4-2019

Modeling sequential and basket-oriented associations for top-K recommendation

Duc-Trong LE DUC TRONG
Singapore Management University, ductrong.le.2014@phdis.smu.edu.sg

MODELING SEQUENTIAL AND BASKET-ORIENTED ASSOCIATIONS FOR TOP-K RECOMMENDATION

LE DUC TRONG

SINGAPORE MANAGEMENT UNIVERSITY

2019

Modeling Sequential and Basket-Oriented Associations for Top-K Recommendation

LE Duc Trong

Submitted to School of Information Systems
in partial fulfillment of the requirements for the
Degree of Doctor of Philosophy in Information Systems

Dissertation Committee:

Hady W. Lauw (Supervisor/Chair)
Associate Professor of Information Systems
Singapore Management University

Yuan Fang
Assistant Professor of Information Systems
Singapore Management University

David Lo
Associate Professor of Information Systems
Singapore Management University

Siu Cheung Hui
Associate Professor
Nayang Technology University

SINGAPORE MANAGEMENT UNIVERSITY
2019

I hereby declare that this PhD dissertation is my original work
and it has been written by me in its entirety.
I have duly acknowledged all the sources of information
which have been used in this dissertation.

This PhD dissertation has also not been submitted for any degree
in any university previously.

LE Duc Trong

24 April 2019

Modeling Sequential and Basket-Oriented Associations

for Top-K Recommendation

by

LE Duc Trong

Abstract

This dissertation addresses a task in recommendation systems, named *top-K recommendation*. Dealing with this problem, we concentrate on modeling item-item association types including *sequential* and *basket-oriented* associations. The first type implies the sequential dependencies between the preceding adoptions and the following adoptions while the second type indicates the correlative dependencies among items concurrently adopted at the same time. The modeling of those associations independently or jointly triggers several new *top-K* recommendation formulations, which are discussed in the four main chapters of the dissertation.

In Chapter 3, we deal with the sequential recommendation problem by modeling sequential associations between consecutive items in single-item sequences. A generative model is built upon Hidden Markov Model (HMM). We modify the traditional HMM by incorporating two latent factor types including: dynamic context and user-biased factors. This modification is motivated by two hypotheses. The first hypothesis is that the transition from one state to the next state is leveraged by dynamic context factors. The second one is based on the intuition that similar users may have analogous preferences on particular items. It expresses that the emission probability might be affected by latent user groups.

In Chapter 4, basket-oriented associations are exploited to solve the basket-completion problem. Suppose that a user is currently holding a basket of items, the task is to recommend him what item should be added into the current basket. We apply Factorization Machines to propose the BFM model, which factorizes the four associations including: user & target item, target item & basket items, among

basket items and user & basket items. Relying on the hypothesis that similar baskets have similar intent, we introduce a constrained variant CBFM considering the similarity as a regularization.

In Chapter 5, *sequential-* and *basket-oriented* associations in contemporaneous basket sequences, are concurrently taken into account for sequential recommendations. A "contemporaneous basket sequence" instance is a pair of "target" (e.g., purchase) and "support" (e.g., click) sequences. To investigate the similarities as well as the differences that users behave in these sequences, we present three twin network architectures. CBS-SN hypothesizes that the user behaviors are exactly the same. In contrast, CBS-CFN takes into account the dissimilarities in modeling long-term sequential dependencies on the two sequences. CBS-DFN considers two different types of sequential dependencies, e.g., short-term on "target" and long-term "support".

In Chapter 6, we introduce and tackle a novel research problem, referred to as *correlation-sensitive next-basket recommendation*, where the objective is to recommend the next basket of correlated items. Assuming baskets share a consistent knowledge on correlative dependencies among items in the form of a correlation matrix, we propose Basket-Sequence Correlation Networks (*Beacon*) that leverages these correlations to enhance the representation of baskets, and subsequently capture sequential associations more accurately. The sequential signal triggered by the last basket is later aggregated with the correlation matrix to generate scores for candidates in the next-basket recommendation.

Contents

List of Figures

List of Tables

Acknowledgments

I gratefully acknowledge the able support and guidance of my supervisor Professor Hady W. Lauw throughout my PhD study. Without his continuous encouragement and careful supervision, this dissertation would never have taken shape. I have received from him a lot of invaluable advice on not only research but also my academic career and personal life.

I am also grateful to my second advisor Professor Yuan Fang for the continuous support since my second year of PhD. His guidance plays a significant role to the completeness of this dissertation.

I highly appreciate useful comments and suggestions from members of my dissertation committee: Professor David Lo and Professor Siu Cheung Hui during my dissertation revision and defense.

My sincere thanks goes to my senior Nguyen Thanh Son, who introduced me to my supervisor and gave me a lot of supports from the first day I joined School of Information Systems. Unforgettable mention about my friends Hoang Tuan Anh, Hoang Ai Phuong, Le Van Minh Tuan for sharing me their inspiration and experience during my PhD time.

I must place on record my thanks to all people in Preferred.AI group: Aghiles Salah, Maksim Tkachenko, Le Duy Dung, Truong Quoc Tuan, Le Trung Hoang, Lee Ween Jiann, Chia Chong Cher, Zhang Ce, Guo Jingyao, for the great moments, discussions and suggestions.

I would like to thank Singapore Management University for the scholarship and financial supports for my study and conference travels.

I express my deep gratitude to my parents, my beloved wife, my sister and my kids. Without their unconditional love and consistent encouragement, I would never accomplish this research work.

Dedicated to my family

Chapter 1

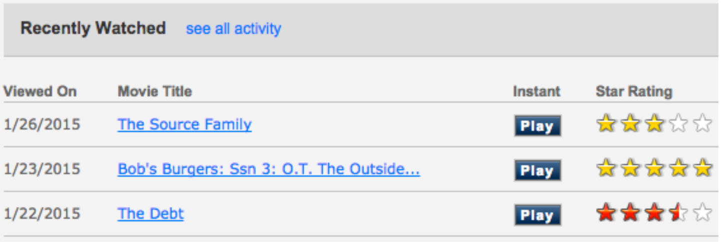
Introduction

1.1 Top-K Recommender Systems

Recommender systems (RS) are embedded in web sites to support the decision-making process of users. Specifically, the primary objective is to help users discover relevant items from enormous collections. ”Item” is a general term, which refers to the object to be recommended. It could be a product in the e-commerce domain, a video in the online streaming context, a location in the check-in site, etc. Additionally, RS also has a role in orienting users to adopt digital content. They are used in marketing campaigns to increase the profit of providers. Currently, RS have been playing more important roles in human daily life via diverse applications. For examples, YouTube recommendations are really useful for over billion of users to find and watch what they need from the chaos of online videos [21]. In the e-commerce domain, Amazon has been integrating with a recommendation engine to increase the satisfaction of customers as well as their profit for almost twenty years [126]. The ever-growing size of social networks (e.g., Twitter, Digg or Facebook) is remarkably contributed by social recommendation algorithms [34, 83, 33]. Foursquare and Yelp apply RS to suggest users point-of-interests where they might want to go [159, 164]. Likewise, there are many other real-life examples [1] summarized in Table 1.1.

System	Item
Amazon.com	Books and other Products
Netflix	DVDs, Streaming Video
Jester	Jokes
GroupLens	News
MovieLens	Movies
last.fm	Songs/Playlists
Google News	News
Google Search	Advertisements, Products
Youtube	Online Video
Facebook	Friends, Advertisements
Pandora	Songs/Playlists
Spotify	Songs/Playlists
Tripadvisor	Travel products
IMDb	Movies
Foursquare	Point-of-interest
Yelp	Point-of-interest

Table 1.1: Real-life Examples of Recommender systems



Recently Watched see all activity

Viewed On	Movie Title	Instant	Star Rating
1/26/2015	The Source Family	Play	★★★☆☆
1/23/2015	Bob's Burgers: Ssn 3: O.T. The Outside...	Play	★★★★★
1/22/2015	The Debt	Play	★★★½☆

Figure 1.1: An Example of Explicit Feedback: Netflix ratings

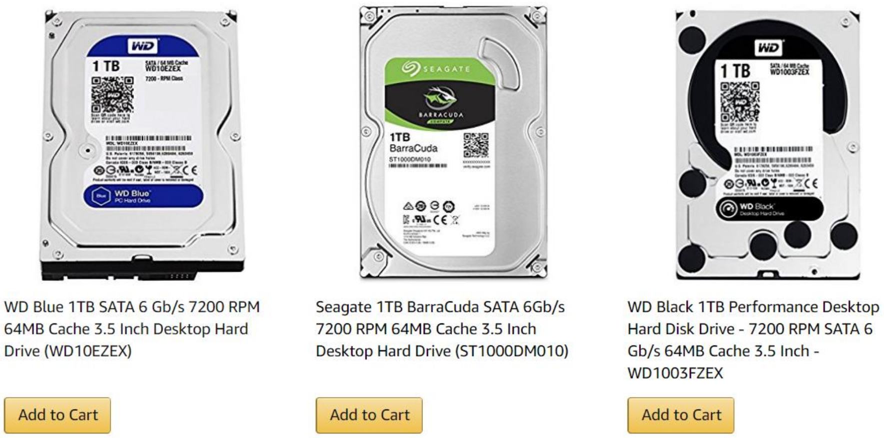
When users interact with RS, they trigger the notion of *”feedback”*. Commonly, there are two types of feedback namely *explicit-* and *implicit feedback* [94]. The former type is often observed in the form of *ratings*. The various rating values illustrate the different appreciation of the user on an item. For example, Netflix utilizes a 5-star ratings system (refer to Figure 1.1), in which 1-star shows an extreme dislike and 5-star indicates a strong interest. Likewise, the *implicit feedback* is often shown in the *binary* form, of which the value 1 represents the adoption, 0 otherwise. For example, a user generates *”1”* feedback when he clicks or purchases an item. The *”0”* feedback is hypothesized for what items he has not adopted. Relying on the two types of feedback, the recommendation task can be tackled either as a rating prediction or a ranking problem [1].

Top-K Recommender Systems represent a typical collection of RS [55, 118], where the recommendation is considered as a ranking problem. Initially, recommender models evaluates the goodness of each item candidate by a real score. A ranking of candidates is constructed by sorting their respective scores. The determination of the top-K highest ranked items is known as *top-K recommendation* [1].

1.2 Objectives

We begin with the question how to build a good *top-K recommender system*. Generally, the fundamental task of RS is to generate effective suggestions for users. The suggestion usually relies on the *understanding* of the historical feedback of users, e.g., clicking, rating, purchasing, etc. The better the *understanding* is, the more reasonable the recommendation is. Typically, the *understanding* could be systematically built by investigating all dependent factors that associated with the item adoption phenomena. This investigation is performed not only on a specific user or item but also across users and items, referred to as *collaborative filtering (CF)*. Previously, numerous works focus on the user-item associations, which emphasize the user-specific factor or personalization [61, 89, 115]. Given a user, the personalization captures his personal preference in adopting items.

In this dissertation, we seek to exploit an orthogonal direction that takes into account item-item associations. This association type reflects the dependency between items, i.e., the adoption of an item might trigger the adoption of other items. In traditional RS, the modeling of item-item associations is primarily based on similarity. Given a user, the recommendation task is to find similar items to those he adopted previously. This strategy is firstly used in *content-based* RS [99]. Items are classified into categories using their descriptive attributes. Items within the same category are similar hence they could be interchangeable options for the recommendation. Another family of similarity-based recommendation algorithms is referred to as *item-based collaborative filtering*. These methods identify pairs of



WD Blue 1TB SATA 6 Gb/s 7200 RPM 64MB Cache 3.5 Inch Desktop Hard Drive (WD10EZEX)

Seagate 1TB BarraCuda SATA 6Gb/s 7200 RPM 64MB Cache 3.5 Inch Desktop Hard Drive (ST1000DM010)

WD Black 1TB Performance Desktop Hard Disk Drive - 7200 RPM SATA 6 Gb/s 64MB Cache 3.5 Inch - WD1003FZEX

Figure 1.2: An Example of Content-based Recommender Systems: Three recommendations for 1TB Hard-drive from Amazon.

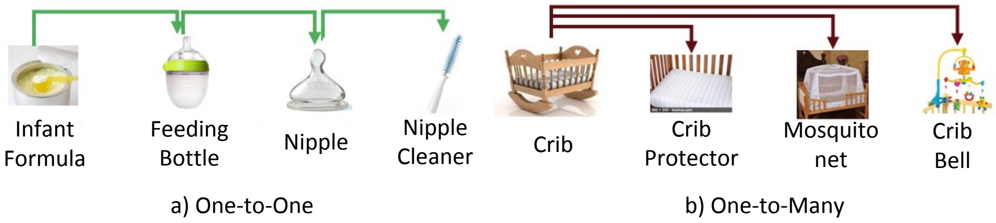


Figure 1.3: An Example of the Sequential Association: Baby stuffs are bought sequentially

items that tend to be rated similarly by users [9, 117]. Based on this knowledge, a user is recommended items that are similar to what he has highly rated. In Figure 1.2, three content-based recommendations of Amazon.com are shown for 1TB hard-drive. They have not only the same type (e.g., hard-drive) but also the similar technical specifications including 7200RPM 6GB/s, 64MB Cache, etc. If a user does not want to buy a Seagate hard-drive, he might have two other options of the Western Digital company.

Besides the *similarity*, there are other dependency types captured in the item-item association such as *sequential-* and *correlative dependencies*. The former type is inferred from **sequential associations** within *sequences*, where the adoptions of users are sorted by the time-aware manner [1]. It indicates the influence of the preceding adoptions on the consecutive adoptions. For example, Figure 1.3 shows interesting examples of sequential dependencies in buying baby stuffs [155]. In

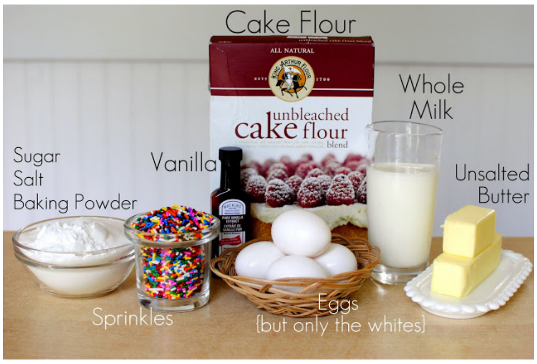


Figure 1.4: An Example of the Basket-oriented Association: Cake ingredients

Figure 1.3(a), it is the "one-to-one" sequential association, in which the current item only triggers the next adoption. Differently, Figure 1.3(b) shows "one-to-many" associations, where the adoption of the first item results in the following items.

Regularly, the order of items within a sequence is indicated via time steps. A time step might flexibly represent a particular timestamp, an hour, a day or a browsing session. Hence, the adoption at a given time-step could be either a single item or a set of items, referring to the notion of *basket*. Associations among items within the same basket, named **basket-oriented associations**, usually imply their complementary relations [85, 137]. Figure 1.4 illustrates necessary ingredients to make a cake from scratch [1]. Several simpler associations of these ingredients, e.g., flour & baking powder or flour & vanilla, can be easily found in real-life shopping baskets.

As the main focus, we investigate the modeling of *sequential-* and *basket-oriented associations* to improve the performance of *top-K recommendations*. In the next section, we will present several novel problem formulations as well as the primary ideas to solve these problems.

1.3 Problems and Challenges

Here, we introduce some potential recommendation problems via modeling *sequential-* or *basket-oriented associations* independently and jointly. If we only take into account the sequential association, we may deal with the sequential recommendation

[1] http://jessicqamaine.blogspot.com/2016/03/cake-ingredients-from-scratch-from.html

problem, which predicts the next item given a sequence of adopted items. Likewise, the independent consideration of *basket-oriented associations* suggests us a potential task named basket-completion (a.k.a. “basket-sensitive”) recommendation, where the goal is to predict an item to add into a given basket of items. In the case of jointly capturing both association types in a give sequence of baskets, a possibility is to recommend the next basket. We will present the overview of these problems in the next four sub-sections.

1.3.1 Sequential Recommendation using Dynamic Factors

Typically, *sequential associations* between items are often found with the notion of sequences. We are interested in modeling this association type to capture the sequential dependency between successive items in a sequence. The dependency is expressed in terms of which other items may be preferred after consuming an item. For instance, a user’s stream of tweets may reveal which topics tend to follow a topic, e.g., commenting on politics upon reading morning news followed by more professional postings during working hours. The sequence of songs one listens to may express a preference for which genre follows another, e.g., more upbeat tempo during a workout followed by slower music while cooling down.

The problem is how to quantify the sequential effect? Given a set of item sequences, we seek a probabilistic model, capturing *sequential associations*, so as to estimate the likelihood of future items in any particular sequence. Each sequence (e.g., a playlist, a stream of tweets) is assumed to have been generated by a single user. To achieve this goal, we turn to probabilistic models for *general* sequences. Regardless of having such models studied in the literature, we build on the foundation of the well-accepted Hidden Markov Model (HMM) [105], which has been shown to be effective in various applications, including speech- and handwriting-recognition, etc. HMM utilizes a number of hidden states. To generate each sequence, we move from one state to another based on *transition* probability. Each item is sampled from the corresponding state’s *emission* probability.

While HMM is fundamentally sound as a basic model for sequences, we identify two significant factors, yet unexploited, which would contribute towards greater effectiveness for modeling sequential preferences. *First*, the generation of an item from a state's emission in HMM is only dependent on the state. However, as we are concerned with *user-generated* sequences, the selection of items may be affected by the personalized association. However, due to the sparsity of information on individual users, we stop short of modeling individual emissions. Rather, we model latent groups, whereby users in the same group share similar preferences over items, i.e., emissions. *Second*, the transition to the next state in HMM is only dependent on the previous state. We posit that *context* in which a transition is about to take place also plays a role. For example, in the scenario of musical playlists, let us suppose that a particular state represents the genre of soft rock. There are different songs in this genre. If a user likes the artist of the current song, she may wish to listen to more songs by the same artist. Otherwise, she may wish to change to a different genre altogether. In this case, the artist is an observed *feature* of the context that may influence the transition dynamically.

1.3.2 Basket-Sensitive Recommendation

In many scenarios, a user adopts more than one item within a session, referred to as a basket of items. The associations among basket-items, also known as *basket-oriented associations*, possibly signify their correlative dependencies. Intuitively, a user buys an item to address a specific need, which frequently could only be fulfilled by multiple related items. When shopping for clothes, a user may be looking for matching top, bottom, and accessories. To make a cake, a user needs flour, milk, eggs, and sugar, among other ingredients. Someone on an errand may wish to visit several places in one trip: dropping mail, collecting laundry, having lunch, and buying groceries. In these cases, the items (e.g., products, places, songs) sought by users are not independent. Given a user who is holding a basket of items, we seek to deal with the basket-completion problem, recommending another item to add to the basket. This observation is relevant to both online and offline scenarios. For instance, an online shopper at Amazon.com, or an offline shopper at an upcoming Amazon Go[2] physical store, may be recommended relevant products based on her current cart. In brick-and-mortar supermarkets, RFID-tagged items and smart shopping carts [158] allow real-time recommendation of items based on a user's smart cart. A basket may also refer to items adopted by a user within a specific period of time, e.g., points of interest visited in a trip. While seeking the latent need represented by a basket of items, recommendation shall still be personalized, as a user may have preferences as to the exact items involved (e.g., brand, size, color).

Considering the basket-completion problem as the next-item recommendation task, we advocate an approach that factorizes basket-level associations. We propose a model that we call BASKET-SENSITIVE FACTORIZATION MACHINE or BFM, which models the recommendation as a function of four types of associations. The first is association between the user and the target item to be recommended (where most matrix factorization approaches stop). In addition, we model association between the target item and each item currently in the basket, association among basket items, and association between the user and each basket item.

While BFM captures the notion of relationship among items within a basket, we further observe relationship among baskets with similar intent. Continuing an earlier example, suppose that a user shops for cake ingredients. At one occasion, the user may already have $\{milk, flour, sugar\}$ in her basket, and we may recommend *eggs*. At another occasion, the user may already have $\{milk, flour, eggs\}$ in her basket, and we thus recommend *sugar*. While we may be recommending different items (*eggs* in one case, and *sugar* in the other), the suggested instances are addressing similar needs. Motivating from the intuitive example, we propose a set of constraints to BFM to make the likelihood of recommendations that eventually belong to the same basket similar. We refer to this second model as CONSTRAINED BFM or CBFM.

[2] https://www.amazon.com/b?node=16008589011

1.3.3 Sequential Recommendation using Contemporaneous Basket Sequences

Recently, Recurrent Neural Networks (RNN) produce state-of-the-art results in learning *long-term sequential dependencies* [74]. However, direct application of RNN to sequential recommendation suffers from two major limitations in modeling choices. First, it models a sequence of one type of actions (e.g., only purchases). Second, it assumes that at each time step, there is only one action (e.g., one item purchased). However, these assumptions may not bear out in some scenarios. For one, there are multiple types of actions resulting from user interaction with a system. In an online marketplace, a user may click on various items under consideration, abandon most, add some to a shopping cart, and put others on a wish list, before an eventual purchase takes place. In a video streaming service, a user may watch some trailers, follow through to watch some shows fully, and later on may rate or review some movies, of which a few might be rated highly. In each case, we are dealing with multiple sequence types (e.g., sequence of clicks and sequence of purchases). Importantly, these sequence types are *contemporaneous*, occurring within a common period of time, and may well be capturing some related underlying behaviors. For instance, to predict what one would purchase, it may be instructive to pay attention to not only what a user has purchased previously, but also what she has clicked in the past. Therefore, we postulate the need for modeling jointly the association between adoptions on these contemporaneous sequences. As each set of contemporaneous sequences is commonly associated with only one user, the essence of this paradigm is learning sequentiality among items across users' sequences, rather than personalization per se.

Additionally, we are not always dealing with a strict ordering of individual items. More frequently, we deal with groups or sessions, whereby there may be sequentiality from one session to another, but the ordering within a session may not be informative. For example, when planning travel, one day we may be searching for airfare, while on another day we may be booking accommodations. When grocery shopping, we may buy for different meal plans on different days. Though not necessarily sequentially ordered, items within a session are probably correlated to some degree, e.g., items of the same meal plan. We refer to such a group or session as "basket", hence the need to model the *correlative dependencies* among basket-items is also raised.

To capture both *sequential-* and *correlative dependencies*, we propose to model *contemporaneous basket sequences* or CBS for short. We just focus on a pair[3] of sequence types: target and support. The *target* sequence refers to high-quality, high-value, and possibly sparser interactions (e.g., purchases) for which we wish to predict the next interaction (e.g., next purchase). The *support* sequence refers to more frequent and informative interactions (e.g., clicks) that would be relevant for predicting the next target item. For example, if purchasing is the target, and clicking is the support, then we are predicting the next purchase by modeling sequence of purchases and sequence of clicks.

We explore dual-RNN structure to represent the two sequence types. Having been generated contemporaneously from the same ecosystem of interactions, the sequence types likely model related phenomena. Instead of two completely different RNN's, we base our CBS framework on the concept of *twin networks*. Analogously to biological twins, they share some commonalities, but to different degrees in different cases. We develop three CBS architectures along the spectrum of commonalities. In all, the two sequence types share a basket encoder to capture in-basket associations among items. They vary in how much sharing occurs at the recurrent units. For CBS-SN (Siamese Networks), the sequence types share a recurrent encoder. For CBS-CFN (Concordant Fraternal Networks), they each have a different recurrent encoder with the same recurrent units. For CBS-DFN (Discordant Fraternal Networks), one sequence type has a recurrent encoder and the other does not, to model different scopes of sequential effects.

[3] While the fundamentals of the proposed modeling would allow further extensions beyond two sequence types, we discourse on only two sequence types for clarity of exposition.

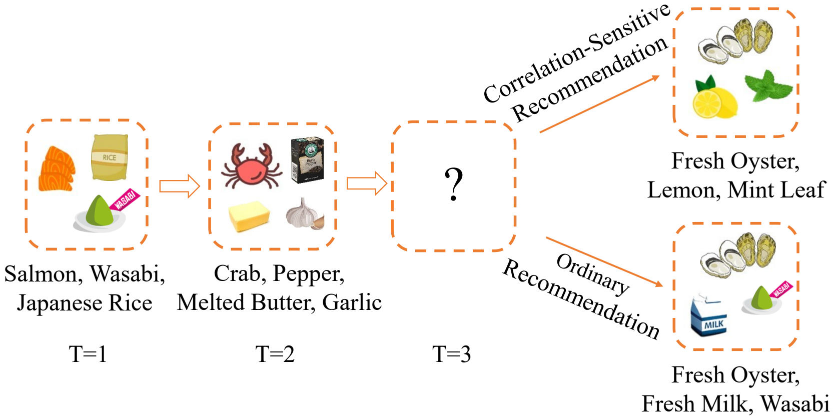


Figure 1.5: Motivating example for correlation-sensitive next-basket recommendation.

1.3.4 Correlation-Sensitive Next-Basket Recommendation

Given a user and his basket-level adoption sequence, his preference is expressed not only via correlative associations among basket items but also sequential association across baskets. Figure 1.5 presents a basket sequence example in the grocery shopping scenario. Each time step respects to a shopping session, where the user purchases a basket of items. He seems to have a firm preference in seafood. In the first session ($T = 1$), the co-occurrence of $\{$*Salmon, Wasabi, Japanese rice*$\}$ somehow implies his latent intention of making *Sushi*. Likewise, the second session ($T = 2$) is about another seafood named *Pepper Crab* with the combination on $\{$*Crab, Pepper, Melted butter, Garlic*$\}$. The question is how to derive insight from his adopted basket sequence and generate the next-basket recommendation?

