) illustrates a category hierarchy, where categories with children (i.e., C_5, C_4) are called *internal categories*. Particularly, C_5 is also named *root category*, whereas categories without children are called *leaf categories*. Dash and solid lines respectively represent the item-category (i.e., an item belongs to a category) and category-category (i.e., parent-child) relationships. Categories in a red dash box comprises a category unit. (b) shows the corresponding regularization coefficients of the corresponding example.

where $\mathbf{U} \in \mathbb{R}^{m \times d}$ and $\mathbf{V} \in \mathbb{R}^{n \times d}$ are the latent factors of users and items, respectively. d is the dimension of latent factors. λ is the regularization coefficient to avoid over-fitting. The unobserved rating for user u_i to item v_j can be estimated by the inner product of the corresponding user and item latent factors, i.e., $\hat{\mathbf{R}}_{ij} = \mathbf{U}_i^T \mathbf{V}_j$.

3.1.2 Modeling Influence of Category Hierarchy

Step by step, we model the influence from a single category to the combinations of categories and finally the entire category hierarchy.

Influence of an Isolated Category. To start, we first define the regularization by an isolated category C_p in the hierarchy as:

$$Dis(C_p) = \sum_{v_j, v_k \in C_p, j<k} \|\mathbf{V}_j - \mathbf{V}_k\|_F^2, \qquad \text{(Eq. 3.2)}$$

where $\|\mathbf{V}_j - \mathbf{V}_k\|_F^2$ is the squared Frobenius norm distance between the latent factors of v_j and v_k belong to category C_p: C_p poses regularization on the cumulation of the pairwise

distance between items belong to it. Thus, $Dis(C_p)$ can be considered as the influence of the isolated category C_p on user-item interactions by regularizing the distance of item latent factors. $Dis(C_p)$ indicates that if two items are characterized by a same category, then the distance of the corresponding item latent factors should be constrained. Theoretically, the distance should be smaller than that of two items which belong to different categories. The definition here only considers the influence of an *isolated* category, while the co-influence of the category hierarchy contributed by the category, i.e., influence of the category *in the hierarchy*, is different from – but based on – the influence of the *isolated* category, which will be illustrated later.

Note that our method models category influence by regularizing item latent factors, and can be straightforwardly transferred to modeling the influence by regularizing user latent factors, or both of them.

Influence of an Isolated Category Unit. Given the above definition, we now model the influence of an isolated combination of categories, on learning item latent factors, by introducing the most important relation among categories in a hierarchy, i.e., *parent-child* (*affiliatedTo*) relation, based on which other relations among categories in the hierarchy such as *siblings*, *ancestors* can be derived. We first define the *category unit*, i.e., $Cu(C_p)$, as the combination of a single parent node C_p and its child nodes, namely:

$$Cu(C_p) = \{C_p\} \cup \{C_c | \forall C_c \in children(C_p)\}. \quad \text{(Eq. 3.3)}$$

Two examples of category units $Cu(C_5)$ and $Cu(C_4)$ are illustrated in the red dash boxes in Figure 3.1(a).

Then we consider the influence of an isolated category unit on learning item latent factors by regularization. For each isolated category unit $Cu(C_p)$, we denote its influence as $\mathbf{I}'(C_p)$, and assign it two parameters g_p, s_p, with the constraint $g_p + s_p = 1$. Parameters g_p and s_p are used to distribute the influence of the category unit to two parts. One is

given by the parent node, weighted by g_p, and the other is given by the child nodes, weighted by s_p. The influence of the isolated category unit, i.e., $\mathbf{I}'(C_p)$, is defined as:

$$\mathbf{I}'(C_p) = g_p Dis(C_p) + s_p(\sum_{\forall C_c \in children(C_p)} Dis(C_c)). \quad \text{(Eq. 3.4)}$$

For example, the influence of the isolated category unit $Cu(C_5)$ in Figure 3.1, i.e., $\mathbf{I}'(C_5)$, is determined by both the influence of the parent node C_5, i.e., $Dis(C_5)$, weighted by g_5, and the influence of its child nodes, i.e., $Dis(C_3)$ and $Dis(C_4)$, weighted by s_5. The overall influence of this isolated category unit is: $\mathbf{I}'(C_5) = g_5 Dis(C_5) + s_5(Dis(C_3) + Dis(C_4))$. Compared with the influence of the isolated category C_5, the influence of category C_5 in $Cu(C_5)$ is different, in that $Dis(C_5)$ is weighted by g_5.

Influence of an Entire Category Hierarchy. Based on the definition of the influence of an *isolated* category unit, we now proceed to model the influence of category unit *in the hierarchy*, thus to formally derive the overall influence of an entire category hierarchy on item latent factors. Note that the influence of a category unit *in the hierarchy* is different from – but based on – the influence of the *isolated* category unit, and can be achieved by recursively defining the regularization of the category unit *in the hierarchy*, which is given by the following formula:

Definition 1 (**Recursive Regularization**)

$$\mathbf{I}(C_p) = \begin{cases} g_p Dis(C_p) & +s_p(\sum_{\forall C_c \in children(C_p)} \mathbf{I}(C_c)), \\ & \textit{if } C_p \textit{ is an internal category}; \\ Dis(C_p), & \textit{if } C_p \textit{ is a leaf category and } |C_p| > 1; \\ 0, & \textit{otherwise}, \end{cases} \quad \text{(Eq. 3.5)}$$

where $|C_p|$ *is the number of items that belong to category* C_p.

From the above definition, we can see the difference between the influence of a category unit *in the hierarchy* $\mathbf{I}(C_p)$ and the influence of an *isolated* category unit $\mathbf{I}'(C_p)$, that is,

Algorithm 1: Recursive Regularization Deduction

Input: category hierarchy $\mathcal{C}$, $g_p, s_p \forall C_p \in \mathcal{C}$

```
foreach C_p ∈ C do
    I(C_p) ← 0;
layer ← #layers of C ;
for l = 0; l ≤ layer; l++ do
    foreach category C_p at layer l of C do
        if C_p is a leaf category (l = 0) and |C_p| > 1 then
            I(C_p) ← Dis(C_p);
        else if C_p is an internal category (l ≠ 0) then
            I(C_p) ← g_p Dis(C_p) + s_p(Σ_{∀C_c ∈ children(C_p)} I(C_c));
I(C) ← I(C_root);
```

$\mathbf{I}(C_p)$ is recursively defined on $\mathbf{I}(C_c)$. Put another way, the influence of a child category is included in the influence of its parent category. Hence, the influence of an entire category hierarchy, denoted by $\mathbf{I}(\mathcal{C})$, is equivalent to that of the root category, as it recursively includes the influence of all categories in the hierarchy. As an example, Eq. 3.6 shows the influence of the category hierarchy in Figure 3.1, given by,

$$\begin{aligned}
&\mathbf{I}(\mathcal{C}) = \mathbf{I}(C_5) \\
=&g_5 Dis(C_5) + s_5(\mathbf{I}(C_4) + \mathbf{I}(C_3)) \\
=&g_5 Dis(C_5) + s_5(g_4 Dis(C_4) + s_4(\mathbf{I}(C_1) + \mathbf{I}(C_2)) + Dis(C_3)) \\
=&g_5 Dis(C_5) + s_5(g_4 Dis(C_4) + s_4 Dis(C_1) + Dis(C_3)).
\end{aligned} \tag{Eq. 3.6}$$

The deduction of recursive regularization of a category hierarchy is shown in Algorithm 1, where the co-influence of categories is modeled as a regularization function parameterized by the weights of each category in the hierarchy. These weights characterize the influence of distinct categories, and can be further learnt from historical user-item interaction data, as we introduce in the next section.

Remark. By recursively weighting and combining category influence over a hierarchy from the root category to the leaves, recursive regularization can model the influence of an arbitrarily deep category hierarchy that can be either balanced or imbalanced.

3.2 The ReMF Framework

In this section, we introduce the novel recommendation framework ReMF, that integrates the recursive regularization into the MF model to exploit category hierarchy. Meanwhile, we present the optimization method and complexity analysis for ReMF.

3.2.1 Integration of Recursive Regularization

By incorporating recursive regularization into MF, the ReMF framework is given by:

Definition 2 (**The ReMF Framework**)

$$\min_{\substack{\mathbf{U},\mathbf{V},\\ g_p,s_p \forall C_p \in \mathcal{C}}} \mathcal{J} = \frac{1}{2}\sum_{i,j} \mathbf{O}_{ij}(\mathbf{R}_{ij} - \mathbf{U}_i \mathbf{V}_j^T)^2 + \frac{\alpha}{2}\mathbf{I}(\mathcal{C}) + \frac{\lambda}{2}(\|\mathbf{U}\|_F^2 + \|\mathbf{V}\|_F^2) \qquad \text{(Eq. 3.7)}$$

where $\mathbf{I}(\mathcal{C})$ *is the recursive regularization term to constrain the distance of item latent factors by modeling the co-influence of categories in the hierarchy traversing from leaf to root category;* α *is a regularization parameter that controls the importance of the recursive regularization, i.e.,* $\mathbf{I}(\mathcal{C})$

Thanks to recursive regularization, ReMF can model the co-influence of categories in the hierarchy to learn item latent factors. It also characterizes the distinct influence of each category, thus helping with the interpretation of the effect of each category in the hierarchy on recommendation, illustrated as follows. Considering the example of Figure 3.1, based on Eq. 3.2 and Eq. 3.6, the category hierarchy influence $\mathbf{I}(\mathcal{C})$ can be rewritten as the following formula:

$$(g_5 + s_5 g_4 + s_5 s_4)\|\mathbf{V}_1 - \mathbf{V}_2\|_F^2 + (g_5 + s_5 g_4)\|\mathbf{V}_1 - \mathbf{V}_3\|_F^2 + \ldots, \qquad \text{(Eq. 3.8)}$$

where the strength of the regularization between v_1, v_2's latent factors is $(g_5 + s_5 g_4 + s_5 s_4)$, and that of v_1, v_3's latent factors is $(g_5 + s_5 g_4)$. In fact, the strength of regularization is the combination of influence of different categories. For simplicity, we assume $g = s = 0.5$ for

each internal category. Therefore, the strength of regularization between v_1, v_2's latent factors is $(g_5 + s_5 g_4 + s_5 s_4) = 1$, from which we could see that the category C_5 has an influence of $g_5 = 0.5$, while its children categories C_4 and C_1 have influence of $s_5 g_4 = 0.25$ and $s_5 s4 = 0.25$, respectively. Then, for v_1, v_3, the strength of regularization between their latent factors is $(g_5 + s_5 g_4) = 0.75$, where the categories C_5, C_4 have influence of $g_5 = 0.5, s_5 g_4 = 0.25$, respectively. The distinct influence of categories on learning item latent factors can therefore be characterized by certain functions of the weights (g, s).

To formally derive category influence on an arbitrary pair of users, we define the *regularization coefficient* $\mathbf{C}_{jk}$ to represent the strength of regularization between v_j and v_k, where a greater value of $\mathbf{C}_{jk}$ indicates a higher correlation between the two items. Hence, $\mathbf{I}(\mathcal{C})$ can be reformulated as:

$$\mathbf{I}(\mathcal{C}) = \sum_{v_j, v_k \in \mathcal{I}, j<k} \mathbf{C}_{jk} \|\mathbf{V}_j - \mathbf{V}_k\|_F^2, \qquad \text{(Eq. 3.9)}$$

We next introduce two theorems for deriving $\mathbf{C}_{jk}$, which is the combination of the influence by different categories on v_j and v_k.

Proposition 1 *The regularization coefficient for any pair of items v_j, v_k (i.e., $\mathbf{C}_{jk}$) belong to the same leaf category is 1:*

$$g_{root} + s_{root}(g_{c_1} + s_{c_1}(g_{c_2} + s_{c_2}(...(g_{c_l} + s_{c_l})))) = 1,$$

where the list $\{C_{root}, C_{c_1}, C_{c_2}, \ldots, C_{c_l}\}$ is the set of the common categories of v_j and v_k, ordered in a sequence from the root category C_{root} to the leaf category C_{c_l}.

Proof. This is straightforward to prove, due to the constraint $g + s = 1$. Considering the example $\{v_1, v_2\}$ in Figure 3.1, the sum of the relevant regularization terms, i.e., $g_5 Dis(C_5), s_5 g_4 Dis(C_4)$ and $s_5 s_4 Dis(C_1)$, in Eq. 3.6 is:

$$\begin{aligned} &(g_5 + s_5(g_4 + s_4))\|\mathbf{V}_1 - \mathbf{V}_2\|_F^2 \\ =&(g_5 + s_5)\|\mathbf{V}_1 - \mathbf{V}_2\|_F^2 = \|\mathbf{V}_1 - \mathbf{V}_2\|_F^2. \end{aligned}$$

□

Proposition 2 *For any pair of items v_j, u_k that do not belong to a common leaf category, the regularization coefficient (i.e., $\mathbf{C}_{jk}$) is:*

$$g_{root} + s_{root}(g_{c_1} + s_{c_1}(g_{c_2} + s_{c_2}(...(g_{c_l})))),$$

where the list $\{C_{root}, C_{c_1}, C_{c_2}, \ldots, C_{c_l}\}$ is the set of the common categories of v_j and v_k, ordered from the root feature C_{root} to the deepest common category C_{c_l}.

Proof. All possible categories that can influence the regularization coefficient of v_j, v_k are their deepest common category, parents and ancestors of the deepest common category. □

According to the above theorems, the value of regularization coefficient always falls into the range of $[0, 1]$, with 1 indicating the full regularization and 0 indicating no regularization. As an example, Figure 3.1(b) shows the regularization coefficients of the category hierarchy in Figure 3.1(a).

These regularization coefficients naturally connect ReMF to network based recommendation methods, which consider pair-wise regularization on users. There are however two essential differences: 1) network-based regularization coefficients are usually hard-coded, while our regularization coefficients are modeled from the category hierarchy structure, and expressed by the function of weights (g, s). And, 2) (g, s), which parametrizes the distinct category influence, is automatically learnt from the historical user-item interaction data, as we will address later.

3.2.2 Optimization and Complexity Analysis

Model Learning. We adopt the SGD scheme [52, 75] to optimize our objective function. The gradients of $\mathbf{U}_i, \mathbf{V}_j$ are given by:

$$\begin{aligned}
\frac{\partial \mathcal{J}}{\partial \mathbf{U}_i} &= -\sum_j \mathbf{O}_{ij}(\mathbf{R}_{ij} - \mathbf{U}_i\mathbf{V}_j^T)\mathbf{V}_j + \lambda \mathbf{U}_i, \\
\frac{\partial \mathcal{J}}{\partial \mathbf{V}_j} &= -\sum_i \mathbf{O}_{ij}(\mathbf{R}_{ij} - \mathbf{U}_i\mathbf{V}_j^T)\mathbf{U}_i + \lambda \mathbf{V}_j + \alpha \sum_{v_j, v_k \in \mathcal{I}, j<k} \mathbf{C}_{jk}(\mathbf{V}_j - \mathbf{V}_k),
\end{aligned} \tag{Eq. 3.10}$$

Algorithm 2: ReMF Model Learning

Input: rating matrix $\mathbf{R}$, category hierarchy $\mathcal{C}$, $d, \gamma, \lambda, \alpha, iter$

Initialize $\mathbf{U}, \mathbf{V}, g_p, s_p$, and $\forall C_p \in \mathcal{C}$;
for $t = 1; t \leq iter; t++$ **do**
  foreach $\mathbf{U}_i \in \mathbf{U}$, $\mathbf{V}_j \in \mathbf{V}$ **do**
    $\mathbf{U}_i^{(t)} \leftarrow \mathbf{U}_i^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial \mathbf{U}_i}$;
    $\mathbf{V}_j^{(t)} \leftarrow \mathbf{V}_j^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial \mathbf{V}_j}$;
  foreach *Internal category in the hierarchy* **do**
    $g_p^{(t)} \leftarrow g_p^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial g_p}$;
    $s_p^{(t)} \leftarrow s_p^{(t-1)} - \gamma \frac{\partial \mathcal{J}}{\partial s_p}$;
  Calculate $\mathcal{J}$ by Algorithm 1 and Definition 2;
  if $\mathcal{J}$ *has converged* **then**
    break;

In terms of (g, s), it can be predefined by domain experts who can fairly quantify the influence of different categories. Instead, we provide an effective data-driven solution that automatically learns (g, s) based on the historical user-item interaction data.

We only need to estimate (g, s) for internal categories in the hierarchy, since the leaf categories do not have children. For an internal category C_p, the gradients of g_p, s_p are equivalent to the multipliers of g_p, s_p in $\mathbf{I}(\mathcal{C})$. Thus, we have:

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial g_p} &= \begin{cases} Dis(C_p), & \text{if } C_p \text{ is root,} \\ \prod_{\forall a: C_a \in ancestors(C_p)} s_a Dis(C_p), & \text{otherwise;} \end{cases} \\ \frac{\partial \mathcal{J}}{\partial s_p} &= \begin{cases} \sum_{\forall C_c \in children(C_p)} \mathbf{I}(C_c), & \text{if } C_p \text{ is root,} \\ \prod_{\forall a: C_a \in ancestors(C_p)} s_a (\sum_{\forall C_c \in children(C_p)} \mathbf{I}(C_c)), & otherwise. \end{cases} \end{aligned} \tag{Eq. 3.11}$$

According to the constraint $g_p + s_p = 1$, we can update g_p (or s_p) using the gradient and the other by $s_p = 1 - g_p$ (or $g_p = 1 - s_p$). The detailed parameter learning process is illustrated in Algorithm 2.

Complexity Analysis. The computational time is mainly taken by evaluating the objective function $\mathcal{J}$ and updating the related variables. The time to compute the $\mathcal{J}$ is $\mathcal{O}(d|\mathbf{R}| + dm^2)$, where $|\mathbf{R}|$ is the number of non-zero observations in the rating matrix $\mathbf{R}$. For all gradients $\frac{\partial \mathcal{J}}{\partial \mathbf{U}_i}, \frac{\partial \mathcal{J}}{\partial \mathbf{V}_j}, \frac{\partial \mathcal{J}}{\partial g_p}, \frac{\partial \mathcal{J}}{\partial s_p}$, the computational time are $\mathcal{O}(d|\mathbf{R}|)$, $\mathcal{O}(d|\mathbf{R}| +$

dn^2), $\mathcal{O}\left(d\sum_{l=0}^{layer-1}\frac{\overline{n}_l(\overline{n}_l-1)n_l}{2}\right)$ and $\mathcal{O}(|s_p|)$, respectively. Wherein $\overline{n}_l$ denotes the average number of items in each node at layer l, n_l denotes the number of nodes at layer l, and $|s_p|$ ($\ll |\mathbf{R}|$) denotes the number of internal nodes. Particularly, we leverage $s_p = (1 - g_p)$ to update s_p. The overall computational complexity of Algorithm 1 is ($\#iteration * \mathcal{O}(d|\mathbf{R}| + dq)$), where $q = max(\sum_{l=0}^{layer-1}\frac{\overline{n}_l(\overline{n}_l-1)n_l}{2}, n^2)$. In real-world applications $\overline{n}_l$ is typically small (e.g., power-law distributed), thus making ReMF scalable to large data sets.

3.3 Empirical Evaluation

In this section, we conduct comprehensive experiments on multiple real-world datasets to evaluate the performance of our proposed models by comparing with a number of state-of-the-art algorithms.

3.3.1 Experimental Setup

Datasets. We collect data of Foursquare check-in's performed over 3 weeks in 4 European capital cities (Amsterdam, London, Paris, Rome) and published on 2 social media platforms (Twitter, Instagram). Table 3.2 shows the statistics about the 8 datasets. We consider users' residence *city*, *country* and *continent* as category information about users[1], as well as a root category *residence location*. We use the method described in [9] to locate users' residence locations. For conciseness, we only analyze the co-influence of *country* and *city*. Overall we consider 121 countries and 2,873 cities. These datasets also contain 3-level category hierarchies to describe the check-in locations. Note that, the category hierarchies of these datasets are all balanced, which means all the users or items are characterized by categories with the same number of levels. To further demonstrate the generalizability of our ReMF, we also test on dataset from Amazon web store [68]. The details about Amazon dataset will be shown in the results and analysis part.

[1]As we mention in Section 3.1.2, our proposed method can be straightforwardly transferred to modeling the influence by regularizing user latent factor. We therefore test the effectiveness of our model on both user and item side.

Table 3.2: Statistics of the datasets for ReMF

		Amsterdam	**Rome**	**Paris**	**London**
Instagram	#Users	4,318	4,081	11,345	12,719
	#POIs	5,768	7,878	14,849	12,892
	#Check-in's	28,142	26,714	80,553	66,092
	Sparsity	99.89%	99.92%	99.95%	99.96%
Twitter	#Users	1,599	1,369	6,521	9,305
	#POIs	3,816	4,876	16,046	14,117
	#Check-in's	8,670	8,727	43,541	48,852
	Sparsity	99.86%	99.87%	99.96%	99.96%

Evaluation. We adopt the standard 5-fold cross validation, and the following 3 metrics for evaluation: MAE and RMSE to measure the error of predicted ratings; and Area Under the ROC Curve (AUC) [35, 131] to measure the quality of predicted ranking of items (ranked according to the predicted ratings). The smaller MAE and RMSE, and the larger AUC, the better the recommendation performance.

Comparison Methods. The following methods are compared: (1) **MF** [75]: matrix factorization method; (2) **CMF** [94]: collective matrix factorization; (3) **TaxMF** [51]: taxonomy-based matrix factorization; (4) **SoReg** [65]: network-based recommendation method incorporating social relations; (5) **FM** [82]: factorization machine; (6) **HieFM**: factorization machine with category hierarchy information.

HieFM is a variation of FM that considers each category path in the hierarchy (from root to leaf nodes) as an additional category in the design vectors of FM. Similar to FM, CMF and TaxMF can also incorporate path-based categories. As FM outperforms CMF and TaxMF [100], we limit our comparison with previous methods exploiting path-based categories to HieFM.

Parameter Settings. We empirically set optimal parameters for each method using a grid search in $\{0.0001, 0.001, 0.01, 0.05\}$ for both λ (including 1-way and 2-way regularization of FM) and the learning rate γ; $\alpha = 0.5$ for CMF; $\beta = 0.01$ for SoReg. For

fair comparison, we set $d = 10$ (the dimension of latent factors) for all the methods, and adopt all categories (i.e., continent, country, and city) as input for TaxMF, CMF, FM and HieFM. HieFM has path-based categories as additional hierarchy information. In SoReg, we model the social relations among users by counting the number of common categories, under the assumption that the commonality establishes implicit social relationships based on the geo-social correlation phenomenon [24]. Without loss of generality, we adopt $f(x) = 1/(1 + x^{-1})$ to map each #check-in $\mathbf{R}_{ij} \in \mathbf{R}$ in POI datasets into the interval $(0, 1)$ [23].