There are a number of research works dealing with the next-basket recommendation problem [110, 147, 161, 141]. Generally, these works have a common limitation that items within recommended basket are usually independent. The recommendation is remarkably influenced by items' popularity while the correlative associations among basket items are not as emphasized. In Figure 1.5, it might trigger the ordinary recommendation as an uncommon combination of $\{$*Fresh Oyster, Fresh Milk, Wasabi*$\}$. Motivating from this scenario, our objective is upgraded to recommend next basket in one stroke with correlated items, referred to as *correlation-sensitive next-basket recommendation*. A basket of $\{$*Fresh Oyster, Lemon, Mint*

Leaf} in Figure 1.5 is a typical example of correlation-sensitive recommendations because items are necessary ingredients of a popular recipe with *fresh oyster*.

Towards this goal, we have to model concurrently basket-oriented and sequential associations in basket sequences before recommending a basket of correlated items. Assume that "predicted" baskets follow the same correlation dependencies built from "observed" baskets. We propose *Beacon*, a hierarchical network architecture of three main layers. The first layer is *Correlation-Sensitive Basket Encoder*. It enhances the determination of a given basket by exploiting not only item biases but also correlative dependencies among items. This is then utilized to infer the basket latent representation. Next, encoded basket vectors of the basket sequence are sequentially fed into the *Sequence Encoder* to capture sequential associations across baskets. The last hidden output of this layer signifies the information about items might be adopted next. In the third layer *Correlation-Sensitive Score Predictor*, we aggregate this with items' biases and correlations to generate the correlation-sensitive next-basket recommendation.

1.4 Organization and Contributions

In the scope of this dissertation, we give efforts on modeling sequential- and basket-oriented associations independently and jointly. We briefly summarize main chapters aligned with our respective contributions as follows:

- In Chapter 3: We exploit the sequential associations between items of sequences to predict the next item adoption (a.k.a. *sequential recommendation*). Concisely, we systematically build a probabilistic model, whereby transitions from one state to another state may be *dynamically* influenced by the context features, and emissions are influenced by latent groups of users. We also describe how to learn the model parameters as well as how to generate next-item predictions. To validate the approach, we evaluate these models comprehensively on both synthetic and real-life datasets. This work was published in

Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD) 2016 [62].

- In Chapter 4: We tackle the basket completion task, recommending a user what item should be added into his current basket of adopted items, based on the *basket-oriented associations*. The first model called BASKET-SENSITIVE FACTORIZATION MACHINE or BFM, models the recommendation as a function of four types of associations namely: the association between the user and the target item to be recommended, the association between the target item and each item currently in the basket, the association among basket items, and the association between the user and each basket item. We investigate empirically which associations are most useful. Upgrading from BFM, we propose a set of constraints to make the likelihood of recommendations that eventually belong to the same basket similar. We call this model as CONSTRAINED BFM or CBFM. The two proposed models are investigated empirically over three real-life datasets. This work was published in Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI) 2017 [63].

- In Chapter 5: The notion of contemporaneous basket sequences motivates us to jointly model both *basket-oriented* and *sequential associations*. We hypothesize that modeling contemporaneous basket sequences could be beneficial for next-item recommendation due to synergies between the target and support sequences. To the best of our knowledge, we are the first to model contemporaneous basket sequences. Dealing with this task, we develop three neural network architectures: CBS-SN, CBS-CFNand CBS-DFN, describe their design and note some learning details clearly. Subsequently, we investigate research questions on the effectiveness of modeling contemporaneous basket sequences on public datasets in different domains. This work has been published in Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI) 2018 [64].

- In Chapter 6: We further improve the modeling of basket sequences for a novel sequential recommendation task named *correlation-sensitive next-basket recommendation*, which proposes to a user the next basket with correlated items. With the assumption that "observed" and "predicted" baskets share a common correlative dependency graph among items, we incorporate this information to improve the representation of baskets as well as the correlation between items in the recommended basket. These ideas are sequentially modeled in a hierarchical network architecture named *Beacon*. We validate the efficiency of *Beacon* via four research questions on three real-life datasets.

Chapter 2

Literature Review

Our primary problem is *top-K recommendation*. Traditional approaches usually rely on modeling user-item associations, generated from the feedback of users on items (e.g., rating, click, etc.). The recommendation is often considered as either a rating prediction task or a ranking problem. The most common prediction framework is matrix factorization [61, 89], where the observed user-by-item rating matrix is factorized into D latent factors. Each user (or item) is represented by a latent vector $v_u \in \mathbb{R}^D$ (or $v_i \in \mathbb{R}^D$). The rating that u gives on i could be computed by $R_{ui} = v_u.v_i$. Improving upon this idea, [140, 47] introduce feature-based matrix factorization models that exploit item's prices to improve the recommendation performance. Another prediction framework is restricted Boltzmann machines [115] based on neural networks. It measures the likelihood u rates i with a particular rating r. [32] introduces a tied variant integrating content information to deal with the *cold-start recommendation problem*. In 2009, Truyen et al. [134] proposes another variant exploiting the similarity and co-occurrence information. Equally important, latent semantic analysis [45, 46] is another direction, which models the association among users, items, and ratings via multinomial probabilities. There are several works based on topic modeling [101, 84, 25], which jointly model ratings and textual information (e.g., reviews, tags). Recently, [40] proposes a neural network-based collaborative filtering framework to model noisy implicit feedback.

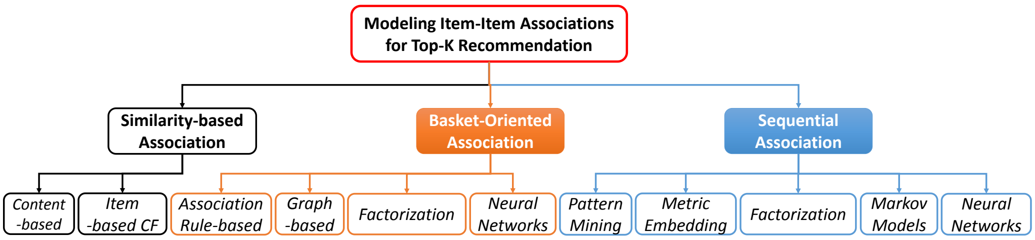


Figure 2.1: The Taxonomy of Modeling Item-Item Associations for Top-K Recommendation

Tackling as the ranking task, [109, 108] present personalized pairwise ranking approaches. Instead of using the actual rating-based ranking, they approximate the probability how likely an item is preferred to another item by a sigmoid estimator. [124] solves this ranking task via optimizing the mean reciprocal rank directly.

In the dissertation, we aim to model the item-item associations for the *top-K recommendation* problem. Modeling these associations helps us to leverage dependent factors affecting the item adoption. Figure 2.1 illustrates our categorization on related approaches. Generally, there are three main association types, namely *similarity-based*, *sequential associations* and *basket-oriented*. In this chapter, we will review the literature on related works of those branches. Especially we concentrate on methods modeling the *sequential-* and *basket-oriented associations* for *top-K recommendation*, which are particularly relevant to this dissertation.

2.1 Modeling Similarity-Based Associations

The objective of this algorithm family is to compute the similarity of items. This similarity could be generated either from intrinsic information (e.g., item attributes) or extrinsic manners (e.g., user ratings). The usage of descriptive information is firstly applied in *content-based* RS [99], where items with similar attributes are grouped into a category. It is commonly used for the *substitutable* recommendation. In another way, the likeness of an item pair is considered in terms of being rated similarly by users [9]. A user is recommended highly-rated items, which are similar to what he adopted. This approach is often referred to as *item-based collaborative* *filtering* [117]. Based on the similarity knowledge, users are suggested candidates that are similar to what they adopted formerly.

Content-based. In this algorithm group, each item can be represented by a vector [99]. This vector could be a structured representation, where each column corresponds to an attribute type (e.g., brand, size) and the respective value is the item's attribute. In another way, it is possible to contain tf-idf values of words in the item's description. Let us denote $\vec{i}, \vec{j}$ to the vector representations of two items i, j respectively. The similarity between i, j is often measured by the *cosine similarity* [73]:

$$sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i}.\vec{j}}{||\vec{i}||_2 * ||\vec{j}||_2}$$

Another direction making use of these vectors is to cluster items into groups of similarity [68]. YouTube researchers cluster videos to make recommendations using the descriptions [23]. [4] mines query logs to generate new query recommendations. The invention of topic modeling [10] triggers several research works on modeling the categorical association. [142] explores relational concepts in the scientific article data to support the finding of relevant works. [29] infers topics from reader histories and content to recommend personalized articles of interest. Generally, these methods have a tendency to exploit the similarity between items for *"substitutable"* recommendations.

Item-based Collaborative Filtering. Unlike the *content-based* approach, the presence of user plays a substantially role in this algorithm family [117]. Here, the explicit feedback (a.k.a. ratings) of users are utilized to compute the similarity between items. Let us denote U as the user set, R_{ui} as the rating of a user u on the $i-$item and $\bar{R}_i$ as the average rating of the i-item. The similarity between two items i, j can be computed by the *Pearson-r* correlation as follows:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{ui} - \bar{R}_i)(R_{uj} - \bar{R}_j)}{\sqrt{\sum_{u \in U}(R_{ui} - \bar{R}_i)^2}\sqrt{\sum_{u \in U}(R_{uj} - \bar{R}_j)^2}}$$

Clearly, different users may have different rating ranges. It is captured in the *adjusted Cosine similarity*, which takes into account the user-bias by the average rating $\bar{R_u}$:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{ui} - \bar{R_u})(R_{uj} - \bar{R_u})}{\sqrt{\sum_{u \in U}(R_{ui} - \bar{R_u})^2}\sqrt{\sum_{u \in U}(R_{uj} - \bar{R_u})^2}}$$

There is another approach [24], named *Conditional Probability-based Similarity*, relies on the item frequencies as follows:

$$sim(i,j) = \frac{\text{Freq}(i,j)}{\text{Freq}(i) \times (\text{Freq}(j))^{\alpha}}$$

where $\alpha \in [0,1]$ is a hyper-parameter. If $\alpha = 0$, it becomes identical to $P(j|i)$.

Inspired by these traditional similarity-based methods, [145] proposes a probabilistic fusion framework, where ratings by similar users towards similar items are utilized to smooth the rating predictions. [52] improves recommendations for cold-start users by combining an item-based model and a trust network as the additional knowledge. Likewise, [28] couples the tagging graph with item-based collaborative filtering for the tag recommendation task.

2.2 Modeling Sequential Associations

Here, we narrow down to review related approaches on modeling sequential associations. This association type implies the sequential dependency of an adoption on its preceding adoptions. In our context, user can adopt either one item or a basket of items at a time step. Hence, single-item and basket sequences are typical input data forms. [103] classifies these approaches as “sequence-aware” RS. Figure 2.2 shows the taxonomy of reviewed techniques and our corresponding research works.

In Table 2.1, related works are listed down and grouped into the five main categories namely *pattern mining, metric embedding, factorization, Markov models* and *neural networks*. These algorithm groups will be thoroughly discussed in next paragraphs.

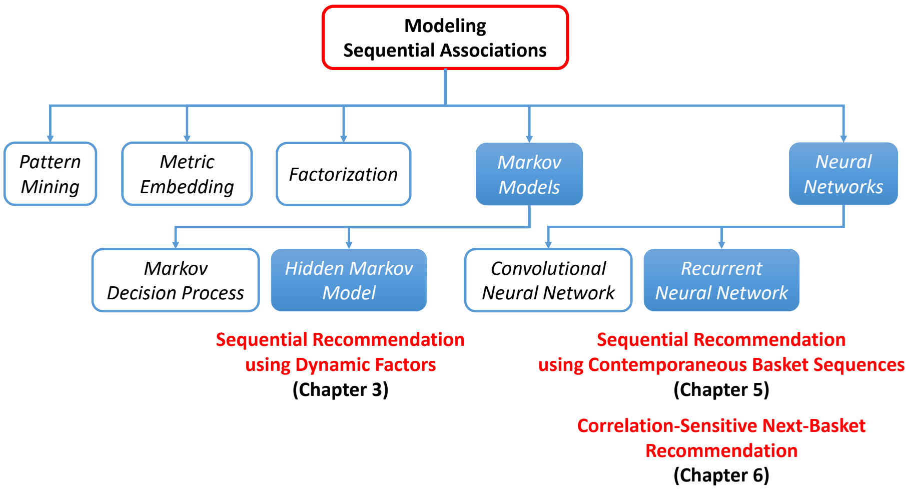

Modeling
Sequential Associations
Pattern Mining
Metric Embedding
Factorization
Markov Models
Neural Networks
Markov Decision Process
Hidden Markov Model
Convolutional Neural Network
Recurrent Neural Network
Sequential Recommendation using Dynamic Factors (Chapter 3)
Sequential Recommendation using Contemporaneous Basket Sequences (Chapter 5)
Correlation-Sensitive Next-Basket Recommendation (Chapter 6)

Figure 2.2: The Taxonomy of Modeling Sequential Associations with Our Corresponding Research

Pattern Mining		Argawal et al. [2]	Rudin et al. [113]
		Mobasher et al. [90]	Sarwar et al. [116]
		Parameswaran et al. [96]	Yap et al. [157]
Metric Embedding		Baeza et al. [5]	Grbovic et al. [31]
		Chen et al. [17, 18]	Tagami et al. [130]
		Feng et al. [27]	Vasile et al. [136]
Factorization		Cheng et al. [20]	Pasricha et al. [97]
		He et al. [37, 39, 38]	Wan et al. [139]
		Liu et al. [82]	Zhao et al. [166]
		Lian et al. [71]	
Markov Models	Markov Decision Process	Brafman et al. [12]	Shani et al. [121]
		Sahoo et al. [114]	Tavakol et al. [132]
	Hidden Markov Model	Hu et al. [48]	Rabiner et al. [105]
		Juang et al. [54]	
Neural Networks	Recurrent Neural Networks	Hidasi et al. [42, 43, 41]	Song et al. [127]
		Lipton et al. [74]	Villatel et al. [138]
		Li et al. [66]	Wu et al. [151]
		Li et al. [70]	Wu et al. [152]
		Liu et al. [77, 79]	Wang et al. [144]
		Neil et al. [92]	Zhang et al. [165]
		Ren et al. [106]	Zhu et al. [168]
	Convolutional Neural Networks	Chen et al. [155]	Tuan et al. [135]
		Liu et al. [80]	Yuan et al. [163]
		Tang et al. [131]	

Table 2.1: Related works on modeling Sequential Associations

Pattern Mining. As a straightforward solution, association rules [2] are exploited to find sequential patterns for recommendations. [90] proposes a scalable framework for the user navigation on Web, mining both sequential and non-sequential pattern from clickstream data. This framework provides effective data structures to store sequential patterns as well as effective real-time recommendations. [96] deals with the *precedence mining* task to suggest students what subjects they might consider next. The novelty compared to the traditional solution, is the proposal of probabilistic approaches. As another variant of mining association rules, [113] improves the next item recommendation task by regulating a trade-off between accuracy on the training set and generalization ability. Meanwhile, [157] presents a personalized sequential pattern mining-based approach using a novel *Competence Score* measure. In general, this algorithm family is simple and easy to implement but it is limited by “known” patterns appearing in training transactions.

Metric Embedding. The main idea of this algorithm group is to learn the item embedding to preserve the sequential dependency, which is presented in terms of transition probabilities. Early, [17] proposes Latent Markov Embedding that each song can be projected in the embedding space. The distance between two songs space is proportional to their transition probability estimated from the playlist data. Subsequently, Chen et al. [18] upgrades this embedding method by constraining not only on sequential orders but also co-occurrences of songs. With the similar idea, [5, 153] generate personalized recommendations for what next-song and mobile applications that user are going to select. Inspired by *word2vec* [88], [31] introduces a neural language-based algorithm to learn user and product embeddings for user-to-product and product-to-product predictions. [136] incorporates products' metadata in learning embeddings. Likewise, [130] infers the latent representations of users from their activities on the Web for the contextual advertisement task. In another domain, there are several works dealing with point-of-interest (POI) data. [27] presents the personalized ranking metric embedding approach for recommending next new point-of-interest. Instead of conserving transition probability values, this approach preserves the relative comparison between these probabilities.

Factorization. This family is an analogous direction to the *metric embedding* approach. Instead of using embedding to approximate the transition probability, it directly factorizes these probabilities into a number of latent factors. [110] conducts joint factorization of the user-by-item rating and item-by-item transition matrices. It is a base to build a more abstractive model, named "Factorization machines" [107]. Relying on the idea in [110], [20] proposes a POI recommendation method using geographical constraints while [37] incorporates latent behavior patterns. Likewise, [82] predicts users' preference transitions over categories of POI and suggests them with locations in the corresponding categories. [71] tackles the next check-in recommendation task by modeling both short- and long-term preferences. Applied to the e-commerce domain, [166] improves the factorization-based sequential model using purchased intervals of items. [139] factorizes monotonic behavior chains (e.g., click, purchase, review) for item recommendation. In the similar way, [39] incorporates item similarities as the additional information in modeling sequential associations. Inspired by *translation-based* techniques, [38, 97] model personalized item sequences for next-item recommendations.

Markov Models. This family is a collection of generative models, of which Hidden Markov Model (HMM) [105] is a representative. It has been shown to be effective in various applications, including speech- and handwriting-recognition [54, 48]. At each time step t of HMM, the user preference is presented by a hidden state variable X_t. X_t is associated emission E and transition probabilities T. The former measures how likelihood items ($V = \{v\}$) are adopted ($P(v|X_t)$ while the latter refers to the switching probability (e.g., $P(X_{t+1} = s'|X_t = s)$) from a hidden state ($X_t = s$) to another hidden state ($X_{t+1} = s'$). In Chapter 3, we seek to model dynamic context-biased transition and user-biased emission. To make the effects of these dynamic factors clear, we build on the foundation of HMM, and focus our comparisons against this base platform. Aside from HMM, there could potentially be different ways to tackle this problem such as probabilistic automata [26] and recurrent neural networks [87]. [114] introduces a variant of HMM, which assumes that users may change their ordinal preferences over time rather than considering dynamic context-bias transition. As a member of the Markovian family, Markov decision processes (MDP) have a similar structure to ours, but their behaviors are different. Given a hidden state $X_t = s$ at time step t, the decision maker can choose any action a (aka policy) from a predefined set. The process responds at the next time step by randomly switching into a new state $X_{t+1} = s'$ with the probability $P(s'|s, a)$, informing the decision maker a corresponding reward $R_a(s, s')$. These probabilities are estimated by seeking "optimal policies" to maximize the responded reward. Considering recommendations as a stochastic sequential decision problem, [12] proposes an MDP-based recommendation system, where items are "actions" and the adoptions of users are "rewards". This approach is somehow closer to the real behavior of users on the Web. However, the computational and representational overhead are its significant problems. [121] is an upgraded version of MDP-based RS, of which the initial states are based on an n-gram predictive model. Another variant [132], named factored MDP, predicts the user's goal (aka topic) within a shopping session. This model is an ensemble of a number of MDP, respecting to the number of items' attribute categories. In each MDP, actions are referred to the options of the item attribute.

Neural Networks. Some variants of neural networks are proposed and applied to model sequences such as Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), etc. Using RNN, [74] shows significant improvements in learning long-term sequential dependencies from sequences. [165] proposes a RNN-based framework solve the click prediction problem in sponsored search systems. To generate recommendations during a browsing session, [42, 41] modify the basic Gated Recurrent Unit, a variant of RNN, to better fit the training data by using session-parallel mini-batches. [66] upgrade this approach with the attention mechanism while [138] adds layer normalization and tied embedding matrix techniques.[70] aggregates the short-term behaviors within a session with users' long-term stable preferences for next-item recommendations. [152, 127] exploit the graph information within the current session while [106] takes into account the repeat consumption behavior. In order to build context-aware RNN-based RS, [43, 77, 79] take into account contextual information such as time, location, weather. [151] introduces the recurrent recommender networks with dual RNNs via fitting concurrently items' and users' rating sequences. Likewise, [144] considers local coherence of nearby feedback to model the dynamics of users and items for the rating prediction problem. [92, 168] upgrades LSTM with a time gate, which explicitly models intervals between time steps. In a different direction, Convolutional Neural Networks (CNN), a powerful model for the image classification task, are also exploited in the recommendation scenarios. Using CNN, [80] solves a click prediction on advertisements. [135] presents another CNN-based model incorporating content features as additional information to enhance the performance of the session-based recommendation task. [131] proposes the Convolutional Sequence Embedding model to tackle the personalized sequential recommendation task. It takes advantage of CNN to learn sequential associations and Latent Factor Model (LFM) to learn personalization. [163] improves this approach by residual learning, the combination of masked filters and 1D dilated convolutions. Dealing with the same task, [155] is the most recent research making use of memory-augmented neural networks, which provide a flexible way to maintain and update user preferences for sequential recommendations.

Hybrid Models. In addition to the above mentioned works, there exist several hybrid methods dealing with the sequential recommendation problem. [16] models the scenario where users "lose interest" over time. [53] takes into account the life stage of a consumer, e.g., products for babies of different ages, while [156] intends to model the evolution that advance ”forward” in event sequences without going "backward". [143] seeks to predict not what, but rather when to recommend an item. In a different goal, [75] tackles the repeat buying behavior prediction using users' and items' features. [19] considers how changes in social relationships over time may affect a user's receptiveness or interest to change. [149] incorporates spatial, temporal and payment information to generate category-level recommendation. [93] proposes *RecWalk*, which performs random walks on inter-item transition probability matrix to infer meaningful relationships for next-item recommendation. [162] presents a multi-order attentive ranking model to capture the union-level item interaction on item sequences, which an item adoption is triggered by a set of previous items. Overall, these models are not considered as our baselines because they capture neither context-aware factors, nor basket sequences.

2.3 Modeling Basket-Oriented Associations

The basket-oriented association signifies the correlation among basket-items. In Chapter 4, we aim to model this association type for the basket-completion recommendation task. It is to recommend a user, currently holding a basket of items, what could be adopted next. Our solution mainly relies on factorizing basket-oriented associations. Figure 2.3 shows the taxonomy of modeling this association type corresponding to our research direction. Specifically, Table 2.2 lists related works, with respect to the four main algorithm groups including *association rule, graph-based, factorization* and *neural networks*. Next, we will give the literature review on these families.

Association Rule. [2] introduces fast algorithms for mining association rules on transactional data. An *association rule* $X \implies Y$ indicates the sequential dependency between two disjoint itemsets X, Y (i.e., $X \cap Y = \emptyset$). These association rules are filtered out to find meaningful patterns using minimal support and minimal confidence thresholds. Let us denote $f(X)$ to the number of transactions containing X and N is the total number of transactions. The support s and confidence c of an association rule $X \implies Y$ are computed by:

$$s(X \implies Y) = \frac{f(X \cup Y)}{N}$$

$$c(X \implies Y) = \frac{f(X \cup Y)}{f(X)}$$

Clearly, the rule is strongly dependent on the union of X and Y, referred to as the associations of items in X, Y. The original aim of association rule mining is to find these insightful associations rather than making recommendations. In the beginning stage, the research focus was mainly on computational efficiency, with pruning strategies such as Apriori [2] or FP-tree [36]. The rules are general rules, and are not personalized. Some works describe ways to use association rules for personalized item recommendation [157]. For experimental comparison in Chapter 4, we follow the approach in [116] as a baseline. Other works inspired by association rules are not directly comparable. [56] utilizes information on product categories for multi-level rules. Taking into account the features of the products in shopping baskets, [35] finds associations rules that match these features directly. [146] considers association rules across baskets, instead of within a basket. [100] exploits bigram rules, with an item each in antecedent and consequent.

Graph-Based. These methods consider the recommendation as a link prediction problem task on graphs. If there exists an edge between two items, it implies the relevance, which is used to generate recommendations. As one of the pioneers, [67] proposes basket-sensitive recommendation using random walks on the user-item bipartite graph. However, it is not designed for personalization or exploiting actual correlations among basket-items. [154] exploits a session-based graph to capture the effects of recent items as well as items across sessions. [30] constructs a *concept graph* from a scientific article corpus to suggest students relevant technical material. [85, 86] learn simultaneously multiple types of relationships such as “complement”, “substitute” in the product graph. In comparison, these works are not considers as our baselines because they are neither basket-sensitive nor personalized.

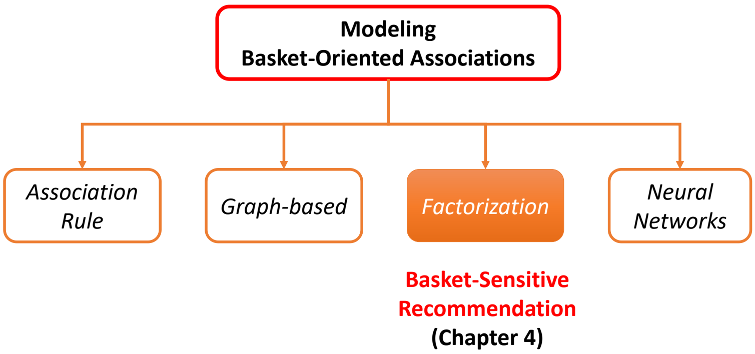


Figure 2.3: The Taxonomy of Modeling Basket-Oriented Associations with Our Corresponding Research

Association Rule	Graph-based	Factorization	Neural Networks
Argawal et al. [2]	Li et al. [67]	Cao et al. [15]	Shen et al. [122]
Han et al. [36]	McAuley et al. [85]	Liu et al. [76]	Hidasi et al. [42, 43]
Han et al. [35]	McAuley et al. [86]	Liu et at. [78]	Hu et al. [49]
Kim et al. [56]	Gordon et al. [30]	Liang et al. [72]	Liu et al. [81]
Sarwar et al. [116]	Xiang et al. [154]	Nguyen et al. [133]	Quadrana et al. [104]
Pradel et al. [100]		Rendle et al. [111]	Wang et al. [148]
Wang et al. [146]			
Yap et al. [157]			

Table 2.2: Related works on modeling Basket-Oriented Associations

Factorization. This group takes the advantage of latent representations, where each item is associated with a latent vector. [72] utilizes item co-occurrences as a form of regularization for matrix factorization. Likewise, [15] relies on the rich signal about item co-occurrences within lists (e.g., playlists, booklists) as a promising factor for jointly recommending items and user generated lists. [133] tackles a similar problem to us, which uses previous clicked items of a user as the context to determine future items what he will click on. Another work using the context-aware factor, [76] claims that conditioning on all context items is not optimal so that they propose a contextual selection strategy for embedding models. Although the aforementioned works exploit the correlation among items, they neither take into account the notion of basket nor the association among context items.