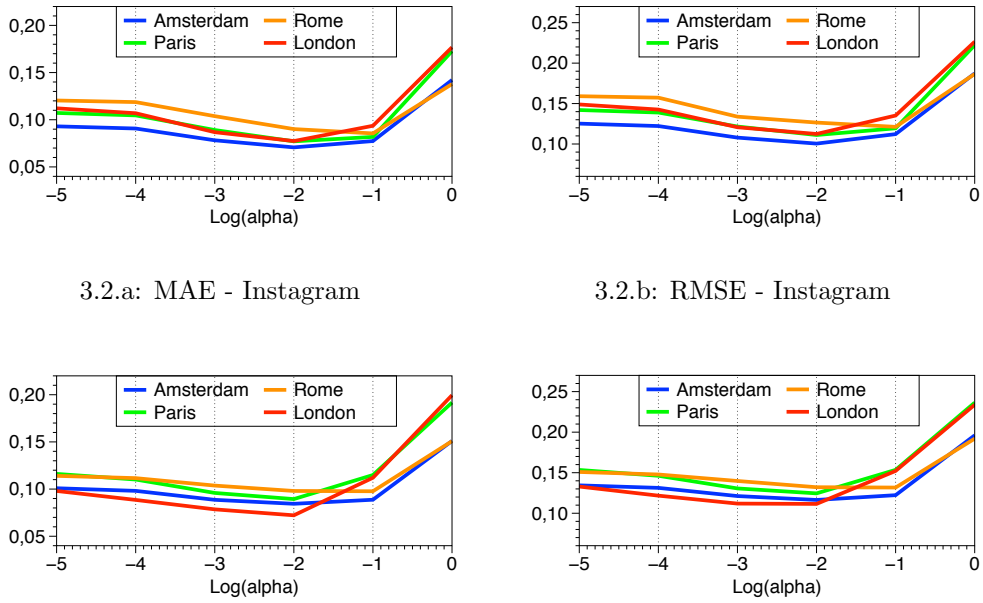
3.3.2 Results and Analysis

We analyze the influence of recursive regularization on ReMF performance, and discuss how the weighting parameters g, s can help the interpretation of recommendation results.

The Impact of α. In ReMF, α controls the strength of recursive regularization of category hierarchy. We apply a grid search in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ to investigate the impact of α on recommendation performance. Results are shown in Figure 3.2. As α varies from small to large, the performance first increases then decreases, with the maximum reached at the range $[10^{-2}, 10^{-1}]$. The performance variations across different datasets suggest the need for data set-specific settings; the similarity in performance variation across α values shows the robustness of ReMF.

Table 3.3: Values of g for continents and top/bottom countries

Continents		**Top countries**		**Bottom countries**	
Name	g	Name	g	Name	g
Europe	0.1837	Portugal	0.6915	Chile	0.0211
America	0.1656	Monaco	0.5813	Thailand	0.0175
Asia	0.1534	Serbia	0.5130	Spain	0.0100
Africa	0.0375	Poland	0.4453	Indonesia	0.0081
Oceania	0.0139	Hungary	0.4141	Belgium	0.0064



3.2.a: MAE - Instagram

3.2.b: RMSE - Instagram

3.2.c: MAE - Twitter

3.2.d: RMSE - Twitter

Figure 3.2: The effects of α on the performance of ReMF

Interpretation from (g, s). We examine (g, s) for the internal categories, i.e., continents and countries, learnt from data. Table 3.3 shows the list of continents and countries ranked according to their g values. Recall that for a continent (country), $g > s$ means that the continent (country) has a stronger effect on user preferences than and its children categories, i.e., countries (cities).

In general the continents have relatively smaller effects on user preferences (with g values all below 0.2), suggesting that continents have weaker effects than their countries. In addition, we observe a big variance in the g values of countries, indicating that different countries have different influence on user preferences. The high variance of countries' g values proves the necessity of parameterizing g, s in recommendation. We then compare the influence of countries and cities on their residents' preferences. As cities of a country and the country comprise a category unit, the influence of a city can be measured by $s = 1 - g$, where g is the influence of the country. We can see from Table 3.3 that most

countries have $g < 0.5$ (only 3 countries have $g > 0.5$), i.e., $s > 0.5$, indicating that the influence of cities in most countries have more influence on their residents' preferences than the countries themselves.

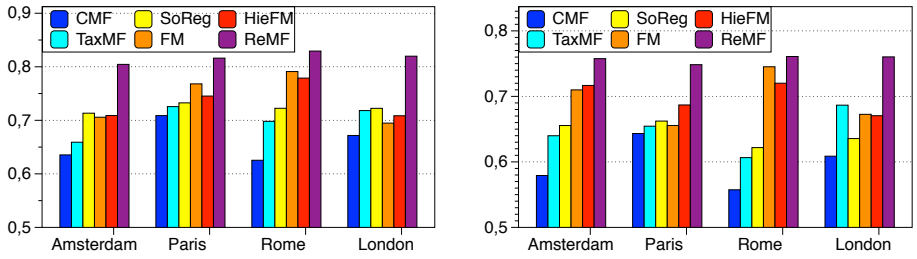
Rating Performance. Two views are created for each dataset during the test process: 1) the "All" view includes all users; while 2) the "Cold start" view indicates that only users with ≤ 5 ratings are involved in the test set.

Table 3.4 compares the performance of the considered recommendation methods for all datasets. Unsurprisingly, the basic matrix factorization model is consistently outperformed by category-aware recommendation methods; this shows that, in the context of the targeted evaluation scenario, the usage of category information about users positively affects recommendation accuracy. In addition, FM outperforms CMF, TaxMF and SoReg. This could be explained by FM considering item-category interactions, in addition to user-item and user-category interactions. HieMF in general outperforms FM, suggesting that information about category relations (paths) can help predict user preferences. ReMF consistently outperforms the methods in the comparison pool, with an average performance gain (w.r.t. the second best method) of **7.20%** (MAE) and **15.07%** (RMSE). Paired t-test shows that the improvements of ReMF on all datasets are significant (p-value < 0.01). Such big improvements clearly show the effectiveness of recursive regularization, and the advantage derived from the full inclusion of information about category relations.

Table 3.4 (data view "Cold start") reports the experimental results of all the comparisons with cold start users. As in the previous case, ReMF achieves the best performance compared with other methods, and significantly outperforms the second best methods in all datasets (p-value < 0.01) by **12.02%** and **17.53%** w.r.t. MAE and RMSE respectively. The relatively larger improvements on the testing view "Cold start" than on

Table 3.4: Performance of all the comparison methods. The best performance for each city is boldfaced; the runner up is labelled with '*'. The improvements by the best method on all datasets are statistically significant (p-value < 0.01).

View	Data		MAE							RMSE						
			MF	**CMF**	**TaxMF**	**SoReg**	**FM**	**HieFM**	**ReMF**	**MF**	**CMF**	**TaxMF**	**SoReg**	**FM**	**HieFM**	**ReMF**
All	Instagram	Ams.	0.196	0.156	0.143	0.104	0.088	0.082*	**0.070**	0.313	0.194	0.193	0.146	0.137	0.135*	**0.100**
		Par.	0.154	0.155	0.142	0.121	0.079*	0.083	**0.077**	0.268	0.192	0.185	0.183	0.129	0.118*	**0.111**
		Rom.	0.255	0.158	0.147	0.136	0.091	0.089*	**0.086**	0.386	0.197	0.186	0.191	0.140	0.139*	**0.121**
		Lon.	0.180	0.156	0.137	0.125	0.083*	0.085	**0.077**	0.296	0.193	0.176	0.184	0.135*	0.140	**0.112**
	Twitter	Ams.	0.226	0.161	0.135	0.123	0.099	0.094*	**0.084**	0.347	0.200	0.172	0.167	0.154	0.145*	**0.116**
		Par.	0.201	0.171	0.155	0.127	0.096	0.094*	**0.089**	0.321	0.214	0.204	0.169	0.141	0.139*	**0.125**
		Rom.	0.268	0.171	0.159	0.135	0.102	0.100*	**0.098**	0.390	0.213	0.203	0.183	0.153	0.147*	**0.131**
		Lon.	0.218	0.166	0.155	0.112	0.093	0.090*	**0.077**	0.308	0.207	0.196	0.154	0.141	0.138*	**0.112**
Cold start	Instagram	Ams.	0.294	0.155	0.146	0.105	0.092	0.089*	**0.071**	0.388	0.193	0.190	0.148	0.144	0.139*	**0.104**
		Par.	0.194	0.154	0.148	0.117	0.085	0.085*	**0.080**	0.311	0.191	0.190	0.171	0.137	0.129*	**0.118**
		Rom.	0.384	0.161	0.152	0.136	0.095	0.094*	**0.081**	0.487	0.199	0.193	0.185	0.146*	0.151	**0.125**
		Lon.	0.303	0.154	0.142	0.122	0.089*	0.090	**0.079**	0.398	0.192	0.182	0.169	0.143	0.143*	**0.116**
	Twitter	Ams.	0.326	0.160	0.143	0.119	0.101	0.097*	**0.085**	0.400	0.198	0.183	0.161	0.156	0.151*	**0.117**
		Par.	0.244	0.171	0.164	0.127	0.101	0.095*	**0.087**	0.376	0.212	0.212	0.171	0.149*	0.150	**0.122**
		Rom.	0.392	0.172	0.168	0.134	0.107	0.107*	**0.099**	0.495	0.213	0.214	0.183	0.156	0.152*	**0.136**
		Lon.	0.330	0.164	0.159	0.113	0.097	0.092*	**0.076**	0.398	0.204	0.201	0.156	0.148	0.144*	**0.109**



3.3.a: AUC – Instagram

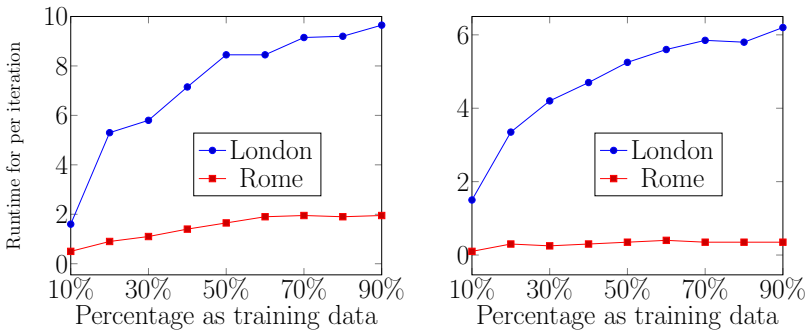
3.3.b: AUC – Twitter

Figure 3.3: AUC of ReMF and the comparative methods

"All" indicates that ReMF has higher capability in coping with the cold start problem compared to the state-of-the-art methods.

Ranking Performance. We further evaluate the ranking quality of items recommended by ReMF and other methods in the comparison pool. Results are shown in Figures 3.3(a-b) for datasets from Instagram and Twitter, respectively. ReMF significantly outperforms the second best method (p-value < 0.01) on all datasets by **9.86%** on average, reaching an averaged AUC of 0.8175 in Instagram and 0.7568 in Twitter. These observations show that the influence of category hierarchy modeled by recursive regularization can effectively complement user-item interaction data in ranking prediction.

Generalizability. We test the performance of ReMF on another task, i.e., product recommendation, using the data from Amazon web store [68]. Different from the POI datasets, here we consider the category hierarchy of items. We focus on the product category of "Clothing, Shoes & Jewelry", having maximal depth of 7, and an unbalanced category hierarchy. An example path in the hierarchy from the root category to the leaf is "Clothing, Shoes & Jewelry → Men → Accessories → Wallets". We uniformly sample the raw data set to include $100,810$ ratings performed by $34,817$ users to $45,716$ items. Table 3.5 compares the performance of ReMF and the other methods in the comparison pool, measured by RMSE, which is more indicative of large errors than MAE. As in



3.4.a: Instagram 3.4.b: Twitter

Figure 3.4: Runtime (seconds/iteration) of ReMF on Instagram and Twitter.

the previous setting, ReMF (in boldfaced) significantly outperforms (p-value < 0.01) the second best method (marked with '*'), i.e., HieFM, by **5.46%** on the testing view of "All" and **7.42%** on "Cold start". These results show that ReMF can be effective in multiple recommendation tasks, and with different topologies of category hierarchy.

Table 3.5: Performance of ReMF on Amazon Dataset

	CMF	**TaxMF**	**SoReg**	**FM**	**HieFM**	**ReMF**
All	1.6356	1.3921	1.3912	1.3899	1.3847*	**1.3091**
Cold start	1.6386	1.4057	1.4054	1.4074	1.4033*	**1.3242**

Complexity Validation. To verify the conclusion made in Complexity Analysis part that the overall computational time is linear with respect to the number of observations in the rating matrix ($|\mathbf{R}|$), we test the runtime of ReMF model and the results are shown in Figure 3.4. It depicts the relationships between the runtime and the size of training data. We randomly select $x\%$ as training data and the rest $(1 - x\%)$ as test data where x is scaled from 10 to 90 with step 10. We observe that with the increase of the training data, the runtime linearly goes up. Note that we only show the results of London and Rome, which possess the largest and smallest data among the four capital cities. Similar observations can be noted for all the other cities. In conclusion, the ReMF model is efficient to scale to very large data sets.

3.4 Summary

Category hierarchies are a common way to capture relations between categories. Yet, the value of this additional information is not fully exploited by state-of-the-art category-aware recommendation methods. This chapter proposes a novel regularization method named recursive regularization for modeling the co-influence of categories in the hierarchy on user-item interactions. Based on this, a new recommendation framework ReMF is proposed to learn hierarchical category influence from historical user-item interaction data. Experimental validation on multiple real-word datasets shows that ReMF can largely outperform state-of-the-art methods, proving the value residing in the exploitation of category hierarchy for better learning user and item latent factors.

Chapter 4

HieVH: Leveraging both Vertical and Horizontal Dimensions of CH

In Chapter 3, we propose ReMF that integrates recursive regularization into matrix factorization to enhance recommendation performance. It mainly focuses on investigating the influence of vertically affiliated categories (i.e., child-parent) in the CH on user-item interactions. The relations of horizontally organized categories (i.e., siblings and cousins) in the hierarchy, however, have only been little investigated. We show in real-world datasets that category relations in horizontal dimension of CH can help explain and further model user-item interactions.

Hence, this chapter contributes a unified recommendation framework HieVH [101] that seamlessly exploits both vertical and horizontal dimensions of CH, to boost recommendation accuracy. To model the vertial dimension, HieVH adapts latent factors of items by adding weighted aggregation of their affiliated categories' latent factors, to better model item latent factors. The weights are automatically learnt from data. Horizontally, category relations are incorporated as regularizers at each layer of the hierarchy, to better model category latent factors. In so doing, through the adaption of item latent factors with category latent vectors in vertical dimension, category relations in horizontal dimension can be inherited by items. Extensive validation demonstrates the superiority

of HieVH against the state-of-the-art. An additional benefit of HieVH is to provide better interpretations of the generated recommendations.

The rest of this chapter is organized as follows. In Section 4.1, we first introduce our metrics for measuring category influence in vertical dimension, and category relations in horizontal dimension of CH. We then apply the proposed metrics to analyze the Amazon Web store data to illustrate the presence of category influence and relations in real-world datasets; Section 4.2 elaborates details of the proposed HieVH framework to enhance the recommendation performance, followed by the experimental results in Section 4.3 and conclusion in Section 4.4.

4.1 Measuring Category Influence and Relations

This section first introduces our metrics for measuring category influence in vertical dimension, and category relations in horizontal dimension of CH. To demonstrate the need for richer CH characterization of user-item interactions for better recommendation, we then apply the proposed metrics to analyze Amazon Web store data.

4.1.1 The Proposed Metrics

Let $\mathcal{U}, \mathcal{I}$ denote the set of users and items, and $\mathcal{C}$ denote the set of categories organized in a hierarchy. r_{ui} is the rating given by user $u \in \mathcal{U}$ to item $i \in \mathcal{I}$. Each item $i \in \mathcal{I}$ is affiliated with a subset of categories $\mathcal{C}(i) = \{f_i^1, f_i^2, \ldots, f_i^L\}$, organized as a path from leaf category f_i^1 to root category f_i^L. Table 4.1 summarizes all the notations utilized in this chapter. Let $P(e_i)$ denote the probability of the event that an item i is rated by a user, defined as,

$$P(e_i) = \frac{|\{u|u \in \mathcal{U}, r_{ui} \neq 0\}|}{|\mathcal{U}|} \quad \text{(Eq. 4.1)}$$

Based on this, we use *item co-occurrence* IC to measure the closeness of two items.

Table 4.1: Mathematical Notations for HieVH

Notation	Description
$\mathcal{U}, \mathcal{I}, \mathcal{C}$	user, item and category set
$\mathcal{C}(i) = \{f_i^1, \cdots, f_i^L\}$	category set for item i in the CH
$P(e_i)$	the probability of an item i rated by a user
$\mathrm{IC}(i, j)$	item co-occurrence
$\overrightarrow{\mathrm{CI}}(f^l)$	the vertical influence of category f^l on items
$\mathrm{CR}(f, g)$	category relation in horizontal dimension
o_{ui}	indicator function
$\theta_u, \theta_i, \theta_f$	user, item and category latent factors
r_{ui}	rating given by user u to item i
$\Phi(\cdot)$	vertical item latent factor adaptive function
$\Psi(\cdot)$	horizontal category regularization function
$\vartheta_{\mathcal{C}(i)}$	influence of categories in $\mathcal{C}(i)$ on item i
α	importance of horizontal category influence
$\Omega(\Theta)$	regularization term to avoid over-fitting
$\mathcal{J}$	objective function of HieVH framework
σ_{ij}, σ_{fg}	item, category relations

Definition 3 (**Item Co-occurrence**)

$$\mathrm{IC}(i, j) = \frac{P(e_i \cap e_j)}{P(e_i) \times P(e_j)} \qquad \text{(Eq. 4.2)}$$

where $P(e_i \cap e_j)$ is the joint probability that both items i and j are rated by a user.

The IC measure can be used to define both category influence in vertical dimension, and category relations in horizontal dimension of the hierarchy, as illustrated below.

Definition 4 (**Category Influence of Vertical Dimension**) *Given the items $i_1, i_2, \ldots$ characterized by a same subset of category path $\mathcal{C}(i_1) = \mathcal{C}(i_2) = \ldots = \{f^1, \ldots, f^L\}$, the influence of an arbitrary category f^l $(1 \leq l \leq L)$ in the path on these items is defined as the following vector,*

$$\overrightarrow{\mathrm{CI}}(f^l) = \frac{1}{|f^l| - 1}\Big[\sum_{j \in f^l, i_1 \neq j} \mathrm{IC}(i_1, j), \sum_{j \in f^l, i_2 \neq j} \mathrm{IC}(i_2, j), \ldots\Big] \qquad \text{(Eq. 4.3)}$$

where each element in the vector is the average IC between the target item and all the other items affiliated to the category. Here we assume that there are at least two items are characterized by category f^l. Therefore, $|f^l| - 1 > 0$ always happens. This definition allows us to test the difference among the influence of categories in the same path. We then define category relations in horizontal dimension, based on item relationships formalized as follows.

Definition 5 (**Item Relationships**) *Items i, j are alternative if $P(e_i|e_j) < P(e_i)$ and $P(e_j|e_i) < P(e_j)$; they are complementary if $P(e_i|e_j) > P(e_i)$ and $P(e_j|e_i) > P(e_j)$; Items i, j are independent if $P(e_i|e_j) = P(e_i)$ and $P(e_j|e_i) = P(e_j)$;*

Two items i and j are therefore *alternative*, if the probability of i being rated given j is rated (e.g., $P(e_i|e_j)$), is lower than that without knowing whether j is rated or not (e.g., $P(e_i)$). Contrarily, they are complementary if the former is larger. Besides, they are independent if the former equals to the latter.

We now turn to the quantification of item relationships, which will be used later for measuring category relations. It turns out that IC can be a proper metric for measuring item relationships, according to the following proposition.

Proposition 3 (**Item Relationships Measured by IC**)

$$\begin{array}{lcl} \text{Items } i \text{ and } j \text{ are alternative} & \Longleftrightarrow & \text{IC} < 1 \\ \text{Items } i \text{ and } j \text{ are independent} & \Longleftrightarrow & \text{IC} = 1 \\ \text{Items } i \text{ and } j \text{ are complementary} & \Longleftrightarrow & \text{IC} > 1 \end{array} \qquad \text{(Eq. 4.4)}$$

A Smaller value of IC (< 1) *indicates a stronger alternative relationship between items i and j; vice versa, a larger value of* IC (> 1) *indicates a stronger complementary relationship between items i and j.*

Proof. Using the relationship between joint and conditional probability, $P(e_i \cap e_j) = P(e_j|e_i) \times P(e_i)$, we have

$$\text{IC}(i,j) = \frac{P(e_i \cap e_j)}{P(e_i) \times P(e_j)} = \frac{P(e_j|e_i) \times P(e_i)}{P(e_i) \times P(e_j)} = \frac{P(e_j|e_i)}{P(e_j)} \qquad \text{(Eq. 4.5)}$$

Similarly, with $P(e_i \cap e_j) = P(e_i|e_j) \times P(e_j)$, we have $\text{IC}(i,j) = \frac{P(e_i|e_j)}{P(e_i)}$. Thus, we can see that if $\text{IC}(i,j) < 1$, then $P(e_j|e_i) < P(e_j)$ and $P(e_i|e_j) < P(e_i)$, vise versa, suggesting an *alternative* relationship between items i, j is equivalent to $\text{IC}(i,j) < 1$. A smaller value of IC would indicate a larger gap between $P(e_j|e_i)$ and $P(e_j)$, $P(e_i|e_j)$ and $P(e_i)$, i.e., a stronger *alternative* relationship; the opposite also holds, i.e., a stronger *alternative* relationship indicates a smaller value of IC. If $\text{IC}(i,j) > 1$, then $P(e_j|e_i) > P(e_j)$ and $P(e_i|e_j) > P(e_i)$, vice versa, suggesting a *complementary* relationship between items i, j is equivalent to $\text{IC}(i,j) > 1$. A larger IC indicates a stronger *complementary* relationship; the opposite also holds. Similarly, if $\text{IC}(i,j) = 1$, then $P(e_j|e_i) = P(e_j)$ and $P(e_i|e_j) = P(e_i)$, vice versa, hence items i, j are *independent* and $\text{IC}(i,j) = 1$ are equivalent. □

The independence between two items provides no additional characterization of user preferences, thus it is neither beneficial for recommendation. With the above metric for measuring item relationships, we now propose the metric for category relations in horizontal dimension.