Neural Networks. The development of neural network-based models triggers state-of-the-art results in numerous applications, which the modeling of basket-oriented associations is not an exception. [122] proposes a structured deep neural network using back-propagation to learn co-occurrence data (e.g., products in transactional data, friends in social networks). Taking into account the order of items within a session (i.e., basket), [42] deals with un-personalized session-based recommendations using recurrent neural networks. Inspired by this work, [104] proposes a personalized variant, which learns from a cross-session sequence. Thoroughly, these works are not comparative to our work in Chapter 4, where the sequential information is unavailable and the personalization is required. Dealing with the same problem to ours, [49] recommends items based on the basket context, which are combined from context items' embeddings. [148, 81] recently applies the attention mechanism to infer the contextual embedding.

Additionally, there are some other efforts in modeling co-occurrences. Incorporating co-occurrence information and category labels, [137] introduces a Siamese CNNs structure to learn compatibility across item categories. *Word2Vec* [88] is invented to capture the correlation of words in sentences. *Product2Vec* [31] is proposed with the similar idea to learn products' embeddings in e-commerce sites to build RS. Incorporating side-information to *Product2Vec*, [136] presents the *Meta-Product2Vec* method to capture the similarities between items.

Hybrid Models. An orthogonal direction is to recommend a user's *next basket*. The key association is *sequence* based on time. One approach is based on integrating matrix factorization and Markov chains [110]. In contrast, our intent in Chapter 4 is to predict which item to be added into the *current* basket, and the key is correlation among items within the basket. Another problem is bundle recommendation [167] to recommend a bundle of items. This is akin to next-basket recommendation, a different scenario from ours. Another is taxonomy-induced associations [120, 59]. There is also similarity- or co-occurrence-induced associations [78, 72]. They recommend items independently, whereas we factor in the items in the user's basket to arrive at the personalized recommendation.

Metric Embedding	Factorization	Recurrent Neural Networks
Wang et al. [147]	Rendle et al. [110]	Bai et al. [7]
	Pathak et al. [98]	Ying et al. [160]
	Wan et al. [141]	Yu et al. [161]

Table 2.3: Related works on modeling jointly Basket-Oriented and Sequential Associations

2.4 Modeling Jointly Sequential- and Basket-Oriented Associations

Table 2.3 summarizes related works of typical approaches on modeling sequences and basket-level associations concurrently for sequential recommendations. With the objective of combining the "long-term" (items a user generally likes) and "short-term" preferences (items frequently consumed within a session), [154] models the problem as random walks in a session-based temporal graph. [160] proposes a hierarchical attention network that combines the long-term preference of adopted session (a.k.a., basket) sequences and the short-term preference of the currently holding basket. Each user (or item) is represented by an embedding vector. In order to generate basket-level recommendations, [147] also introduces a hierarchical aggregation architecture between users and the most recent adopted basket. The representation of a basket is inferred by aggregating the representations of its items. Using the similar idea, [161] presents another RNN-based approach dealing with the next-basket recommendation problem. Recently, [141] proposes a personalized model which infers the representation of a given basket via its pairwise relationships (e.g., item-item, user-item). [7] improves this approach by incorporating item attribute information in a hierarchical attentive structure.

In Chapter 5, we focus on modeling a *pair* of contemporaneous basket sequences. Each sequence is built *across* sessions (each session is a basket). Most of the mentioned works are preoccupied with only *one* sequence type, and that sequence consists of *individual* items (not baskets). They are not comparable to our work, as they model neither baskets, nor contemporaneous sequences. In the experiment section of Chapter 5, we will compare to some of these baselines having one sequence type to investigate the effects of modeling a pair of sequence types. In a different direction, there are instances where “twin” networks are applied for other purposes than next-item recommendation. These applications include question answering [22], text similarity [91], and image matching [58]. In such cases, the two distinct inputs are assumed to have largely similar meanings. In our case, the relationship may be asymmetric, with one being the target sequence (to predict its next item), and the other the support sequence (to assist in predicting the target sequence). For another instance, in neural machine translation [6, 129], the objective is to transform a sequence in one language to another language. In our case, the objective is not transformation, but rather predicting the next item in the target sequence.

Generally, the mentioned works deal with the same task, known as *next-basket recomendation*, where items are independently recommended. This limitation motivates us to consider a novel formulation, referred to as *correlation-sensitive next-basket recommendation* in Chapter 6. Modeling users' preferences from basket sequences, we aim to recommend the next basket with correlated item. Our research is somehow related to the bundle recommendation problem [112]. [167] introduces a construction algorithm, where items are sequentially added with the constraint on selected ones. [98] starts from a ranking problem on existing bundles of games to infer the embeddings of items and users. These embeddings are later exploited to generate a bundle recommendation. As a different formulation, [102, 119] present their research on recommending packages to groups of users. Comparing to ours, these works neglect either correlative associations in recommended baskets or sequential associations. In some sense, our work in Chapter 6 relates to the *list-wise recommendation* problem [51, 125]. However, their focus is items' ranking rather than modeling correlative dependencies per se.

Chapter 3

Sequential Recommendation using Dynamic Factors

Users express their preferences via the adoptions of items such as clicking, rating or liking, etc. The notion of sequence is generated when considering time-sensitive on the user-item adoptions. Given a sequence, associations between items somehow imply "sequential dependency" (a.k.a. "sequential preference"), manifests itself in scenarios such as song or video playlists, topics one reads or writes about in social media, etc. Modeling these associations is helpful to predict the next item expected to appear beyond this sequence (e.g., what video to watch next, what song to listen next, etc). This task is known as the sequential recommendation. Moreover, the common approach (before Deep Learning becomes popular) to model sequential preferences relies primarily on the first-order sequential dependency, i.e., which item follows another item. However, there are other important factors, due to either the user or the context, which may dynamically affect the way a sequence unfolds. Therefore, it raises the need to develop generative modeling of sequences, incorporating dynamic user-biased emission and context-biased transition for sequential preference.

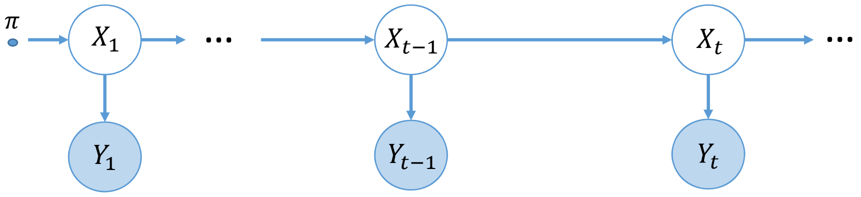


Figure 3.1: A standard HMM for sequential preferences

Preliminaries. Towards capturing sequential preferences, our model builds upon HMM. The standard HMM assumes a series of discrete time steps $t = 1, 2, \ldots$, where an item Y_t can be observed at step t. To model the sequential effect in this series of observed items, HMM employs a Markov chain over a *latent* finite state space across the time steps. As illustrated in Figure 3.1, at each time step t a latent state X_t is transitioned from the previous state X_{t-1} in a Markovian manner, i.e., $P(X_t|X_{t-1}, X_{t-2}, \ldots, X_1) \equiv P(X_t|X_{t-1})$, known as the *transition* probability.

Formally, consider an HMM with a set of observable items $\mathcal{Y}$ and a set of latent states $\mathcal{X}$. It can be fully specified by a triplet of parameters $\theta = (\pi, A, B)$, such that $\forall x, u \in \mathcal{X}, y \in \mathcal{Y}, t \in \{1, 2, \ldots\}$,

- π is the initial state distribution with $\pi_x \triangleq P(X_1 = x)$;
- A is the transition matrix with $A_{xu} = P(X_t = u|X_{t-1} = x)$;
- B is the emission matrix with $B_{xy} = P(Y_t = y|X_t = x)$.

Given a sequence of items $Y_1, \ldots, Y_t$, the optimal parameters θ^* can be learned by maximizing likelihood (Eq (3.1)). Note that we can easily extend the likelihood function to accommodate multiple sequences, but for simplicity we only demonstrate with a single sequence throughout the technical discussion. Moreover, given θ^* and a sequence of items $Y_1, \ldots, Y_t$, the next item y^* can be predicted by maximum a posteriori probability (Eq (3.2)). Both learning and prediction can be efficiently solved using the forward-backward algorithm [105].

$$\theta^* = \arg\max_\theta P(Y_1, ..., Y_t; \theta) \tag{3.1}$$

$$y^* = \arg\max_y P(Y_{t+1} = y|Y_1, \ldots, Y_t; \theta^*) \tag{3.2}$$

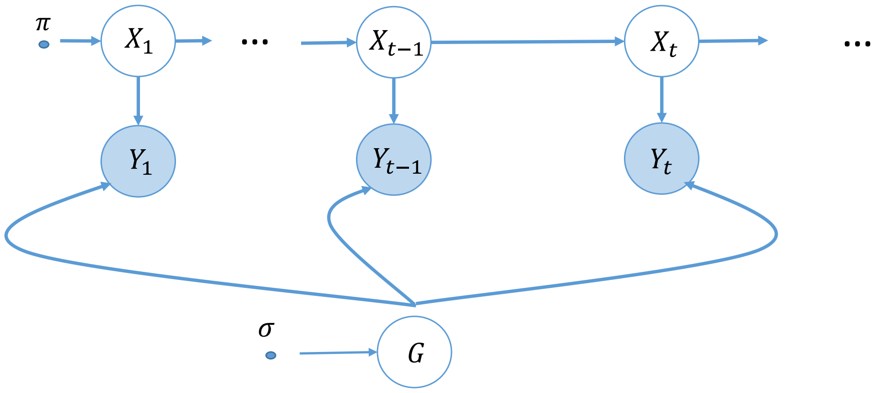


Figure 3.2: Sequential models with dynamic user-biased emissions

3.1 Models

In a standard HMM, item emission probabilities are invariant across users, and state transition probabilities are independent of contexts at different times. However, these assumptions often deviate from real-world scenarios, in which different users and contexts may have important bearing on emissions and transitions. In this section, we model dynamic emissions and transitions respectively, and ultimately jointly, to better capture sequential preferences.

3.1.1 Modeling Dynamic User-Biased Emissions (SEQ-E)

It is often attractive to consider personalized preferences [110], where different user sequences may exhibit different emissions even though they share a similar transition. For instance, while two users both transit from soft rock to hard rock in their respective playlist, they might still choose songs of different artists in each genre. As another example, two users both transit from spring to summer in their apparel purchases, but still prefer different brands in each season. However, a fully personalized model catered to every individual user is often impractical due to inadequate training data for each user. We hypothesize that there exist different groups such that users across groups manifest different emission probabilities, whereas users in the same group share the same emission probabilities.

In Figure 3.2, we introduce a variable G_u to represent the group assignment of each user u. For simplicity, our technical formulation presents a single sequence and hence only one user. Thus, we omit the user notation u when no ambiguity arises. Assuming a set of groups $\mathcal{G}$, the new model can be formally specified by the parameters (π, σ, A, B), such that $\forall x \in \mathcal{X}, y \in \mathcal{Y}, g \in \mathcal{G}, t \in \{1, 2, \ldots\}$,

- π and A are the same as in a standard HMM;
- σ is the group distribution with $\sigma_g = P(G = g)$;
- B is the new emission tensor with $B_{gxy} = P(Y_t = y | X_t = x, G = g)$.

3.1.2 Modeling Dynamic Context-Biased Transitions (SEQ-T)

In standard HMM, the transition matrix is invariant over time. In real-world applications, this assumption may not hold. The transition probability may change depending on contexts that vary with time. Consider modeling a playlist of songs, where the transitions between genres are captured. The transition probabilities could be influenced by characteristics of the current song (e.g., artist, lyrics and sentiment). A fan of the current artist may break her usual pattern of genre transition and stick to genres by the same artist for the next few songs. As another example, a user purchasing apparels throughout the year may follow seasonal transitions. If satisfied with certain qualities (e.g., material and style) of past purchases, she may buy more such apparels out of season to secure discounts, breaking the usual seasonal pattern. We call such characteristics *context features*.

It is infeasible to differentiate transition probabilities by individual context features directly, which would blow up the parameter space and thus pose serious computational and data sparsity obstacles. Instead, we propose to model a single context factor that directly influences the next transition. The context factor, being latent, manifests itself through the observable context features.

As illustrated in Figure 3.3, consider a set of context features $F = \{F^1, F^2, \ldots\}$. As feature values vary over time, let $F_t = (F_t^1, F_t^2, \ldots)$ denote the feature vector at

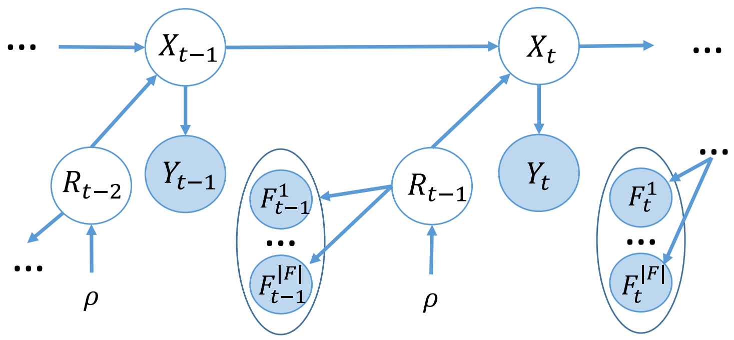


Figure 3.3: Sequential models with dynamic context-biased transitions

time t. Each feature F^i takes a set of values $\mathcal{F}^i$, i.e., $F_t^i \in \mathcal{F}^i, \forall i \in \{1, ..., |F|\}, t \in \{1, 2, \ldots\}$. Similarly, let R_t denote the latent context factor at time t, and $\mathcal{R}$ denote the set of context factor levels, i.e., $R_t \in \mathcal{R}, \forall t \in \{1, 2, \ldots\}$. Finally, the model can be specified by the parameters (π, ρ, A, B, C), such that $\forall x, u \in \mathcal{X}, i \in \{1, \ldots, |F|\}, f \in \mathcal{F}_i, t \in \{1, 2, \ldots\}$,

- π and B are the same as in a standard HMM;
- ρ is the distribution of the latent context factor with $\rho_r = P(R_t = r)$;
- C is the feature probability matrix with $C_{rif} = P(F_t^i = f|R_t = r)$;
- A is the new transition tensor with $A_{rxu} = P(X_t = u|X_{t-1} = x, R_{t-1} = r)$.

3.1.3 Joint Model (SEQ*)

As discussed, user groups and context features can dynamically bias the emission and transition probabilities, respectively. Here, we consider both users and contexts in a joint model, as shown in Figure 3.4. Accounting for all the parameters defined earlier, the joint model is specified by a six-tuple $\theta = (\pi, \sigma, \rho, A, B, C)$. The algorithm for learning and inference will be discussed in the next section.

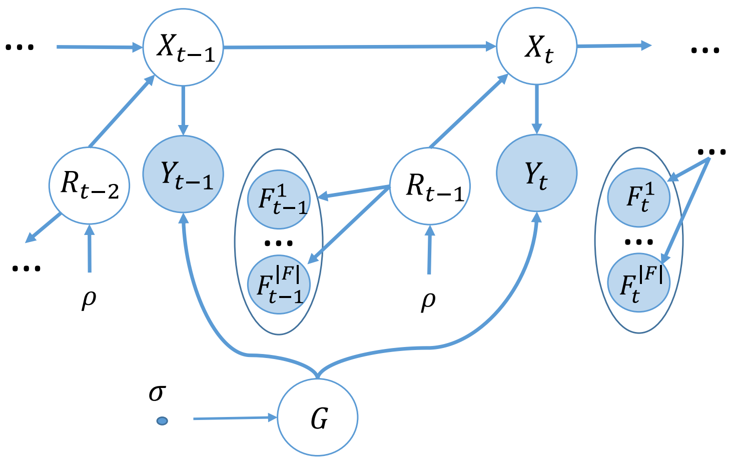


Figure 3.4: Sequential models with both dynamic user-biased emissions and context-biased transitions

3.2 Learning and Prediction

We now present efficient learning and prediction algorithms for the joint model. Note that the user and context-biased models are only degenerate cases of the joint model— the former assumes one context factor level (i.e., $|\mathcal{R}| = 1$) and no features (i.e., $F = \emptyset$), whereas the latter assumes one user group (i.e., $|\mathcal{G}| = 1$).

3.2.1 Parameter Learning

The goal of learning is to optimize the parameters $\theta = (\pi, \sigma, \rho, A, B, C)$ through maximum likelihood, given the observed items and features. Consider a sequence of $T > 1$ time steps. Let $\underline{Y} \triangleq (Y_1, \ldots, Y_T)$ as a shorthand; and similarly for $\underline{F}, \underline{X}, \underline{R}$. Subsequently, the optimal parameters can be obtained as follows.

$$\theta^* = \arg\max_\theta \log P(\underline{Y}, \underline{F}; \theta) \tag{3.3}$$

We demonstrate with one sequence for simpler notations. The algorithm can be trivially extended to enable multiple sequences as briefly described later.

Expectation Maximization (EM). We apply the EM algorithm to solve the above optimization problem. Each iteration consists of two steps below.

- *E-step*. Given parameters θ' from the last iteration (or random ones in the first iteration), calculate the expectation of the log likelihood function:

$$Q(\theta|\theta') = \textstyle\sum_{\underline{X},G,\underline{R}} P(\underline{X}, G, \underline{R}|\underline{Y}, \underline{F}; \theta') \log P(\underline{Y}, \underline{F}, \underline{X}, G, \underline{R}; \theta') \tag{3.4}$$

- *M-step*. Update the parametes $\theta = \arg\max_\theta Q(\theta|\theta')$.

Given the graphical model in Figure 3.4, the joint probability $P(\underline{Y}, \underline{F}, \underline{X}, G, \underline{R})$ can be factorized as

$$P(G)P(X_1) \cdot \prod_{t=1}^{T} \left(P(Y_t|G, X_t)P(R_t) \prod_{i=1}^{|F|} P(F_t^i|R_t) \right) \cdot \prod_{t=1}^{T-1} P(X_{t+1}|X_t, R_t). \tag{3.5}$$

Maximizing the expectation $Q(\theta|\theta')$ is equivalent to maximize the following, assuming that $Y_t = y_t$ and $F_t^i = f_t^i$ are observed, $\forall t \in \{1, \ldots, T\}, i \in \{1, \ldots, |F|\}$.

$$\begin{aligned}
&\textstyle\sum_{x\in\mathcal{X}} P(X_1 = x|\underline{Y}, \underline{F}; \theta') \log \pi_x + \sum_{g\in\mathcal{G}} P(G = g|\underline{Y}, \underline{F}; \theta') \log \sigma_g \\
&+ \textstyle\sum_{t=1}^{T} \sum_{r\in\mathcal{R}} P(R_t = r|\underline{Y}, \underline{F}; \theta') \log \rho_r \\
&+ \textstyle\sum_{t=1}^{T-1} \sum_{x\in\mathcal{X}} \sum_{u\in\mathcal{X}} \sum_{r\in\mathcal{R}} P(R_t = r, X_t = x, X_{t+1} = u|\underline{Y}, \underline{F}; \theta') \log A_{rxu} \\
&+ \textstyle\sum_{t=1}^{T} \sum_{x\in\mathcal{X}} \sum_{g\in\mathcal{G}} P(X_t = x, G = g|\underline{Y}, \underline{F}; \theta') \log B_{gxy_t} \\
&+ \textstyle\sum_{t=1}^{T} \sum_{i=1}^{|F|} \sum_{r\in\mathcal{R}} P(R_t = r|\underline{Y}, \underline{F}; \theta') \log C_{rif_t^i}
\end{aligned} \tag{3.6}$$

The optimization problem is further constrained by laws of probability, such that $\sum_{x\in\mathcal{X}} \pi_x = 1, \sum_{g\in\mathcal{G}} \sigma_g = 1, \sum_{r\in\mathcal{R}} \rho_r = 1, \sum_{u\in\mathcal{X}} A_{rxu} = 1, \sum_{y\in\mathcal{Y}} B_{gxy} = 1$ and $\sum_{f\in\mathcal{F}^i} C_{rif} = 1$. Applying Lagrange multipliers, we can derive the following updating rules.

$$\begin{aligned}
\pi_x &= \frac{P(X_1 = x|\underline{Y}, \underline{F}; \theta')}{1} = \frac{\sum_{g\in\mathcal{G}} \sum_{r\in\mathcal{R}} \gamma_{gxr}(1)}{1}, \qquad (3.7)\\
\sigma_g &= \frac{P(G = g|\underline{Y}, \underline{F}; \theta')}{1} = \frac{\sum_{x\in\mathcal{X}} \sum_{r\in\mathcal{R}} \gamma_{gxr}(1)}{1},\\
\rho_r &= \frac{\sum_{t=1}^{T} P(R_t = r|\underline{Y}, \underline{F}; \theta')}{\sum_{t=1}^{T} \sum_{k\in\mathcal{R}} P(R_t = k|\underline{Y}, \underline{F}; \theta')} = \frac{\sum_{g\in\mathcal{G}} \sum_{x\in\mathcal{X}} \sum_{t=1}^{T} \gamma_{gxr}(t)}{T},\\
A_{rxu} &= \frac{\sum_{t=1}^{T-1} P(R_t = r, X_t = x, X_{t+1} = u|\underline{Y}, \underline{F}; \theta')}{\sum_{t=1}^{T-1} P(R_t = r, X_t = x|\underline{Y}, \underline{F}; \theta')} = \frac{\sum_{t=1}^{T-1} \sum_{g\in\mathcal{G}} \xi_{gxur}(t)}{\sum_{t=1}^{T-1} \sum_{g\in\mathcal{G}} \gamma_{gxr}(t)},\\
B_{gxy} &= \frac{\sum_{t=1}^{T} P(X_t = x, G = g|\underline{Y}, \underline{F}; \theta') I(y_t = y)}{\sum_{t=1}^{T} P(X_t = x, G = g|\underline{Y}, \underline{F}; \theta')} = \frac{\sum_{t=1}^{T} \sum_{r\in\mathcal{R}} \gamma_{gxr}(t) I(y_t = y)}{\sum_{t=1}^{T} \sum_{r\in\mathcal{R}} \gamma_{gxr}(t)},\\
C_{rif} &= \frac{\sum_{t=1}^{T} P(R_t = r|\underline{Y}, \underline{F}; \theta') I(f_t^i = f)}{\sum_{t=1}^{T} P(R_t = r|\underline{Y}, \underline{F}; \theta')} = \frac{\sum_{t=1}^{T} \sum_{g\in\mathcal{G}} \sum_{x\in\mathcal{X}} \gamma_{gxr}(t) I(f_t^i = f)}{\sum_{t=1}^{T} \sum_{g\in\mathcal{G}} \sum_{x\in\mathcal{X}} \gamma_{gxr}(t)},
\end{aligned}$$

where $I(\cdot)$ is an indicator function and

$$\gamma_{gxr}(t) \triangleq P(G = g, X_t = x, R_t = r|\underline{Y}, \underline{F}; \theta'), \qquad (3.8)$$

$$\xi_{gxur}(t) \triangleq P(G = g, X_t = x, X_{t+1} = u, R_t = r|\underline{Y}, \underline{F}; \theta'). \qquad (3.9)$$

Note that, to account for multiple sequences, in each updating rule we need to respectively sum up the denominator and numerator over all the sequences.

Inference. To efficiently apply the updating rules, we must solve the inference problems for $\gamma_{gxr}(t)$ and $\xi_{gxur}(t)$ in Eq (3.8) and (3.9). Towards these two goals, similar to the forward-backward algorithm [105] for the standard HMM, we first need to support the efficient computation of the below probabilities.

$$\alpha_{gxr}(t) = P(Y_1, \ldots, Y_t, F_1, ..., F_t, X_t = x, G = g, R_t = r; \theta') \qquad (3.10)$$

$$\beta_{gxr}(t) = P(Y_{t+1}, ..., Y_T, F_{t+1}, ..., F_T|X_t = x, G = g, R_t = r; \theta') \qquad (3.11)$$

Let $\theta' = (\pi', \sigma', \rho', A', B', C')$ and $C'(r,t) = \prod_{i=1}^{|F|} C'_{rif_t^i}$, both probabilities can be computed recursively, as follows.

$$\alpha_{gxr}(t) = \begin{cases} \pi'_x \sigma'_g \rho'_r C'(r,1) B'_{gxy_1}, & t = 1 \\ \rho'_r C'(r,t) B'_{gxy_t} \sum_{u \in \mathcal{X}} \sum_{k \in \mathcal{R}} \alpha_{guk}(t-1) A'_{kux}, & \text{else} \end{cases} \tag{3.12}$$

$$\beta_{gxr}(t) = \begin{cases} B'_{gxy_T} C'(r,T), & t = T-1 \\ \sum_{k \in \mathcal{R}} \rho'_k C'(k,t+1) \sum_{u \in \mathcal{X}} B'_{guy_{t+1}} A'_{rxu} \beta_{guk}(t+1), & \text{else} \end{cases} \tag{3.13}$$

Subsequently, $\gamma_{gxr}(t)$ and $\xi_{gxur}(t)$ can be further computed.

$$\xi_{gxur}(t) = \frac{\alpha_{gxr}(t) A'_{xur} B'_{guy_{t+1}} \sum_{k \in \mathcal{R}} \beta_{guk}(t+1) \rho'_k C'(k,t+1)}{\sum_{h \in \mathcal{G}} \sum_{v \in \mathcal{X}} \sum_{k \in \mathcal{R}} \alpha_{hvk}(T)} \tag{3.14}$$

$$\gamma_{gxr}(t) = \begin{cases} \sum_{x \in \mathcal{X}} \xi_{gxur}(t) & t = T \\ \sum_{u \in \mathcal{X}} \xi_{gxur}(t) & \text{else} \end{cases} \tag{3.15}$$

3.2.2 Item Prediction

Once the parameters are learnt, we can predict the next item of a user given her existing sequence of items $\{Y_1, Y_2, ..., Y_t\}$ and context features $\{F_1, F_2, ..., F_t\}$. In particular, her next item y^* can be chosen by maximum a posteriori estimation:

$$\begin{aligned} y^* &= \arg\max_y P(Y_{t+1} = y | Y_1, \dots, Y_t, F_1, ..., F_t) \\ &= \arg\max_y P(Y_1, \dots, Y_t, Y_{t+1} = y, F_1, ..., F_t) \\ &= \arg\max_y P(Y_1, \dots, Y_t, Y_{t+1} = y, F_1, ..., F_t, F_{t+1}) / P(F_{t+1}) \\ &= \arg\max_y \sum_{g \in \mathcal{G}} \sum_{x \in \mathcal{X}} \sum_{r \in \mathcal{R}} \alpha_{gxr}(t+1). \end{aligned} \tag{3.16}$$

While we do not observe features at time $t+1$, in the above we can adopt any value for F_{t+1} which does not affect the prediction. Instead of picking the best candidate item, we can rank all the candidates and suggest the top-K items.