Definition 6 (**Category Relations in Horizontal Dimension**) *The relation of two categories f and g is given by the following formula,*

$$\text{CR}(f,g) = \frac{1}{|f| \times |g|} \sum\nolimits_{i \in f} \sum\nolimits_{j \in g} \text{IC}(i,j) \qquad \text{(Eq. 4.6)}$$

CR is defined as the average IC between all pairs of items, where the two items in each pair are characterized respectively by the two categories. Similar to IC, $\text{CR}(f,g) < 1$, $\text{CR}(f,g) > 1, \text{CR}(f,g) = 1$ indicate that categories f and g are *alternative*, *complementary* and independent, respectively.

4.1.2 Analysis in Real-world Data

We now show the presence of category influence and relations in the Amazon Web store data: Clothing, Shoes & Jewelry. The details of the dataset are deferred to the Experi-

mental Results section. For demonstration purpose, we only show the results of the top-3 layers of the hierarchy.

The category hierarchy contains 116 categories at Layer 1, thus having 116 paths in vertical dimension. Comparing the influence of categories in the same path, we find that 74.33% of category influence at Layer 1 is significantly larger than that at Layer 2, and that 72.57% of the influence at Layer 2 is significantly larger than that at Layer 3, with paired t-test and p-value < 0.01.

The root layer (Layer 3) contains two categories, Women's Clothing and Men's Clothing. It can be observed from Figure 4.1 (log-scaled, i.e., $x = log(\text{CR})$) at 3 layers of the hierarchy.) that the two categories have an *alternative* relation, indicating that women's and men's clothing are generally not purchased together. For Layer 2, we observe that the category relations are evenly distributed on the side $log(\text{CR}) < 0$ and $log(\text{CR}) > 0$, indicating that both *alternative* and *complementary* category relations exist at this layer. An example of *complementary* categories is Women's Watches, Men's Watches, suggesting that women's and men's watches are usually purchased together, e.g., for couples, despite of the fact that women's clothing and men's clothing are *alternative*. When looking at category relations at Layer 1, we can see that the relations among most categories are *complementary*, e.g., women's active clothing and women's athletic shoes. Overall, as a general trend, more *complementary* relations can be observed in lower layers than upper layers, suggesting that customers tend to buy items characterized differently by fine-grained categories to match each other.

4.2 The HieVH Framework

This section describes the HieVH framework – that seamlessly exploits both vertical and horizontal dimensions of CH to enhance recommendation performance.

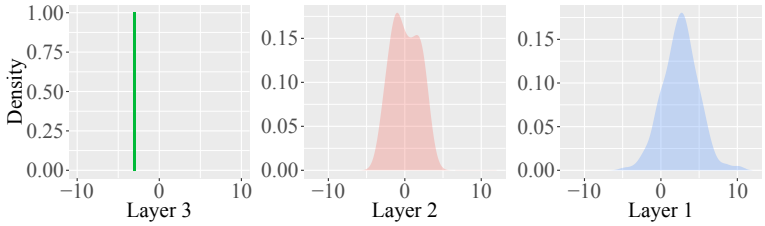


Figure 4.1: Distribution of category relations

The Basic Model. Our method is built on the latent factor model (LFM), where each user and item in the high dimensional user-item interaction space are mapped into a low dimensional space. We generalize the basic LFM to seamlessly integrate both vertical and horizontal dimensions of CH by minimizing the following equation:

$$\mathcal{J} = \overbrace{\sum_{o_{ui}\neq 0} C\big(r_{ui}, <\theta_u, \overline{\theta}_i>\big)}^{\text{cost function}} + \overbrace{\alpha \sum_{f,g\in\mathcal{C}} \Psi(\theta_f, \theta_g) + \Omega(\Theta)}^{\text{regularizers}} \qquad \text{(Eq. 4.7)}$$

where $o_{ui} = 1$ if user u rates item i, otherwise 0; $\theta_u, \theta_i, \theta_f, \theta_g \in \mathbb{R}^d$ are the latent factors of user u, item i and categories f and g, respectively; d is the dimension of latent factors; $C(\cdot)$ is a convex cost function (e.g., quadratic function) measuring the difference between the real rating r_{ui} and the predicted rating, i.e., the inner product of θ_u and $\overline{\theta}_i$; and $\overline{\theta}_i = \Phi(\theta_i, \theta_f)$ is the adaptive item latent factor considering the influence of categories in vertical dimension on item latent factors through function Φ; Ψ is the regularization function to constrain the difference between θ_f and θ_g based on the relations among categories in horizontal dimension; α controls the importance of Ψ; $\Omega(\Theta)$ with $\Theta = \{\lambda, \theta_u, \theta_i, \theta_f, \theta_g\}$ are regularizers to avoid over-fitting; λ is the regularization hyperparameter. The main challenge is how to effectively formulate the functions Φ, Ψ by integrating the influence and relations of categories in the two dimensions of CH.

4.2.1 Modeling Vertical Dimension

Categories are vertically affiliated in the hierarchy. Based on the results shown in subsection 4.1.2, we observe that an item i is characterized by all the affiliated categories

$\mathcal{C}(i) = \{f_i^1, f_i^2, \ldots, f_i^L\}$, organized as a path in the hierarchy with different degrees. Hence, we formulate the function $\Phi(\theta_i, \theta_f)$ to adapt the latent factor of item i, i.e., θ_i, by adding to it the latent factors of its affiliated categories, i.e., $\mathcal{C}(i)$ in the hierarchy,

$$\overline{\theta}_i = \Phi(\theta_i, \theta_f, \vartheta_f) = \theta_i + \left[\vartheta_{f^1}, \vartheta_{f^2}, \cdots, \vartheta_{f^L}\right] \begin{bmatrix} -\theta_{f^1}- \\ -\theta_{f^2}- \\ \cdots \\ -\theta_{f^L}- \end{bmatrix}_{L \times d} \quad \text{(Eq. 4.8)}$$

where $\vartheta_{\mathcal{C}(i)} = [\vartheta_{f^1}, \vartheta_{f^2}, \ldots, \vartheta_{f^L}]$ is the parameter vector, indicating the different influence of categories in $\mathcal{C}(i)$ on item i. It can be automatically learnt by our model. θ_{f^l} $(1 \leq l \leq L)$ is the latent vector of category $f^l \in \mathcal{C}(i)$.

In this equation, any items, e.g., i and j, that belong to the same category set, i.e., $\mathcal{C}(i) = \mathcal{C}(j)$, share the same parameter vector, i.e., $\vartheta_{\mathcal{C}(i)} = \vartheta_{\mathcal{C}(j)} = [\vartheta_{f^1}, \vartheta_{f^2}, \ldots, \vartheta_{f^L}]$. That is to say, the categories organized in a same path influence all items belonging to the leaf category in that path. In this way, we reduce the number of parameters and avoid over-fitting. The number of parameter vectors is the total number of the unduplicated category paths in CH, which is equal to the size of leaf category set. Note that, in the adaptive function Φ, a good estimation of category latent factors is essential to accurately adapt item latent factors, which can be facilitated by considering horizontal dimension of CH, as given below.

4.2.2 Modeling Horizontal Dimension

From the perspective of horizontal dimension, categories are organized as siblings or cousins at the same layer of the hierarchy. Data analysis on real-world datasets shows the presence of two types of category relations, i.e., *alternative* and *complementary*, which are highly useful to better model category latent factors.

Hence, we incorporate such kind of category relations by assuming that in each Layer l $(1 \leq l \leq L)$ of the hierarchy: if two categories are *alternative*, then the distance of

their latent factors should be large; if *complementary*, the distance of their latent factors should be small. Based on the above assumption, we devise the following regularizer to better model category latent factors,

$$\Psi(\theta_f, \theta_g) = \sum\nolimits_{l=1}^{L} \sum\nolimits_{f,g \in \mathcal{C}^l, f<g} \sigma_{fg} \|\theta_f - \theta_g\|_F^2 \qquad \text{(Eq. 4.9)}$$

where $\mathcal{C}^l$ is the category set at Layer l of the hierarchy; $\sigma_{fg} = log(\text{CR}(f, g))$ with $\sigma_{fg} < 0, \sigma_{fg} > 0, \sigma_{fg} = 0$ indicating categories f and g are *alternative*, *complementary* and *independent*, respectively. Through adaptive function Φ adding latent vectors of affiliated categories to their items' latent factors, category relations in horizontal dimension are inherited by items. Consequently, the better estimated category latent factors can more accurately adapt item latent factors.

Similarly, we also incorporate item relationships to help better model item latent factors by assuming that if two items are *alternative*, the distance of their latent factors should be large; if *complementary*, it should be small. Based on this, we design the following regularizer,

$$\Psi(\theta_f, \theta_g; \theta_i, \theta_j) = \Psi(\theta_f, \theta_g) + \sum\nolimits_{i,j \in \mathcal{I}, i<j} \sigma_{ij} \|\theta_i - \theta_j\|_F^2$$

where $\sigma_{ij} = log(\text{IC}(i, j))$ with $\sigma_{ij} < 0, \sigma_{ij} > 0, \sigma_{ij} = 0$ indicating items i and j are alternative, complementary and independent. Therefore $\Psi(\theta_f, \theta_g; \theta_i, \theta_j)$ combines the regularizations of both horizontal category relations and item relationships, aiming at restricting the distance of both category latent factors and item latent factors.

Note that σ_{fg}, σ_{ij} seamlessly accommodate our assumptions illustrated below: if two categories f, g are alternative, then we have CR < 1, thus $\sigma_{fg} < 0$. In this case, minimizing Ψ leads to large distance between θ_f and θ_g; if f, g are complementary, then we have CR > 1, thus $\sigma_{fg} > 0$. In this case, minimizing Ψ leads to small distance between θ_f and θ_g; if f, g are independent, then CR $= 1$, thus $\sigma_{fg} = 0$. In this case, the independent category relations are not considered in Ψ. σ_{ij} holds similar properties as σ_{fg}.

Once the category and item relationships are incorporated into the objective function $\mathcal{J}$, we can more accurately model category and item latent factors in function Φ, thus can ultimately better model user-item interactions.

Remark. HieVH seamlessly integrates the modeling of both vertical and horizontal dimensions of CH. Though in this chapter we focus on item CH, HieVH can as well accommodate user CH. It is noteworthy to remark how HieVH is able to handle arbitrarily imbalanced CH, thus making its application suitable to a wide variety of application scenarios. Specifically, it first determines the depth of a category as the number of layers from the root category to this category in a top-down fashion. Then, the categories that have the same depth are on the same layer.

4.2.3 Optimization and Complexity Analysis

Model Learning. We adopt the widely utilized stochastic gradient descent (SGD) method to optimize HieVH. The update rules of all the variables are given by Eq. 4.10. The optimization process is demonstrated in Algorithm 3, which is mainly composed of parameter update (line 3-12).

$$
\begin{aligned}
&\nabla\mathcal{J}(\theta_u) = \sum\nolimits_{i\in\mathcal{I}} o_{ui}\left(<\theta_u,\overline{\theta}_i> - r_{ui}\right)\overline{\theta}_i + \lambda\theta_u \\
&\nabla\mathcal{J}(\theta_i) = \sum_{u\in\mathcal{U}} o_{ui}\left(<\theta_u,\overline{\theta}_i> - r_{ui}\right)\theta_u + \lambda\theta_i + \alpha\sum_{i,j\in\mathcal{I},i<j}\sigma_{ij}(\theta_i-\theta_j) \\
&\forall f\in\mathcal{C}^l, l=\{1,2,\cdots,L\}, \\
&\nabla\mathcal{J}(\theta_f) = \sum\nolimits_{u\in\mathcal{U}}\sum\nolimits_{i\in\mathcal{I},f\in\mathcal{C}(i)}\vartheta_f o_{ui}\left(<\theta_u,\overline{\theta}_i> - r_{ui}\right)\theta_u \\
&\qquad\qquad + \lambda\theta_f + \alpha\sum\nolimits_{f,g\in\mathcal{C}^l,f<g}\sigma_{fg}(\theta_f-\theta_g) \\
&\nabla\mathcal{J}(\vartheta_f) = \sum_{u\in\mathcal{U}}\sum_{i\in\mathcal{I},f\in\mathcal{C}(i)} o_{ui}\left(<\theta_u,\overline{\theta}_i> - r_{ui}\right)<\theta_u,\theta_f> + \lambda\vartheta_f
\end{aligned}
\qquad \text{(Eq. 4.10)}
$$

Complexity Analysis. The computational time is mainly taken by evaluating the objective function $\mathcal{J}$ and updating the relevant variables. The time to compute $\mathcal{J}$ is

Algorithm 3: HieVH Optimization Process

Input: rating matrix $\mathbf{R}$, category hierarchy $\mathcal{C}$, $d, \alpha, \lambda, \gamma, Iter$

Initialize θ, ϑ with small values;
$L \leftarrow$ the highest layer of $\mathcal{C}$;
// Parameter update for $\mathcal{J}$
for $t = 1; t \leq Iter; t++$ **do**
  foreach $u \in \mathcal{U}, i \in \mathcal{I}$ **do**
    $\theta_u^{(t)} \leftarrow \theta_u^{(t-1)} - \gamma \nabla \mathcal{J}(\theta_u)$;
    $\theta_i^{(t)} \leftarrow \theta_i^{(t-1)} - \gamma \nabla \mathcal{J}(\theta_i)$;
    for $l = 1; l \leq L; l++$ **do**
      foreach $f \in \{\mathcal{C}^l \cap \mathcal{C}(i)\}$ **do**
        $\theta_f^{(t)} \leftarrow \theta_f^{(t-1)} - \gamma \nabla \mathcal{J}(\theta_f)$;
        $\vartheta_f^{(t)} \leftarrow \vartheta_f^{(t-1)} - \gamma \nabla \mathcal{J}(\vartheta_f)$;
  if $\mathcal{J}$ *has converged* **then**
    break;

$\mathcal{O}(d|\mathbf{R}| + dn^2)$, where $|\mathbf{R}|$ is the number of non-zero observations in the rating matrix $\mathbf{R}$, and n is the number of items. For the gradients $\nabla \mathcal{J}(\theta_u), \nabla \mathcal{J}(\theta_i), \nabla \mathcal{J}(\theta_f), \nabla \mathcal{J}(\vartheta_f)$, the computational time are $\mathcal{O}(d|\mathbf{R}|)$, $\mathcal{O}\left(d|\mathbf{R}| + d\frac{n(n-1)}{2}\right)$, $\mathcal{O}\left(d|\mathcal{C}||\overline{\mathbf{R}}| + d\sum_{l=1}^{L} \frac{|\overline{\mathcal{C}}|(|\overline{\mathcal{C}}|-1)}{2}\right)$, $\mathcal{O}(L|\mathcal{C}^1||\overline{\mathbf{R}}|)$, respectively. Wherein $|\mathcal{C}|$ is the total number of categories in the hierarchy; $|\overline{\mathbf{R}}|$ is the average number of ratings under each category; $|\overline{\mathcal{C}}|$ is the average number of categories at each layer of the hierarchy. Generally due to $L < |\overline{\mathcal{C}}| < |\mathcal{C}^1| < |\mathcal{C}| \ll n$ and $|\overline{\mathbf{R}}| < |\mathbf{R}|$, the overall computational complexity of Algorithm 3 is $(Iter \times \mathcal{O}(d|\mathbf{R}| + dn^2))$. In summary, our proposed framework is scalable to large datasets.

4.3 Experimental Results

In this section, we conduct comprehensive experiments on multiple real-world datasets to evaluate the performance of our proposed models by comparing with a number of state-of-the-art algorithms.

Table 4.2: Descriptive statistics of the datasets for HieVH

Data	#users	#items	#ratings	#features	#layers
Clothing	36,000	42,201	60,141	2,764	7
Electronics	43,234	38,766	77,962	1,292	6
CDs & Vinyl	33,868	36,320	71,872	1,293	6
Home & Kitchen	44,519	37,445	73,820	2,002	5

4.3.1 Experimental Setup

Datasets. To validate the effectiveness of the proposed HieVH, we use the Amazon Web store dataset [68]. This dataset has recently been applied for evaluating recommendation methods incorporating CH [33, 117]. Similar to these works, we consider the Clothing Shoes & Jewelry dataset; to evaluate the generalizability of HieVH, we further consider three other datasets in different domains, including Electronics, CDs & Vinyl, and Home & Kitchen. The CHs of all the datasets are imbalanced. Note that we do not use the datasets utilized in Chapter 3, i.e., Instagram and Twitter, as the CH contained in these two datasets is balanced. While we intent to demonstrate the ability of HieVH to model the unbalanced CH, we thus select Amazon web store dataset directly. We uniformly sample the datasets to balance their sizes for cross-dataset comparison. Table 4.2 reports the statistics of the datasets.

Comparison Methods. We compare with six state-of-the-art algorithms, 1) MF [75]: matrix factorization model; 2) CMF [94]: collective matrix factorization; 3) FM [82]: factorization machine; 4) TaxMF [16, 51]: taxonomy hierarchy based matrix factorization; 5) Sherlock [33]: visually-aware recommendation model with the incorporation of category hierarchy; 6) ReMF ([117] in Chapter 3): our recursive regularization based matrix factorization. Methods 2-3 only utilize categories in CH without considering the structure. Methods 4-6 are all based on CH. Besides, three variants of our proposed framework are compared. A) HieV: only considers vertical category *affiliatedTo* influence of CH; B) HieVC: exploits vertical category influence and horizontal *complementary*

category relation of CH; C) HieVH: integrates both vertical category influence and horizontal *complementary* & *alternative* category relations of CH.

Evaluation. Standard 5-fold cross validation is adopted to evaluate all the methods. The Area Under the ROC Curve (AUC) is used as the evaluation metric. Larger AUC indicates better recommendation performance. Note that we focus on investigating the ranking results of our proposed method, instead of the rating prediction results. Therefore, we do not adopt mean absolute error (MAE) and root mean square error (RMSE) as evaluation metrics, which are typically used to measure the rating prediction results.

Parameter Settings. Optimal parameter settings have been empirically estimated. We set $d = 10$ and apply a grid search in $\{0.001, 0.01, 0.1\}$ for γ, λ and 1/2-way regularization of FM; $\alpha = 0.5, 0.01$ for CMF and ReMF, respectively; for Sherlock, we use the same settings as suggested in [33].

4.3.2 Results and Analysis

Impact of α. In HieVH, α controls the importance of category relations in the horizontal dimension of FH. We apply grid search in $\{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$ to investigate the impact of α on recommendation performance. Results are shown in Figure 4.2. As α varies from small to large, the performance first increases then decreases, with the maximum reached at the range $[10^{-3}, 10^{-2}]$. The performance variations across datasets suggest the need for dataset-specific settings; the similarity in performance variation across α values demonstrates the robustness of HieVH.

Comparative Results. Table 4.3 summarizes the performance of all comparison methods across all datasets, where two views are created for each dataset: 'All Users' indicates all users are considered in the test data; while 'Cold Start' indicates only users with ≤ 5 ratings are involved in the test data. Several interesting observations can be noted.

Table 4.3: Performance (AUC) of all the comparison methods. The best performance is highlighted in bold; the second best performance of other methods is marked by '*'; the column 'Improve' indicates the relative improvements that the proposed HieVH achieves w.r.t. the '*' results

Datasets	**Cases**	**MF**	**CMF**	**FM**	**TaxMF**	**Sherlock**	**ReMF**	**HieV**	**HieVC**	**HieVH**	**Improve**
Clothing.	All Users	0.5455	0.5646	0.6826	0.6509	0.6747	0.7015*	0.7160	0.7291	**0.7375**	5.13%
	Cold Start	0.5426	0.5667	0.6629	0.6493	0.6702	0.7032*	0.7124	0.7284	**0.7352**	4.55%
Electronics	All Users	0.5555	0.5762	0.6839	0.6569	0.6915	0.7337*	0.7512	0.7672	**0.7748**	5.60%
	Cold Start	0.5526	0.5735	0.6831	0.6475	0.6982	0.7305*	0.7474	0.7658	**0.7741**	5.97%
CDs& Vinyl	All Users	0.5478	0.5622	0.6356	0.6905	0.7082	0.7249*	0.7328	0.7516	**0.7600**	4.84%
	Cold Start	0.5433	0.5609	0.6231	0.6881	0.7076	0.7243*	0.7315	0.7514	**0.7588**	4.76%
Home & Kit.	All Users	0.5420	0.5545	0.6938	0.6469	0.6938	0.7279*	0.7456	0.7574	**0.7667**	5.33%
	Cold Start	0.5395	0.5562	0.6915	0.6511	0.6973	0.7275*	0.7412	0.7554	**0.7650**	5.15%

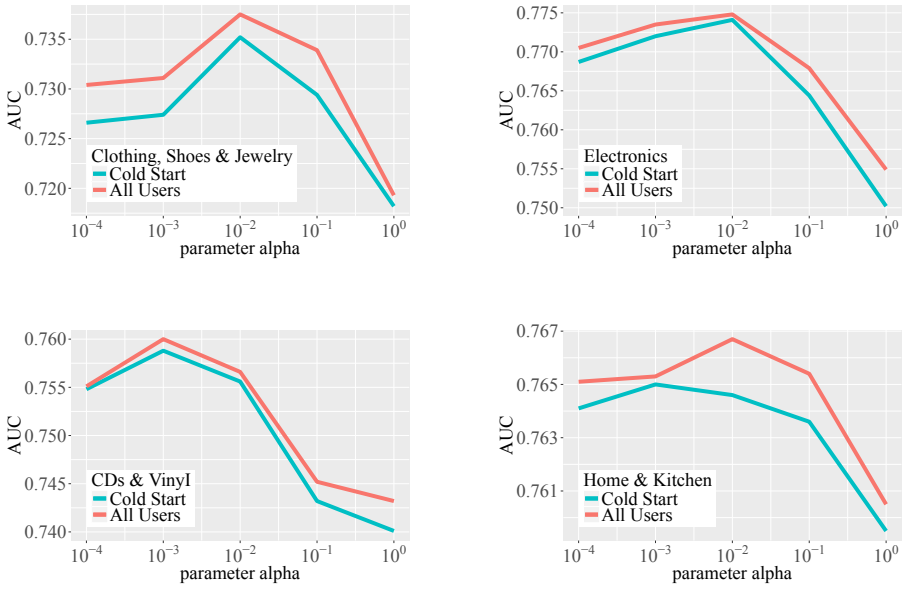


Figure 4.2: The impact of parameter α on HieVH

Compared with all other methods incorporating CH, MF considering no auxiliary information performs the worst, indicating the effectiveness of category-aware recommendation. The methods originally designed for the flat category structure, including CMF and FM, generally perform worse than the CH based methods (TaxMF, Sherlock and ReMF). Since CH needs to be converted into a flat structure when applied into CMF and FM, the result demonstrates that useful information is lost in the conversion. FM outperforms CMF and even some CH based methods. This could be explained by FM further considering user-category interactions, in addition to the user-item and item-category interactions, as in CMF.

Among the three state-of-the-art CH based methods, Sherlock performs better than TaxMF, but worse than ReMF. The reason behind is that TaxMF views the influence of categories in different layers of CH identically, whereas Sherlock weights the influence of categories in different layers differently. However, the weights are defined manually. In contrast, ReMF automatically learns such influence by a parameterized regularization

Table 4.4: Cp and Ap of the Clothing Hierarchy

Approaches	Layer 1		Layer 2	
	Cp	Ap	Cp	Ap
ReMF	87.62%	12.38%	75.23%	24.76%
HieV	88.89%	11.11%	76.19%	23.89%
HieVC	92.62%	7.38%	84.62%	15.38%
HieVH	95.65%	4.35%	89.35%	10.65%

traversing from root to leaf categories.

We now compare the three variants of our proposed framework – HieV, HieVC and HieVH, with our ReMF method in Chapter 3. By considering the vertical influence of category *affiliatedTo* relation only, HieV performs slightly better than ReMF. The possible explanation behind is that in HieV, item latent factors are directly adapted by the affiliated category latent factors; whereas in ReMF, latent factors of items are regularized by those of items that share common ancestor categories, which means items are indirectly influenced by their affiliated categories. In other words, the adaption of item latent factors in HieV is more straightforward than that in ReMF, thus more effective. HieVC upgrades HieV by adding *complementary* category relation in the horizontal dimension; HieVC is then promoted to HieVH by further incorporating *alternative* category relation. In results, HieVC performs better than HieV, but worse than HieVH, implying that both *complementary* and *alternative* category relations among horizontally organized categories help improve recommendation accuracy.

Overall, when compared with all the other comparison methods across all the datasets, HieVH achieves the best performance. The improvements w.r.t. all users and cold start are 5.23%, 5.11% on average, respectively, which are statistically significant (Paired t-test, p-value < 0.001). This implies that the recommendation performance can be further enhanced by appropriately considering both vertical and horizontal dimensions of CH.

Interpretations by HieVH. We now analyze how the incorporation of category relations can better explain user-item interactions. To this end, we first derive for each user

Figure 4.3: The example of recommendations generated by HieVH

the category relations between the rated items (i.e., training data), and the correctly recommended items (i.e., intersection between recommended items and test data). We calculate the percentage of complementarity Cp and alternativity Ap among these relationships for each user. Good recommendations would result in a high percentage of complementary items and low percentage of alternative items.

Table 4.4 shows the average Cp and Ap for all users in the test data at the top-3 layers of the Clothing hierarchy (Layer 3 excluded since only an alternative relation exists). We could see that from ReMF, HieV to HieVC, HieVH, with *complementary* and *alternative* category relations considered, Cp increases, and Ap decreases in both layers. Among all the methods, HieVH achieves the highest Cp, and lowest Ap, with significant improvements over HieVC (Paired t-test, p-value < 0.01). This clearly indicates that by incorporating the two types of category relations in the horizontal dimension of CH, the recommendations better approximate real user preferences. Example users to whom the recommendations benefit from category relations generated by HieVH are shown in Figure 4.3. The recommendations to users u_1 and u_2 are better because of the *complementarity* among fashion clothing that u_1 is more fond of, and among athletic clothing that u_2 instead is more fond of, and the *alternativity* between fashion and athletic clothing. Similarly, the recommendations for u_3 are provided because of her interests in fashion clothing and lingerie. For user u_4, it is discovered that he likes clothing collocation, i.e., the *complementarity* of the items he purchased.

4.4 Summary

Category hierarchy is well-known to enhance the recommendation performance, as emphasized in Chapter 3, where we propose a recursive regularization based matrix factorization model, i.e., ReMF, to accommodate the influence of category *affiliatedTo* relation in vertical dimension of CH, so as to improve the recommendation performance. While ReMF ignores category relations in horizontal dimension of CH. In this chapter, we first show the presence of vertical category influence and horizontal relations in real-world datasets based on our proposed metrics. Then we design HieVH to seamlessly exploit both the vertical and horizontal dimensions of category hierarchy for better recommendation. Experimental results on four real-world datasets show that HieVH consistently outperforms state-of-the-art counterparts. Furthermore, HieVH provides better interpretations of the generated recommendations.

Chapter 5

MRLR: Recommendation with Multi-level Representation Learning

In Chapter 4, we propose HieVH that incorporates category relations in both vertical and horizontal dimensions of CH into latent factor model to further boost recommendation accuracy. In the recent years, representation learning (RL) has proven to be more effective than latent factor model (LFM) in capturing local item relationships by modeling item co-occurrence in individual user's interaction record. However, we argue that the value of RL for recommendation has not reached the full potential. Existing RL based recommendation methods either neglect personalization or overlook the multi-level organizations of items for fine-grained item relationships.

In this chapter, we therefore design a unified Bayesian framework MRLR [102] to learn user and item embeddings from a multi-level item organization, thus capturing item relationships in different levels of granularity. Specifically, we first extend the original item embedding method to a more generic Bayesian framework, under which we then fuse the likelihood function of user-specific pairwise item ranking. This unified framework can thus benefit from user and item RL while reaching the goal of personalized recommendation. To fully exploit RL, we further extend the framework to multi-level RL by introducing item category as the intermediate level of item organization between individual items and items rated by the same user, so as to capture fine-grained item

relationships. Extensive validation on multiple real-world datasets demonstrates that MRLR consistently outperforms state-of-the-art algorithms.

This chapter is organized as follows. In Section 5.1, we first formulate recommendation as a personalized ranking problem, based on which the objective function is proposed. We then propose the MRLR framework in Section 5.2. Specifically, we present the details of modeling personalization and multi-level item organization step by step; later we introduce the optimization method and complexity analysis for the proposed framework. Section 5.3 describes the experimental results, followed by the conclusion in Section 5.4.

5.1 Problem Formulation and Objective Function

Suppose we have m users $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, and n items $\mathcal{I} = \{v_1, v_2, \ldots, v_n\}$. We use the binary user feedback matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$. If the interaction (rating) from u_p to v_i is observed, indicating u_p prefers v_i, then $\mathbf{R}_{pi} = 1$; otherwise 0. $\mathcal{I}^+_{u_p}$ is the set of items that user u_p prefers. $\mathcal{D}_r = \{(u_p, v_i, v_j) | u_p \in \mathcal{U}, v_i \in \mathcal{I}^+_{u_p}, v_j \in \mathcal{I} \backslash \mathcal{I}^+_{u_p}\}$ is the set of user-specific ranking triples indicating u_p prefers v_i to v_j, where $\mathcal{I} \backslash \mathcal{I}^+_{u_p}$ denotes the set of items that u_p has no interaction with. $\mathcal{D}_c = \{(u_p, v_i, v_k) | u_p \in \mathcal{U}, v_i, v_k \in \mathcal{I}^+_{u_p}\}$ is the set of item co-rated triples indicating u_p prefers both v_i and v_k. For each user, we aim to provide a personalized ranking list of top-K items that she has not interacted with. Table 5.1 summarizes all the notations utilized in this chapter.

More specifically, our goal is to design a unified multi-level RL framework (MRLR) to learn user and item embeddings from both item co-rated relationships and user-specific ranked lists of items, thus to benefit from user and item RL, as well as to reach the goal of personalized recommendation. We define the objective function of MRLR using a Bayesian framework, maximizing the following posterior probability,

$$P(\Theta|\mathcal{D}) \propto P(\mathcal{D}|\Theta)P(\Theta) \propto P(\mathcal{D}_c, \mathcal{D}_r|\Theta)P(\Theta) \qquad \text{(Eq. 5.1)}$$

Table 5.1: Mathematical Notations for MRLR

Notation	Description
$\mathcal{U}, \mathcal{I}, \mathcal{C}$	user, item, category set
$\mathbf{R}_{pi}$	rating given by user u_p to item v_i
$\mathcal{I}_{u_p}^{+}$	set of items that u_p prefers
$\mathcal{D}_r$	set of user-specific ranking triples
$\mathcal{D}_c$	set of item co-rated triples
$\mathbf{u}_p, \mathbf{v}_k, \mathbf{c}_l$	user, item and category latent factors
α_1	importance of personalization
α_2	importance of item co-rated relationships
α_3	importance of category influence
λ_Θ	regularization coefficient
$\Omega(\Theta)$	regularization term to avoid over-fitting
$\mathcal{J}$	objective function of MRLR framework

where Θ is the set of parameters in MRLR, $\mathcal{D}$ is the observed data. It is proportional to maximizing the likelihood of the observed triples given the embeddings, i.e., $P(\mathcal{D}|\Theta)$. We define the likelihood function as the joint probability of item co-rated triples and user-specific ranking triples, i.e., $P(\mathcal{D}_c, \mathcal{D}_r|\Theta)$. Assuming the item co-rated triples and user-specific ranking triples are conditionally independent, the joint probability is then reformulated as follows:

$$\begin{aligned} P(\mathcal{D}_c, \mathcal{D}_r|\Theta) =& P(\mathcal{D}_c|\Theta)P(\mathcal{D}_r|\Theta) \\ =& \prod_{(u_p,v_i,v_k)\in\mathcal{D}_c} P((u_p, v_i, v_k)|\Theta) \prod_{(u_p,v_i,v_j)\in\mathcal{D}_r} P((u_p, v_i, v_j)|\Theta) \end{aligned} \quad \text{(Eq. 5.2)}$$

where $P((u_p, v_i, v_k)|\Theta), P((u_p, v_i, v_j)|\Theta)$ denote the conditional probability of item co-rated triples and user-specific ranking triples, respectively. Hence, the MRLR framework seamlessly fuses the two components: (1) item co-rated triples for better user and item embedding; (2) user-specific ranking triples for personalized ranking. Besides, through multi-level RL, MRLR can fully exploit RL from a multi-level item organization, i.e., items in user-specific ranked list, items in a same category, and individual items, to capture item relationships in different levels of granularity for better recommendation.

5.2 The MRLR Framework

This section first presents the proposed multi-level RL framework (MRLR) to achieve the goal of personalized recommendation. Then the optimization method and complexity analysis of MRLR are elaborated in detail.

5.2.1 Modeling Personalized Recommendation

Modeling User and Item Embedding. For each user u_p and item $v_i \in \mathcal{I}_{u_p}^+$, the Skip-gram method [70, 71] aims at predicting the probability of item $v_k \in \mathcal{I}, (i \neq k)$ also preferred by u_p, i.e., $P(v_k|v_i)$, which is calculated by the softmax function:

$$P(v_k|v_i, \Theta) = \frac{exp(\mathbf{v}_i^T \mathbf{v}'_k)}{\sum_{v_g \in \mathcal{I}} exp(\mathbf{v}_i^T \mathbf{v}'_g)} \quad \text{(Eq. 5.3)}$$

where $\mathbf{v}_i, \mathbf{v}_k, \mathbf{v}_g$ are embeddings of items v_i, v_k, v_g, respectively.

To allow for personalization, we model user u_p's preference towards item v_k by a similar softmax function:

$$P(v_k|u_p, \Theta) = \frac{exp(\mathbf{u}_p^T \mathbf{v}'_k)}{\sum_{v_g \in \mathcal{I}} exp(\mathbf{u}_p^T \mathbf{v}'_g)} \quad \text{(Eq. 5.4)}$$

where $\mathbf{u}_p$ denotes the user embedding of u_p.

We now model the item co-rated triples $P((u_p, v_i, v_k)|\Theta)$. It should properly accommodate both the item co-rated relationships (Eq. 5.3), and personalization (Eq. 5.4). Instead of directly optimizing $P((u_p, v_i, v_k)|\Theta)$, we optimize the conditional probability $P(v_k|(u_p, v_i), \Theta)$, $P(v_i|(u_p, v_k), \Theta)$ and $P(u_p|(v_i, v_k), \Theta)$. Since we aim to recommend items to given users, we do not need to model $P(u_p|(v_i, v_k), \Theta)$. We take $P(v_k|(u_p, v_i)\Theta)$ for example. Inspired by document RL in NLP [55], the user and item embeddings $\mathbf{u}_p, \mathbf{v}_i$ are summed as the new condition to predict the probability of v_k rated by u_p, given by,

$$P(v_k|(u_p, v_i), \Theta) = \frac{exp(\alpha_1 \mathbf{u}_p^T \mathbf{v}'_k + \alpha_2 \mathbf{v}_i^T \mathbf{v}'_k)}{\sum_{v_g \in \mathcal{I}} exp(\alpha_1 \mathbf{u}_p^T \mathbf{v}'_g + \alpha_2 \mathbf{v}_i^T \mathbf{v}'_g)} \quad \text{(Eq. 5.5)}$$

where $\alpha_1 + \alpha_2 = 1.0$; $exp(\alpha_1 \mathbf{u}_p^T \mathbf{v}'_k + \alpha_2 \mathbf{v}_i^T \mathbf{v}'_k)$ aims to take into account both the personalized aspect by the term $\mathbf{u}_p^T \mathbf{v}'_k$, and item co-rated relationships by the term $\mathbf{v}_i^T \mathbf{v}'_k$. We model $P(v_i|(u_p, v_k), \Theta)$ in a similar way.

Modeling Personalized Ranking. It has proven that recommendation is better modeled as a personalized ranking problem than the rating prediction one [83, 111]. Existing RL methods, however, optimize towards predicting user preferences over individual items (i.e., rating prediction), instead of predicting user preferences over a list of items (i.e., personalized ranking). We now proceed to model the user-specific ranking triples $P((u_p, v_i, v_j)|\Theta)$, to achieve the goal of personalized ranking. Similarly, we optimize the conditional probability of $P((v_j, v_i)|u_p, \Theta)$ and $P(u_p|(v_j, v_i), \Theta)$ instead of $P((u_p, v_i, v_j)|\Theta)$. As our goal is to recommend items, we only consider $P((v_j, v_i)|u_p, \Theta)$, which involves a user's preference over a pair of items. Based on Eq. 5.4, we further deduce a user's preference on a pair of items. As the triple (u_p, v_i, v_j) indicates that u_p prefers v_i to v_j, it means that for u_p, we should maximize the probability that v_i is preferred by u_p but v_j is not favored by u_p. We denote such probability by $P((\neg v_j, v_i)|u_p, \Theta)$, which is defined as below:

$$P((\neg v_j, v_i)|u_p, \Theta) = \frac{exp(\mathbf{u}_p^T \mathbf{v}'_i - \mathbf{u}_p^T \mathbf{v}'_j)}{\sum_{v_h, v_g \in \mathcal{I}} exp(\mathbf{u}_p^T \mathbf{v}'_h - \mathbf{u}_p^T \mathbf{v}'_g)} \quad \text{(Eq. 5.6)}$$

where the term $exp(\mathbf{u}_p^T \mathbf{v}'_i - \mathbf{u}_p^T \mathbf{v}'_j)$ denotes the preference difference of user u_p towards item v_i and item v_j.

As we illustrate in Chapter 3 and 4, item category information has proven to be effective to enhance recommendation performance, as items in a same category inherit similar characteristics, and users may have close preference towards the associated items in the same category. Analogically, item category can be also integrated into our proposed approach to achieve a multi-level representation learning framework, thus to further boost recommendation accuracy, as we will elaborate in the next subsection.

5.2.2 Modeling Multi-level Item Organization

We further consider multi-level granularity of item organizations to capture fine-grained item relationships. Specifically, we introduce item category as the intermediate level between items in the same user-specific ranked list and individual items. The rationale behind is that items in a same category generally share similar characteristics.

To integrate the influence of item category for better recommendation, we extend our framework to multi-level RL. The item embedding is thus reformulated as,

$$\overline{\mathbf{v}}_i = \mathbf{v}_i + \frac{\alpha_3}{|\mathcal{C}_{v_i}|} \sum\nolimits_{c_l \in \mathcal{C}_{v_i}} \mathbf{c}_l \tag{Eq. 5.7}$$

where $\mathcal{C}_{v_i}$ is the set of categories that v_i belongs to; $|\mathcal{C}_{v_i}|$ is the size of $\mathcal{C}_{v_i}$; $\mathbf{c}_l$ is the embedding for category c_l. By replacing the item embedding in Eq. 5.5 and Eq. 5.6, the category RL can adapt item embedding, which is similar as what we do in the vertical dimension of HieVH in Chapter 4, serving as the intermediate level RL. α_3 controls the strength of the influence of category embeddings on item embeddings. MRLR can now capture fine-grained relationships of items in local context (i.e., item co-rated relationships), in the same category, and in user-specific ranked item list.

5.2.3 Optimization and Complexity Analysis

Model Learning. Optimizing our MRLR framework is proportional to minimizing the negative log-likelihood function, given by,

$$\begin{aligned} \min_{\Theta} \mathcal{J} = - \sum_{(u_p, v_i, v_k) \in \mathcal{D}_c} & logP((u_p, v_i, v_k)|\Theta) - \\ & \sum_{(u_p, v_i, v_j) \in \mathcal{D}_r} logP((u_p, v_i, v_j)|\Theta) + \lambda_\Theta \Omega(\Theta) \end{aligned} \tag{Eq. 5.8}$$

where $\Omega(\Theta)$ is the regularizer to prevent over-fitting, and λ_Θ is the regularization coefficient. To solve the optimization problem, we apply the stochastic gradient descent (SGD) method to the objective function $\mathcal{J}$.

Approximation of softmax function. It is impractical to directly adopt the softmax functions $P(v_k|(u_p, v_i), \Theta)$, $P(v_i|(u_p, v_k), \Theta)$ and $P((\neg v_j, v_i)|u_p, \Theta)$ to optimize our framework, since the cost of computing the denominators of these functions is proportional to the total number of items (n), which is considerably huge in real-world applications. To accelerate the speed, we adopt negative sampling proposed in [71]. Take $P(v_k|(u_p, v_i), \Theta)$ as an example, which can be approximated via negative sampling as follows:

$$\begin{aligned} &P(v_k|(u_p, v_i), \Theta) = \sigma(\mathbf{u}_p^T \mathbf{v}'_k + \mathbf{v}_i^T \mathbf{v}'_k) \\ &\prod_{g=1}^{N} \mathbb{E}_{(u_p, v_i, v_g) \sim P(\mathcal{D}_c^-)} \sigma(-(\mathbf{u}_p^T \mathbf{v}'_g + \mathbf{v}_i^T \mathbf{v}'_g)) \end{aligned} \tag{Eq. 5.9}$$

where $\sigma(x) = 1/(1 + exp(-x))$ is the sigmoid function; $\mathcal{D}_c^- = \mathcal{D}_r$ is the opposite triple set of $\mathcal{D}_c$; $P(\mathcal{D}_c^-)$ is a function randomly sampling instances from $\mathcal{D}_c^-$. N is the number of negative instances to be drawn per positive instance. The idea behind negative sampling is that we want to maximize the similarity between v_k and (u_p, v_i) and minimize the similarity between a randomly sampled item v_g and (u_p, v_i). In this way, we can approximately maximize $P(v_k|(u_p, v_i), \Theta)$.

Similarly, $P(v_i|(u_p, v_k), \Theta), P((\neg v_j, v_i)|u_p, \Theta)$ are also approximated via negative sampling. One issue we should deal with is that computing the numerators of the softmax function $P((\neg v_j, v_i)|u_p, \Theta)$ is also very expensive, as we have at least $\mathcal{O}(mn * \min(|\mathcal{I}_{u_1}^+|, \cdots, |\mathcal{I}_{u_m}^+|))$ training triples in $\mathcal{D}_r$, where $|\mathcal{I}_{u_m}^+|$ is the size of $\mathcal{I}_{u_m}^+$. Following Bayesian personalized ranking model (BPR) [83], we thus randomly sample user-specific ranking triples instead of using all the triples. This is an optimal trade-off between efficiency and accuracy, and detailed discussion can be noted in [83]. The optimization process is shown in Algorithm 4, which is mainly composed of two steps, i.e., negative sampling (line 3-5), and parameter update (line 6-13).

Complexity Analysis. The computational time is mainly taken by evaluating the objective function $\mathcal{J}$ and updating the related variables. The time to compute $\mathcal{J}$ is

Algorithm 4: The optimization of MRLR

Input: $\mathbf{R}, \mathcal{C}, \lambda_\Theta, \alpha, \gamma, d, iter$

Initialize $\Theta = \{\mathbf{u}, \mathbf{v}, \mathbf{c}\}$ with small values;
Randomly sample (u_p, v_i, v_j) for $\mathcal{D}_r$;
// Negative sampling procedure
foreach $(u_p, v_i, v_k) \in \mathcal{D}_c$, *and* $(u_p, v_i, v_j) \in \mathcal{D}_r$ **do**
    Draw N negative instances from the distribution $P(\mathcal{D}_c^-)$;
    Draw N negative instances from the distribution $P(\mathcal{D}_r^-)$;
// Parameter update
for $t = 1; t \leq iter; t++$ **do**
    foreach $(u_p, v_i, v_k) \in \mathcal{D}_c$, *and* $(u_p, v_i, v_j) \in \mathcal{D}_r$ **do**
        $\mathbf{u}_p^{(t)} \leftarrow \mathbf{u}_p^{(t-1)} - \gamma \nabla \mathcal{J}(\mathbf{u}_p)$;
        $\mathbf{v}^{(t)} \leftarrow \mathbf{v}^{(t-1)} - \gamma \nabla \mathcal{J}(\mathbf{v}), \mathbf{v}{=}\{\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k, \mathbf{v}_g, \mathbf{v}_h\}$;
        for $l = 1; l \leq |\mathcal{C}_v|; l++$ **do**
            $\mathbf{c}_l^{(t)} \leftarrow \mathbf{c}_l^{(t-1)} - \gamma \nabla \mathcal{J}(\mathbf{c}_l)$;
    if $\mathcal{J}$ *has converged* **then**
        break;

$\mathcal{O}\big(d|\mathcal{D}_c| + d|\mathcal{D}_r|\big)$, where d is the dimension of embeddings, and $|\mathcal{D}_c|, |\mathcal{D}_r|$ are the sizes of item co-rated triples and user-specific ranking triples, respectively. For all gradients $\nabla \mathcal{J}(\mathbf{u}_p), \nabla \mathcal{J}(\mathbf{v}_i), \nabla \mathcal{J}(\mathbf{c}_l)$, the computational time are $\mathcal{O}\big(d|\mathcal{D}_c| + d|\mathcal{D}_r|\big)$, $\mathcal{O}\big(d|\mathcal{D}_c| + d|\mathcal{D}_r|\big)$ and $\mathcal{O}\big(d(|\mathcal{D}_c| + |\mathcal{D}_r|)|\mathcal{C}_{v_i}|\big)$, respectively. $|\mathcal{C}_{v_i}|$ is generally no larger than 10 in real-world applications [117]. Hence, the overall computational complexity is $(\#iteration * \mathcal{O}(d|\mathcal{D}_c| + d|\mathcal{D}_r|))$. Specifically, $|\mathcal{D}_c| \leq mq(q-1)/2$, where $q = \max(|\mathcal{I}_{u_1}^+|, \cdots, |\mathcal{I}_{u_m}^+|)$. In real-world, q is typically small (e.g., power-law distribution). For $\mathcal{D}_r$, as illustrated before, we adopt the random sampling method to reduce its number. To sum up, MRLR is scalable to large datasets.