3.2.3 Complexity Analysis

We conduct a complexity analysis for learning the joint model SEQ*. Consider one sequence of length T with $|\mathcal{X}|$ states, $|\mathcal{Y}|$ items, $|\mathcal{G}|$ user groups, $|\mathcal{R}|$ context factor levels, $|F|$ features and $|\mathcal{F}|$ values for each feature. For this one sequence, the complexity of one iteration of the EM is contributed by three main steps:

- *Step 1:* Calculate α, β: $O\,(T|\mathcal{G}||\mathcal{X}||\mathcal{R}|^2(|\mathcal{X}| + |F|))$. Because $\rho'_r, C'(r,t)$ in Eq (3.12) are independent of g, x, u, k while $\rho'_k, C'(k,t+1)$ in Eq (3.13) are independent of g, x, u, r, we further simplify this to: $O\,(T|\mathcal{R}|(|\mathcal{G}||\mathcal{X}|^2|\mathcal{R}| + |F|))$

- *Step 2:* Calculate ξ, γ using α, β: $O\,(T|\mathcal{G}||\mathcal{X}|^2|\mathcal{R}|^2|F|)$. As $\rho'_k C'(k,t+1)$ in Eq (3.14) is independent of g, x, u, r, we have: $O\,(T|\mathcal{R}|(|\mathcal{G}||\mathcal{X}|^2|\mathcal{R}| + |F|))$.

- *Step 3:* Update θ using γ, ξ: $O\,(T|\mathcal{G}||\mathcal{X}||\mathcal{R}|(|\mathcal{X}| + |F|))$. As y in B_{gxy} of Eq (3.7) is independent of g, x, r, we first compute the denominator, and update a normalized score to y in the B_{gxy} while computing the numerator. Likewise, i, f in C_{rif} are independent of g, x, r. Thus, we can transform it to: $O\,(T|\mathcal{R}|(|\mathcal{G}||\mathcal{X}|^2 + |F|))$.

The overall complexity of SEQ* is $O\,(T|\mathcal{R}|(|\mathcal{G}||\mathcal{X}|^2|\mathcal{R}| + |F|))$ for one sequence, one iteration. The complexities of other variants are (by substitution):

- HMM with $|\mathcal{G}| = |\mathcal{R}| = 1, |F| = |\mathcal{F}| = 0$: $O\,(T|\mathcal{X}|^2)$

- SEQ-E with $|\mathcal{R}| = 1, |F| = |\mathcal{F}| = 0$: $O\,(T|\mathcal{G}||\mathcal{X}|^2)$

- SEQ-T with $|\mathcal{G}| = 1$: $O\,(T|\mathcal{R}|(|\mathcal{X}|^2|\mathcal{R}| + |F|))$

The result implies that the running times of our proposed models are quadratic in the number of states and context factor levels, while linear in all the other variables. HMM is also quadratic in the number of states. Comparing to HMM with the same number of states, our joint model incurs a quadratic increase in complexity only in the number of context factor levels (which is typically small), and merely a linear increase in the number of groups and context features.

3.3 Experiments

The objective of experiments is to evaluate *effectiveness*. We first look into a synthetic dataset to investigate whether context-biased transition and user-biased emission could have been simulated by increasing the number of HMM's states. Next, we experiment with two real-life, publicly available datasets, to investigate whether the models result in significant improvements over the baseline.

3.3.1 Setup

We elaborate on the general setup here, and describe the specifics of each dataset later in the appropriate sections. Each dataset has a set of sequences. We create random splits of 80:20 ratio of training versus testing. In this sequential preference setting, a sequence (a user) is in either training or testing, but not necessarily in both. This is different from a fully personalized ordinal preference setting (a different framework altogether), where a user would be represented in both sets.

Task. For each sequence in the testing set, given the sequence excepts the last item, we seek to predict the last item. Each method generates a top-K recommendation, which is evaluated against the held-out ground-truth last item.

Comparative Methods. Since we build our dynamic context and user factors upon HMM, it is the most appropriate baseline. To investigate the contribution of user-biased emission and context-based transition *separately*, we compare the two models SEQ-E and SEQ-T respectively against the baseline. To see their contributions *jointly*, we further compare SEQ* against the baseline. In addition, we include the result of the frequency-based method POP as a reference, which simply choose the most popular item in the training data.

Metrics. We rely on two conventional metrics for top-K recommendation. Inspired by a similar evaluation task in [142], the first metric we use is *Recall@K*.

$$Recall@K = \frac{\text{number of sequences with the ground truth item in the top } K}{\text{total number of sequences in the testing set}}$$

If we assume the ground truth item to be the only true answer, average precision can be measured similarly (dividing by K) and would show the same trend as recall. In the experiments, we primarily study top 1% recommendation, i.e., *Recall@1%*, but will present results for several other K's as well. Actually, it is not clear that the other items in the top-K would really be rejected by a user [142]. Instead of precision, we rely on another metric.

The second metric is Mean Reciprocal Rank or *MRR*, defined as follows.

$$MRR = \frac{1}{|S_{\text{test}}|} \times \sum_{s \in S_{\text{test}}} \frac{1}{\text{rank of target item for sequence } s}$$

We prefer a method that places the ground-truth item higher in the top-K recommendation list. Because the contribution of a very low rank is vanishingly small, we cut the list off at 200, i.e., ranks $\geq$ 200 contribute zero to *MRR*. Realistically, a recommendation list longer than 200 is unlikely in realistic scenarios.

For each dataset, we create five random training/testing splits. For each "fold", we run the models ten times with different random initializations (but with common seeds across comparative methods for parity). For each method, we average the *Recall@K* and *MRR* across the fifty readings. All comparisons are verified by one-sided paired-sample Student's *t*-test at 0.05 significance level.

3.3.2 Synthetic Dataset

We begin with experiments on a synthetic dataset, for two reasons. First, one advantage of a synthetic dataset is the knowledge of the actual parameters (e.g., transition and emission probabilities), which allows us to verify our model's ability to recover these parameters. Second, we seek to verify whether the effects of context-biased transition and user-biased emission could have been simulated by increasing the number of hidden states of traditional sequence model HMM.

Dataset. We define a synthetic dataset with the following configuration: 2 groups ($|\mathcal{G}| = 2$), 2 states ($|\mathcal{X}| = 2$), 2 context factor levels ($|\mathcal{R}|$ = 2), 4 items

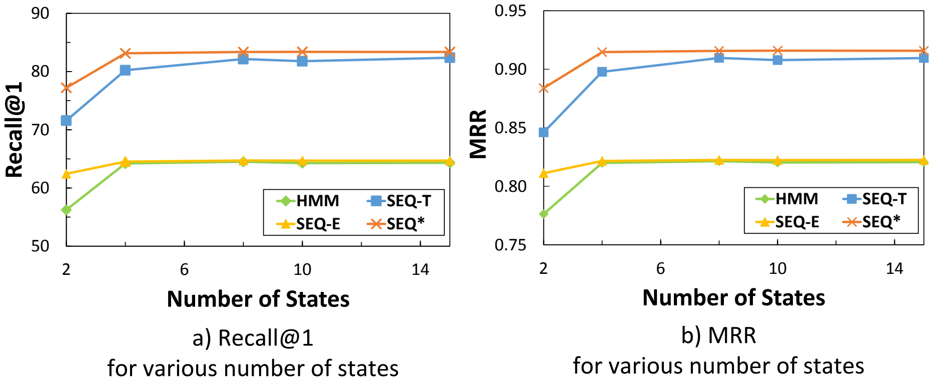


Figure 3.5: Performance of comparative methods on Synthetic Data for *Recall@1* and *MRR*

($|\mathcal{Y}| = 4$), 4 features ($|F|$ = 4) each with 2 feature values (present or absent).

Here, we discuss the key ideas. A six-tuple $\theta = (\pi, \sigma, \rho, A, B, C)$ is specified as follows: $\pi = [0.8, 0.2]$, $\sigma = [0.9, 0.1]$, $\rho = [0.3, 0.7]$. The transition tensor A is such that we induce self-transition to the same state for the first context factor level, and switching to the other state for the second context factor level. The emission tensor B is such that the four (state, group) combinations each tend to generate one of the four items. The feature matrix C is such that each context factor level is mainly associated with two of the four features.

We then generate 10 thousand sequences, each of length 10 ($T = 10$). For each sequence, we first draw a group according to σ. At time $t = 1$, we draw the first hidden state X_1 from π, followed by drawing the first item Y_1 from B. We also draw a context factor level from ρ and generate features via C. For time $t = 2, \ldots, 10$, we follow the same process, but each hidden state is now drawn from A according to the previous state and context factor level at time $t - 1$.

Results. We run the four comparative methods on this synthetic dataset, fixing the context factor levels and groups to 2 for the relevant methods, while varying the number of states. Figure 3.5(a) shows the results in terms of *Recall@1*, i.e., the ability of each method in recommending the ground truth item as the top prediction. There are several crucial observations. *First*, the proposed model SEQ* outperforms the rest, attaining recall close to 85%, while the baseline HMM hovers around 65%. SEQ* also outperforms SEQ-T and SEQ-E.

Second, as we increase the number of states, most models initially increase in performance and then converge. Evidently, increasing the number of states alone does not lift the baseline HMM to the same level of performance as SEQ* or SEQ-T, indicating the effect of context-biased transition. Meanwhile, though SEQ-E and HMM are similar (due to inability to model context factor), SEQ* is slightly better than SEQ-T, indicating the contribution of user-biased emission. Figure 3.5(b) shows the results for *MRR*, showing similar trends and observations.

Supplementary. Here, we first describe the original parameters for the synthetic data generation, and then show the parameters learned by each method.

Original Parameter Values. To generate a synthetic dataset, we construct a simple model with 2 latent groups ($|\mathcal{G}| = 2$), 2 latent states ($|\mathcal{X}| = 2$), 2 levels of context factor ($|\mathcal{R}|$ = 2), 4 items ($|\mathcal{Y}| = 4$), 4 features ($|F|$ = 4) each with 2 binary feature values.

The six-tuple parameter $\theta = (\pi, \sigma, \rho, A, B, C)$ is specified such that:

- Each (group, state) combination predominately generates 1 of the 4 items.
- Each context factor level is characterized by a pair of context features. One context factor level predominately supports self-transition to the same state. The other level predominately supports switching to the other state.

The original parameter values used during synthetic data generation are shown in the *Original Values* column of Table 3.1.

Learned Parameter Values. We run each comparative model on the generated synthetic dataset with 10 thousand sequences, each of length 10. Table 3.1 also shows the parameter values learned by each model.

HMM: It seems to take the advantage of the grouping probability to create hidden states and aggregate the emission probabilities of the two groups. Its transition probability also favors that of the *majority* context factor level.

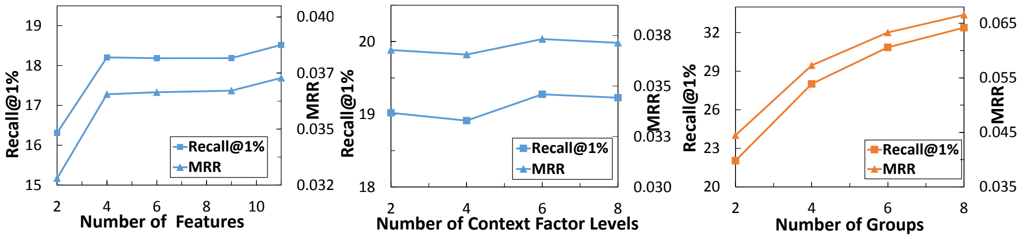
Parameter	Original Values	Learned Values			
		HMM	SEQ-E	SEQ-T	SEQ*
$\pi = [\pi_0, \pi_1]$	[0.80, 0.20]	[0.90, 0.10]	[0.75, 0.25]	[0.80, 0.20]	[0.80, 0.20]
$\sigma = [\sigma_0, \sigma_1]$	[0.90, 0.10]	N.A.	[0.90, 0.10]	N.A.	[0.90, 0.10]
$\rho = [\rho_0, \rho_1]$	[0.30, 0.70]	N.A.	N.A.	[0.30, 0.70]	[0.30, 0.70]
$A = [A_0, A_1]$ $A_i =$ $\begin{bmatrix} A_{i00} & A_{i01} \\ A_{i10} & A_{i11} \end{bmatrix}$	$A_0 =$ $\begin{bmatrix} 0.010 & 0.990 \\ 0.700 & 0.300 \end{bmatrix}$ $A_1 =$ $\begin{bmatrix} 0.990 & 0.010 \\ 0.300 & 0.700 \end{bmatrix}$	$A_0 =$ $\begin{bmatrix} 0.999 & 0.001 \\ 0.001 & 0.999 \end{bmatrix}$	$A_0 =$ $\begin{bmatrix} 0.670 & 0.330 \\ 0.280 & 0.720 \end{bmatrix}$	$A_0 =$ $\begin{bmatrix} 0.008 & 0.992 \\ 0.703 & 0.297 \end{bmatrix}$ $A_1 =$ $\begin{bmatrix} 0.994 & 0.006 \\ 0.292 & 0.708 \end{bmatrix}$	$A_0 =$ $\begin{bmatrix} 0.004 & 0.996 \\ 0.696 & 0.304 \end{bmatrix}$ $A_1 =$ $\begin{bmatrix} 0.993 & 0.007 \\ 0.293 & 0.707 \end{bmatrix}$
$B = [B_0, B_1]$ $B_i =$ $\begin{bmatrix} B_{i00} & B_{i01} \\ B_{i02} & B_{i03} \\ B_{i10} & B_{i11} \\ B_{i12} & B_{i13} \end{bmatrix}$	$B_0 =$ $\begin{bmatrix} 0.991 & 0.003 \\ 0.003 & 0.003 \\ 0.003 & 0.991 \\ 0.003 & 0.003 \end{bmatrix}$ $B_1 =$ $\begin{bmatrix} 0.003 & 0.003 \\ 0.991 & 0.003 \\ 0.003 & 0.003 \\ 0.003 & 0.991 \end{bmatrix}$	$B_0 =$ $\begin{bmatrix} 0.605 & 0.389 \\ 0.003 & 0.003 \\ 0.002 & 0.003 \\ 0.605 & 0.390 \end{bmatrix}$	$B_0 =$ $\begin{bmatrix} 0.989 & 0.006 \\ 0.003 & 0.002 \\ 0.202 & 0.790 \\ 0.005 & 0.003 \end{bmatrix}$ $B_1 =$ $\begin{bmatrix} 0.002 & 0.002 \\ 0.980 & 0.018 \\ 0.003 & 0.005 \\ 0.028 & 0.784 \end{bmatrix}$	$B_0 =$ $\begin{bmatrix} 0.891 & 0.004 \\ 0.101 & 0.004 \\ 0.008 & 0.890 \\ 0.003 & 0.099 \end{bmatrix}$	$B_0 =$ $\begin{bmatrix} 0.991 & 0.002 \\ 0.003 & 0.004 \\ 0.010 & 0.984 \\ 0.003 & 0.003 \end{bmatrix}$ $B_1 =$ $\begin{bmatrix} 0.002 & 0.003 \\ 0.990 & 0.005 \\ 0.002 & 0.004 \\ 0.014 & 0.980 \end{bmatrix}$
$C = [C_0, C_1]$ $C_i =$ $\begin{bmatrix} C_{i00} & C_{i01} \\ C_{i10} & C_{i11} \\ C_{i20} & C_{i21} \\ C_{i30} & C_{i31} \end{bmatrix}$	$C_0 =$ $\begin{bmatrix} 0.10 & 0.90 \\ 0.20 & 0.80 \\ 0.90 & 0.10 \\ 0.90 & 0.10 \end{bmatrix}$ $C_1 =$ $\begin{bmatrix} 0.90 & 0.10 \\ 0.90 & 0.10 \\ 0.10 & 0.90 \\ 0.30 & 0.70 \end{bmatrix}$	N.A.	N.A.	$C_0 =$ $\begin{bmatrix} 0.09 & 0.91 \\ 0.19 & 0.81 \\ 0.90 & 0.10 \\ 0.90 & 0.10 \end{bmatrix}$ $C_1 =$ $\begin{bmatrix} 0.90 & 0.10 \\ 0.90 & 0.10 \\ 0.08 & 0.92 \\ 0.28 & 0.72 \end{bmatrix}$	$C_0 =$ $\begin{bmatrix} 0.09 & 0.91 \\ 0.19 & 0.81 \\ 0.90 & 0.10 \\ 0.90 & 0.10 \end{bmatrix}$ $C_1 =$ $\begin{bmatrix} 0.90 & 0.10 \\ 0.90 & 0.10 \\ 0.08 & 0.92 \\ 0.28 & 0.72 \end{bmatrix}$

Table 3.1: Synthetic Parameters: Original Values vs. Learned Values by Various Models

SEQ-E: It learns the group distribution σ well, but the initial distribution of hidden states π is a bit off. This affects the emission probability. The transition probability amounts to an aggregation of the two context factor levels.

SEQ-T: It can recover most of the parameters, except the emission probability due to its not taking into account the user bias.

SEQ*: Importantly, SEQ* is the only model that can virtually recover all of the original parameter values with very light noises.



a) Vary the number of features for the SEQ-T model

b) Vary the number of context factor levels for the SEQ-T model

c) Vary the number of groups for the SEQ-E model

Figure 3.6: Effects of features, context factors & groups on Yes.com

3.3.3 Real-Life Datasets

We now investigate the performance of the comparative methods on real-life, publicly available datasets covering two different domains: song playlists from online radio station Yes.com, and hashtag sequences from users' Twitter streams.

Playlists from Yes.com. We utilize the *yes_small* dataset[1] collected by [17]. The dataset includes about 430 thousand playlists, involving 3168 songs. Noticeably, the majority of playlits has length which is shorter than 30. To keep the playlist lengths relatively balanced, we filter out playlists with fewer than two songs and retain up to the first thirty songs in each playlist. Finally, we have 250 thousand playlists (sequences) consisting of 3168 unique songs (items).

Features. We study the effect of features on the context-biased transition model SEQ-T. Each song may have tags. There are 250 unique tags. We group tags with similar meanings (e.g.,"male vocals" and "male vocalist"). As the first feature, we use a binary feature of whether the current song and the previous song shares at least one tag. For additional features, we use the most popular tags. Note that we never assume knowledge of the tags of the song to be predicted. Figure 3.6(a) shows the performance of SEQ-T, with two context factor levels, for various number of features. Figure 3.6(a) has dual vertical axes for *Recall@1%* (left) and *MRR* (right) respectively. The trends for both metrics are similar: performance initially goes up and then stabilizes. In subsequent experiments, we use eleven features (similarity feature and ten most popular tags).

[1] http://www.cs.cornell.edu/~shuochen/lme/data_page.html

		POP	HMM	SEQ-T	SEQ-E	SEQ*	Imp.
5 States	Recall@1%	6.8	13.8	$18.4^{\dagger}$	$22.0^{\S}$	$24.1^{\dagger\S}$	+10.3
	Recall@50	9.6	19.2	$25.1^{\dagger}$	$29.5^{\S}$	$32.1^{\dagger\S}$	+13.0
	Recall@100	16.2	29.3	$37.0^{\dagger}$	$42.6^{\S}$	$46.1^{\dagger\S}$	+16.8
10 States	Recall@1%	6.8	22.3	$23.2^{\dagger}$	$27.8^{\S}$	$28.6^{\dagger\S}$	+6.3
	Recall@50	9.6	30.0	$31.1^{\dagger}$	$36.9^{\S}$	$38.1^{\dagger\S}$	+8.1
	Recall@100	16.2	43.4	$44.9^{\dagger}$	$52.1^{\S}$	$53.5^{\dagger\S}$	+10.2
15 States	Recall@1%	6.8	26.1	$26.5^{\dagger}$	$30.1^{\S}$	$30.6^{\dagger\S}$	+4.5
	Recall@50	9.6	34.7	$35.5^{\dagger}$	$39.4^{\S}$	$40.2^{\dagger\S}$	+5.5
	Recall@100	16.2	49.3	$50.8^{\dagger}$	$55.1^{\S}$	$56.3^{\dagger\S}$	+7.0

Table 3.2: Performance of comparative methods on Yes.com for *Recall@K*

	POP	HMM	SEQ-T	SEQ-E	SEQ*	Imp.
5 States	0.014	0.028	$0.037^{\dagger}$	$0.044^{\S}$	$0.049^{\dagger\S}$	+0.021
10 States	0.014	0.045	$0.047^{\dagger}$	$0.057^{\S}$	$0.059^{\dagger\S}$	+0.014
15 States	0.014	0.053	$0.054^{\dagger}$	$0.062^{\S}$	$0.063^{\S}$	+0.009

Table 3.3: Performance of comparative methods on Yes.com for *MRR*

Context Factor. We then vary the number of context factor levels of SEQ-T (with eleven features). Figure 3.6(b) shows that for this dataset, there is not much gain from increasing the number of context factor levels beyond two. Therefore, for greater efficiency, subsequently we experiment with two context factor levels.

Latent Groups. We turn to the effect of latent groups on the user-biased emission model SEQ-E. Figure 3.6(c) shows the effect of increasing latent groups. More groups lead to better performance. Because of the diversity among sequences, having more groups increases the flexibility in modeling emissions while still sharing transitions. For the subsequent comparison to the baseline, we will experiment with two latent groups, as the earlier comparison has shown that the results with higher number of groups would be even higher.

Comparison to Baseline. We now compare the proposed models SEQ-T, SEQ-E, and SEQ* to the baseline HMM. Table 3.2 shows a comparison in terms of *Recall@K* for 5, 10, and 15 states. In addition to *Recall@1%* (corresponding to top 31), we also show results for *Recall@50* and *Recall@100*. The symbol † denotes statistical significance due to the effect of context-biased transition. In other words,

		POP	HMM	SEQ-T	SEQ-E	SEQ*	Imp.
5 States	Recall@1%	8.4	16.9	$17.1^\dagger$	$20.6^\S$	$21.0^{\dagger\S}$	+4.1
	Recall@50	16.1	28.3	$28.6^\dagger$	$33.2^\S$	$33.7^{\dagger\S}$	+5.4
	Recall@100	25.5	40.6	$40.9^\dagger$	$46.0^\S$	$46.5^{\dagger\S}$	+5.9
10 States	Recall@1%	8.4	21.8	$22.0^\dagger$	$26.5^\S$	$26.9^{\dagger\S}$	+5.1
	Recall@50	16.1	34.2	$34.4^\dagger$	$39.4^\S$	$39.8^{\dagger\S}$	+5.7
	Recall@100	25.5	47.2	$47.4^\dagger$	$52.0^\S$	$52.4^\S$	+5.2
15 States	Recall@1%	8.4	25.2	$25.3^\dagger$	$29.9^\S$	$30.0^{\dagger\S}$	+4.8
	Recall@50	16.1	38.1	$38.2^\dagger$	$43.1^\S$	$43.3^{\dagger\S}$	+5.1
	Recall@100	25.5	51.2	$51.3^\dagger$	$55.2^\S$	$55.3^{\dagger\S}$	+4.1

Table 3.4: Performance of comparative methods on Twitter.com for *Recall@K*

the outperformance of SEQ-T over HMM, and that of SEQ* over SEQ-E, are significant. The symbol § denotes statistical significance due to the effect of user-biased emission, i.e., the outperformance of SEQ-E over HMM, and that of SEQ* over SEQ-T, are significant. Finally, our overall model SEQ* is significantly better than the baseline HMM in all cases. The absolute improvement of the former over the latter in additional percentage terms is shown in the *Imp.* column. For all models, more states generally translate to better performance, and the improvements are somewhat smaller but still significant. Table 3.3 shows a comparison in terms of *MRR*, where similar observations hold.

Hashtag Sequences from Twitter.com. We conduct similar experiments on the Twitter dataset[2] [69]. There are 130 thousand users. In our scenario, each sequence corresponds to the hashtags of a user. The average length of our dataset is 19. If a tweet has multiple hashtags, we retain the most popular one, so as to maintain the sequence among tweets. Similarly to the treatment of stop words and infrequent words in document modeling, we filter out hashtags that are too popular (frequency $\geq$ 25000) or relatively infrequent (frequency $\leq$ 1000). Finally, we obtain 114 thousand sequences involving 2121 unique hashtags. Similarly to Yes.com, we run the models for two levels of context factor and two latent groups, but with seven features extracted from the tweet of the current hashtag (not the one to be predicted):

[2] `https://wiki.cites.illinois.edu/wiki/display/forward/Dataset-UDI-TwitterCrawl-Aug2012`

	POP	HMM	SEQ-T	SEQ-E	SEQ*	Imp.
5 States	0.019	0.045	$0.046^{\dagger}$	$0.062^{\S}$	$0.063^{\dagger\S}$	+0.0183
10 States	0.019	0.063	0.064	$0.084^{\S}$	$0.086^{\dagger\S}$	+0.0227
15 States	0.019	0.076	$0.078^{\dagger}$	$0.100^{\S}$	$0.101^{\dagger\S}$	+0.0246

Table 3.5: Performance of comparative methods on Twitter.com for *MRR*

number of retweets, number of hashtags, time intervals to the previous one and two tweets, time interval to the next tweet, and edit distances with the previous one and two observations.

The task is essentially predicting the next hashtag in a sequence. In brief, Tables 3.4 and 3.5 support that the improvements due to context-biased transition (†) and user-biased emission (§) are mostly significant. Importantly, the overall improvements by SEQ* over the baseline HMM (*Imp.* column) are consistent and hold up across 5, 10, and 15 states for both *Recall@K* and *MRR*.

Computational efficiency is not the main focus of experiments. We comment briefly on the running times. For the Twitter dataset, the average learning time per iteration on Intel Xeon CPU X5460 3.16GHz with 32GB RAM for our models with 15 states, 2 groups, 2 context factor levels are 2, 3, and 6 minutes for SEQ-E, SEQ-T and SEQ* respectively. HMM requires less than a minute.

3.4 Summary

In this chapter, we develop a generative model for sequences, which models two types of dynamic factors. First, transition from one state to the next may be affected by context factor. This results in SEQ-T model, with context-biased transition. Second, we seek to incorporate how different latent user groups may have preferences for certain items. This results in SEQ-E model, with user-biased emission. Finally, we unify these two factors into a joint model SEQ*. Experiments on both synthetic and real-life datasets support the case that these dynamic factors contribute towards better performance than the baseline HMM (statistically significant) in terms of top-K recommendation for sequences.

Chapter 4

Basket-Sensitive Recommendation

We address the basket-completion problem where the user is currently holding a basket of items, and the task is to recommend an item to be added to the basket. Given a basket, the correlation (i.e., *basket-oriented associations*) between current items may imply an underlying latent need, e.g., ingredients to prepare some dishes, spare parts of some devices. Thus, it is important that a recommended item is relevant not only to the user, but also to the existing items in the basket. Towards this goal, the need for *basket-sensitive* personalized item recommendations becomes remarkably pronounced.

4.1 Models

Consider N users $U = \{u_1, u_2, ..., u_N\}$ and M items $V = \{v_1, v_2, ..., v_M\}$. Given a user $u_i \in U$, a basket $B_i \subset V$ is defined as a subset of items that u_i is currently "holding". We refer to them as *basket items*. For instance, these could be items in the user's shopping cart or places already visited by the user on that day. Our objective is to recommend a *target item* $v_j \in V \setminus B_i$ (or a ranked list of items) to u_i. We seek to learn a real-valued function $F(u_i, B_i, v_j; \Theta)$, such that if $F(u_i, B_i, v_j; \Theta) > F(u_i, B_i, v_{j'}; \Theta)$, then the target item v_j is preferable to $v_{j'}$, and would be more likely to be recommended to u_i. Θ denotes the parameters of the function.

4.1.1 Basket-Sensitive Factorization Machine (BFM)

We develop our proposed BFM model by incorporating various types of useful associations.

User and Target Item. For personalized item recommendation, the basic type of association is between the user and the target item. This is a fundamental building block in most matrix factorization techniques [61]. Building upon this foundation, we include in Θ, a latent vector $x_i \in \mathbb{R}^K$ in K dimensions for each user $u_i \in U$, as well as a latent vector $y_j \in \mathbb{R}^K$ for each target item $v_j \in V$. $F(u_i, B_i, v_j; \Theta)$ is assumed to be proportional to ${x_i}^T y_j$.

$$F(u_i, B_i, v_j; \Theta) \propto {x_i}^T y_j \tag{4.1}$$

Matrix factorization essentially assumes that the basket items are irrelevant, implying that a user chooses items independently of one another. In practice, we expect that items in a basket may be associated with one another.

Basket Item and Target Item. It is important to model the associations between items in the basket and the target item. For instance, a supermarket shopper who is picking up ingredients for curry would be recommended items differently from when she is picking up ingredients for cake. To model the influence of a basket item on the choice of the target item, we further include in Θ a latent vector $z_j \in \mathbb{R}^K$ for every item $v_j \in V$. As opposed to y_j that models v_j's behavior as a target item, z_j models v_j's behavior as a basket item. Without prior knowledge, we assume that all items in the basket would have an influence on the choice of the target item.

$$F(u_i, B_i, v_j; \Theta) \propto \sum_{v_k \in B_i} {y_j}^T z_k \tag{4.2}$$

Among Basket Items. A basket of items may not always share a strong association among themselves. Possibly, a shopper may buy a complete set of ingredients for some dish. On another occasion, the shopper may pick up loose ends, resulting in a basket of less related items. The strength of association among current basket-items may influence the choice of the target item. We model this association as well.

$$F(u_i, B_i, v_j; \Theta) \propto \sum_{(v_k \neq v_{k'}) \in B_i} {z_k}^T z_{k'} \tag{4.3}$$

User and Basket Item. For completeness, we also model the association between the user and each basket item.

$$F(u_i, B_i, v_j; \Theta) \propto \sum_{v_k \in B_i} {x_i}^T z_k \tag{4.4}$$

This association is potentially redundant if the associations between the user and the target item, as well as between the target item and basket items are already modeled.

Overall Function. We now encapsulate the above association types into one overall function as follows.

$$\begin{aligned} F(u_i, B_i, v_j; \Theta) \ &\propto \gamma_1 \cdot {x_i}^T y_j + \gamma_2 \cdot \sum_{v_k \in B_i} {y_j}^T z_k \\ &+ \gamma_3 \cdot \sum_{(v_k \neq v_{k'}) \in B_i} {z_k}^T z_{k'} + \gamma_4 \cdot \sum_{v_k \in B_i} {x_i}^T z_k \end{aligned} \tag{4.5}$$

For flexibility in whether to incorporate an association type, we indicate each association type with a binary variable $\gamma_1, \gamma_2, \gamma_3, \gamma_4 \in \{0, 1\}$ to be specified according to each application scenario. We will experiment with different combinations of association types to see which are most useful.

Prediction. Once the parameters Θ are learned, given a user u_i and a basket B_i, we construct a recommendation list of target items in the order of decreasing $F(u_i, B_i, v_j; \Theta)$.

Parameter Learning. Given a set of tuples T, where each $t = \langle u_i, B_i, v_j, \delta \rangle \in T$ connotes a user u_i holding a basket B_i. If $\delta = 1$, the user ends up adopting a target item v_j. If $\delta = -1$, the user does not adopt v_j. A user may have multiple tuples in T. The goal is to learn the parameters in Θ, i.e., $\{x_i\}_{u_i \in U}$, and $\{y_j, z_j\}_{v_j \in V}$ to maximize the likelihood of observing T.

We make the interesting observation that the model parameters can be mapped into a factorization machine or FM [107]. Let $\mathbf{h}$ be a vector of length p, with binary elements, i.e., $h_i \in \{0, 1\}$. A second-order FM is as follows.

$$\mathcal{F}(\mathbf{h}) = \mu_0 + \sum_{i=1}^{p} \mu_i h_i + \sum_{i=1}^{p} \sum_{j=i+1}^{p} h_i h_j ({\phi_i}^T \phi_j) \tag{4.6}$$

The parameters include the global bias μ_0 and a bias coefficient μ_i for each component. Each $\phi_i \in \mathbb{R}^K$ is a K-dimensional latent vector associated with the i^{th} component.

We transform our model into the appropriate factorization machine. For $t = \langle u_i, B_i, v_j, \delta \rangle \in T$, we construct a binary vector $\mathbf{h}^t$ of length p, where $p = N + 2M$. The first N terms in $\mathbf{h}^t$ are for the presence of a user. We have $h_i^t = 1$. The next M terms in $\mathbf{h}^t$ are for the target item. We have $h_{N+j}^t = 1$. The last M terms are for the basket items. For each basket item $v_k \in B_i$, we have $h_{N+M+k}^t = 1$. All other elements of $\mathbf{h}^t$ are zeros. The latent vectors of this factorization machine stand for those of BFM. ϕ_i stands for a user latent vector x when $i \leq N$, for a target item latent vector y when $N < i \leq N + M$, and for a basket item latent vector z when $N + M < i$.

For BFM, the function $F(u_i, B_i, v_j; \Theta)$ in Eq (4.5) is effectively transformed into Eq (4.7). Θ denotes the ϕ_i's that also stand for x, y, z's. The addition of biases μ_0 and μ_i's is appropriate, and it is a common practice in matrix factorization-based recommendation [61].

$$\begin{aligned}\mathcal{F}(\mathbf{h}; \Theta) =& \mu_0 + \sum_{i=1}^{p} \mu_i h_i + \gamma_1 \sum_{i=1}^{N} \sum_{j=N+1}^{N+M} h_i h_j ({\phi_i}^T \phi_j) + \gamma_2 \sum_{i=N+1}^{N+M} \sum_{j=N+M+1}^{p} h_i h_j ({\phi_i}^T \phi_j) \\ &+ \gamma_3 \sum_{i=N+M+1}^{p} \sum_{j=i+1}^{p} h_i h_j ({\phi_i}^T \phi_j) + \gamma_4 \sum_{i=1}^{N} \sum_{j=N+M+1}^{p} h_i h_j ({\phi_i}^T \phi_j) \end{aligned} \tag{4.7}$$

To learn from training data T, we would like $\mathcal{F}(\mathbf{h}^t; \Theta)$ to be high when $t.\delta = 1$, and to be low when $t.\delta = -1$. To penalize errors during training, we adopt the following optimization criterion incorporating a logistic loss function.

$$\text{OPT\_BFM}(T) = \text{argmin}_{\Theta} \left[\sum_{t \in T} - \ln(\sigma(\mathcal{F}(\mathbf{h}^t; \Theta) \times t.\delta)) + \sum_{\theta \in \Theta} \lambda_\theta \theta^2 \right] \tag{4.8}$$

where $\sigma(a) = 1/(1+e^{-a})$ is the sigmoid function, and $\lambda_\theta \in \mathbb{R}^+$ is the regularization coefficient for θ.

The parameters could be estimated via several methods, e.g., stochastic gradient descent (*SGD*), alternating least-squares and Markov Chain Monte Carlo [107].

4.1.2 Constrained BFM or CBFM

We describe CONSTRAINED BFM or CBFM that incorporates constraints relating baskets of similar intent. Intuitively, if a user shops for a number of items to fulfil a need, conceivably on different occasions the user may put items in different sequences and construct different intermediate baskets that make up the same collection of items. For example, if the intent is served by four items v_1, v_2, v_3, v_4, on one occasion when a user's basket contains $\{v_1, v_2, v_3\}$, we would recommend v_4, while on a different occasion when the user's basket contains $\{v_1, v_3, v_4\}$, we would recommend v_2. Because the recommendations go on to serve the same intent for the user, we postulate that their likelihoods should be similar.

Definition 1. TUPLES OF THE SAME INTENT *We say that two tuples t_1 and t_2 in the training data T have the same intent if the following conditions hold:*

- *t_1 and t_2 concern the same user,*
- *the union of the basket items and the target item is identical between t_1 & t_2,*
- *both are positive examples, i.e., $t_1.\delta = 1$ and $t_2.\delta = 1$.*

Given two tuples t_1 and t_2 of the same intent, we seek to minimize the difference between their function values.

$$\left(\mathcal{F}(\mathbf{h}^{t_1}; \Theta) - \mathcal{F}(\mathbf{h}^{t_2}; \Theta)\right)^2 \tag{4.9}$$

Different pairs of tuples with the same intent may have different degrees of correlation, which we model by the Point-wise Mutual Information (PMI) [11] of their target items. Suppose for two same-intent tuples t_1 and t_2, their target items are v_1 and v_2. The PMI is the joint probability of v_1 and v_2, estimated through their joint co-occurrence across transactions, divided by the marginal probabilities of v_1 and v_2 respectively, as shown in Eq (4.10). The higher the PMI the more likely two items appear in the same basket.

$$PMI(t_1, t_2) = \ln \frac{P(v_1, v_2)}{P(v_1)(v_2)} \tag{4.10}$$

In practice, there may be more than two tuples sharing the same intent. For a collection of such tuples from positive examples, the objective is to learn high scores. Imposing similarity across all pairs of such tuples may have the unintended effect of making them equally low, instead of equally high. Therefore, we would only add the constraint between a tuple t and its same-intent tuple t^m that has the maximum score.

We now define the optimization criterion for the CONSTRAINED BFM or CBFM, as shown in Eq (4.11) below.

$$\begin{aligned}\text{OPT\_CBFM}(T) = \text{argmin}_\Theta \Bigg[\sum_{\theta \in \Theta} \lambda_\theta \theta^2 + \sum_{t \in T} \Big\{ &- \ln(\sigma(\mathcal{F}(\mathbf{h}^t; \Theta) \times t.\delta)) \\ &+ \frac{\alpha}{2} \times PMI(t, t^m) \times \big(\mathcal{F}(\mathbf{h}^t; \Theta) - \mathcal{F}(\mathbf{h}^{t^m}; \Theta)\big)^2 \Big\} \Bigg]\end{aligned} \tag{4.11}$$

The objective function of CBFM subsumes that of BFM. It still has the logistic loss function of BFM. It also features the constraints. α is a coefficient controlling the strength of the constraint vis-á-vis the logistic loss, to be tuned empirically. As we are most concerned with strongly-correlated tuples, we apply the constraint only for positive tuples having positive PMI; otherwise it is zero with no effect.

The inference for CBFM's logistic loss is similar to that of BFM. The key difference is the constraint component. The overall gradient of parameters consists of the gradient due to the logistic loss, as well as the gradient due to the derivative the constraint component. The latter is as follows.

$$\begin{aligned}&\frac{\partial}{\partial\theta}\mathcal{F}(\mathbf{h}^{t};\Theta)-\mathcal{F}(\mathbf{h}^{t^{m}};\Theta))^{2} \\ &\qquad =2(\mathcal{F}(\mathbf{h}^{t^{m}};\Theta)-\mathcal{F}(\mathbf{h}^{t};\Theta))\frac{\partial}{\partial\theta}(\mathcal{F}(\mathbf{h}^{t^{m}};\Theta)-\mathcal{F}(\mathbf{h}^{t};\Theta))\end{aligned} \tag{4.12}$$

Subsequently, we perform the following update iteratively in the SGD algorithm, where η is the learning rate.

$$\begin{aligned}\theta \leftarrow &\theta-\eta\Bigg[t.\delta\times(\sigma(\mathcal{F}(\mathbf{h}^{t};\Theta)\times t.\delta)-1)\frac{\partial}{\partial\theta}\mathcal{F}(\mathbf{h}^{t};\Theta)+2\lambda_{\theta}\theta \\ &+\alpha\times PMI(t,t^{m})\times(\mathcal{F}(\mathbf{h}^{t^{m}};\Theta)-\mathcal{F}(\mathbf{h}^{t};\Theta))\times\frac{\partial}{\partial\theta}(\mathcal{F}(\mathbf{h}^{t^{m}};\Theta)-\mathcal{F}(\mathbf{h}^{t};\Theta))\Bigg]\end{aligned} \tag{4.13}$$

The complexity of CBFM learning is similar to BFM, $\mathcal{O}(K\cdot|T|\cdot\bar{s})$, where K is the vector dimensionality, $|T|$ is the size of the training data, and $\bar{s}$ is the average basket size in T. Compared to BFM, CBFM takes additional comparisons to find the maximum instance t^{m}. The extra calculation is linearly proportional to the size of the respective basket.

4.2 Experiments

The objective of experiments is to investigate the effectiveness of the BFM and CBFM.

4.2.1 Setup

Datasets. We experiment with three public real-life datasets from different domains bearing basket-like associations. The dataset sizes are summarized in Table 4.1.

TaFeng[1]: This is a retail market dataset. There are a series of transactions, where each transaction involves a user and multiple grocery items. The hypothesis is that

[1] http://recsyswiki.com/wiki/Grocery_shopping_datasets

Dataset	#Users	#Items	#Transactions	Average #Items per Transaction
TaFeng	11711	11035	71447	7.3
BeiRen	9245	5581	87224	6.1
Foursquare	1548	3619	31377	2.7

Table 4.1: Statistics for TaFeng, BeiRen & Foursquare.

items in a basket may be related as they go towards household needs.

BeiRen[2]: This comes from a large retailer in China, capturing the period from 2012 to 2013. Similar to *TaFeng*, each transaction contains a set of items bought by a given user.

Foursquare[3]: This consists of users' check-ins at various points of interest in Singapore [164]. We treat the check-ins within the same day as a transaction. The hypothesis is that these check-ins involve related purposes. Previous works on point-of-interest [164] rely on modeling temporal or sequential associations. That is not the scope of our work, which is modeling in-basket associations.

Similar pre-processing is applied on all datasets. For sufficient statistics, we filter out items bought by too few users, i.e., 10 users for TaFeng & BeiRen and 5 users for Foursquare corresponding to their data sizes. As our focus is on modeling associations, we remove items that behave like "stop words" with presence in a large fraction of transactions (more than 5%). There are merely 2 or 3 such items in TaFeng and BeiRen respectively and none in Foursquare. As transactions with single items do not contain item-item association, we retain only transactions with more than 2 items. We also filter out users with fewer than 3 transactions, which is the minimum needed to have a training/validation/testing split.

Training, Validation & Testing. We further split the transactions as follows. For each user, we sort her transactions chronologically. The last transaction will be part of the testing set. The second-last transaction will be part of the validation set. The rest will be part of the training set.

For each transaction, we induce positive tuples in the form of $t = \langle u_i, B_i, v_j, 1\rangle$.

[2] http://www.brjt.cn

[3] http://www.ntu.edu.sg/home/gaocong/datacode.htm

For each item v_j in the transaction of user u_i, we hide v_j as the item to be predicted. The remaining items observed in that transaction will form the basket B_i. Hence, a transaction containing n items will result in n positive tuples. In addition, as we discuss previously in Section 4.1.2, these n tuples are said to have the same intent.

As the datasets only have positive examples, following [95], we create negative examples by sampling. From each positive tuple $t = \langle u_i, B_i, v_j, 1\rangle$, we can create a negative tuple $t^\neg = \langle u_i, B_i^\neg, v_j^\neg, -1\rangle$. As $v_j^\neg$, we randomly pick an item never selected by the user. $B_i^\neg$ contains items that never co-occur with either the user, $v_j^\neg$ or other items in $B_i^\neg$. For parity, we have $|B_i| = |B_i^\neg|$. As we expect there are more items that a user does not prefer than those that a user does, we have twice as many negative tuples. Training set has both positive and negative tuples for learning, while validation and testing sets consist of only positive tuples.

Evaluation Task & Metrics. The evaluation task is top-n recommendations. For each tuple $t = \langle u_i, B_i, v_j, 1\rangle$ in the testing set, we hide the observed item v_j, and require each model to produce a ranked list of items for u_i based on B_i. A list that ranks the observed v_j higher is better. For the proposed models, the performance numbers are averaged across 25 runs with different random initializations.

We rely on two evaluation metrics frequently used for top-n recommendation [110]. The first metric is Half-Life Utility or HLU [13]. It measures how likely a user will adopt an item at a ranking position k. [13] proposed this probability as $2^{\frac{1-k}{\beta-1}}$ with β as the half-life parameter. In our context, HLU is defined:

$$HLU = \frac{1}{|T_{\text{test}}|} \times C \times \sum_{t \in T_{\text{test}}} 2^{\frac{1-r_t}{\beta-1}}$$

where T_{test} is the testing set, and r_t is the rank of the item v_j in tuple t in the list. C is the scaling parameter. Following [110], we set $\beta = 5$ and $C = 100$.

The second metric is recall (*R@n*), defined as the percentage of testing instances with the ground truth item in top-n. The higher the percentage, the better the model is. In experiments, we primarily investigate top-10 recommendations, i.e., *R@10*,

Association				TaFeng		BeiRen		Foursquare	
γ_1	γ_2	γ_3	γ_4	HLU	R@10 (%)	HLU	R@10 (%)	HLU	R@10 (%)
1	0	0	0	0.06	0.10	1.94	3.16	5.45	8.29
1	1	0	0	1.47†	2.27†	3.35†	5.11†	8.11†	11.98†
1	1	1	0	**2.14**†§	**3.41**†§	**3.75**†§	5.77†	**8.51**†§	**12.48**†§
1	1	0	1	1.64†	2.59†	3.59†	5.54†	7.08†	10.50†
1	1	1	1	2.08†	3.31†	3.74†	**5.78**†	8.02†	11.84†

Table 4.2: Performance Comparison for BFM with Various Association Types on TaFeng, BeiRen and Foursquare.The four association types include γ_1: User & Target, γ_2: Target & Basket, γ_3: Basket & Basket, γ_4: User & Basket. The first row is akin to the traditional FM model without using basket-sensitive information.

but will show performances for several other top-n as well. Precision may not be suitable, as the unobserved items may not necessarily be negative examples, but rather simply unlabeled positive examples [142].

4.2.2 Results

BFM **with Various Association Types.** First, we investigate several combinations of association types to determine what constitutes a good configuration for BFM. We implement BFM in Java based on libFM [4]. For these experiments, we use latent factor dimension $K = 8$ and regularization parameter $\lambda_\theta = 0.01$, which are also the defaults of libFM. The various numbers of latent factor dimensions K are empirically investigated in the last experiment. The initial learning rate η is 0.0001 for TaFeng, BeiRen and 0.001 Foursquare respectively to reflect their relative sparsity. We further apply the *Bold-Driver* adaptive learning rate [8].

Table 4.2 shows BFM of different configurations. The first configuration $[\gamma_1, \gamma_2, \gamma_3, \gamma_4] = [1, 0, 0, 0]$ has only associations between each user and the target item. This is a factorization machine (FM) akin to matrix factorization, which does not feature any basket effects. The second configuration $[1, 1, 0, 0]$ adds associations between each basket item to the target item, with a higher performance than FM in terms of HLU and $R@10$, implying that basket items indeed have an influence on

[4] http://www.libfm.org

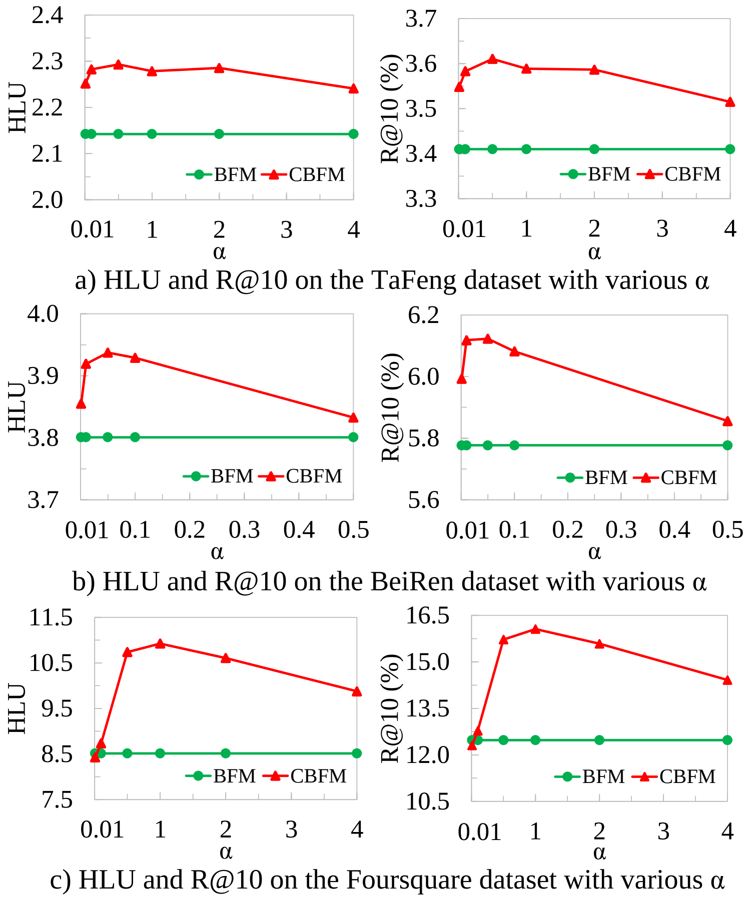


Figure 4.1: Performance Comparison for BFM and CBFM for Various α on TaFeng, BeiRen and Foursquare

the target item. We further experiment with several more configurations. It emerges that the best configuration is $[1, 1, 1, 0]$, which includes associations among basket items, but excludes those between users and basket items.

Table 4.2 shows that models with basket associations ($\exists j \in [2, 4]$ and $\gamma_j = 1$) are better than FM. The symbol † indicates that paired samples t-test shows statistical significance (at 0.05 level) in the improvements over FM. That the best configuration $[1, 1, 1, 0]$ shows statistically significant improvements over the second-best configuration $[1, 1, 1, 1]$ is indicated by the symbol $\S$. Therefore, we will use $[1, 1, 1, 0]$ as the default for BFM.

Effect of Constraint. We take the best configuration of BFM, and add the basket-level constraint to form CBFM. Figure 4.1(a) illustrates the CBFM's HLU and $R@10$ on TaFeng when we vary α. BFM is equivalent to CBFM when $\alpha = 0$.