5.3 Experimental Results

In this section, we conduct comprehensive experiments on multiple datasets to evaluate the effectiveness of our proposed approach against other state-of-the-art counterparts. We further provide an insightful analysis based on the experimental results.

5.3.1 Experimental Setup

Datasets. Following Chapter 3 and Chapter 4, we also adopt the Amazon Web store data [68] in this chapter, which contains a series of datasets from various domains (e.g., clothing, electronics). To evaluate the effectiveness of MRLR, we choose four datasets, including Clothing, Electronics, Sports, Home. Besides user-item interactions, the datasets also include the categories that each item belongs to. We uniformly sample the datasets, to balance their sizes in the same order of magnitude for cross-dataset comparison. Table 5.2 reports the statistics of the datasets.

Table 5.2: Statistics of the datasets for MRLR

Datasets	#Users	#Items	#Ratings	#Categories
Clothing	29,550	50,677	181,993	1,764
Electronics	59,457	64,348	518,291	1,292
Sports	28,708	46,315	237,578	1,293
Home	37,884	50,948	313,871	2,002

Comparison Methods. We compare with seven state-of-the-art algorithms, 1) MF [75]: matrix factorization model aiming at rating prediction; 2) BPR [83]: Bayesian personalized ranking focusing on item ranking; 3) FM [82]: factorization machine fusing item category. We only compare with FM, as it generally outperforms other LFM based methods; 4) Item2Vec [4]: item embedding based method; 5) Meta-Prod2Vec [104]: integrates item category based on Item2Vec; 6) CoFactor [58]: jointly factorizes user-item rating matrix and item co-rated matrix; 7) User2Vec [26]: considers the user as a global context while learning item embedding; Moreover, four variants of our proposed framework are also compared, a) RL: RL model only considering user and item embedding ; b) PR: personalized ranking model; c) RLR: the RL model combining models a) and b); d) MRLR: multi-level RL model with multi-level item organizations based on c).

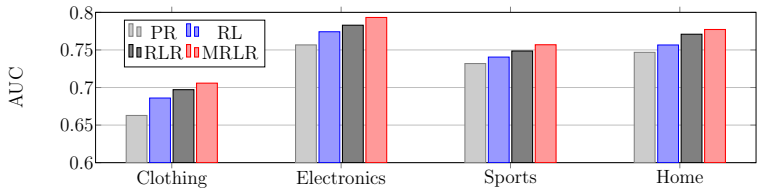


Figure 5.1: The results of our four variants for MRLR

Evaluation. Standard 5-fold cross validation is adopted to evaluate all the methods. The Area Under the ROC Curve (AUC) is used as the evaluation metric. Larger AUC indicates better recommendation performance.

Parameter Settings. We empirically find out the optimal parameter settings for all method. We set $d = 10$. We apply a grid search in $\{0.001, 0.01, 0.1, 1.0\}$ for the learning rate γ, λ_Θ and 1/2-way regularization of FM, and a grid search in $\{1, 5, 10, 20, 50\}$ for the number of negative instances N.

5.3.2 Results and Analysis

Results of Variants. The performance of our four variants is depicted by Figure 5.1. RLR outperforms both PR and RL by 3.54% and 1.42% in AUC respectively (both significant, Paired t-test with p-value $< .01$), showing the effectiveness of both representation learning and personalized ranking. MRLR combining RLR with multi-level item organizations, performs the best among the four variants – with 1.12% lift in AUC compared to RLR (p-value $< .01$), indicating the benefit of considering fine-grained item relationships.

Impacts of Parameter α. Parameters α_1, α_2 control the importance of personalization and item co-occurrence relationships as shown in Eq. 5.5. α_3 controls the effect of item category for adapting item embedding as shown in Eq. 5.7. We apply a grid search ranging from 0 to 1 with step 0.1 to investigate their impacts. As $\alpha_1 + \alpha_2 = 1$, we only study the impacts of α_1, α_3, and we fix one and vary the other each time. The results

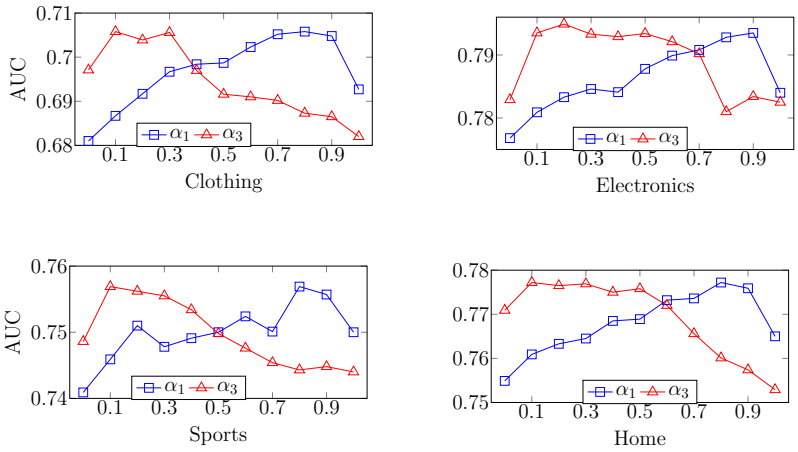


Figure 5.2: The effects of parameters α_1 and α_3 on MRLR

are described by Figure 5.2. For the four datasets, as α_1 varies from small to large, the performance first increases then decreases, with the maximum reached at around 0.8. This indicates that user preferences play an important role in item recommendation. In terms of α_3, we observe that the optimal settings range from 0.1 to 0.2, denoting a substantial contribution of item category in recommendation. The similarity in performance variation across α_1, α_3 values on the four datasets demonstrates the robustness of MRLR.

Visualization of Embeddings. MRLR framework can generate meaningful embeddings that help interpret recommendation results. To show this, we visualize the embeddings of users, items and categories learnt by MRLR in a two dimensional space using t-SNE [66]. Figure 5.3 illustrates the results of two examples in the Clothing dataset, which is the visualization of user (red dot), item (blue triangle), and category (brown square) embeddings in a two dimensional space, where left-pointing triangles are rated items; right-pointing triangles are recommended items; the category of an item is labelled by a rectangle whose color is the same as its belonging category. For conciseness, we do not visualize the other datasets, however, similar observations as below can be obtained: 1) the rated items and the recommended items are generally clustered. This indicates certain similarity among the rated items and the recommended items to the

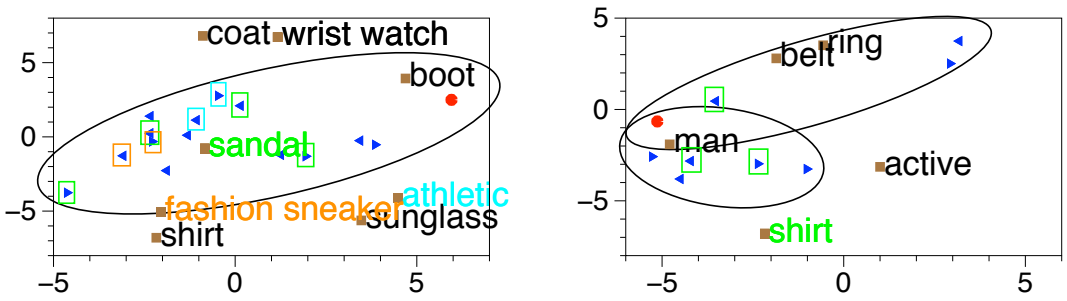


Figure 5.3: Visualization of embeddings for MRLR

same user. 2) each cluster is located at the side of the user, and the user is represented as an endpoint of these clusters, indicating that user preference can be manifested as the direction along which the rated items are clustered. This suggests that the recommendations are determined by both rated items and user preferences. Finally, we note that the categories of recommended items are overlapped with those of the rated items. For instance, for the user in the right plot the overlapped category is Shirts, indicating user preference over shirts. For the user in the left plot the overlapped categories are Athletic, Fashion Sneakers, and Sandals, indicating that the user has a more diverse set of preferences. These observations show that MRLR can capture meaningful item relationships in multiple levels of item organizations – individual items, items in the same category, and items rated by the same user.

Comparative Results. Table 5.3 summarizes the performance of all comparison methods. Two views are considered: 'All Users' indicates all users are considered in the test data; while 'Cold Start' indicates only users with less than 5 ratings are involved in the test data. Several interesting findings are observed as follows.

Among the latent factor model based methods (MF, BPR and FM), MF performs the worst, as it is the basic rating prediction method without considering any auxiliary information. FM incorporates item category as auxiliary input, significantly outperforms

Table 5.3: Performance (AUC) of all the comparison methods, where the best performance is highlighted in bold; the second best performance of other methods is marked by '*'; the column 'Improve' indicates the improvements of our proposed MRLR approach relative to the '*' results

Datasets	**Cases**	**MF**	**BPR**	**FM**	**Item2Vec**	**MP2Vec**	**CoFactor**	**User2Vec**	**MRLR**	**Improve**
Clothing	All Users	0.5255	0.6151	0.5972	0.6429	0.6600*	0.6012	0.6249	**0.7058**	6.94%
	Cold Start	0.5291	0.6135	0.5969	0.6426	0.6602*	0.5984	0.6203	**0.7022**	6.36%
Electronics	All Users	0.6595	0.7178	0.7066	0.7529	0.7604*	0.7000	0.7121	**0.7932**	4.31%
	Cold Start	0.6558	0.7161	0.7010	0.7535	0.7631*	0.6937	0.7107	**0.7935**	3.98%
Sports	All Users	0.6136	0.6992	0.6856	0.7015	0.7148*	0.6693	0.6852	**0.7569**	5.89%
	Cold Start	0.6175	0.7013	0.6861	0.7063	0.7149*	0.6679	0.6883	**0.7541**	5.48%
Home	All Users	0.6319	0.6930	0.6795	0.7297	0.7455*	0.6737	0.6969	**0.7772**	4.25%
	Cold Start	0.6408	0.6911	0.6841	0.7317	0.7449*	0.6731	0.6917	**0.7763**	4.22%

MF, indicating the effectiveness of item category for better recommendation. Interestingly, the performance of FM is worse than that of BPR. This verifies that personalized ranking is more effective than rating prediction in real-world recommendation scenarios.

The RL methods, including Item2Vec, MetaProd2Vec, CoFactor and User2Vec, generally perform better than latent factor based methods, despite being rating prediction models. This confirms that representation learning is more effective than latent factor models for recommendation. Among them, Item2Vec performs worse than MetaProd2Vec. This observation further confirms the previous conclusion that item category is useful to improve recommendation performance.

CoFactor and User2Vec consider personalization in addition to item embedding. CoFactor is equivalent to the CMF method as it simultaneously factorizes user-item and item-item co-occurrence matrices with shared item latent factors, while User2Vec adopts CBOW to integrate personalization. Theoretically, the performance of the two methods should be better than that of Item2Vec, since they can provide users with personalized item recommendation list. We empirically find that User2Vec outperforms CoFactor, but both are slightly worse than Item2Vec. However, our proposed variant RL with Skip-gram outperforms Item2Vec, by 6.37% on average (Figure 5.1). Hence, we conjecture that considering personalization with Item2Vec helps improve recommendation performance, but CMF, CBOW are less effective than Skip-gram in incorporating item co-occurrence relationships with personalization.

Overall, compared with all the other methods, MRLR performs the best by learning user and item embeddings from a multi-level item organization, i.e., items in user-specific ranked list, items in the same category, and individual items. The improvements w.r.t. 'All User' and 'Cold Start' are 5.35%, 5.01% on average (both with p-value $< .01$), respectively. This implies that recommendation performance can be further enhanced by appropriately considering multi-level representation learning and personalized ranking.

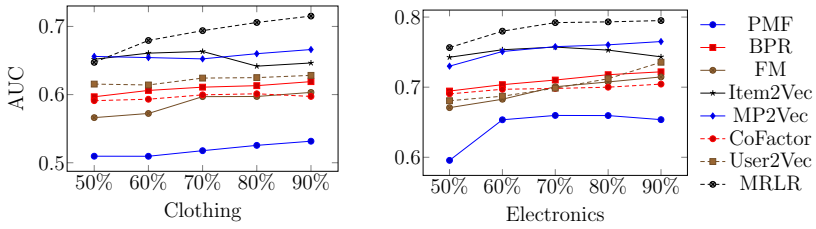


Figure 5.4: Impacts of data sparsity on the performance for MRLR

Impacts of Data Sparsity. We further study the impacts of data sparsity on the recommendation performance. Figure 5.4 depicts the variation of performance of all methods on Clothing & Electronics when the percentage of training data size w.r.t. the overall data size increases from 50% to 90%. We observe that MRLR consistently outperforms other methods across all levels of data sparsity. Furthermore, the performance of MRLR with data sparsity at 60% is better than that of any of other methods with data sparsity at 90%. Such observations also hold in other datasets, showing that MRLR can achieve better performance even with high data sparsity.

Generalizability. To evaluate the generalizability of MRLR, we further collect data of Foursqure check-in performed over 3 weeks in 4 European capital cities (Amsterdam, London, Paris, Rome), published on Instagram (31,872 users perform 198,801 check-in at 41,387 locations that belong to 492 categories) and Twitter (18,522 users; 109,790 check-in; 38,855 locations; 482 categories). Figure 5.5 compares the performance of MRLR and the other methods. As in the previous setting, MRLR significantly outperforms (p-value < 0.01) the second best method MetaProd2Vec by 5.10% on 'All Users' and 5.62% on 'Cold Start'. These results demonstrate that the proposed MRLR framework can be effective in multiple recommendation tasks.

Complexity Validation. To verify the conclusion made in Complexity Analysis part that the overall computational time is linear with respect to the number of observations in the rating matrix ($|\mathbf{R}|$), we test the runtime of MRLR model and the results are shown

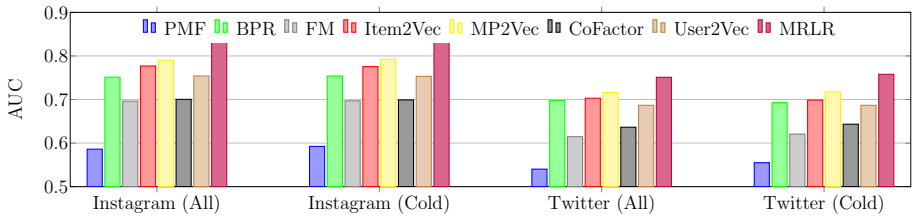


Figure 5.5: Comparative results on Instagram and Twitter for MRLR

in Figure 5.6. It depicts the relationships between the runtime and the size of training data. We randomly select $x\%$ as training data and the rest $(1 - x\%)$ as test data where x is scaled from 10 to 90 with step 10. We observe that with the increase of the training data, the runtime linearly goes up. Note that we only show the results of Clothing and Electronics, which possess the smallest and largest data among the four datasets. Similar observations can be noted for all the other datasets. In conclusion, MRLR is efficient to scale to very large data sets.

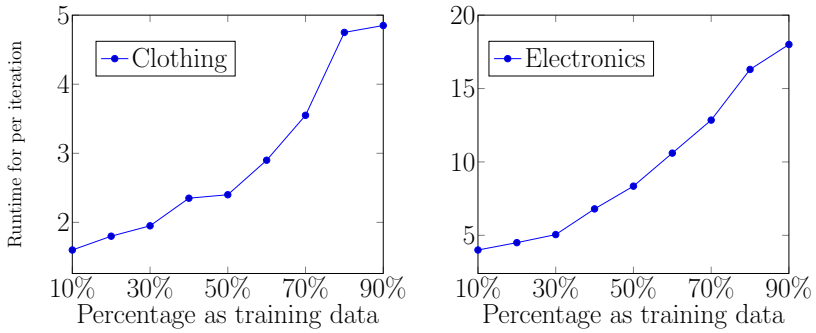


Figure 5.6: Runtime (seconds/iteration) of MRLR on Clothing and Electronics.

5.4 Summary

Representation learning (RL) has drawn much attention in recommendation, due to its effectiveness in capturing local item relationships. However, all existing RL based recommendation models either neglect personalization or overlook multi-level organizations of items for fine-grained item relationships. Therefore, this chapter proposes a multi-level

RL framework for personalized recommendation – MRLR, which learns user and item embeddings from a multi-level item organization for better recommendation. MRLR, therefore, benefits from RL as well as achieves the goal of personalized recommendation. Empirical validation on multiple real-world datasets shows that MRLR significantly outperforms state-of-the-art algorithms.

Chapter 6

RKGE: Recommendation with Knowledge Graph Embedding

In Chapters 3, 4 and 5, we focus on incorporating category hierarchy into latent factor model or representation learning model to help resolve *data sparsity* and *cold start* problems in recommendation, so as to achieve high quality recommendations. With the development of semantic web, the knowledge graph (KG), as an auxiliary data source, has proven to be effective in uncovering fine-grained item relationships by providing heterogeneous information related to items, i.e., different types of entities and relations, thus facilitating to infer user preferences towards items from different angles. While existing recommendation methods mainly rely on heavy and tedious feature engineering processes to manually extract features from the KG.

In this chapter, we propose RKGE, a KG embedding recommendation approach based on a novel recurrent network architecture that automatically learns semantic representations of entities and paths for effective recommendation. In particular, to learn the relations between a pair of entities, RKGE first mines all the paths linking paired entities which carry different semantics, in an automatic fashion. It then encodes all paths between the entity pair through a batch of RNNs to learn the semantic representations of entities and paths, and seamlessly integrates them into recommendation. Furthermore, it

Table 6.1: Mathematical Notations for RKGE

Notations	Descriptions
$\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$	user set
$\mathcal{I} = \{v_1, v_2, \ldots, v_n\}$	item set
$\mathbf{R} \in \mathbb{R}^{m \times n}$	user-item rating matrix
$r_{ij}, \tilde{r}_{ij}$	u_i's observed/estimated rating to v_j
$\mathcal{G} = (\mathcal{E}, \mathcal{L})$	the knowledge graph
$\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$	entity set
$\mathcal{R} = \{r_1, r_2, \ldots, r_g\}$	entity relation set
$\mathcal{P}(e_i, e_j) = \{p_1, p_2, \ldots p_h\}$	paths between e_i and e_j
$\mathbf{u}_i, \mathbf{v}_j, \mathbf{p}_l$	embeddings for u_i, v_j, p_l
$\mathbf{W}, \mathbf{H}$	transformation parameters of LSTM
$\mathbf{h}_{lt}$	hidden state for path p_l at step t
$\mathbf{c}_{lt}$	cell state for path p_l at step t
$\mathcal{J}$	loss function

employs a pooling operator to discriminate the importances of different paths in characterizing user preferences over items. Experimental results on multiple real-world datasets demonstrate the superiority of RKGE against state-of-the-art algorithms. In addition, we show that RKGE provides meaningful explanations for the recommendation results.

The rest of this chapter is organized as follows: We first formalize the recommendation problem in Section 6.1. Section 6.2 proposes the RKGE recommendation framework, which is composed of four components, i.e., semantic path mining, recurrent network architecture, saliency determination and recommendation generation. We then present the experimental results in Section 6.3, followed by the conclusion in Section 6.4.

6.1 Problem Formulation

This section formalizes the recommendation problem investigated in this chapter. Suppose we have m users $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, and n items $\mathcal{I} = \{v_1, v_2, \ldots, v_n\}$. The binary

matrix $\mathbf{R} \in \mathbb{R}^{m \times n}$ records the historical user-item interactions, with the $\mathbf{R}_{ij} = 1$ indicating that u_i prefers v_j; otherwise 0. We use *entity* as a generic term to represent all the objects (e.g., user, item, genre, actor) that can be mapped into a KG, denoted as $\mathcal{G}$. The definition of KG is given as below. The sets of entities and entity relations are denoted by $\mathcal{E} = \{e_1, e_2, \ldots, e_k\}$ and $\mathcal{R} = \{r_1, r_2, \ldots, r_g\}$, respectively. $\mathcal{P}(e_i, e_j) = \{p_1, p_2, \ldots p_h\}$ represents the set of connected paths between entities e_i and e_j. Table 6.1 summarizes all the notations utilized in this chapter.

Definition 7 ***Knowledge Graph.*** *KG is defined as a directed graph $\mathcal{G} = (\mathcal{E}, \mathcal{L})$ with an entity type mapping function $\phi : \mathcal{E} \rightarrow \mathcal{A}$ and a link type mapping function $\psi : \mathcal{L} \rightarrow \mathcal{R}$. Each entity $e \in \mathcal{E}$ belongs to an entity type $\phi(e) \in \mathcal{A}$, and each link $l \in \mathcal{L}$ belongs to a relation type $\psi(l) \in \mathcal{R}$.*

The KG investigated in this study can be considered as a heterogeneous information network, as there are more than one types of entities and entity relations included, i.e., $|\mathcal{A}| > 1$ and/or $|\mathcal{R}| > 1$. Figure 1.2 provides a toy example for KG in the movie domain, where entities include user, movie and the corresponding attributes (e.g., genre, actor and director); links describe the relations between entities (e.g., "rating" behavior and "acting" behavior). Without loss of generality, in our study we also includes historical user-item interaction record in the KG. Given the KG as input, our goal is to exploit its heterogeneous information to help learn high quality representations of both users and items, which are then used for generating better recommendations. The extracted representations are expected to fully capture the semantic meanings of entities and entity relations encoded in KG. To achieve this goal, we propose the recurrent knowledge graph embedding framework (RKGE), which will be elaborated in the next section.

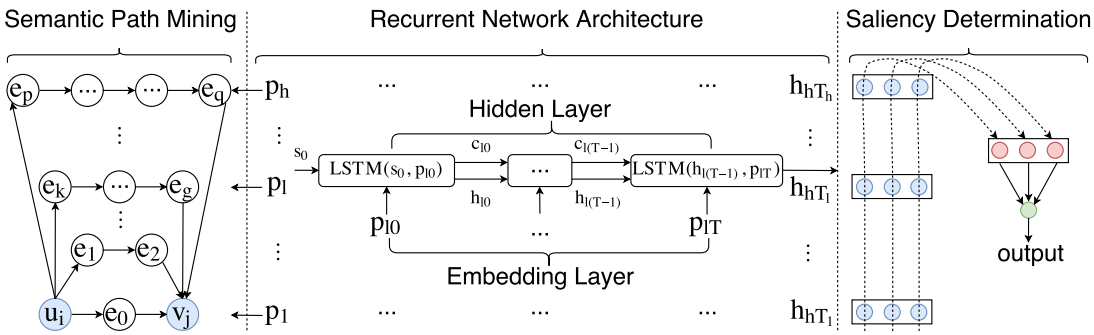


Figure 6.1: The framework of RKGE

6.2 The RKGE Framework

This section presents the details of our proposed Recurrent Knowledge Graph Embedding based framework (RKGE), which aims at leveraging the KG to help achieve better recommendation performance, followed by the end-to-end model learning process. The overall framework of RKGE is illustrated by Figure 6.1. It consists of four modules: 1) semantic path mining, to efficiently discover paths between paired entities; 2) recurrent network architecture, to seamlessly encode semantic paths via a batch of RNNs; 3) saliency determination, to automatically distinguish different path saliency through a pooling operation; and 4) recommendation generation, to provide each user a ranked list of items she will be interested in. Note that Figure 6.1 only describes the case of a user-item pair. We do not show the recommendation generation module as it involves multiple items.

6.2.1 Semantic Path Mining

To fully exploit the entity relations encoded in KG, we first mine paths with different semantics between entities, which are then seamlessly incorporated into a novel recurrent network architecture for effective recommendation.

Due to the large volume and complexity of KG, there can be a large number of connected paths between two entities that may contain different entity types and relation

types in different orders and with various lengths. It is unrealistic to make use of all the paths to model entity relations on account of high computational cost. Besides, the utilization of some paths that do not carry sufficient semantics may bring in much noise, thus degenerating recommendation accuracy. Two strategies are therefore devised to help select salient paths between entities, so as to increase the efficiency as well as to reduce the time complexity of the proposed framework:

Strategy 1 *We only take into account user-to-item paths $\mathcal{P}(u_i, v_j)$ that connect user u_i with all items that u_i has interacted with, i.e., $\{v_j | r_{ij} > 0\}$. These paths are most helpful for effective recommendation given our goal to recommend items to users. Moreover, they further include those relevant item-to-item and user-to-user paths as subsequences of the user to item paths.*

Strategy 2 *We enumerate paths with a length constraint, i.e., only the paths with length less than a threshold are used by our framework. As pointed out by Sun et al. [98], paths with relatively short length are good enough to model entity relations, whereas longer paths may bring in remote neighbors and lose semantic meanings, thus introducing much noise.*

We will investigate how lengths of paths can affect recommendation performance in our new context of knowledge embedding based approach and show in our experiment that similar results also hold. Guided by the above two strategies, RKGE can mine the qualified paths with different semantics that connect the entity pairs (i.e., user-item) in an automatic fashion, instead of manually designed and extracted features (e.g., meta paths) from KG. These paths will be further processed by recurrent networks to automatically learn their semantic representations for recommendation, as will be introduced next.

6.2.2 Recurrent Network Architecture

By regarding the directed path between user and item as a sequence, where entities in the path correspond to the elements in the sequence, we naturally consider employing

RNN to encode the path with various lengths, so as to accurately model the semantic relations and to achieve better user and item representations. This is mainly attributed to two essential reasons: 1) RNN is effective in modeling sequences with varying lengths; 2) RNN is capable of modeling the semantics of not only entities but also the entire paths between entity pairs.

Given that multiple paths may connect two entities, we devise a novel recurrent network architecture to capture all possible relations between any two entities. The recurrent network architecture comprises a batch of RNNs, with each single RNN learning the semantic representation of an individual path.

Formally, for an entity pair (u_i, v_j) in KG, assume there are h connected directed paths of different lengths, i.e., $\mathcal{P}(u_i, v_j) = \{p_1, p_2, \cdots, p_h\}$. Given as input an arbitrary path p_l with length T in the format of $p_l = e_0 \xrightarrow{r_1} e_1 \xrightarrow{r_2} e_2 \cdots \xrightarrow{r_T} e_T$ with $e_0 = u_i, e_T = v_j$, RNN encodes the entire path by learning a semantic representation for each entity and learning one single representation for the whole path. These goals are achieved by two layers, respectively the embedding layer and the hidden layer, as illustrated in Figure 6.1.

Embedding Layer. For each entity e_k in $\mathcal{P}(u_i, v_j)$, the embedding layer learns a distributed representation p_{lt} that maps e_t to a low-dimensional vector, with each element of the vector representing the affinity of this entity to a latent topic, thus capturing the semantic meaning of the entity. Overall, given the full path as input, the embedding layer will learn an embedded representation of the path as $\mathbf{p}_l = \{\mathbf{p}_{l0}, \mathbf{p}_{l1}, \mathbf{p}_{l2}, \cdots, \mathbf{p}_{lT}\}$, in which each element denotes the representation (embedding) of the corresponding entity in p_l. This new representation will then be fed as input to the hidden layer to learn a single representation that encodes the entire path.

Hidden Layer. To learn the path representation, the hidden layer considers both the embeddings of entities in the path and the order of these entities. It takes a flow-based approach to encode the sequence from the beginning entity of the path e_0 to the ending

entity e_T: in each step t, it learns a hidden state h_{lt} that encodes the subsequence from e_0 to e_t, which is then used as input together with the embedding of e_{t+1} (i.e., $p_{l(t+1)}$) to learn the hidden state of the next time step, i.e., $h_{l(t+1)}$. The final state h_{lT} will encode the entire path, thus is used as the representation of the path.

In order to better control the information flows through the path, we adopt Long Short-Term Memory (LSTM) [37], which has gained much popularity among different RNN models due to its strong capability in dealing with the gradient vanishing problem [126], making it effective in preserving historical information in a sequence. This is achieved by the two types of *gates* used by LSTM: the forget gate and the input gate.

LSTM first decides what information of path p_l to throw away from the previous cell state $\mathbf{c}_{l(t-1)}$, which is achieved by a sigmoid layer called the "forget gate layer", given by:

$$f_{lt} = \sigma\left(\mathbf{W}_f \cdot \mathbf{h}_{l(t-1)} + \mathbf{H}_f \cdot \mathbf{p}_{lt} + b_f\right) \tag{Eq. 6.1}$$

where f_{lt} ranges from 0 to 1, and 0 means completely get rid of the state while 1 represents entirely keep the state; $\mathbf{W}_f$ and $\mathbf{H}_f$ are respectively the linear transformation parameters for the previous and current steps; b_f is the bias term; $\mathbf{h}_{l(t-1)}$ is the hidden state in the previous step of path p_l, and $\mathbf{p}_{lt}$ is the representation of the current step, i.e., the t^{th} entity, for path p_l,

LSTM then determines what new information of path p_l to store in the cell state, which consists of two parts: a sigmoid layer called "input gate layer" to decide which values to update; a tanh layer to create a vector of new candidate values $\tilde{\mathbf{c}}_{lt}$, given as:

$$\begin{aligned} i_{lt} &= \sigma\left(\mathbf{W}_i \cdot \mathbf{h}_{l(t-1)} + \mathbf{H}_i \cdot \mathbf{p}_{lt} + b_i\right) \\ \tilde{\mathbf{c}}_{lt} &= \tanh\left(\mathbf{W}_c \cdot \mathbf{h}_{l(t-1)} + \mathbf{H}_c \cdot \mathbf{p}_{lt} + b_c\right) \end{aligned} \tag{Eq. 6.2}$$

We proceed to update the old cell state $\mathbf{c}_{l(t-1)}$ into the new cell state $\mathbf{c}_{lt}$ by balancing the previous and current information of path p_l via the gates, defined by,

$$\mathbf{c}_{lt} = f_{lt} \cdot \mathbf{c}_{l(t-1)} + i_{lt} \cdot \tilde{\mathbf{c}}_{lt} \tag{Eq. 6.3}$$

Finally, the output is also based on a sigmoid layer that decides which parts of the cell state to output, formulated by:

$$o_{lt} = \sigma\left(\mathbf{W}_o \cdot \mathbf{h}_{l(t-1)} + \mathbf{H}_o \cdot \mathbf{p}_{lt} + b_o\right) \quad \text{(Eq. 6.4)}$$

The hidden state of path p_l at step t is thus defined by,

$$\mathbf{h}_{lt} = o_{lt} \cdot \tanh(\mathbf{c}_{lt}) \quad \text{(Eq. 6.5)}$$

After incorporating each of the qualified paths (the total number is h) between u_i and v_j into the corresponding LSTM model, we obtain the hidden representations of the paths, i.e., the representations of the entity relations of u_i and v_j. These hidden states are then utilized for learning better representations for u_i, v_j.

6.2.3 Saliency Determination

As there are h paths linking u_i and v_j, different paths have different lengths and carry different semantic meanings. This plays different roles in modeling the relations between u_i and v_j. Previous work has shown that the paths with shorter lengths have more important influences than the longer ones, as the shorter paths usually indicate a stronger connectivity with clearer semantics, whereas the longer ones suggest a weaker proximity [98]. This however may not always hold in the real situations. Even the paths with the same lengths may have different impacts on the relations of u_i and v_j. Therefore, we design a data-driven method via conducting a pooling operation [56] to help distinguish the different path importances.

Generally, the pooling operation is used to combine vectors resulting from different windows into a single vector with the same dimension, by taking the max value observed in each dimension of the vectors over the different windows. The intention is to focus on the most important "features" of these vectors, which is just particularly suitable for our case, i.e., determining the path saliency on modeling entity relations.

For the h connected paths in $\mathcal{P}(u_i, v_j) = \{p_1, p_2, \cdots, p_h\}$, we denote their last hidden states by the recurrent network architecture as $\mathbf{h}_{1T_1}, \mathbf{h}_{2T_2}, \cdots, \mathbf{h}_{hT_h}$, where T_h is the last step of p_h as well as the length of p_h. Based on this, we add a *max pooling* layer, resulting in an aggregated hidden state $\mathbf{h}$,

$$\mathbf{h}[j] = \max_{1 \leq i \leq h} \mathbf{h}_{iT_i}[j] \tag{Eq. 6.6}$$

where $\mathbf{h}[j]$ is the value of the j^{th} dimension of $\mathbf{h}$. From the above equation we can see that the effect of the max-pooling operation is to get the most salient feature across all the connected paths. The hidden state $\mathbf{h}$ is thus an aggregated representation of the paths in which each dimension reflects the most salient information. Furthermore, to avoid the aggregated hidden state $\mathbf{h}$ being dominated by a certain $\mathbf{h}_{iT_i}$, e.g., a single path in $\mathcal{P}(u_i, v_j)$, we also perform an *average pooling* operation [8] towards the last hidden states of all the paths, where the average values in each dimension are retained instead of the maximum one. Their respective performance is evaluated in the experiments section.

6.2.4 Recommendation Generation

Through the pooling operation, we obtain a final hidden state of all the paths between u_i and v_j, i.e., the aggregated effects of paths on the relations of u_i and v_j. To further precisely quantify the relation (proximity) of u_i and v_j, i.e., $\tilde{r}_{ij}$, we adopt a fully-connected layer after the pooling layer, given by,

$$\tilde{r}_{ij} = f(\mathbf{h}) = \sigma\left(\mathbf{W}_r \cdot \mathbf{h} + b_r\right) \tag{Eq. 6.7}$$

where $\mathbf{W}_r$ is regression coefficient and b_r is the bias term. We adopt a sigmoid function $\sigma(\cdot)$ to control the range of $f(\mathbf{h})$ into $[0, 1]$.

Once the model training process is finished, better representations of users and items can be achieved through encoding the connected paths between them by RKGE. Thus,

Algorithm 5: RKGE Optimization

Input: rating matrix $\mathbf{R}$, the knowledge graph $\mathcal{G}$, in_dim, hidden_dim, max_path_length, γ, Iter

Initialize the embeddings of $e \in \mathcal{G}$ with small values;
`// Semantic Path Mining`
Built the graph $\mathcal{G}$ with Python Networkx;
foreach $u_i \in \mathcal{U}$ **do**
    Based on $\mathbf{R}$, get positive pairs (u_i, v_j);
    Randomly sample to generate negative pairs (u_i, v_k);
    $(u, v) \leftarrow (u_i, v_j) \cup (u_i, v_k)$;
    foreach *(u, v) pair* **do**
        Mine connected paths $\mathcal{P}(u, v)$;
`// Recurrent Network Architecture`
for $t = 1; t \leq Iter; t++$ **do**
    foreach (u, v) *pair* **do**
        `// Recurrent Network Batch`
        for $p_l \in \mathcal{P}(u, v)$ **do**
            $\mathbf{h}_{lT_l} \leftarrow$ based on Equ. (1-4);
            Combine$(\mathbf{h}) \leftarrow \mathbf{h}_{lT_l}$;
        `// Saliency Determination`
        $\mathbf{h} \leftarrow pool\big(\text{Combine }(\mathbf{h})\big)$ based on Equ. (5);
        Calculate $\tilde{r}_{ij}$ based on Equ. (6);
    Update parameters by back propagation through time;

during the testing process, we calculate the proximity score of users and their unrated items via the inner product [86] of the corresponding embeddings, which is given by,

$$s(u_i, v_j) = \langle \mathbf{u}_i, \mathbf{v}_j \rangle \qquad \text{(Eq. 6.8)}$$

Finally, we rank the items based on the computed proximity score, and then recommend the top-K items with the highest score to u_i.

6.2.5 End-to-End Parameter Learning

Given the training data $\mathcal{D}_{\text{train}}$, which contains N instances in the form of $(u_i, v_j, r_{ij}, \mathcal{P}(u_i, v_j))$, RKGE learns the involved parameters by minimizing the following loss function:

$$\mathcal{J} = \frac{1}{N} \sum\nolimits_{r_{ij} \in \mathcal{D}_{\text{train}}} BCELoss(\tilde{r}_{ij}, r_{ij}) \qquad \text{(Eq. 6.9)}$$

where BCELoss(·) is the Binary Cross Entropy between the target and the predicted values, which is similar as the Cross Entropy for multiple classification problem. The details of BCELoss(·) can be found by the provided link[1]. Since all the modules and the above loss function are analytically differentiable, RKGE can be readily trained in an end-to-end manner. In the learning process, parameters are updated by the back propagation through time (BPTT) algorithm [113] in the recurrent layers of Recurrent Network Architecture Module, and by normal back-propagation in other parts. We adopt RMSprop [103] to adaptively update the learning rate, which has proven to be highly effective to train neural networks. To prevent over-fitting, dropout [96] is employed to randomly drop hidden units of the network in each iteration during the training process. Besides, we randomly sample unrated items for each user as the negative instances, the number of which is the same with her rated items. The paths connected the negative instances are also incorporated into RKGE to help balance the model learning process.

The detailed optimization process is described by Algorithm 5, which is mainly composed of two modules: the semantic path mining (lines 2-8) and the recurrent network architecture (lines 9-16) module, including the recurrent network batch (lines 11-13) and saliency determination (lines 14-15). The computational complexity of RKGE is acceptable. The training time for the two evaluation datasets, i.e., IM-1M and Yelp2013 is less than 2 hours. The details about the datasets are deferred to the experiments section.

6.3 Experimental Results

This section conducts an extensive empirical study on multiple real-world datasets to validate the effectiveness of the proposed RKGE framework. We also provide a detailed analysis according to the experimental results.

[1] `https://pytorch.org/docs/master/nn.html?highlight=bceloss#torch.nn.BCELoss`

6.3.1 Experimental Setup

Datasets. To demonstrate the effectiveness of our proposed recommendation framework RKGE, we adopt two widely used real-world datasets from different domains (movie and local business) for the empirical study. Here we do not employ the Amazon web store datasets as utilized in Chapters 3, 4 and 5, as the knowledge graph information about these datasets is unavailable.

- **IM-1M**: The first dataset is IM-1M, which is built by combing MovieLens 1M[2] and the corresponding IMDB[3] datasets. MovieLens 1M is a personalized movie rating dataset, which consists of 1M ratings (ranging from 1 to 5) with 6,040 users and 3,706 movies; IMDB contains movie auxiliary information, such as genre, actor, director, etc. The two datasets are linked by the titles and release dates of movies. After mapping the two datasets, we have 6,040 users and 3,382 movies, and 756,684 ratings in the final dataset.

- **Yelp2013**: The second dataset is Yelp, which is the Yelp Challenge Dataset[4] released by Yelp[5] and is available now at Kaggle[6]. This dataset contains user check-ins to local business, together with user reviews and local business information network (e.g., category, location). Yelp is much sparser than IM-1M, thus the performances of all the methods are expected to decline accordingly on Yelp.

We process the two datasets in accordance with literature [122, 12] as follows: if a user provides a rating towards a movie, or wrote a review for a business, we set the feedback as 1, otherwise it would be set to 0. We split the feedback of IM-1M in the order of their

[2]http://grouplens.org/datasets/movielens/
[3]http://www.imdb.com/
[4]https://www.yelp.com/dataset_challenge
[5]http://www.yelp.com/
[6]https://www.kaggle.com/c/yelp-recsys-2013/data

Table 6.2: Statistics of the datasets for RKGE

	Datasets	**IM-ML**	**Yelp2013**
User-item interaction	#Users	6,040	37,940
	#Items	3,382	11,516
	#Ratings	756,684	229,178
	Data Density	3.704%	0.0052%
Knowledge graph	#Entities	18,920	46,606
	#Entity Types	11	7
	#Links	800,261	302,937
	#Link Types	10	6
	Graph Density	0.447%	0.028%

timestamps, and use the older 80% of feedback as training data and the more recent 20% as test data. With Yelp we utilize the original training and test sets in the published version. The overall statistics of the two datasets are summarized in Table 6.2.

Comparison Methods. For the selection of comparison algorithms, we mainly focus on the KG based approaches and also include some baseline methods. We compare with the following state-of-the-art recommendation algorithms, 1) **MostPop**: recommends the most popular items to all users without personalization; 2) **BPRMF** [83]: Bayesian personalized ranking model based on matrix factorization (MF) [75], which is the basic latent factor model aiming at item ranking; 3) **LIBFM** [82]: factorization machine is a classic feature based latent factor model, to which we feed items' attributes in the knowledge graph as raw features; 4) **HeteRS** [79, 80]: the graph based recommendation approach integrating the knowledge graph via Markov chain; 5) **HeteRec** [122]: latent factor model by incorporating meta path for personalized recommendation; 6) **GraphLF** [12]: the knowledge graph based approach using Personalized PageRank to help infer user preferences through logic reasoning; 7) **CKE** [127]: the recently proposed state-of-the-art collaborative knowledge graph embedding based approach that learns better item latent representations with the help of the knowledge graph.

For fair comparison, we remove the textual and visual embedding modules from the original CKE model, due to the unavailability of such auxiliary information in the evaluation datasets. Furthermore, we take FM as the representative feature based latent factor model, as its performance is generally better than other counterparts, such as SVDFeature [16], collective matrix factorization (CMF) [94] and tensor factorization (TF) [48]. We do not compare with other previous meta path based approaches (e.g., HeteCF [64], SimMF [90]), since they are generally outperformed by both HeteRec and GraphLF.

Evaluation Metrics. By following state-of-the-art works [122, 12] leveraging the KG for effective recommendation, we evaluate the performance of all the approaches using the standard metrics, including Precision at $N = \{1, 5, 10\}$, i.e., $Prec@N$, and the top-10 Mean Reciprocal Rank [105].

Parameter Settings. We empirically find out the optimal parameter settings for each comparison method. For all the methods, we apply a grid search in $\{5, 10, 20, 50, 100, 200\}$ to find the optimal settings for the dimension of the latent factor d. A grid search in $\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$ is applied for the learning rate and regularization coefficient (including the 1/2-way regularization of FM). For HeteRS, we set $\beta = 10^4$ on the two datasets. For HeteRec and CKE, a grid search in $\{5, 10, 20, 50, 100, 200\}$ is applied to find out the best settings for the number of negative samples; other parameters in these methods are set as suggested by the original papers. For RKGE, the number of hidden units is set to 16 and 32 on IM-ML and Yelp2013, respectively, which is selected from the option set $\{8, 16, 32, 64, 128\}$ based on a held-out validation set. To avoid potential over-fitting, the dropout value is validated from the option set $\{0.00, 0.25, 0.50\}$.

6.3.2 Impacts of Parameters

Impacts of Different Path Lengths. Paths with different lengths capture different semantic meanings, which help infer user preferences from different perspectives and

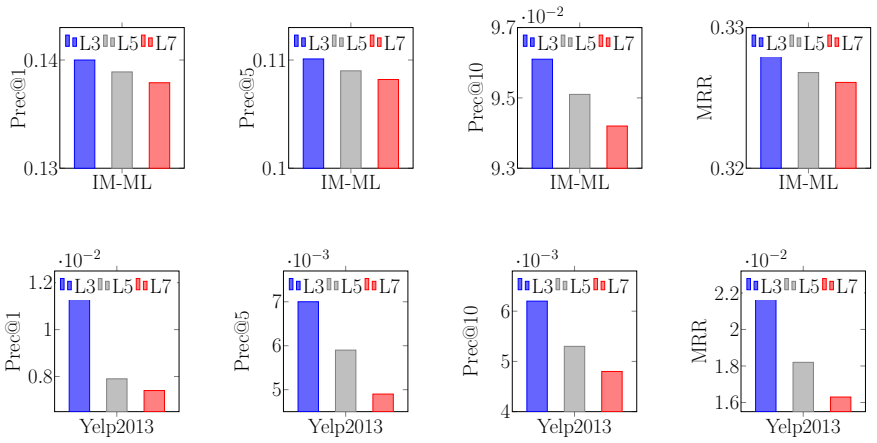


Figure 6.2: The impacts of different path lengths on RKGE

allow to generate different recommendations. Our hypothesis is that paths with relatively short lengths are more beneficial for modeling entity relation as they carry clearer and interpretable semantic meanings. This has been verified in meta-path based method [98]. Here we study if the same result holds when applying our KG embedding based approach. To empirically study the impact of path length on recommendation accuracy, we incorporate paths with different lengths into the proposed RKGE model. We test with lengths L = {3, 5, 7}, as RKGE is aimed at modeling indirect paths between users and items, and these paths are of odd lengths since items can only be indirectly linked via their attribute entities. Figure 6.2 depicts the results on the two datasets. From the results, we can observe that as the path lengths increase, the performance (including Pre@1, 5, 10 and MRR) of RKGE decreases gradually on both datasets. This verifies our intuition and confirms previous findings in the new context of KG embedding based approach.