Dataset	Model	HLU	R@n (%)		
			10	20	50
TaFeng	CBFM	**2.29**‡	**3.61**‡	**6.00**‡	**11.11**‡
	BFM	2.14	3.41	5.77	10.79
	ASR	1.97	2.72	3.56	4.83
BeiRen	CBFM	**3.89**‡	**6.12**‡	**10.42**‡	**19.04**‡
	BFM	3.75	5.78	9.91	18.79
	ASR	3.74	5.31	7.60	11.56
Foursquare	CBFM	**10.92**‡	**16.06**‡	**21.83**‡	**30.81**‡
	BFM	8.51	12.48	17.86	26.41
	ASR	6.54	10.36	12.56	15.64

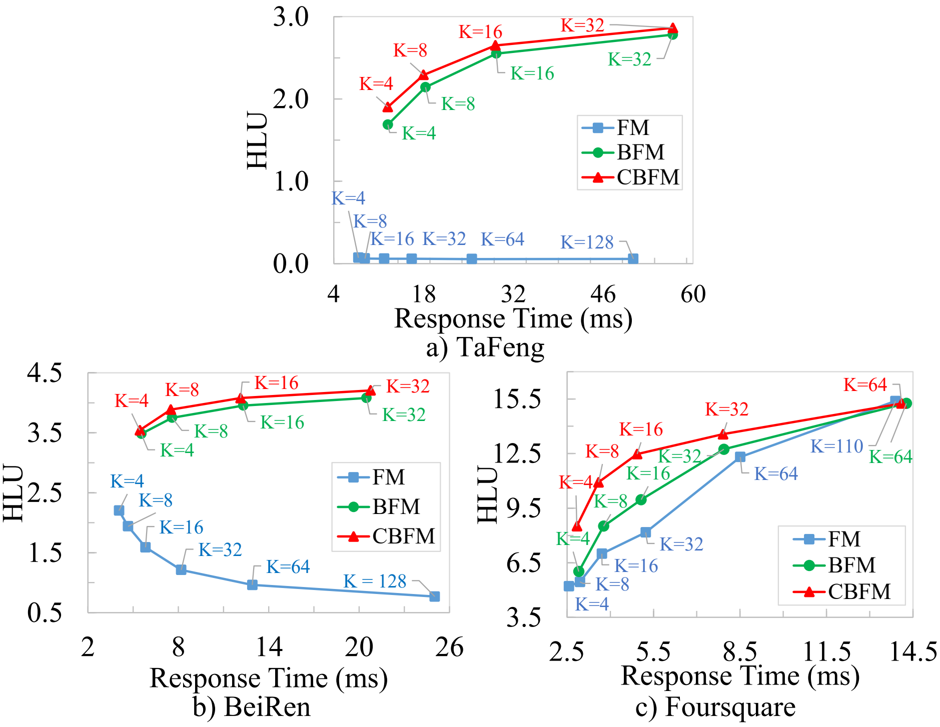
Table 4.3: Performance Comparison to Association Rules (ASR) on TaFeng, BeiRen and Foursquare.

The performance generally rises and then falls. The best on TaFeng is $\alpha = 0.5$. Figure 4.1(b) is for BeiRen, where the best configuration is $\alpha = 0.05$. Finally, Figure 4.1(c) shows the corresponding results on Foursquare, with best performance at $\alpha = 1$. Subsequently, we will use these α.

Comparison to Association Rules. We include a comparison to a baseline based on association rules [116]. First, we learn association rules from the training data, with minimum support of 10 on TaFeng, BeiRen and 5 on Foursquare (the same filters as for our training data). For a user and a basket, a rule is applicable if the antecedent items are contained in the basket, and have been adopted by the user previously. For each target item, if there are multiple applicable rules, we use the rule with maximum confidence. We then construct a ranked list in decreasing order of confidence.

Table 4.3 shows a comparison to the association rule-based baseline ASR. In addition to HLU and $R@10$, we also show $R@20$ and $R@50$. CBFM and BFM both outperform ASR across all measures. We hypothesize this is due to their use of factorization that allows them to discover other latent associations among items. The symbol ‡ indicates the statistically significant (at 0.05 level) improvement of CBFM over BFM.

Model Complexity and Response Time. The goal of recommender systems is to provide users with relevant results in a timely and responsive manner [60]. To



a) TaFeng

b) BeiRen

c) Foursquare

Figure 4.2: Half-life Utility and Response Times

retrieve the top-n recommendation at run time, we need to evaluate the prediction score for all possible items in the inventory [3]. One setting that has a direct effect on retrieval speed is the number of latent factors. We define response time as the time required to do the prediction computation for all items (required for top-n). Timing is based on a PC with Intel Core i5 3.2GHz with 8GB RAM.

Figure 4.2 demonstrates how HLU and response time are affected by different number of latent factors K. For CBFM, we tune the α based on the validation set for each K. We see a trend that increasing K leads to higher HLU but also higher response times. For TaFeng and BeiRen, Figures 4.2(a) and 4.2(b) show that CBFM has a slight gap over BFM throughout (statistically significant at 0.05 level). FM has relatively low performance, probably due to significant basket effects and data sparsity. FM does better in Foursquare, as shown in Figure 4.2(c). For very high number of latent factors (which also result in higher response times), eventually the models achieve similar performance. It might be triggered by the strong effect of personalized preferences. Importantly, CBFM shows a good trade-off behavior.

Dataset	K	Response Time (ms)			HLU			R@10 (%)		
		CBFM	BFM	FM	CBFM	BFM	FM	CBFM	BFM	FM
TaFeng	4	12.4	12.5	7.9	1.90	1.69	0.07	3.00	2.70	0.12
	8	18.0	18.3	8.9	2.29	2.14	0.06	3.61	3.43	0.10
	16	29.2	29.3	11.9	2.65	2.55	0.06	4.16	4.03	0.10
	32	56.9	56.9	16.2	2.86	2.78	0.06	4.49	4.40	0.10
BeiRen	4	5.44	5.54	4.06	3.54	3.48	2.20	5.53	5.30	3.71
	8	7.49	7.58	4.66	3.89	3.75	1.94	6.11	5.75	3.06
	16	12.13	12.3	5.82	4.08	3.95	1.59	6.39	6.10	2.49
	32	20.76	20.49	8.19	4.21	4.08	1.21	6.61	6.31	1.69
Foursquare	4	2.84	2.91	2.57	8.51	6.02	5.22	12.43	9.10	7.76
	8	3.59	3.77	2.95	10.92	8.51	5.45	15.96	12.10	8.41
	16	4.93	5.07	3.71	12.47	9.96	7.00	18.27	15.02	10.53
	32	7.91	7.94	5.23	13.56	12.73	8.17	20.04	18.69	12.62

Table 4.4: Response Times, Half-life Utility (HLU) and Recall@10 for various K on TaFeng, BeiRen, and Foursquare.

For most response times, especially for the fast response times, it has significantly better HLU.

Considering the alignment on K, Table 4.4 shows that FM is a bit faster than CBFM, BFM in term of response times. The increase of K triggers the increase of response times. However, CBFM and BFM significantly outperform FM in terms of HLU and $R@10$. These observations are consistent to what are observed on the three datasets in Figure 4.2.

4.3 Summary

In this chapter, we investigate recommendation models that take into account of a user's current basket in making personalized recommendations. We propose two models: BFM that incorporates various association types, and CBFM that further integrates constraints for baskets with similar intent. Experiments show some improvements over factorization machine that does not model basket associations, and association rules that do not benefit from latent associations discovered by factorization.

Chapter 5

Sequential Recommendation using Contemporaneous Basket Sequences

The user interaction with online websites frequently leaves a heterogeneous and contemporaneous trail of actions (e.g., clicks, bookmarks, purchases). Given a sequence of a particular type (e.g., purchases)– referred to as the target sequence, modeling *sequential dependencies* is still a substantial solution to the next-item recommendation task. With the development of Deep Learning techniques, deep neural networks (e.g., Recurrent Neural Networks) are useful to capture the *long-term* sequential preference. However, the problem becomes more difficult if there are multiple items (a basket) adopted at each time step. Items within a basket potentially indicate latent associations, referred to as *basket-oriented association*. Additionally, the next item prediction on the target sequence may be helped by also learning from another *contemporaneous* sequence (e.g., clicks), referred as the supporting sequence. In this chapter, we introduce three twin network structures modeling associations between contemporaneous sequences to solve the next-item recommendation task.

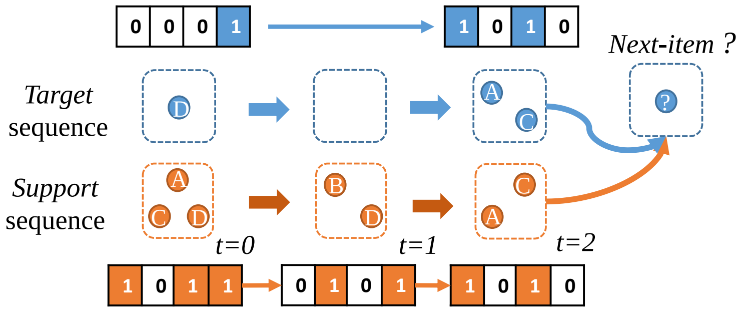


Figure 5.1: Example representations of target and support sequences

5.1 Models

Here, we describe the framework for contemporaneous basket sequences. We begin with the notations and problem formulation, before elaborating the proposed architectures.

Let $V = \{v_0, v_1, ..., v_{N-1}\}$ denote the set of items under consideration. $B_t \subset V$ denotes a basket of items "adopted" at time step t. Adoption could mean purchasing, clicking, reviewing, or any other binary indication of preference. Another equivalent representation of B_t is a binary vector of length N, i.e., $X_t =< x_0, x_1, ..., x_{N-1} > \in \{0, 1\}^N$, whereby $x_i = 1$ when $v_i \in B_t$, and 0 otherwise.

The data is a set of sequence pairs $D = \{\langle T_j, S_j \rangle\}_{j=0}^{M-1}$. For the j^{th} instance, T_j is its target sequence, while S_j is its support sequence. Both T_j and S_j are represented as sequences of baskets/binary vectors $\{X_t\}$. For example, D may concern M users, whereby T_j comprises the sequence of baskets purchased by user j, and S_j comprises her sequence of clicks. For ease of illustration, and without loss of generality, subsequently we may use "purchase" or "target" interchangeably, as well as "click" or "support". Generally, the objective is to learn a model that uses information from both target and support sequences to predict the next item in the target sequence. The two sequence types are *contemporaneous*, i.e., occurring over a common time period. More precisely, the time period covered by T_j overlaps with S_j, but the last basket of S_j does not occur later than the next-item to be predicted for T_j, to avoid using future information to predict a past event. Figure 5.1 illustrates one instance of a pair of sequence types for four items A to D. In the first time step, the user "clicks" on A, C, D (represented as $[1, 0, 1, 1]$), eventually "purchasing" D (represented as $[0, 0, 0, 1]$). The subsequent time steps involve baskets of different items.

Since we are dealing with sequences, we build on the foundation of RNNs, known for its capacity for generating sequential data. However, since we need to model two sequence types simultaneously, we investigate dual-RNN architectures or *twin networks*. We develop three such architectures, which differ in the degree of parameter sharing between the two sequence types. Their etymologies are inspired by biological terms describing twins [44]. CBS-SN is named after "Siamese twins"or identical twins with 100% gene sharing, to signify how the two sequence types will be modeled by identical RNNs. CBS-CFN and CBS-DFN are named after "fraternal twins" that on average share 50% of their genes, to signify both similarities and differences between the sequence types.

5.1.1 CBS with Siamese Networks (CBS-SN)

Our first model CBS-SN is based on the idea of Siamese networks. This structure contains twin networks that receive two distinct inputs, have their parameters tied so as to constrain the two inputs to the same feature space, and are conjoined together (concatenated) at the top layer [14]. The specific realization depends on the problem scenario.

Figure 5.2 illustrates the architecture of CBS-SN. We describe it layer by layer. The bottom layer is the *basket encoder*. In each time step, we have a basket/binary vector X_t. We hypothesize that there are correlated items that may co-occur within baskets. To capture the correlative information, we utilize a dense (fully connected) layer to map a basket's binary vector X_t into its hidden representation b_t as follows:

$$b_t = f(\Theta_b X_t + \Omega_b) \tag{5.1}$$

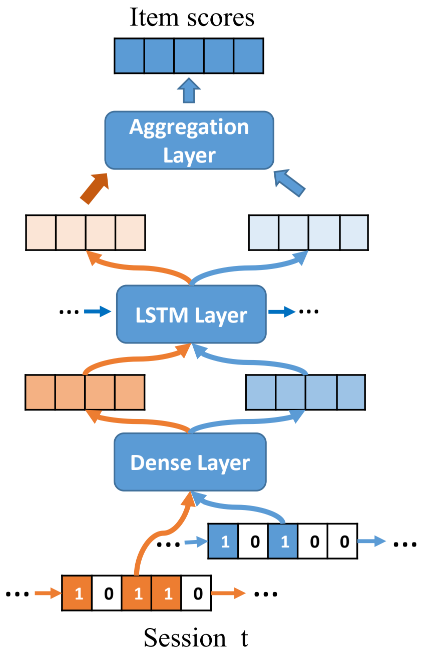


Figure 5.2: Modeling Contemporaneous Basket Sequences with Siamese Networks (CBS-SN)

where f is an activation function, L is the number of latent dimensions in the dense layer; and $\Theta_b \in \mathbb{R}^{L \times N}, \Omega_b \in \mathbb{R}^L$; are parameters to be learned.

The middle layer is the *recurrent encoder* based on LSTM. It seeks to capture the sequential effect by feeding the basket representation b_t into a recurrent layer. The hidden recurrent representation h_t at the time step t is computed as follows:

$$h_t = g(\Phi_b b_t + \Phi_h h_{t-1} + \Omega_h) \tag{5.2}$$

where g is an activation function, H is the number of hidden recurrent units; and $\Phi_b \in \mathbb{R}^{H \times L}, \Phi_h \in \mathbb{R}^{H \times H}, \Omega_h \in \mathbb{R}^H$; are parameters to be learned.

The final layer is the *aggregation layer*. The assumption of CBS-SN is that the target sequence and the support sequence are distinct manifestations of the same underlying phenomenon. Therefore, the two sequence types share the same basket encoder and LSTM recurrent encoder. Let $\hat{h}_T$ be the last hidden recurrent representation for the target sequence, and correspondingly $\hat{h}_S$ for the support sequence. In this Siamese networks-inspired structure, the aggregated layer is as follows:

$$\begin{aligned} \hat{h} &= concat(\hat{h}_T, \hat{h}_S) \\ r_{\text{agg}} &= W\hat{h} + \Omega_c \end{aligned} \tag{5.3}$$

where $W \in \mathbb{R}^{N \times 2H}, \Omega_c \in \mathbb{R}^N$ are parameters to be learned. The scores of next-item candidates are computed as a function of this aggregated representation:

$$Y = \sigma(r_{\text{agg}}) \tag{5.4}$$

where $Y \in \mathbb{R}^{1 \times N}$, the *softmax* function $\sigma(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}}$.

The output is the vector Y of length N, where each element is the likelihood of each item to be the next adoption.

5.1.2 CBS with Concordant Fraternal Networks (CBS-CFN)

The earlier assumption that the sequence types reflect the same underlying phenomenon may be too strong. For instance, purchases and clicks are related, in that some clicks lead to purchases. However, clicking or browsing actions are low-cost and easy to undo, as opposed to purchases that require a larger commitment of resources. Therefore, they may reflect different sequential behaviors.

As illustrated in Figure 5.3, our second model CBS-CFN leverages on two distinct recurrent encoders: *LSTM Layer 1* for the support sequence and *LSTM Layer 2* for the target sequence. They still share the same basket encoder, as in-basket associations are likely to still be similar in both cases. From partial sharing of parameters, different recurrent encoders but same basket encoder, arises the notion of *fraternal* networks. The term *concordant* signifies the same size of LSTMs, capturing longer-term sequentiality in both sequence types.

Because CBS-CFN assumes the target sequence and support sequence are distinct, we would like to aggregate them in a way that allows their contributions to

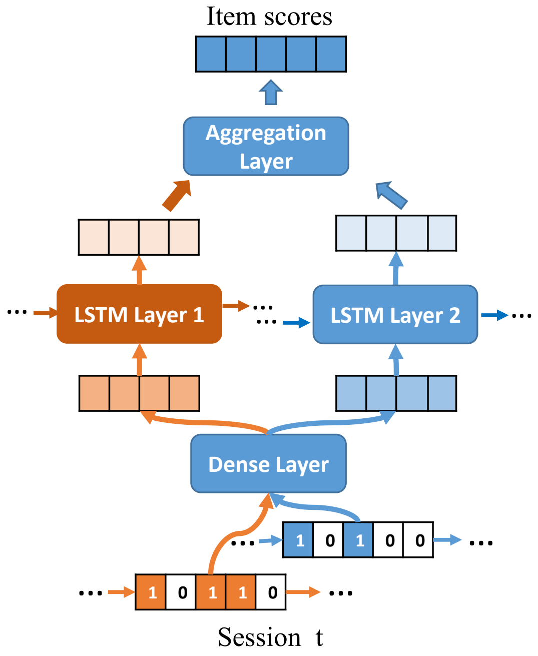


Figure 5.3: Modeling Contemporaneous Basket Sequences with Concordant Fraternal Networks (CBS-CFN)

be weighted accordingly. The last hidden recurrent representations $\hat{h}_T$ and $\hat{h}_S$ are aggregated as follows:

$$r_{\text{agg}} = W_1\hat{h}_S + W_2\hat{h}_T + \Omega_c \tag{5.5}$$

where $W_1, W_2 \in \mathbb{R}^{N\times H}, \Omega_c \in \mathbb{R}^N$ are parameters to be learned. The output Y is computed as in Eq (5.4).

5.1.3 CBS with Discordant Fraternal Networks (CBS-DFN)

The previous model seeks to capture distinct sequence types of the same sequential dependencies. In some scenarios, it may be appropriate to capture different scopes of sequential dependency. For instance, browsing and clicking may have longer-term dependency than purchases. The Figure 5.4 illustrates our third model CBS-DFN, where the support sequence has a recurrent encoder to learn longer-term recurrence relations, but the target sequence relies on shorter-term relations

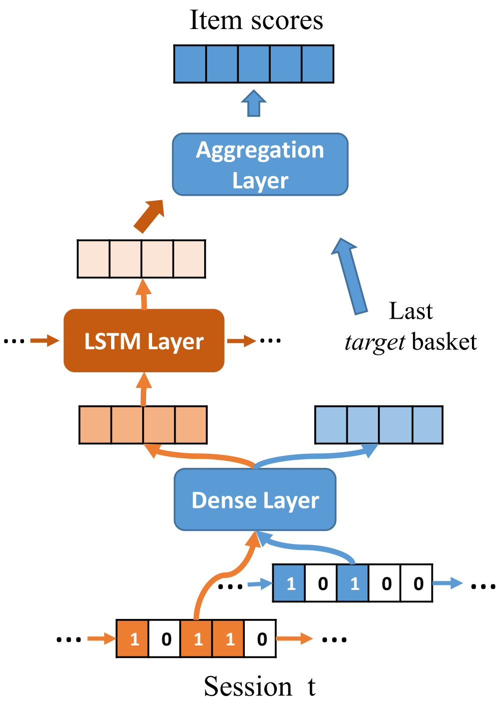


Figure 5.4: Modeling Contemporaneous Basket Sequences with Discordant Fraternal Networks (CBS-DFN)

and directly makes use of the output of the bottom layer (basket encoder). The term *discordant* refers to this varying treatment. The aggregated operation is as follows:

$$r_{\text{agg}} = W_1 \hat{h}_S + W_2 \hat{b}_T + \Omega_c \tag{5.6}$$

where $\hat{b}_T$ is the hidden representation of the last basket, $W_1 \in \mathbb{R}^{N \times H}, W_2 \in \mathbb{R}^{N \times L}, \Omega_c \in \mathbb{R}^N$ are parameters. The output Y is computed as in Eq (5.4).

5.2 Experiments

We delve into several research questions on the utility of modeling longer sequences, as opposed to short-term dependencies; and the utility of modeling two contemporaneous sequence types, as opposed to relying on one sequence type.

Dataset		#Sequence	#Item	#Average Length	#Average Basket Size
Alibaba	Support	23740	13498	11.2	5.5
	Target			5.3	1.8
MovieLens	Support	189858	8202	34.5	2.5
	Target			16.6	1.8

Table 5.1: Statistics for Alibaba, MovieLens

5.2.1 Setup

Datasets. We experiment with two public real-life datasets of different domains. The statistic is summarized in the Table 5.1.

Alibaba[1]: Alibaba provided mobile shopping data for the period from 18/11/2014 to 18/12/2014. For each user, we construct her session sequence, where each session contains the items she adopted within a day. From session sequences, we generate contemporaneous basket sequences of the two adoption types: *click* as support and *purchase* as target.

MovieLens[2]: We filter out ratings in the last three years from 01/2006 to 01/2009. We build a session sequence for each user, where each session represents what movies she adopted in a specific day. Here, the two adoption types are *selecting a movie to rate* as support (akin to clicking), and *assigning a movie a high rating* as target (akin to purchasing). High rating is at least 4.5 out of 5.

Preprocessing. We filter out infrequent items, i.e., fewer than 50 clicks for Alibaba or 20 ratings for MovieLens. Since the aim is to model sequences, we filter out sequences with less than 2 baskets. Sequences are separated chronologically by three non-overlapping periods, denoted as $(P_{\text{train}}, P_{\text{validate}}, P_{\text{test}})$. They are $(29, 1, 1)$ day(s) for Alibaba and $(31, 3, 3)$ months the for MovieLens. Following [110], we seek to recommend new items, and so ignore item candidates that have occurred in the most recent basket.

Evaluation Task & Metrics. The task is evaluated via top-K recommendations. For each testing sequence pair $\langle S, T\rangle$, we hide the last $target$ basket B to create $|B|$

[1] https://tianchi.aliyun.com/dataset/dataDetail?dataId=46

[2] https://grouplens.org/datasets/movielens/10m

testing instances with the ground-truth. We utilize two conventional metrics for top-K recommendations. The first metric is recall (*Recall@K*), defined as the percentage of testing instances with the ground truth item in top$-K$. In experiments, we mainly rely on the top-10 recommendations, i.e., *Recall@10*, but will later show other top$-K$ performances as well. To evaluate the overall ranking performance, the second metric is Mean Reciprocal Rank (*MRR*), computed as follows:

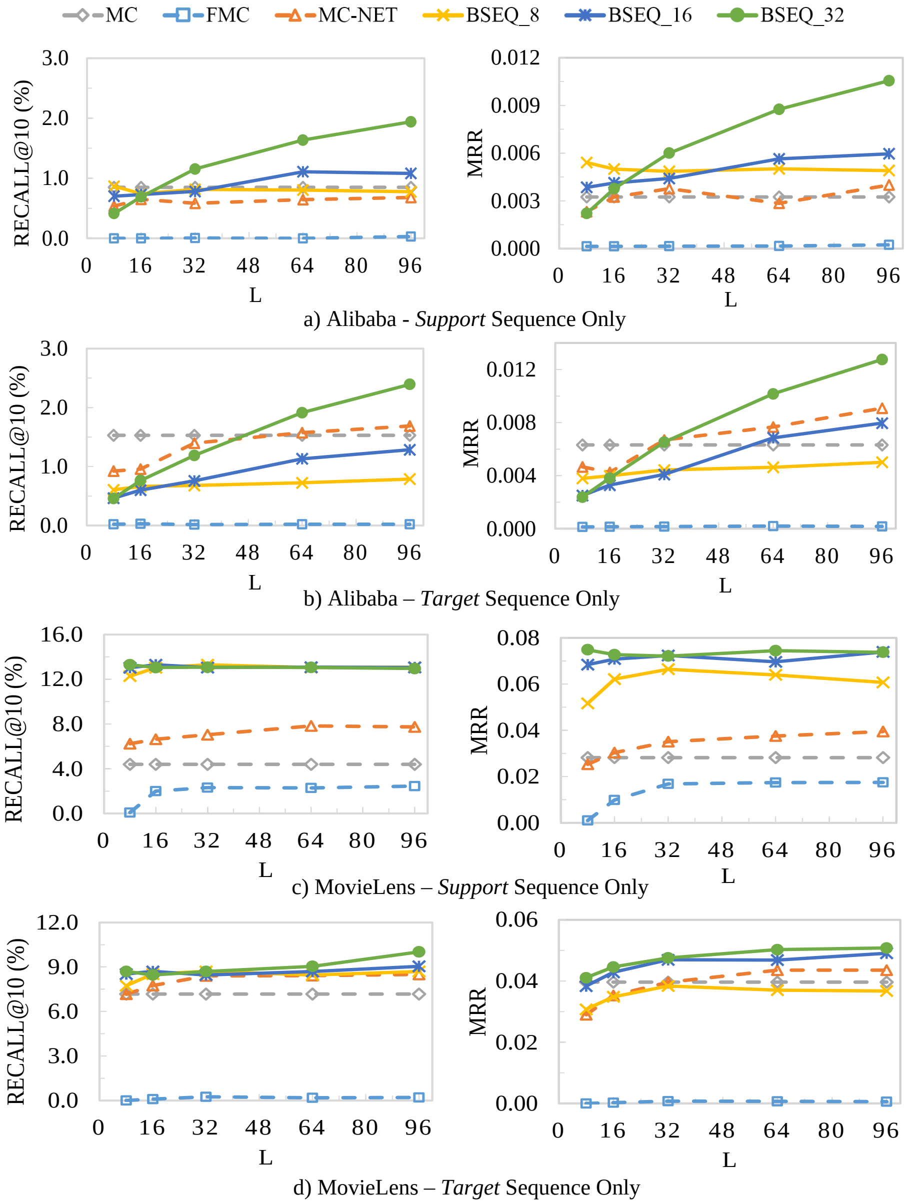
$$\mathit{MRR} = \frac{\sum_{B} \sum_{v \in B} \frac{1}{\text{rank of } v \text{ for } (S, T \backslash B)}}{\#\text{total testing instances}} \tag{5.7}$$

We cut the recommendation list off at 200 because the rest contribute almost zero to *MRR*. The performances are averaged across 30 runs with different random initializations. Comparisons are supported by one-tailed paired-sample Student's t-test at 0.05 significance level.