Impacts of Different Pooling Operations. To determine the saliency of different paths between two entities, pooling operations are adopted in the proposed RKGE model; max-pooling and average-pooling are the most widely utilized operations. Max-pooling focuses on the most important paths, whereas the average-pooling aims at aggregating

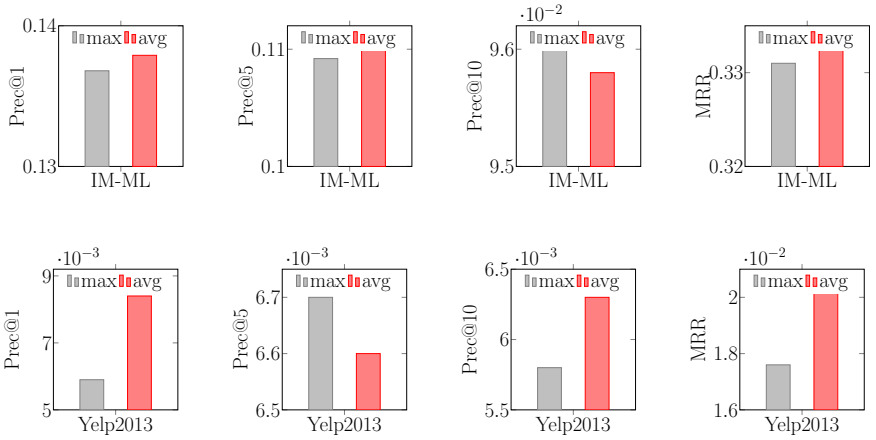


Figure 6.3: The impacts of different pooling strategies on RKGE

the impacts of all paths. Hence, average-pooling avoids the results being dominated by a certain path. To study their respective potential for recommendation, we design RKGE with different pooling versions. Figure 6.3 illustrates their performance on the two datasets, from which we can note that the performance of average-pooling is generally better than that of max-pooling. This supports the intuition that user preferences towards items are determined by combinations of heterogeneous factors and highlights the importance of a method that can make full use of these factors.

Interpretability of RKGE on Recommendation. By fully capturing the semantics of entities and entity relations encoded in the KG, RKGE can not only provide effective recommendations, but also possess a better interpretability for user-item interactions. To verify this, for each user, we first map her rated items (i.e., items in the training data) and the correctly recommended items (i.e., the interaction between items in her top-10 recommendation list and test data) into the KG, and then check whether there are semantic paths linking those items. For conciseness, we do not show the results for all the users. Figure 6.4 displays the results of a randomly sampled user, say Bob, on the IM-ML dataset. Note that similar observations can be also obtained for other users on the other two datasets.

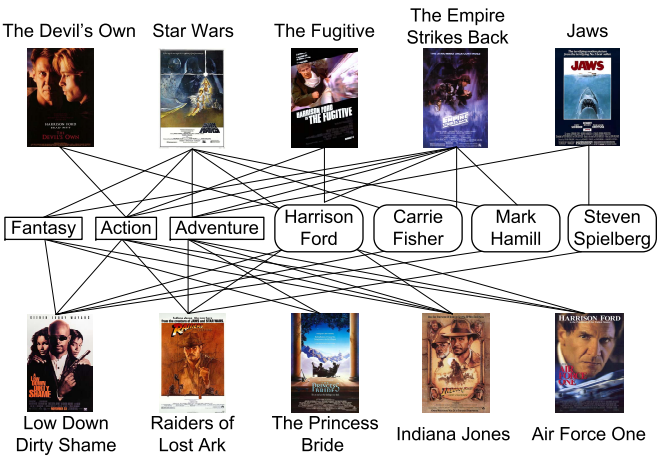


Figure 6.4: The interpretability of RKGE on the recommendation

In Figure 6.4, the items above are Bob's rated items, whereas the items below are the correctly recommended ones. For simplicity, we only keep four types of entities in the KG, i.e., movie, genre, actor and director. Several interesting findings are obtained. First, the correctly recommended movies are all connected to Bob's rated movies either by genre (e.g., "Low Down Dirty Shame" – "Star Wars"), actors (e.g., "Air Force One" – "The Devil's Own"), or directors (e.g., "Raiders of Lost Ark" – "Jaws"). This suggests that RKGE can well infer Bob's specific preference from different perspectives with the help of KG, based on which generates correct recommendations. Second, most of the correctly recommended movies are connected with rated movies by different types of paths, instead of a single one. For example, "Raiders of Lost Ark" is connected with "Jaws" by "Adventure" and "Steven Spielberg"; "Air Force One" links with "Star Wars" by "Action", "Adventure" and "Harrison Ford". This implies that user-item interactions are co-influenced by different paths, possibly with different degrees, and that their joint effects can be effectively captured by RKGE, allowing it to provide recommendations with high interpretability.

Table 6.3: Performance of all comparison methods on the two real-world datasets. The best performance is boldfaced; and the runner up is labeled with '*'; The column 'Improve' indicates the relative improvements that our proposed approach RKGE achieves w.r.t. the best performance of methods proposed by others

View	Datasets	Metrics	MostPop	BPRMF	LIBFM	NCF	HeteRS	HeteRec	GraphLF	CKE	RKGE	Improve
All Users	IM-1M	Prec@1	0.0118	0.0409	0.0459	0.0450	0.0689	0.0764	0.1069*	0.0954	**0.1396**	30.58%
		Prec@5	0.0064	0.0438	0.0525	0.0482	0.0528	0.0579	0.0360	0.0781*	**0.1092**	39.82%
		Prec@10	0.0081	0.0441	0.0456	0.0485	0.0475	0.0488	0.0581	0.0682*	**0.0861**	26.25%
		MRR	0.0245	0.1234	0.1412	0.1360	0.1600	0.1737	0.1524	0.2440*	**0.3056**	25.25%
	Yelp	Prec@1	0.0003	0.0051	0.0054	0.0056	0.0047	0.0072	0.0083	0.0084*	**0.0113**	34.52%
		Prec@5	0.0007	0.0058	0.0059	0.0060*	0.0052	0.0050	0.0054	0.0057	**0.0070**	16.67%
		Prec@10	0.0004	0.0052	0.0054	0.0055	0.0031	0.0039	0.0056*	0.0053	**0.0062**	10.71%
		MRR	0.0010	0.0162	0.0167	0.0178	0.0116	0.0151	0.0189*	0.0178	**0.0218**	15.34%
Cold Start	IM-1M	Prec@1	0.0028	0.0171	0.0330	0.0188	0.0405	0.0573	0.0677	0.0687*	**0.0809**	17.76%
		Prec@5	0.0017	0.0191	0.0203	0.0210	0.0428	0.0428	0.0267	0.0432*	**0.0481**	11.34%
		Prec@10	0.0013	0.0205	0.0273	0.0225	0.0370	0.0402	0.0422*	0.0372	**0.0467**	10.66%
		MRR	0.0050	0.0438	0.0457	0.0481	0.1239	0.1355	0.1188	0.1408*	**0.1521**	8.03%
	Yelp	Prec@1	0.0004	0.0028	0.0042	0.0031	0.0037	0.0053	0.0061	**0.0067**	0.0061*	-8.96%
		Prec@5	0.0003	0.0037	0.0040	0.0041	0.0035	0.0035	0.0038	0.0052*	**0.0056**	7.69%
		Prec@10	0.0003	0.0031	0.0039	0.0034	0.0031	0.0032	0.0047*	0.0043	**0.0055**	17.02%
		MRR	0.0006	0.0098	0.0104	0.0108	0.0097	0.0113	0.0141	**0.0151**	0.0149*	-1.32%

6.3.3 Comparative Results

Table 6.3 summarizes the performance of all comparison methods across the two real-world datasets, where two views are created for each dataset: 'All Users' indicates all the users are considered in the test data; while 'Cold Start' indicates only users with less than 5 ratings are involved in the test data. A number of interesting observations can be noted from the results for the two views.

Performance on All Users. As the basic recommendation approaches considering no auxiliary information, MostPop and BPRMF perform worse than other methods. This helps confirm the usefulness of KGs for recommendation. We further observe that BPRMF highly outperforms MostPop, which is mainly because BPRMF is a personalized recommendation method via learning individual user's preference, whereas MostPop is a simple and non-personalized one.

By incorporating item attributes in the KG as raw features, LIBFM performs better than BPRMF, sometimes even better than existing KG based methods (e.g., HeteRS, HeteRec on Yelp), suggesting its superiority in utilizing auxiliary information for effective recommendation. Despite this, LIBFM models entity interactions in a linear fashion, thus is intrinsically limited by its expressive power for capturing complex patterns. Well-designed neural network based methods are more capable of modeling complex entity relations, as shown by the performance of NCF. Although merely considering user-item interaction data, NCF sometimes even performs better than LIBFM, demonstrating the effectiveness of neural architecture.

In terms of the methods specifically designed for KGs, HeteRec outperforms HeteRS. The possible reason is that HeteRS is a graph based method built upon random walk, thus failing to explicitly capture the semantics of entities and entity relations encoded in KG, whereas HeteRec is a latent factor model based approach which exploits the relation

heterogeneity in the KG by introducing meta paths. This verifies that semantic paths in KG indeed facilitate to generate effective recommendation. GraphLF is also based on random walk, yet it combines the strength of latent factorization and logic reasoning, thus achieving better performance than HeteRS and HeteRec. By learning item semantic representations from KGs, CKE generally performs the best among the four existing KG based methods (i.e., HeteRS, HeteRec, GraphLF and CKE), implying the effectiveness of network embedding for better recommendation. However, it ignores the relations of paired entities linked by paths, thus failing to capture the full semantics encoded by KGs.

Overall, when compared with all the other comparison methods across the two datasets, our proposed RKGE consistently achieves the best performance. The improvements w.r.t. Precision and MRR are 26.42%, 20.30% on average, respectively (Paired t-test, p-value < 0.01). This implies that the recommendation performance can be further boosted by appropriately combining the strengths of network embedding and semantic path mining on KGs.

Performance on Cold Start. Similar observations with "All Users" can be seen on "Cold Start". As in the previous setting, RKGE significantly outperforms (p-value $<$ 0.01) the best existing method by 9.25% and 3.36% for Precision and MRR on IM-1M, respectively. While on Yelp, the case is slightly different. The performance of RKGE is worse than CKE on some metrics (e.g., Prec@1, MRR). This is possibly due to the extremely low graph density of Yelp compared with IM-1M, leading to insufficient semantic paths for RKGE to take advantage of.

We further analyze the robustness of the compared methods for cold start recommendation. We observe that most methods are vulnerable to cold start users. This is because these methods learn user preference based on historical interactions, which contains very limited information for cold start users. Interestingly, CKE and RKGE consistently outperform other methods, implying the robustness of KG embedding for cold start users.

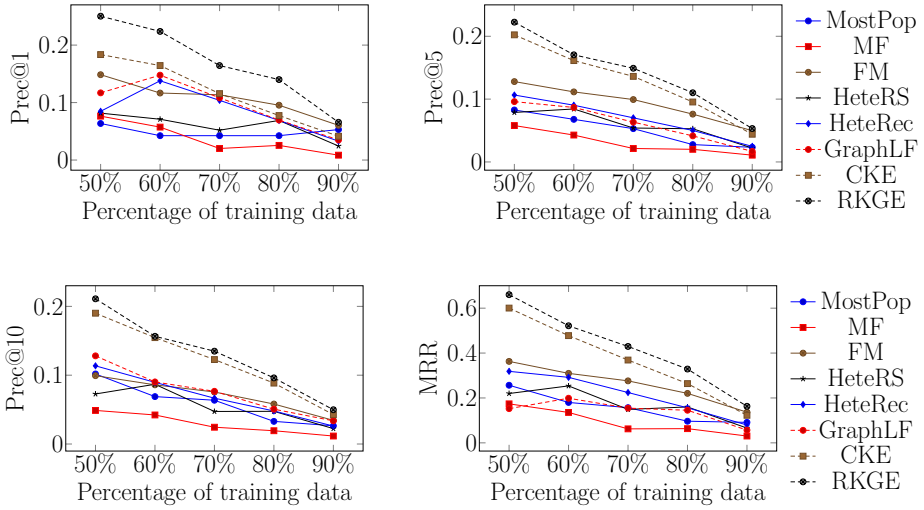


Figure 6.5: Impacts of data sparsity on RKGE for IM-1M

RKGE further outperforms CKE, showing that the usage of path semantics by RKGE can effectively capture users' preferences even from their limited historical interaction records. Besides, we also notice that the improvement ratios of RKGE on "All Users" are larger than those on "Cold Start". This could be explained by the fact that different from other cold-start cases where cold-start users possess similar amount of external information as warm-start users, in recommendation with KGs, cold-start users have very limited number of paths linking entities in KGs. Therefore, the recommendation with KGs for cold-start users is more difficult.

Performance on Data Sparsity. We further investigate the impact of data sparsity on the recommendation performance. Figures 6.5 and 6.6 depict the variation of performance for all methods when the percentage of training data size w.r.t. the overall data size increases from 50% to 90% with step 10% on the two datasets, respectively.

As the percentage of training data increases, the ranking performance for all the methods across the two datasets decreases gradually. The models can be well trained with more training data available, thus the recommendation performance should be improved.

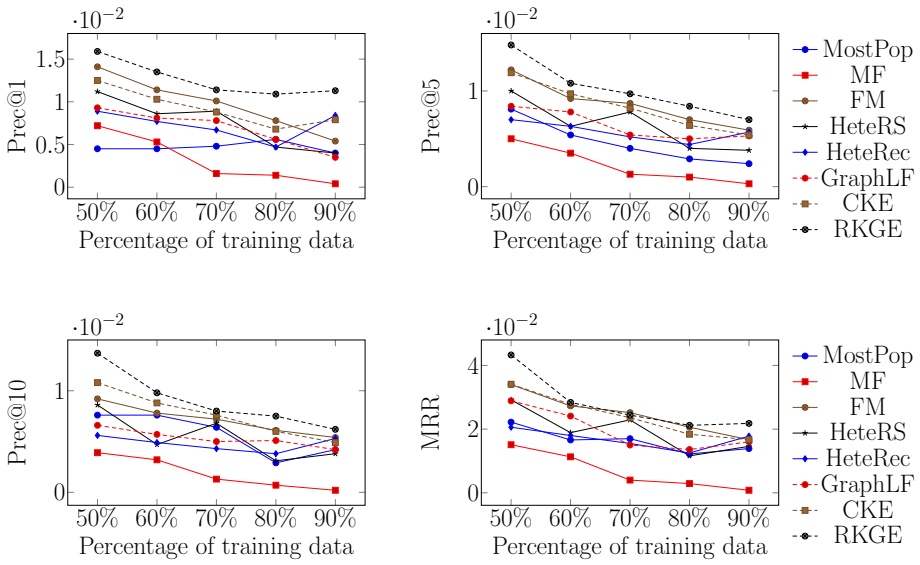


Figure 6.6: Impacts of data sparsity on RKGE for Yelp

This is also held by the scenario when doing rating prediction [102]. As more training data comes, the prediction error (e.g., Mean Absolute Error) goes down. However, for the item ranking problem, the case is a little bit different. One major possible explanation for this phenomenon is that although the ranking model might be well trained with more training data, the test data decreases correspondingly. Therefore, the possibility that the right items are hit from the test data also goes down. Another reason to explain the performance variation of the KG based methods might be that with the help of auxiliary information in the KG, they can achieve a better recommendation performance even with a higher data sparsity. However, as the training data becomes denser, the user behavior pattern can be better reflected by the historical data, and the auxiliary information in KG may introduce some noise, thus deteriorating the accuracy of recommendation.

Despite of the performance variation, our proposed approach RKGE consistently outperforms the other comparison methods across all levels of data sparsity on the two datasets. The improvements are statistically significant (Paired t-test, p-value < 0.01). This implies that RKGE has higher capability in coping with the data sparsity problem

compared to the state-of-the-art methods.

6.4 Summary

The knowledge graph has recently attracted a considerable amount of interest from the recommendation community due to its help in dealing with *data sparsity* and *cold start* problems. This chapter proposes a knowledge graph embedding recommendation framework – RKGE – based on a novel recurrent network architecture for high quality recommendation. RKGE can not only learn the semantic representation of different types of entities, but also automatically capture entity relations encoded in the knowledge graph. Extensive validation on two real-world datasets demonstrates the superiority of RKGE against other state-of-the-art recommendation algorithms.

Chapter 7

Conclusions and Future Work

In this chapter, we first summarize the contributions of this dissertation, and then point out several promising directions for future work.

7.1 Summary of Contributions

Recommendation is a fundamental task to enable personalized information filtering, thus to mitigate the *information overload* problem. The goal is to learn user preferences from historical user-item interactions, based on which recommend relevant items. Traditional recommendation technique, i.e., collaborative filtering (CF), however, inherently suffer from data sparsity and cold start problems. Therefore, in this dissertation, we aim to exploit item relationships to alleviate the concerned issues of CF, and investigate three specific recommendation problems.

In real applications, items are often organized by category, which enables category to become a quite prevalent way to define item relationships. Item category has proven to be helpful in generating effective recommendations, as users tend to have similar preferences towards items that belong to the same category. Existing work mainly focuses on categories that are organized in a flat structure, where categories are independent and in a same level. In fact, category can be often organized in a richer knowledge structure - category hierarchy (CH), to reflect the inherent correlations among different categories.

In Chapter 3, we therefore investigate the problem of category hierarchy based recommendation. To better employ category *affiliatedTo* relation in the vertical dimension of CH, we propose a novel recommendation framework ReMF, that integrates recursive regularization into matrix factorization model to better learn user and item latent factors. ReMF not only models the co-influence of hierarchically organized categories on user-item interactions, but also learns the strength of such co-influence from historical user-item interaction data, thus to improve recommendation performance. We evaluate the proposed method on multiple real-world datasets, and the experimental results show that ReMF significantly outperforms a number of baselines.

Chapter 4 goes deeper into the category hierarchy based recommendation problem. Based on Chapter 3, we further consider category relations in horizontal dimension of CH, i.e., *alternative* and *complementary*, together with category *affiliatedTo* relation in vertical dimension. The result is a unified matrix factorization based recommendation framework HieVH that seamlessly fuses both dimensions of CH, to help achieve better recommendation performance. Specifically, HieVH adapts latent factors of items by adding weighted aggregation of their affiliated categories latent factors, to better model item latent factors. The weights are automatically learnt from data. Horizontally, category relations, i.e., *alternative* and *complementary*, are incorporated as regularizers at each layer of the hierarchy, to better model category latent factors. In doing so, through the adaption of item latent factors with category latent vectors in vertical dimension, category relations in horizontal dimension can be inherited by items. Extensive validation demonstrates the superiority of HieVH against the state-of-the-art. An additional benefit of HieVH is to provide better interpretations of the generated recommendations.

Recently, representation learning (RL) has proven to be more effective than matrix factorization model in capturing local item relationships by modeling the item co-occurrence in individual user's interaction record. However, we contend that the potential

of RL for recommendation has not been fully exploited. Existing RL based recommendation methods either ignore personalization or the possible multi-level organizations of items for uncovering fine-grained item relationships, which could help achieve high quality recommendation performance.

In Chapter 5, we thus concentrate on the problem of representation learning based recommendation. In order to reach full exploitation of RL for effective recommendation, we contribute a multi-level RL method – MRLR – for personalized recommendation. Particularly, we first extend the original item embedding method to a more generic Bayesian framework, under which we then fuse the likelihood function of user-specific pairwise item ranking. This unified framework can then learn user and item embedding from both item co-occurrence relationships and user-specific ranked lists of items, benefiting from user and item RL while reaching the goal of personalized recommendation. Inspired by paragraphs in NLP as the intermediate level of word organization between individual words and documents, we further extend the framework to multi-level RL, i.e., category RL as the intermediate level of individual item RL and RL of items that rated by a same user, so as to capture fine-grained item relationships. Empirical validation on multiple real-world datasets shows that MRLR achieves better recommendation performance than the state-of-the-art algorithms.

With the development of semantic web, the knowledge graph (KG) has attracted much attention in the community of recommender systems, due to its efficiency in resolving data sparsity and cold start problems. It greatly helps uncover fine-grained item relationships by providing heterogeneous information related to items, i.e., different types of features and relations, thus facilitating to infer user preference towards items from various perspectives. Existing recommendation methods mainly rely on heavy and tedious feature engineering processes to manually extract features from the KG, limiting the enhancement of recommendation performance.

Chapter 6, therefore, aims at addressing the problem of knowledge graph based recommendation. We seek for a new data-driven method that does not rely on handcrafted features, yet can capture both semantics of entities and paths encoded in KG for effective recommendation. To this end, we proposes a unified recurrent knowledge graph embedding framework RKGE, which is able to not only learn semantic representations of entities and paths in a fully automatic way, but also to automatically discriminate the importance of different paths for recommendation. In order to learn the relations between a pair of entities, RKGE first mines all the paths linking paired entities which carry different semantics, without predefining the specific types of the path. It then encodes all paths between the entity pair through a batch of RNNs, with each path modeled by a single RNN. RKGE is thus flexible in capturing different numbers of paths with various lengths that connect entity pairs. The different effects of paths are then learned through a pooling operation, which further discriminates the importance of different paths and aggregates their effects for learning user preferences. Extensive validation on two real-world datasets demonstrates the superiority of RKGE against other counterparts.

To sum up, in this dissertation, we contribute to a series of novel recommendation approaches by exploiting item relationships for more effective recommendation. We achieve this goal from two different angles: 1) for the representation of item relationships, we investigate from item category with flat structure to category hierarchy and then to the knowledge graph; 2) for techniques, we go from latent factor models to representation learning methods, and then deep learning methods to help capture more and more fine-grained item relationships step by step.

7.2 Future Work

There are multiple potential directions for future work. In this dissertation, we present two potential and promising directions:

Temporal Context. One possible direction is to further accommodate temporal context into recommendation, thus adapting our proposed approaches to be dynamic recommendation models. In reality, user preferences often evolve over time, affected by dynamic user inclinations, item perception and popularity. Temporal context therefore has been recognized as an important type of information for modeling the dynamics of user preferences. It has extensive applications, ranging from movie recommendation [6], music recommendation [51], to location recommendation [125]. However, the incorporation of temporal context into our proposed recommendation approaches is non-trivial due to the following challenges:

- First, as our methods exploit item relationships (e.g., category hierarchy, the knowledge graph) for better recommendation, it is essential to first figure out how the two factors (i.e., item auxiliary information and temporal context) co-influence the final recommendation performance, and then design efficient frameworks to appropriately fuse the two factors in a unified manner. To resolve this issue, we may conduct extensive data analysis on multiple real-world datasets to get some insightful guidelines for the model design.