Learning Details. To learn parameters, we seek to minimize the cross-entropy loss based the output of the Eq (5.4). All neural networks are trained in 20 epochs of batch-size 32 by the *Adam* optimizer with the learning rate 0.001. The dense layer use the *ReLU* activation function to only keep positive weights. In the recurrent layer, $LSTM$ unit is applied with a 0.3 dropout probability. Additionally, we also measure the *Recall@10* for both training and validating sets. The performance on validation is used to decide whether to save learned models. After each epoch, its model is kept if the validation's accuracy is better than the previous epoch. Finally, the best model is used to generate top-K prediction on the testing set.

5.2.2 Research Questions

RQ1. Is modeling sequential data useful for next-item recommendation? To focus on the effect of sequence itself, rather than the effect of joining contemporaneous sequences, we first create a single-sequence variant of CBS, which we call *Basket Sequences* or BSEQ. We compare it to another approach that models only short-term transitions based on the *Markov chain* (MC) property. The first baseline MC



a) Alibaba - *Support* Sequence Only

b) Alibaba – *Target* Sequence Only

c) MovieLens – *Support* Sequence Only

d) MovieLens – *Target* Sequence Only

Figure 5.5: Performance Comparison of Next-Item Recommendations using *Markov Chain* vs. *RNN* on Alibaba, MovieLens

generates conditional probabilities of the next-item given the previous item. The probability of the next-item given the previous basket is the average over the transition probabilities from each basket item to the next item. The second baseline FMC factorizes the MC conditional probabilities to reduce the sparsity [110]. We use the *LibFM*[3] library to learn this model. The third baseline MC-NET is a simple neural network that feeds the last basket representation from the basket encoder to predict

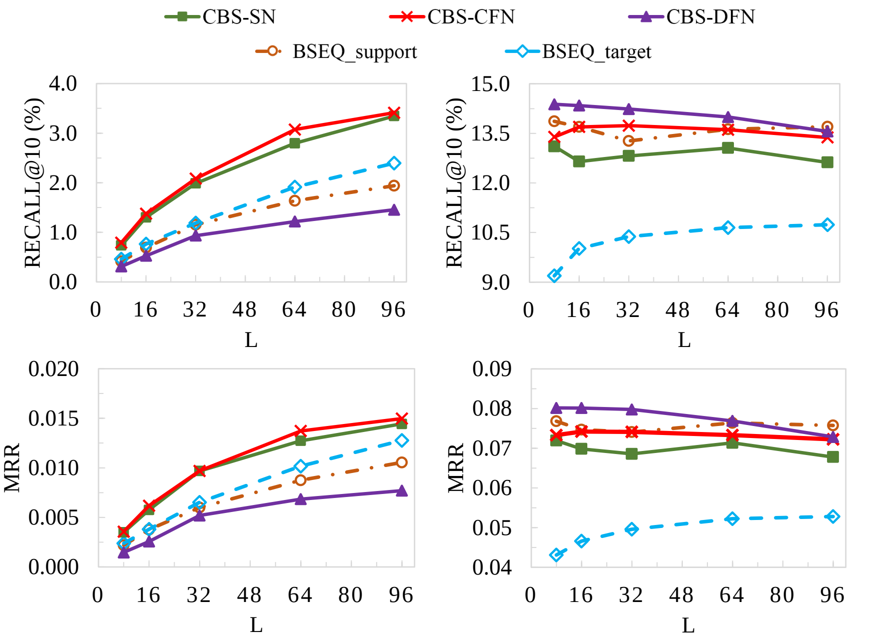
[3] http://www.libfm.org

the next item. For each sequence type, we train MC, FMC, MC-NET, BSEQ with various latent dimensions $L \in \{8, 16, 32, 64, 96\}$. BSEQ is investigated with three settings of the hidden state size $H \in \{8, 16, 32\}$, resulting in BSEQ_8, BSEQ_16, and BSEQ_32.

On Alibaba, Figure 5.5(a) shows the performance of the models when using only the support sequence, while Figure 5.5(b) shows the same for the target sequence only. The three *Markov*-based models are not influenced much by various latent dimensions L, except MC-NET on the *target* sequence. Because we are predicting for the target sequence, it is reasonable that the *Markov*-based models perform better when learnt on the target sequence than the support sequence. Importantly, not only are the three variants of BSEQ more sensitive to different latent dimensions, but they also show better results given sufficient L. BSEQ_32 is the best variant with a consistent improvement trend, which verifies the presence of longer-term dependencies in Alibaba sequences.

Figures 5.5(c) and (d) demonstrate the performances on MovieLens. We can draw similar conclusions as before on the strength of the BSEQ variants over the *Markov* baselines. Interestingly, the gap between the two families is larger on the *support* than on the *target*. This indicates a stronger longer-term dependency on the *support* sequences, while the robustness of the *Markov* baselines on the *target* sequences implies greater effect of short-term transitions on MovieLens's target sequence. Overall, BSEQ_32 still shows the best performance, and will be used subsequently for future comparisons.

RQ2. Is modeling contemporaneous sequences useful? We consider two model families: the single-sequence BSEQ and the dual-sequence CBS. Both run under the same setting of the hidden state size $(H = 32)$ and latent dimensions $L \in \{8, 16, 32, 64, 96\}$. In Figure 5.6(a), CBS-SN and CBS-CFN improve significantly upon the BSEQ models on Alibaba. Modeling contemporaneous basket sequences gives more information and supportive evidence by taking advantage of the long-term dependencies from both sequences types. The CBS-DFN model does not



a) Recall@10 and MRR on Alibaba b) Recall@10 and MRR on MovieLens

Figure 5.6: Performance Comparison of the BSEQ and CBS models on Alibaba, MovieLens.

work as well, which could be explained by the loss of information from confining the target sequence only to the most recent basket. This short-term dependency is different to the actual sequential dependency reflecting on the target sequence. Therefore, it triggers in the disagreement in the aggregation layer.

Figure 5.6(b) illustrates the performance of the two model families on MovieLens dataset. The observations of the two BSEQ models are consistent with what we found in the previous experiments. The longer-term dependency is stronger on the *support* sequence and weaker in the *target* sequence. This is the appropriate scenario for CBS-DFN, which shows the biggest improvements over BSEQ_support.

Generally, the fraternal networks (CBS-CFN and CBS-DFN) tend to perform better than the Siamese networks (CBS-SN) because the former could more flexibly fit the two sequence types. Between CBS-CFN and CBS-DFN, CBS-CFN has an advantage when there is more long-term dependency in the target sequence whilst CBS-DFN has an advantage when there is more short-term dependency in the target sequence.

Dataset	Model	L	MRR	$Recall@K$ (%)		
				10	20	50
Alibaba	POP	-	0.004	0.51	0.68	1.19
	DRM$_{support}$	64	0.011	2.14	2.91	4.02
	DRM$_{target}$	32	0.004	0.72	1.19	1.98
	BSEQ$_{support}$	96	0.011	1.94	2.54	3.92
	BSEQ$_{target}$	96	0.013	2.39	3.14	4.56
	CBS-SN	96	0.014	3.34	4.13	5.43
	CBS-CFN	96	**0.015**‡§	**3.41**‡§	**4.36**‡§	**5.60**‡§
	CBS-DFN	96	0.008	1.45	1.90	3.02
MovieLens	POP	-	0.006	1.79	2.89	6.58
	DRM$_{support}$	96	0.002	0.37	0.71	1.53
	DRM$_{target}$	16	0.001	0.20	0.36	0.77
	BSEQ$_{support}$	8	0.075	13.87	18.65	28.50
	BSEQ$_{target}$	64	0.050	10.65	15.55	25.95
	CBS-SN	8	0.070	13.11	17.66	27.67
	CBS-CFN	32	0.072	13.73	18.80	**29.65**‡§
	CBS-DFN	8	**0.078**‡§	**14.38**‡§	**19.39**‡§	29.33

Table 5.2: Best Performance Comparison on Alibaba, MovieLens. The symbols $\ddagger, \S$ denote the statistically significant improvements of our best model over the BSEQ and DRM models respectively

RQ3. How does the proposed CBS models perform against other baselines? We summarize the best performance of our proposed models as compared to baselines in Table 5.2. POP recommends items based on popularity. DRM is the recently proposed dynamic recurrent model [161], which is a state-of-the-art baseline capable of modeling basket sequences of a single type. By design, it is limited to fixing the same number of latent dimensions and hidden state size $(H = L)$. We consider the same setting $H = 32$ for BSEQ and CBS. For BSEQ, CBS, and DRM, we tune $L \in \{8, 16, 32, 64, 96\}$ for their respective best performance and indicate the chosen L in Table 5.2. For Alibaba, CBS-CFN significantly outperforms the baselines, i.e., BSEQ (+15%/ +43%), DRM (+36%/ +59%) in terms of $HLU/Recall@10$ as relative improvements over their best variant. This implies the effectiveness on modeling contemporaneous sequences. With sustainable gaps (+7%/+2%) in comparing to CBS-SN, CBS-CFN shows the benefits of giving the sequence types some flexibility in the recurrent layers. For MovieLens, CBS-DFN is the best-performing model with significantly relative improvements over BSEQ (+4%/ +4%) and CBS-SN (+11%/ +10%). It is 38 times better than DRM, which might be converged at a local optima. Yet, these improvements again confirm the usefulness of contemporaneous sequences.

5.3 Summary

In this chapter, we address the next-item recommendation by modeling contemporaneous basket sequences. We introduce three architectures based on twin networks, which vary in the degree of similarity or parameter sharing across the two sequence types. Experiments show that there is utility to modeling sequential data, with two sequence types better than one. The two sequence types may benefit from some flexibility in their parameters and size, as supported by the good performance of the fraternal variants over the Siamese variant.

Chapter 6

Correlation-Sensitive Next-Basket Recommendation

The notion of *basket-level* adoptions is observed in various real-life scenarios. For examples, people purchase a set of items within an online shopping session, or book a package of Point-of-Interests (POIs) for a trip, or assign a set of tags to a bookmark url, etc. Taking into account the time-sensitive manner, a user creates a basket-level adoption sequence from his active sessions. There might exist basket-oriented associations among basket items and sequential associations across sequence baskets. Being different from the previous chapter, we seek to model basket sequences for the next-basket recommendation task, which has got a lot of attraction recently. However, basket items are often suggested independently and without their correlations. This limitation motivates us to deal with a novel research problem, referred to as *correlation-sensitive next-basket recommendation*, i.e., next basket with correlated items. Towards this goal, we develop a hierarchical network named *Beacon* to jointly model the two associations on basket sequences for next-basket recommendations.

6.1 Model

6.1.1 Basket-Sequence Correlation Networks (*Beacon*)

In this section, we introduce the problem formulation on modeling basket sequences for correlation-sensitive next-basket recommendation. Figure 6.1 demonstrates our proposed framework named Basket-Sequence Correlation Networks (*Beacon*), which consists of three main components.

First, let us denote $V = \{v_0, v_1, ..., v_{N-1}\}$ as the set of N items. Each basket $B \subset V$ is transformed into an equivalent binary vector $X =< x_0, x_1, ..., x_{N-1} > \in \{0, 1\}^N$, whereby $x_i = 1$ if the item v_i appears in B and 0 otherwise.

Correlation-Sensitive Basket Encoder. Apparently, there are two primary factors that trigger the presence of an item in the basket B_t at a given time t, including its bias and correlative dependencies with other basket items. Taking into account those information may help enhance the representation of B_t. Let us denote $z_t \in \mathbb{R}^N$ as the intermediate representation of B_t, we have:

$$z_t = X \odot \Omega_V + X\hat{A} \tag{6.1}$$

where $\odot$ is the element-wise product; $\Omega_V \in \mathbb{R}^N$ are item bias parameters to be learned; $\hat{A}$ is a correlation matrix built from "observed" basket. Each cell in $\hat{A}$ represents how strong two items are correlated. The construction of $\hat{A}$ will be presented in Section 6.1.3.

Generally, not all correlative associations are useful. Weak associations might become noises which affect the representation of B_t. Therefore, we introduce $\eta_A \in \mathbb{R}$, a single-value parameter to cancel out these noises automatically. The intermediate representation z_t could be transformed as:

$$z_t = X \odot \Omega_V + ReLU(X\hat{A} - \eta_A \mathbf{1}) \tag{6.2}$$

where $\mathbf{1}$ is a vector of ones. Subsequently, z_t is fed into a fully-connected layer to

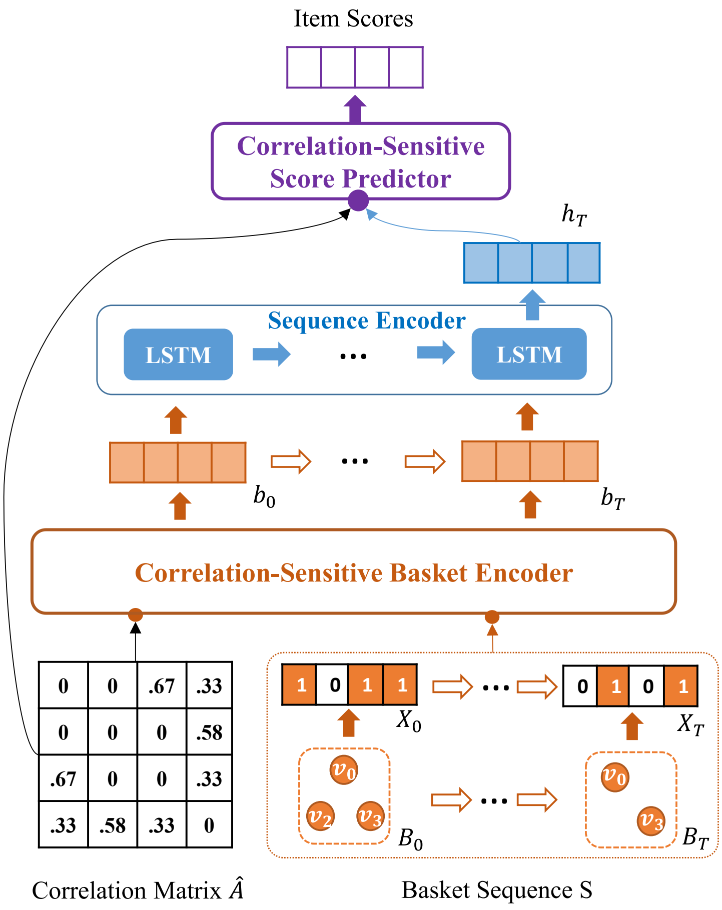


Figure 6.1: The architecture of *Beacon* model

infer the $D-$dimensional latent representation as follows:

$$b_t = ReLU(\Phi_z z_t + \Omega_z) \tag{6.3}$$

where $\Phi_z \in \mathbb{R}^{D\times N}, \Omega_z \in \mathbb{R}^D$ are parameters to be learned.

Sequence Encoder. Given a basket sequence length T: $S = \{B_0, ..., B_T\}$, the recurrent hidden output h_t at time t is computed by:

$$h_t = tanh(\Phi_b b_t + \Phi_h h_{t-1} + \Omega_h) \tag{6.4}$$

where $\Phi_b \in \mathbb{R}^{H\times D}, \Phi_h \in \mathbb{R}^{H\times H}, \Omega_h \in \mathbb{R}^H$ are parameters to be learned.

Correlation-Sensitive Score Predictor. Suppose that h_T is the last hidden output of S via the sequence encoder, the sequential signal for next-item adoption is estimated by:

$$p_V = \sigma(\Phi_S h_T) \tag{6.5}$$

where σ is sigmoid function, $\Phi_S \in \mathbb{R}^{N \times H}$ is a parameter to be learned. In order to recommend a basket with correlated items, p_V is then aggregated with item biases and correlative dependencies like Eq (6.1). Here, we assume that "predicted" baskets consistently follow correlative dependencies in $\hat{A}$ as "observed" ones. The threshold parameter η_A is not applied to cover all possible combinations in the recommendation phase. The aggregated vector $y^S \in \mathbb{R}^N$ is as follows:

$$y^S = p_V \odot \Omega_V + p_V \hat{A} \tag{6.6}$$

Empirically, we found that y^S is more about basket-sensitive associations rather than sequence-sensitive associations. The sequential information p_V might be absorbed by $\Omega_V, \hat{A}$ so that different sequences have the same recommendation. We flexibly control basket-oriented and sequential factors using a hyper-parameter $\alpha \in [0, 1]$:

$$o^S = \alpha.y^S + (1 - \alpha).p_V \tag{6.7}$$

where $o^S \in \mathbb{R}^N$, each o_i^S indicates the score of item v_i.

Correlation-Sensitive Next-basket recommendation. Once all parameters are learned, for each basket sequence S of length T, we can produce o^S to measure how likely items could be selected for the next-basket recommendation. Given B_{T+1} as the "predicted" basket, we expect its items would have higher scores than others in o^S. Let us denote $\hat{r}^S$ as the predicted ranking of item v_i based on o_i^S ($\hat{r}_i^S \in \{1, 2, ..., |V|\}$). Ideally, items in B_{T+1} should have ranks in the range $[1, |B_{T+1}|]$, which are the top positions.

In practice, the size of B_{T+1} is unknown so that we approximate the prediction for B_{T+1} as a basket of size K:

$$B_{T+1}|K = \{v_i | \hat{r}_i^S \leq K\} \tag{6.8}$$

6.1.2 Learning Strategy.

Let us denote $S' = S \setminus B_T$, $V' = V \setminus \{v_i \in B_T\}$ and $\hat{r}^{S'}$ is the respective item ranking for the next basket recommendation. Inspired by [123], we approximate the ranking $\hat{r}_i^{S'} \approx {}^1\!/_{\sigma(o_i^{S'})}$ with σ is the sigmoid function. The log loss $\mathcal{L}_S$ of the next-basket recommendation task on S is estimated by:

$$\begin{aligned}\mathcal{L}_S &= -\sum_{v_i \in B_T} [\ln \sigma(o_i^{S'}) + \sum_{v_j \in V'} \ln(1 - \sigma(o_j^{S'} - o_i^{S'}))] \\ &= -\sum_{v_i \in B_T} \ln \sigma(o_i^{S'}) - \sum_{v_i \in B_T} \sum_{v_j \in V'} \ln(1 - \sigma(o_j^{S'} - o_i^{S'}))\end{aligned} \tag{6.9}$$

The main idea of Eq (6.9) is to favor the rankings of adopted basket items (i.e., the first summation) as well as penalizing others ranked higher than them (i.e., the second summation). However, $|V'| \gg 1$ triggers the unbalance between the two summations to the loss. $\mathcal{L}_S$ is transformed as follows:

$$\mathcal{L}_S = -\sum_{v_i \in B_T} \ln \sigma(o_i^{S'}) - \frac{|B_T|}{|V'|} \sum_{v_j \in V'} \ln(1 - \sigma(o_j^{S'} - o_m^{S'})) \tag{6.10}$$

where $o_m^{S'} = \min\{o_i^{S'} | v_i \in B_T\}$;

Inference. Given the training basket sequence set $\mathcal{S}$, we seek to minimize the total log loss to infer parameters:

$$\theta^* = \text{argmin}_{\theta \in \Theta} \sum_{S \in \mathcal{S}} \mathcal{L}_S \tag{6.11}$$

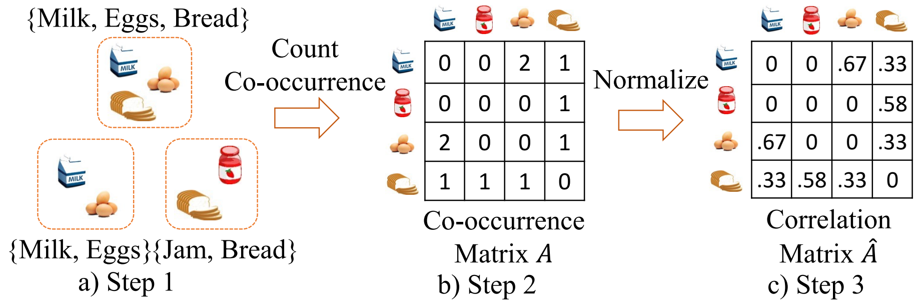


Figure 6.2: A 3-step construction of the correlation matrix $\hat{A}$, built on three baskets $\{$*Milk, Eggs*$\}$, $\{$*Jam, Bread*$\}$, $\{$*Milk, Eggs, Bread*$\}$.

6.1.3 Correlation Matrix

As mentioned earlier, correlative matrix is constructed from "observed" baskets. It is utilized to leverage quantitatively correlative dependencies between item pairs. Hence, it is required to satisfy the following two properties:

- Item pair with frequent co-occurrence should have a higher correlative score than less frequent pairs.
- Item pair with an exclusive connection should have a higher correlative score than non-exclusive pairs.

The first requirement leads us to consider *symmetric co-occurrence matrix*, denoted as $A \in \mathbb{R}^{N \times N}$, a simple way to represents co-occurrences. Considering a set of three baskets including $\{$*Milk, Eggs*$\}$, $\{$*Jam, Bread*$\}$, $\{$*Milk, Eggs, Bread*$\}$, Figure 6.2(b) illustrates how A preserves the co-occurrence information. *Milk* and *Eggs* co-occur in two baskets hence the respective cells have value of 2. Likewise, $\{$*Jam, Bread*$\}$, $\{$*Milk, Bread*$\}$ and $\{$*Eggs, Bread*$\}$ have score of 1. Other pairs that have never correlated are set 0. However, the exclusive pair $\{$*Jam, Bread*$\}$ has the same score as the non-exclusive ones namely $\{$*Milk, Bread*$\}$, $\{$*Eggs, Bread*$\}$. This violates the second condition.

Inspired by [57], the violation could be removed by normalizing A using the the Laplacian technique as follows:

$$\hat{A} = \tilde{D}^{-\frac{1}{2}} A \tilde{D}^{-\frac{1}{2}} \tag{6.12}$$

where $\tilde{D}$ is the degree matrix, $\tilde{D}_{ii} = \sum_j A_{ij}$. Continuing the previous example, we have got the normalized correlation matrix $\hat{A}$ as shown in Figure 6.2(c). It is clear that $\hat{A}$ satisfies both conditions. Yet, $\{$*Milk, Eggs*$\}$ has the highest score 0.67 while $\{$*Jam, Bread*$\}$ shows a higher score (i.e., 0.58) than the two other pairs $\{$*Milk, Bread*$\}$, $\{$*Eggs, Bread*$\}$ with the score of 0.33.

Furthermore, in some cases, the correlation matrix could be too sparse to provide useful associations. We may consider higher-order correlations up to the N-th order, i.e., $\hat{A} + \sum_{n=2}^{N} \mu^{n-1} \text{Norm}(\hat{A}^n)$, where $\mu \in (0, 1)$ is a discount factor for higher orders, and $\text{Norm}(\cdot)$ sets the diagonal to zero and applies the same normalization in Eq (6.12).

6.2 Experiments

Here, we study four research questions to investigate the efficiency of *Beacon* for the correlation-sensitive next-basket recommendation task.

6.2.1 Setup

Dataset. We conduct experiments on three public real-life datasets of three different domains as follows:

TaFeng[1]: It is a grocery shopping dataset containing users' purchased transactions from November 2000 to February 2001. Each transaction is considered as a basket of purchased items. Given a user, the basket sequence is built by sorting his basket-level adoptions chronologically.

[1] http://www.bigdatalab.ac.cn/benchmark/bm/dd?data=Ta-Feng

Dataset	#Sequence	#Item	Average Length	Average Basket Size
TaFeng	77209	9964	7.0	5.9
Delicious	61908	6520	21.4	3.8
Foursquare	100980	5527	22.2	1.8

Table 6.1: Statistics for TaFeng, Delicious, Foursquare datasets

Delicious[2]: This data is shared in Recsys 2011. It contains social networking, bookmarking and tagging information from Delicious social bookmarking system. Here, each bookmark is associated with a basket of tag assingments. A bookmark list of a given user will generate a basket sequence of tags.

Foursquare[3]: This is Singaporean check-ins data from Foursquare, collected from August 2010 to July 2011 [164]. We define a basket as the set of check-ins within the same day. The basket sequence of a given user is constructed via active days of the period.

Pre-processing. For sufficient recommendations, we filter out all users and items, which do not satisfy the k-core property, i.e., each user adopts at least k items and each item is adopted by at least k users. The values of k are 10, 5, 5 for TaFeng, Delicious and Foursquare respectively. Additionally, we also filter out basket sequences with fewer than 2 baskets. In order to create train/validation/test sets, sequences are cut chronologically into three non-overlapping periods $(t_{\text{train}}, t_{\text{val}}, t_{\text{test}})$, i.e., (3, 0.5, 0.5) months for TaFeng, (80, 2, 2) months for Delicious and (10, 0.5, 0.5) months for Foursquare. For the train and validation sets, we generate all sub-sequences of the basket sequences having more than 3 baskets. Sub-sequences with more than 30 baskets are shorten with the prefix cut off. For the testing set, all items appeared in the most recent basket are ignored to facilitate new-item recommendations [111]. The statistics after preprocessing are described in Table 6.1.

Correlation Matrix. We construct the input correlation matrix according to Section 6.1.3. Based on the validation set, we choose the first-order correlation for Delicious and Foursquare whilst adopting the higher-order correlation for TaFeng

[2] https://grouplens.org/datasets/hetrec-2011

[3] http://www.ntu.edu.sg/home/gaocong/datacode.htm

with $N = 5$ and $\mu = 0.85$.