- Second, it is necessary to find out an efficient solution to better capture the temporal dynamics, thus helping achieve high-quality recommendation. Most of the existing methods [22, 51, 53, 79, 126] model the temporal dynamics by extending the latent factor model (LFM) with handcrafted features, so as to describe certain temporal patterns of user-item interactions. They either bin user-item interactions into time windows, assuming similar user behavioral patterns in the same window [51, 126], or adopt a time decay function to under-weight the interactions occurring deeper into the past [53, 79]. The handcrafted features, though proven to be effective, cannot capture complex temporal patterns in reality [114]. More importantly,

these methods cannot automatically select important interaction records in user-item interaction history when modeling user preferences. This greatly limits their application in real-world scenarios where user-item interactions can often happen accidentally. This problem may be solved by the recurrent neural network (RNN), which captures both the latent structures in historical user-item interactions – through hidden units – and their dynamics along the temporal domain. Unlike LFM based methods, RNN is nonparametric, thus can learn inherent dynamics that are more complex and suitable for making recommendations.

Deep Learning Methods. Another direction is pointed out from the perspective of techniques. In recent years, the revolutionary advances of deep learning technique have gained significant attention in various domains, such as speech recognition, image analysis and natural language processing, with recommender systems being no exceptions [129]. Deep learning provides a better understanding of user's demands, item's characteristics and historical interactions between them, thus can be adapted to further enhance recommendation performance. Actually, the RKGE model proposed in Chapter 6 is an adaptation and application of RNN, which is just a classic deep learning based method. The empirical study on real-world datasets has demonstrated the superiority of deep learning based methods against other state-of-the-art recommendation algorithms.

Similarly, other prevalent deep learning based methods, e.g., convolutional neural network (CNN), can also be applied into the area of recommender systems. Considering an on-line hotel booking system, customers may generally select several hotels as candidates according to the location and price at first when booking hotels. In order to make the final decision, they possibly compare the images of candidate hotels posted by the operators, and finally choose the one in their preferred styles (e.g., furniture, color). In this scenario, we could first exploit CNN to extract latent features from the hotel

images, which then can be incorporated into recommendation models for better recommendation service. This idea can be applied to any systems with images available (e.g., clothing, restaurants), and images as a type of auxiliary source data could be also helpful to address data sparsity and cold start issues of recommender systems.

List of Publications

- Yun Liu, Huihuai Qiu, Guibing Guo, **Zhu Sun**, Jie Zhang, Hai Thanh Nguyen. BPRH: Bayesian Personalized Ranking for Heterogeneous Implicit Feedback. Information Sciences, 2018.

- **Zhu Sun**, Jie Yang, Jie Zhang, Alessandro Bozzon, Yu Chen, Chi Xu. MRLR: Multi-level Representation Learning for Personalized Ranking in Recommendation. Proceedings of 26th International Conference on Artificial Intelligence (IJCAI), 2017.

- **Zhu Sun**, Jie Yang, Jie Zhang, Alessandro Bozzon. Exploiting both Vertical and Horizontal Dimensions of Feature Hierarchy for Effective Recommendation. Proceedings of 31st AAAI Conference on Artificial Intelligence (AAAI), 2017.

- Chang Xu, Jie Zhang, **Zhu Sun**. Online Reputation Fraud Campaign Detection in User Ratings. Proceedings of 26th International Conference on Artificial Intelligence (IJCAI), 2017.

- Wenjie Pei, Jie Yang, **Zhu Sun**, Jie Zhang, Alessandro Bozzon, David Tax. Interactive Attention-Gated Recurrent Networks for Recommendation. Proceedings of 26th ACM International Conference on Information and Knowledge Management (CIKM), 2017.

- **Zhu Sun**, Guibing Guo, Jie Zhang. Learning Hierarchical Category Influence on both Users and Items for Effective Recommendation. Proceedings of 32nd ACM Symposium on Applied Computing (SAC), 2017.

- Jie Yang, **Zhu Sun**, Jie Yang, Alessandro Bozzon, Jie Zhang. Learning Hierarchical Feature Influence for Recommendation by Recursive Regularization. Proceedings of 10th ACM Conference on Recommender Systems (RecSys), 2016.
- **Zhu Sun**, Guibing Guo, Jie Zhang. Exploiting Implicit Item Relationships for Recommender Systems. Proceedings of 23rd International Conference on User Modeling, Adaptation and Personalization (UMAP), 2015.

List of Submissions

- **Zhu Sun**, Jie Yang, Jie Zhang, Alessandro Bozzon, Longkai Huang, Chi Xu. Recurrent Knowledge Graph Embedding for Effective Recommendation. RecSys, 2018.
- Jie Yang, **Zhu Sun**, Jie Zhang, Alessandro Bozzon. Exploiting Feature Hierarchy for Effective Recommendation. IEEE Transactions on Knowledge and Data Engineering (TKDE), 2018.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6):734–749, 2005.

[2] Amir Albadvi and Mohammad Shahbazi. A hybrid recommendation technique based on product category attributes. *Expert Systems with Applications*, 36(9):11480–11488, 2009.

[3] Y Bao, H Fang, and J Zhang. Leveraging decomposed trust in probabilistic matrix factorization for effective recommendation. In *AAAI*, page 350, 2014.

[4] Oren Barkan and Noam Koenigstein. Item2vec: neural item embedding for collaborative filtering. In *International Workshop on MLSP*, pages 1–6. IEEE, 2016.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *JMLR*, 3(Feb):1137–1155, 2003.

[6] James Bennett, Stan Lanning, et al. The netflix prize. 2007:35, 2007.

[7] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *JMLR*, 3(Jan):993–1022, 2003.

[8] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. A convolutional neural network for modelling sentences. In *ACL*, 2014.

[9] Stefano Bocconi, Alessandro Bozzon, Achilleas Psyllidis, Christiaan Titos Bolivar, and Geert-Jan Houben. Social glass: A platform for urban analytics and decision-making through heterogeneous social data. In *WWW*, pages 175–178. ACM, 2015.

[10] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, pages 2787–2795, 2013.

[11] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[12] Rose Catherine and William Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *RecSys*, pages 325–332. ACM, 2016.

[13] Soumen Chakrabarti. Dynamic personalized pagerank in entity-relation graphs. In *WWW*, pages 571–580. ACM, 2007.

[14] Sneha Chaudhari, Amos Azaria, and Tom Mitchell. An entity graph based recommender system. *AI Communications*, (Preprint):1–9, 2017.

[15] Sonny Han Seng Chee, Jiawei Han, and Ke Wang. Rectree: An efficient collaborative filtering method. In *DaWaK*, volume 1, pages 141–151. Springer, 2001.

[16] Tianqi Chen, Weinan Zhang, Qiuxia Lu, Kailong Chen, Zhao Zheng, and Yong Yu. Svdfeature: a toolkit for feature-based collaborative filtering. *JMLR*, 13(Dec):3619–3622, 2012.

[17] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *JMLR*, 12(Aug):2493–2537, 2011.

[18] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *RecSys*, pages 191–198. ACM, 2016.

[19] Ramesh Dommeti. Neighborhood based methods for collaborative filtering. *A Case Study, I*, pages 1–5, 2009.

[20] Hui Fang, Guibing Guo, and Jie Zhang. Multi-faceted trust and distrust prediction for recommender systems. *Decision Support Systems*, 71:37–47, 2015.

[21] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *TKDE*, 19(3):355–369, 2007.

[22] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *RecSys*, pages 93–100. ACM, 2013.

[23] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Content-aware point of interest recommendation on location-based social networks. In *AAAI*, pages 1721–1727. Citeseer, 2015.

[24] Huiji Gao, Jiliang Tang, and Huan Liu. gscorr: modeling geo-social correlations for new check-ins on location-based social networks. In *CIKM*, pages 1582–1586. ACM, 2012.

[25] László Grad-Gyenge, Peter Filzmoser, and Hannes Werthner. Recommendations on a knowledge graph. In *International Workshop on MLRec*, pages 13–20, 2015.

[26] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. E-commerce in your inbox: Product recommendations at scale. In *KDD*, pages 1809–1818. ACM, 2015.

[27] Guibing Guo. Integrating trust and similarity to ameliorate the data sparsity and cold start for recommender systems. In *RecSys*, pages 451–454. ACM, 2013.

[28] Guibing Guo, Jie Zhang, and Daniel Thalmann. A simple but effective method to incorporate trusted neighbors in recommender systems. In *UMAP*, pages 114–125. Springer, 2012.

[29] Guibing Guo, Jie Zhang, and Daniel Thalmann. Merging trust in collaborative filtering to alleviate data sparsity and cold start. *KBS*, 57:57–68, 2014.

[30] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. A novel bayesian similarity measure for recommender systems. In *IJCAI*, pages 2619–2625, 2013.

[31] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. *KBS*, 74:14–27, 2015.

[32] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *AAAI*, pages 123–129, 2015.

[33] Ruining He, Chunbin Lin, Jianguo Wang, and Julian McAuley. Sherlock: sparse hierarchical embeddings for visually-aware one-class collaborative filtering. In *IJCAI*, pages 3740–3746, 2016.

[34] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, pages 173–182, 2017.

[35] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *TOIS*, 22(1):5–53, 2004.

[36] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[37] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[38] Thomas Hofmann. Latent semantic models for collaborative filtering. *TOIS*, 22(1):89–115, 2004.

[39] Seyed Abbas Hosseini, Keivan Alizadeh, Ali Khodadadi, Ali Arabzadeh, Mehrdad Farajtabar, Hongyuan Zha, and Hamid R Rabiee. Recurrent poisson factorization for temporal recommendation. *arXiv preprint arXiv:1703.01442*, 2017.

[40] Longke Hu, Aixin Sun, and Yong Liu. Your neighbors affect your ratings: on geographical neighborhood influence to rating prediction. In *SIGIR*, pages 345–354. ACM, 2014.

[41] Zhiting Hu, Poyao Huang, Yuntian Deng, Yingkai Gao, and Eric P Xing. Entity hierarchy embedding. In *ACL-IJCNLP*, pages 1292–1300, 2015.

[42] Won-Seok Hwang, Ho-Jong Lee, Sang-Wook Kim, and Minsoo Lee. On using category experts for improving the performance and accuracy in recommender systems. In *CIKM*, pages 2355–2358. ACM, 2012.

[43] Mohsen Jamali and Martin Ester. A matrix factorization technique with trust propagation for recommendation in social networks. In *RecSys*, pages 135–142. ACM, 2010.

[44] Rodolphe Jenatton, Julien Mairal, Francis R Bach, and Guillaume R Obozinski. Proximal methods for sparse hierarchical dictionary learning. In *ICML*, pages 487–494, 2010.

[45] Ke Ji, Hong Shen, Hui Tian, Yanbo Wu, and Jun Wu. Two-phase layered learning recommendation via category structure. In *PAKDD*, pages 13–24. Springer, 2014.

[46] How Jing and Alexander J Smola. Neural survival recommender. In *WSDM*, pages 515–524. ACM, 2017.

[47] Bhargav Kanagal, Amr Ahmed, Sandeep Pandey, Vanja Josifovski, Jeff Yuan, and Lluis Garcia-Pueyo. Supercharging recommender systems using taxonomies for learning user purchase behavior. *VLDB*, 5(10):956–967, 2012.

[48] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys*, pages 79–86. ACM, 2010.

[49] Choonho Kim and Juntae Kim. A recommendation algorithm using multi-level association rules. In *WI*, pages 524–527. IEEE, 2003.

[50] Seyoung Kim and Eric P Xing. Tree-guided group lasso for multi-task regression with structured sparsity. In *ICML*, 2010.

[51] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *RecSys*, pages 165–172. ACM, 2011.

[52] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, pages 426–434. ACM, 2008.

[53] Yehuda Koren. Collaborative filtering with temporal dynamics. In *KDD*, pages 447–456. ACM, 2009.

[54] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *RecSys*, pages 61–68. ACM, 2009.

[55] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *ICML*, pages 1188–1196, 2014.

[56] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of Brain Theory and Neural Networks*, 3361(10):1995, 1995.

[57] Daniel Lemire and Anna Maclachlan. Slope one predictors for online rating-based collaborative filtering. In *SDM*, pages 471–475. SIAM, 2005.

[58] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *RecSys*, pages 59–66. ACM, 2016.

[59] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252. ACM, 2010.

[60] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187, 2015.

[61] Christoph Lippert, Stefan Hagen Weber, Yi Huang, Volker Tresp, Matthias Schubert, and Hans-Peter Kriegel. Relation prediction in multi-relational domains using matrix factorization. In *NIPS Workshop: SISO*, 2008.

[62] Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. Personalized point-of-interest recommendation by mining users' preference transition. In *CIKM*, pages 733–738. ACM, 2013.

[63] Kai Lu, Guanyuan Zhang, Rui Li, Shuai Zhang, and Bin Wang. Exploiting and exploring hierarchical structure in music recommendation. In *AIRS*, pages 211–225. Springer, 2012.

[64] Chen Luo, Wei Pang, Zhe Wang, and Chenghua Lin. Hete-cf: Social-based collaborative filtering recommendation using heterogeneous relations. In *ICDM*, pages 917–922. IEEE, 2014.

[65] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. Recommender systems with social regularization. In *WSDM*, pages 287–296. ACM, 2011.

[66] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008.

[67] Benjamin Markines, Ciro Cattuto, Filippo Menczer, Dominik Benz, Andreas Hotho, and Gerd Stumme. Evaluating similarity measures for emergent semantics of social tagging. In *WWW*, pages 641–650. ACM, 2009.

[68] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52. ACM, 2015.

[69] Aditya Krishna Menon, Krishna-Prasad Chitrapura, Sachin Garg, Deepak Agarwal, and Nagaraj Kota. Response prediction using collaborative filtering with hierarchies and side-information. In *KDD*, pages 141–149. ACM, 2011.

[70] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[71] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[72] Koji Miyahara and Michael J Pazzani. Collaborative filtering with the simple bayesian classifier. In *PRICAI*, pages 679–689. Springer, 2000.

[73] Koji Miyahara and Michael J Pazzani. Improvement of collaborative filtering with the simple bayesian classifier. *Information Processing Society of Japan*, 43(11), 2002.

[74] Andriy Mnih. Taxonomy-informed latent factor models for implicit feedback. In *KDD Cup*, pages 169–181, 2011.

[75] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2008.

[76] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[77] Wenjie Pei, Jie Yang, Zhu Sun, Jie Zhang, Alessandro Bozzon, and David MJ Tax. Interacting attention-gated recurrent networks for recommendation. In *CIKM*, 2017.

[78] David M Pennock, Eric Horvitz, Steve Lawrence, and C Lee Giles. Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach. In *UAI*, pages 473–480. Morgan Kaufmann Publishers Inc., 2000.

[79] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. A general graph-based model for recommendation in event-based social networks. In *ICDE*, pages 567–578. IEEE, 2015.

[80] Tuan-Anh Nguyen Pham, Xutao Li, Gao Cong, and Zhenjie Zhang. A general recommendation model for heterogeneous networks. *TKDE*, 28(12):3140–3153, 2016.

[81] M Ross Quillan. Semantic memory. In *Semantic Information Processing*, pages 227–270. MIT Press, Cambridge, MA, 1968.

[82] Steffen Rendle. Factorization machines. In *ICDM*, pages 995–1000. IEEE, 2010.

[83] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, pages 452–461. AUAI Press, 2009.

[84] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[85] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *EC*, pages 158–167. ACM, 2000.

[86] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pages 285–295. ACM, 2001.

[87] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *CIT*, volume 1, 2002.

[88] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. Autorec: Autoencoders meet collaborative filtering. In *WWW*, pages 111–112. ACM, 2015.

[89] Mohak Sharma, P Krishna Reddy, R Uday Kiran, and Thirumalaisamy Ragunathan. Improving the performance of recommender system by exploiting the categories of products. In *International Workshop on Databases in Networked Information Systems*, pages 137–146. Springer, 2011.

[90] Chuan Shi, Jian Liu, Fuzhen Zhuang, S Yu Philip, and Bin Wu. Integrating heterogeneous information via flexible regularization framework for recommendation. *KAIS*, 49(3):835–859, 2016.

[91] Chuan Shi, Zhiqiang Zhang, Ping Luo, Philip S Yu, Yading Yue, and Bin Wu. Semantic path based personalized recommendation on weighted heterogeneous information networks. In *CIKM*, pages 453–462. ACM, 2015.

[92] Yue Shi, Martha Larson, and Alan Hanjalic. Tags as bridges between domains: Improving recommendation with tag-induced cross-domain collaborative filtering. *UMAP*, pages 305–316, 2011.

[93] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *CSUR*, 47(1):3, 2014.

[94] Ajit P Singh and Geoffrey J Gordon. Relational learning via collective matrix factorization. In *KDD*, pages 650–658. ACM, 2008.

[95] Alexander J Smola and Risi Kondor. Kernels and regularization on graphs. In *COLT*, volume 2777, pages 144–158. Springer, 2003.

[96] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 15(1):1929–1958, 2014.

[97] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009:4, 2009.

[98] Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S Yu, and Tianyi Wu. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *VLDB*, 4(11):992–1003, 2011.

[99] Zhu Sun, Guibing Guo, and Jie Zhang. Exploiting implicit item relationships for recommender systems. In *UMAP*, pages 252–264. Springer, 2015.

[100] Zhu Sun, Guibing Guo, and Jie Zhang. Learning hierarchical category influence on both users and items for effective recommendation. In *SAC*, pages 1679–1684. ACM, 2017.

[101] Zhu Sun, Jie Yang, Jie Zhang, and Alessandro Bozzon. Exploiting both vertical and horizontal dimensions of feature hierarchy for effective recommendation. In *AAAI*, pages 189–195, 2017.

[102] Zhu Sun, Jie Yang, Jie Zhang, Alessandro Bozzon, Yu Chen, and Chi Xu. Mrlr: Multi-level representation learning for personalized ranking in recommendation. In *IJCAI*, pages 2807–2813. IJCAI, 2017.

[103] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2):26–31, 2012.

[104] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *RecSys*, pages 225–232. ACM, 2016.

[105] Ellen M Voorhees et al. The trec-8 question answering track report. 99:77–82, 1999.

[106] Slobodan Vucetic and Zoran Obradovic. Collaborative filtering using a regression-based approach. *KIS*, 7(1):1–22, 2005.

[107] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. Collaborative deep learning for recommender systems. In *KDD*, pages 1235–1244. ACM, 2015.

[108] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR*, pages 501–508. ACM, 2006.

[109] Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. Exploring implicit hierarchical structures for recommender systems. In *IJCAI*, pages 1813–1819, 2015.

[110] Yueyang Wang, Yuanfang Xia, Siliang Tang, Fei Wu, and Yueting Zhuang. Flickr group recommendation with auxiliary information in heterogeneous information networks. *Multimedia Systems*, pages 1–10, 2016.

[111] Markus Weimer, Alexandros Karatzoglou, Quoc V Le, and Alex J Smola. Cofi rank-maximum margin matrix factorization for collaborative ranking. In *NIPS*, pages 1593–1600, 2008.

[112] Li-Tung Weng, Yue Xu, Yuefeng Li, and Richi Nayak. Exploiting item taxonomy for solving cold-start problem in recommendation making. In *ICTAI*, volume 2, pages 113–120. IEEE, 2008.

[113] Paul J Werbos. Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339–356, 1988.

[114] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *WSDM*, pages 495–503. ACM, 2017.

[115] Gui-Rong Xue, Chenxi Lin, Qiang Yang, WenSi Xi, Hua-Jun Zeng, Yong Yu, and Zheng Chen. Scalable collaborative filtering using cluster-based smoothing. In *SIGIR*, pages 114–121. ACM, 2005.

[116] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. Social collaborative filtering by trust. In *IJCAI*, pages 2747–2753, 2013.

[117] Jie Yang, Zhu Sun, Alessandro Bozzon, and Jie Zhang. Learning hierarchical feature influence for recommendation by recursive regularization. In *RecSys*, pages 51–58. ACM, 2016.

[118] Xiwang Yang, Harald Steck, and Yong Liu. Circle-based recommendation in online social networks. In *KDD*, pages 1267–1275. ACM, 2012.

[119] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alexander J Smola, and Eduard H Hovy. Hierarchical attention networks for document classification. In *HLT-NAACL*, pages 1480–1489, 2016.

[120] Xiao Yu, Quanquan Gu, Mianwei Zhou, and Jiawei Han. Citation prediction in heterogeneous bibliographic networks. In *SDM*, pages 1119–1130. SIAM, 2012.

[121] Xiao Yu, Xiang Ren, Quanquan Gu, Yizhou Sun, and Jiawei Han. Collaborative filtering with entity similarity regularization in heterogeneous information networks. *IJCAI-HINA*, 27, 2013.

[122] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *WSDM*, pages 283–292. ACM, 2014.

[123] Xiao Yu, Xiang Ren, Yizhou Sun, Bradley Sturt, Urvashi Khandelwal, Quanquan Gu, Brandon Norick, and Jiawei Han. Recommendation in heterogeneous information networks with implicit user feedback. In *RecSys*, pages 347–350. ACM, 2013.

[124] Quan Yuan, Li Chen, and Shiwan Zhao. Factorization vs. regularization: fusing heterogeneous social relationships in top-n recommendation. In *RecSys*, pages 245–252. ACM, 2011.

[125] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *SIGIR*, pages 363–372. ACM, 2013.

[126] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[127] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. Collaborative knowledge base embedding for recommender systems. In *KDD*, pages 353–362. ACM, 2016.

[128] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *SDM*, pages 549–553. SIAM, 2006.

[129] Shuai Zhang, Lina Yao, and Aixin Sun. Deep learning based recommender system: A survey and new perspectives. *arXiv preprint arXiv:1707.07435*, 2017.

[130] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. Taxonomy discovery for personalized recommendation. In *WSDM*, pages 243–252. ACM, 2014.

[131] Tong Zhao, Julian McAuley, and Irwin King. Improving latent factor models via personalized feature projection for one class recommendation. In *CIKM*, pages 821–830. ACM, 2015.

[132] Jing Zheng, Jian Liu, Chuan Shi, Fuzhen Zhuang, Jingzhi Li, and Bin Wu. Recommendation in heterogeneous information network via dual similarity regularization. *Data Science and Analytics*, 3(1):35–48, 2017.

[133] Cai-Nicolas Ziegler, Georg Lausen, and Lars Schmidt-Thieme. Taxonomy-driven computation of product recommendations. In *CIKM*, pages 406–415. ACM, 2004.