Evaluation metrics. Given a testing sequence S, we take out the last basket B as the ground-truth. All models are required to generate the next-basket recommendation relied on the preceding basket sequence $S' = S \backslash \{B\}$. This recommendation is then compared to the ground-truth basket B to investigate the top-K recommendation performance. We consider the two conventional metrics. The first metric is F-measure ($F1@K$), computed via recall ($R@K$), precision ($P@K$) as:

$$\begin{aligned}
\text{Top@K} &= \{v_i | \hat{r}_i^S \leq K\} \\
\text{R@K} &= \frac{|\text{Top@K} \cap B|}{|B|} \\
\text{P@K} &= \frac{|\text{Top@K} \cap B|}{K} \\
\text{F1@K} &= \frac{2 \times \text{P@K} \times \text{R@K}}{(\text{P@K} + \text{R@K})}
\end{aligned}$$

To evaluate the overall ranking performance, we consider the second metric, Half-life utility (HLU) a.k.a. ’Breese score’ [13]:

$$\text{HLU} = C \times \frac{\sum_{v_i \in B} 2^{\frac{1-\hat{r}_i^S}{\beta-1}}}{\sum_{r=1}^{|B|} 2^{\frac{1-r}{\beta-1}}}$$

where the scaling hyper-parameter C, the half-life hyper-parameter β are set to 100, 5 respectively. We also use another ranking metric, known as Mean reciprocal ranking (MRR):

$$\text{MRR} = \frac{1}{|B|} \times \sum_{v_i \in B} \frac{1}{\hat{r}_i^S}$$

The measurements are averaged across all test baskets using 10 runs with different random initialization. Comparisons are supported by two-tailed paired-sample Students t-test at 0.05 significance level.

Learning details. With the objective of minimizing the log loss in Eq (6.11), all models are trained in 15 epochs of batch-size 32. We use the RMSProp optimizer

Dataset	Model	D	H	F1@K (%)		HLU	MRR
				5	10		
TaFeng	POP	-	-	4.66	4.02	6.64	0.040
	MC	-	-	4.11	3.61	5.78	0.033
	MCN	32	-	4.56	4.02	6.34	0.031
	DRM	8	-	5.85	4.90	6.96	0.030
	BSEQ	32	16	4.48	4.04	6.34	0.031
	Beacon	8	64	**6.36**†‡§	**5.26**†‡§	**7.87**†‡§	**0.041**†‡§
Delicious	POP	-	-	3.88	4.04	6.05	0.035
	MC	-	-	4.27	4.59	6.52	0.035
	MCN	32	-	4.20	4.59	6.50	0.035
	DRM	32	-	3.13	3.47	4.93	0.028
	BSEQ	64	32	3.86	3.97	5.95	0.034
	Beacon	64	64	**4.93**†‡§	**5.47**†‡§	**7.76**†‡§	**0.042**†‡§
Foursquare	POP	-	-	2.73	2.90	4.84	0.030
	MC	-	-	3.58	3.43	5.53	0.030
	MCN	64	-	3.09	2.89	5.08	0.030
	DRM	64	-	2.84	3.00	4.98	0.030
	BSEQ	64	32	2.80	2.89	4.82	0.029
	Beacon	64	64	**3.61**†‡§	**3.59**†‡§	**6.32**†‡§	**0.040**†‡§

Table 6.2: Best Performance Comparison between *Beacon* versus baselines on TaFeng, Delicious and Foursquare. $\dagger, \ddagger, \S$ represent statistically significant improvements of *Beacon* over MCN, DRM, BSEQ respectively.

with the learning rate 0.001 to update gradients. The LSTM layer is applied with a 0.3 dropout probability. A grid search is performed on various latent dimension $D \in \{8, 16, 32, 64\}$ and hidden state size $H \in \{16, 32, 64\}$. Additionally, the log loss on the validation set is used to decide whether to save learned models. The best model is selected to evaluate the testing set.

6.2.2 Research Questions

RQ1. Does *Beacon* outperform baselines? In Table 6.2, we report the best performance of *Beacon* as compared to several baselines. POP is the popularity-based recommendation approach while MC recommends items relied on 1st-order Markov-chain dependencies. As recent works in modeling basket sequences for the next basket recommendation task, [161]) proposes DRM, a dynamic recurrent model, where a basket representation is aggregated by its items' embedding via a pooling layer; [65] introduces BSEQ to capture long-term sequential dependencies, each basket is encoded directly from a binary vector using a dense layer. MCN is a simple variant of BSEQ, which only model 1st-order Markov-chain dependencies. All models are tuned using the same set of latent dimension $D \in \{8, 16, 32, 64\}$ and recurrent hidden unit $H = \{16, 32, 64\}$. We use $\alpha = 0.5$ as the default setting for *Beacon*. The best setting in terms of $F1@5$ on the testing set is indicated in Table 6.2. In addition to the main metric, $F1@10, HLU$ and MRR are also given.

For TaFeng, the item popularity seems to be an important factor since POP results in better metrics than MC, MCN and BSEQ. Beyond the popularity, DRM and *Beacon* show advantages in capturing associations between basket items. Yet, *Beacon* is the best-performing model with significant relative improvements in terms of $F1@5$ over the three state-of-the-art baselines BSEQ(+42%), DRM(+9%) and MCN(+39%). Considering ranking metrics, there are similar observations on both HLU, MRR in comparing *Beacon* to BSEQ(+24%,+32%), DRM(+13%,+37%) and MCN(+24%,+32%).

For Delicious and Foursquare, Markov-based models (MC & MCN) do better than other baselines. It might imply that items in a testing basket are strongly dependent on the most recent basket. Overall, *Beacon* significantly outperforms all baselines across the three measurements. In terms of $F1@5$, the relative improvements of *Beacon* over BSEQ, DRM and MCN on Delicious are +17%, +58%, +28% respectively. Likewise, these values are +17%, +27%, +29% on Foursquare. We observe the consistent enhancement of *Beacon* over the baselines on the two remaining metrics HLU, MRR.

RQ2. Is the learning of item bias Ω_V helpful? With the objective of investigating the effectiveness of Ω_V in modeling baskets, we consider two *Beacon* variants without using the correlative information $(\hat{A}, \eta_A)$. One variant fixes the item bias Ω_V a vector of ones whilst the other learns the item bias Ω_V. Their best performances on TaFeng, Delicious & Foursquare are shown in Table 6.3. It is clear that the learning of item bias Ω_V generates significant improvements over the fixed one.

Dataset	Item Bias Ω_V	F1@K (%) 5	F1@K (%) 10	HLU	MRR
TaFeng	Fixed	3.87	3.44	5.13	0.025
	Learned	**5.78**‡	**4.86**‡	**7.18**‡	**0.033**‡
Delicious	Fixed	4.02	4.43	6.38	0.029
	Learned	**4.67**‡	**5.10**‡	**7.15**‡	**0.034**‡
Foursquare	Fixed	2.98	3.29	5.39	0.033
	Learned	**3.58**‡	**3.52**‡	**6.16**‡	**0.038**‡

Table 6.3: Performance Comparison of *Beacon* with fixed & learned Ω_V on TaFeng, Delicious and Foursquare. $\ddagger$ denotes a statistically significant improvement.

Dataset	Model Correlations?	F1@K (%) 5	F1@K (%) 10	HLU	MRR
TaFeng	No	5.78	4.86	7.18	0.033
	Yes	**6.36**§	**5.26**§	**7.87**§	**0.041**§
Delicious	No	4.67	5.10	7.15	0.034
	Yes	**4.94**§	**5.47**§	**7.76**§	**0.042**§
Foursquare	No	3.58	3.52	6.16	0.038
	Yes	**3.61**	**3.59**§	**6.32**§	**0.040**§

Table 6.4: Performance Comparison of *Beacon* with & without modeling Correlations on TaFeng, Delicious and Foursquare. $\S$ denotes a statistically significant improvement.

RQ3. Is the modeling of the correlation matrix useful? We compare *Beacon* to its variant that does not incorporate the correlation matrix and only learns the item bias Ω_V. Table 6.4 summarizes their best performance on TaFeng, Delicious & Foursquare in terms of $F1@5$. Other metrics namely $F1@10, HLU$ and MRR are also provided for references. There is a similar observation across all datasets, which *Beacon* triggers sustainable improvements over the compared variant. In other words, the modeling of correlations benefits to next-basket recommendations.

RQ4. How does the hyper-parameter α affect the performance? According to Eq (6.7), α is to balance between basket-level and sequential associations. The higher α somehow emphasizes more on endogenous attributes within baskets while the smaller value favors exogenous effects across baskets. We vary α and report the best performance of *Beacon* for each α value in terms of $F1@5$ on validation sets. The performances on HLU, MRR are also provided for references. To maintain the similar relative significance in comparison, we plot the performance of *Beacon*

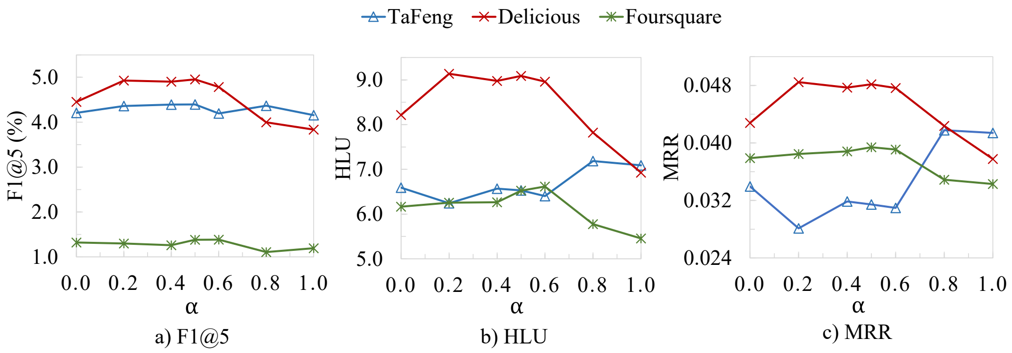


Figure 6.3: Best Performance Comparison of *Beacon* for various α on TaFeng, Delicious and Foursquare.

on TaFeng, Delicious and Foursquare in the same graph as shown in Figure 6.3.

For TaFeng, there are no remarkable differences on $F1@5$ for various α values. The comparable performance at $\alpha = 0.5$ and $\alpha = 0.8$ are relatively better than others. However, $\alpha = 0.8$ results in a significant improvement in terms of HLU, MRR. It signifies that items in testing baskets are ranked higher within top-5. This ranking is triggered by the stronger emphasis on important factors including item bias and correlations. For Delicious, we witness the *rise* and *fall* trend. There are similar results for $\alpha \in [0.2, 0.6]$. The lower performance for $\alpha = 0.0$ or $\alpha \geq 0.8$ reckons that both basket-oriented and sequential associations are useful. Likewise, we witness the similar observation for Foursquare.

6.3 Summary

In this chapter, we address the correlation-sensitive next-basket recommendation problem, which recommends the next basket with correlated items. Assuming baskets share a consistent knowledge on correlative dependencies among items, we propose *Beacon* that utilizes the correlation information to enhance the representation of baskets as well as the next-basket prediction. Experimental results on three public real-life datasets show the helpfulness of exploiting correlative dependencies in making next-basket recommendations.

Chapter 7

Conclusions & Future Work

7.1 Conclusions

In this dissertation, we present our research on modeling sequential and basket-oriented associations for *top-K recommendation*. We formulate and propose effective solutions for various recommendation problems, where the two association types are exploited separately and jointly.

In Chapter 3, we deal with the sequential recommendation task based on dynamic context factors. A generative model that takes into account sequential associations between consecutive items is built upon Hidden Markov Model (HMM). First, we incorporate dynamic context factors with the hypothesis that they might help to leverage the transition from one state to the next state. Additionally, we modify the emission probability with the presence of latent user groups. The reason is that cohorts of similar users may have analogous preferences on particular items. Evaluation on both artifact and real-life datasets shows statistically significant improvement of the proposed model over HMM for *top-K recommendations*.

In Chapter 4, our focus is to model the basket-oriented associations for the basket-completion problem. Given a user holding several items in his current basket, the task is to recommend the user what item should be added next. As the first solution, we propose BFM based on *Factorization Machines*, that simply consolidate various user-item and item-item association types. Extending this model, we introduce the constrained variant CBFM that regularizes similar intent baskets. Experiments on three real-life datasets reveal the effectiveness of our approach over closely related baselines including association rules and factorization machine (without modeling basket-oriented associations).

In Chapter 5, both association types are concurrently modeled for the next-item recommendation task with the notion of contemporaneous basket sequences. In our context, being contemporaneousness is considered as a pair of basket sequences of two different actions (e.g., click & purchase), which are created by a same user within the same time period. We present three twin network architectures that leverage the similarities and dissimilarities of user behaviors on the two sequences. Experiments on two real-life datasets show that the modeling of sequential and basket-oriented associations in contemporaneous basket sequences help to improve the *top-K recommendation* performance.

In Chapter 6, it is a further exploration in jointly modeling the two associations but for a novel task named *correlation-sensitive next-basket recommendation*, which aims to recommend next basket with correlated items. We hypothesize that items in both “observed” and “predicted” baskets follow a consistent correlative dependency matrix. The correlation is informative not only for the representation of baskets but also for making next-basket predictions. We propose *Beacon*, a hierarchical network architecture, which incorporates the matrix via two components including *Correlation-Sensitive Basket Encoder* and *Correlation-Sensitive Score Predictor*. Finally, we conduct extensive experiments on three real-life datasets of different domains. The results show *Beacon* can benefit from the correlation to create significant improvements over baselines for the next-basket recommendation problem.

7.2 Future Work

With the objective of improving further the research on modeling *sequential-* and *basket-oriented* associations for Top-K recommendation, we propose a few potential research directions as our future work.

Contemporaneous Actions. Within a period of time, users express their attention via various type of actions, e.g., click, like/favorite, add-to-cart, purchase items, etc. The co-occurrence might imply some latent causalities among these actions [139]. In Chapter 5, we present a research on modeling concurrently a pair of basket sequences of only two different action types (i.e., click and purchase). As a possible direction, we may upgrade the proposed models to be able to take into account the remaining action types (e.g., like or add-to-cart), which are useful to better model the preference of users. Additionally, different action types have different importance, hence there is another need to leverage their respective contemporaneous basket sequences in the aggregation layer before making recommendations. With these motivations, it is necessary to propose a better generic framework to not only capturing all contemporaneous actions but also aggregating them efficiently for the next-item recommendation task.

Multi-Intention Baskets with Memory Networks. Chapter 6 shows our exploration on modeling basket sequences for next-basket recommendations, which the basket-oriented associations are leveraged using the item-item correlation matrix constructed from observed baskets. Through the correlation-sensitive basket encoder, each basket is represented by an latent representation, which implies an underlying intention. However, there might have multiple intentions triggered in a single basket. Items belonged to an intention are remarkably correlated while items of different intentions are somehow independent. For examples, a user buys a basket of {*beef, black pepper, fresh milk, bread*}. The association of {*beef, black pepper*} signifies his intention on making beef steak for dinner while {*fresh milk, bread*} may be for his breakfast. Other associations seems to be uncommon in real-life scenarios. In order to model multiple intentions, it also raises a need to upgrade the RNN-based sequence encoder, due to the limitation in performing memorization [150, 128]. Recently, there are several works that applied memory networks [155, 50] in generating sequential recommendations. They not only improve the recommendation performance but also increase the interpretability of suggestions. Inspired by these interesting works, we aim to exploit memory networks on modeling multi-intention basket sequences for the next-basket recommendation task.

Bibliography

[1] Charu C Aggarwal et al. *Recommender systems*. Springer, 2011.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *VLDB*, volume 1215, pages 487–499, 1994.

[3] Yoram Bachrach, Yehuda Finkelstein, Ran Gilad-Bachrach, Liran Katzir, Noam Koenigstein, Nir Nice, and Ulrich Paquet. Speeding up the xbox recommender system using a euclidean transformation for inner-product spaces. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 257–264. ACM, 2014.

[4] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *International Conference on Extending Database Technology*, pages 588–596. Springer, 2004.

[5] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. Predicting the next app that you are going to use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, pages 285–294. ACM, 2015.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Proceedings of The 2nd International Conference on Learning Representations*, 2015.

[7] Ting Bai, Jian-Yun Nie, Wayne Xin Zhao, Yutao Zhu, Pan Du, and Ji-Rong Wen. An attribute-aware neural attentive model for next basket recommen-

dation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1201–1204. ACM, 2018.

[8] Roberto Battiti. Accelerated backpropagation learning: Two optimization methods. *Complex Systems*, 3(4):331–342, 1989.

[9] Robert M Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 7–14, 2007.

[10] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[11] Gerlof Bouma. Normalized (pointwise) mutual information in collocation extraction. In *Biennial GSCL Conference*, volume 156, 2009.

[12] Ronen I Brafman, David Heckerman, and Guy Shani. Recommendation as a stochastic sequential decision problem. In *ICAPS*, pages 164–173, 2003.

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

[14] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744, 1994.

[15] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, and Tat-Seng Chua. Embedding factorization models for jointly recommending items and user generated lists. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 585–594. ACM, 2017.

[16] Jun Chen, Chaokun Wang, and Jianmin Wang. A personalized interest-forgetting markov model for recommendations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 16–22, 2015.

[17] Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722, 2012.

[18] Shuo Chen, Jiexun Xu, and Thorsten Joachims. Multi-space probabilistic sequence modeling. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 865–873. ACM, 2013.

[19] Wei Chen, Wynne Hsu, and Mong Li Lee. Modeling user's receptiveness over time for recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 373–382. ACM, 2013.

[20] Chen Cheng, Haiqin Yang, Michael R Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, volume 13, pages 2605–2611, 2013.

[21] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198. ACM, 2016.

[22] Arpita Das, Harish Yenala, Manoj Chinnakotla, and Manish Shrivastava. Together we stand: Siamese networks for similar question retrieval. In *ACL*, volume 1, pages 378–387, 2016.

[23] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Liv-

ingston, et al. The youtube video recommendation system. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 293–296. ACM, 2010.

[24] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.

[25] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 193–202. ACM, 2014.

[26] Pierre Dupont, François Denis, and Yann Esposito. Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms. *Pattern recognition*, 38(9):1349–1371, 2005.

[27] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. Personalized ranking metric embedding for next new poi recommendation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[28] Jonathan Gemmell, Thomas Schimoler, Maryam Ramezani, Laura Christiansen, and Bamshad Mobasher. Improving folkrank with item-based collaborative filtering. *Recommender Systems & the Social Web*, 2009.

[29] Prem K Gopalan, Laurent Charlin, and David Blei. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*, pages 3176–3184, 2014.

[30] Jonathan Gordon, Linhong Zhu, Aram Galstyan, Prem Natarajan, and Gully Burns. Modeling concept dependencies in a scientific corpus. In *Proceedings*

of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), volume 1, pages 866–875, 2016.

[31] 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 *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1809–1818. ACM, 2015.

[32] Asela Gunawardana and Christopher Meek. Tied boltzmann machines for cold start recommendations. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 19–26. ACM, 2008.

[33] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Zadeh. Wtf: The who to follow service at twitter. In *Proceedings of the 22nd international conference on World Wide Web*, pages 505–514. ACM, 2013.

[34] Ido Guy, Naama Zwerdling, Inbal Ronen, David Carmel, and Erel Uziel. Social media recommendation based on people and tags. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 194–201. ACM, 2010.

[35] Eui-Hong Sam Han and George Karypis. Feature-based recommendation system. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 446–452. ACM, 2005.

[36] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *SIGMOD*, pages 1–12, 2000.

[37] Jing He, Xin Li, Lejian Liao, Dandan Song, and William K Cheung. Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. In *The Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[38] Ruining He, Wang-Cheng Kang, and Julian McAuley. Translation-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*, pages 161–169. ACM, 2017.

[39] Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 191–200. IEEE, 2016.

[40] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[41] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on Information & Knowledge Management*, 2018.

[42] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings of The 3rd International Conference on Learning Representations*, 2016.

[43] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the ACM Conference on Recommender Systems*, pages 241–248, 2016.

[44] Chantal Hoekstra, Zhen Zhen Zhao, Cornelius B Lambalk, Gonneke Willemsen, Nicholas G Martin, Dorret I Boomsma, and Grant W Montgomery. Dizygotic twinning. *Human reproduction update*, 14(1):37–47, 2007.

[45] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[46] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

[47] Diane Hu, Raphael Louca, Liangjie Hong, and Julian McAuley. Learning within-session budgets from browsing trajectories for item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 2018.

[48] Jianying Hu, Michael K Brown, and William Turin. Hmm based online handwriting recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 18(10):1039–1045, 1996.

[49] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. Diversifying personalized recommendation with user-session context.

[50] Jin Huang, Wayne Xin Zhao, Hong-Jian Dou, Ji-Rong Wen, and Edward Y. Chang. Improving sequential recommendation with knowledge-enhanced memory networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2018.

[51] Shanshan Huang, Shuaiqiang Wang, Tie-Yan Liu, Jun Ma, Zhumin Chen, and Jari Veijalainen. Listwise collaborative filtering. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 343–352. ACM, 2015.

[52] Mohsen Jamali and Martin Ester. Trustwalker: a random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 397–406. ACM, 2009.

[53] Peng Jiang, Yadong Zhu, Yi Zhang, and Quan Yuan. Life-stage prediction for product recommendation in e-commerce. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1879–1888. ACM, 2015.

[54] BH Juang and LR Rabiner. Hidden markov models for speech recognition. *Technometrics*, pages 251–272, 1991.

[55] George Karypis. Evaluation of item-based top-n recommendation algorithms. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 247–254. ACM, 2001.

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

[57] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *Proceedings of The 4th International Conference on Learning Representations*, 2017.

[58] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, 2015.

[59] Noam Koenigstein, Gideon Dror, and Yehuda Koren. Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In *Proceedings of the ACM Conference on Recommender Systems*, pages 165–172, 2011.

[60] Noam Koenigstein, Parikshit Ram, and Yuval Shavitt. Efficient retrieval of recommendations in a matrix factorization framework. In *Proceedings of the 21st ACM international conference on Information & Knowledge Management*, pages 535–544, 2012.

[61] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 42(8), 2009.

[62] Duc-Trong Le, Yuan Fang, and Hady W Lauw. Modeling sequential preferences with dynamic user and context factors. In *ECML-PKDD*, pages 145–161, 2016.

[63] Duc-Trong Le, Hady Wirawan LAUW, and Yuan Fang. Basket-sensitive personalized item recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 2060–2066, 2017.

[64] Duc-Trong Le, Hady Wirawan LAUW, and Yuan Fang. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018.

[65] Duc Trong Le, Hady Wirawan Lauw, and Yuan Fang. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. IJCAI, 2018.

[66] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 26th ACM international conference on Information & Knowledge Management*, pages 1419–1428. ACM, 2017.

[67] Ming Li, Benjamin M Dias, Ian Jarman, Wael El-Deredy, and Paulo JG Lisboa. Grocery shopping recommendations based on basket-sensitive random walk. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1215–1224, 2009.

[68] Qing Li and Byeong Man Kim. Clustering approach for hybrid recommender system. In *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on*, pages 33–38. IEEE, 2003.

[69] Rui Li, Shengjie Wang, Hongbo Deng, Rui Wang, and Kevin Chen-Chuan Chang. Towards social user profiling: unified and discriminative influ-

ence model for inferring home locations. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1023–1031. ACM, 2012.

[70] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1734–1743. ACM, 2018.

[71] Defu Lian, Vincent W Zheng, and Xing Xie. Collaborative filtering meets next check-in location prediction. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 231–232. ACM, 2013.

[72] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the ACM Conference on Recommender Systems*, pages 59–66, 2016.

[73] Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

[74] Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv:1506.00019*, 2015.

[75] Guimei Liu, Tam T Nguyen, Gang Zhao, Wei Zha, Jianbo Yang, Jianneng Cao, Min Wu, Peilin Zhao, and Wei Chen. Repeat buyer prediction for e-commerce. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 155–164. ACM, 2016.

[76] Liping Liu, Francisco Ruiz, Susan Athey, and David Blei. Context selection for embedding models. In *Advances in Neural Information Processing Systems*, pages 4819–4828, 2017.

[77] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. Context-aware sequential recommendation. In *Proceedings of the 16th International Conference on Data Mining*, pages 1053–1058, 2016.

[78] Qiang Liu, Shu Wu, and Liang Wang. Collaborative prediction for multi-entity interaction with hierarchical representation. In *Proceedings of the 24th ACM international conference on Information & Knowledge Management*, pages 613–622, 2015.

[79] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[80] Qiang Liu, Feng Yu, Shu Wu, and Liang Wang. A convolutional click prediction model. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1743–1746. ACM, 2015.

[81] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. Stamp: short-term attention/memory priority model for session-based recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1831–1839. ACM, 2018.

[82] Xin Liu, Yong Liu, Karl Aberer, and Chunyan Miao. Personalized point-of-interest recommendation by mining users' preference transition. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 733–738. ACM, 2013.

[83] Ashwin Machanavajjhala, Aleksandra Korolova, and Atish Das Sarma. Personalized social recommendations: accurate or private. *Proceedings of the VLDB Endowment*, 4(7):440–450, 2011.

[84] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172. ACM, 2013.

[85] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2015.

[86] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[87] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[88] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[89] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[90] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Proceedings of hte IEEE International Conference on Data Mining*, pages 669–672. IEEE, 2002.

[91] Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. Learning text similarity with siamese recurrent networks. In *Workshop on Representation Learning for NLP*, pages 148–157, 2016.

[92] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: accelerating recurrent network training for long or event-based sequences. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 3889–3897. Curran Associates Inc., 2016.

[93] Athanasios N Nikolakopoulos and George Karypis. Recwalk: Nearly uncoupled random walks for top-n recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 150–158. ACM, 2019.

[94] Douglas W Oard, Jinmook Kim, et al. Implicit feedback for recommender systems.

[95] Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 8th International Conference on Data Mining*, pages 502–511, 2008.

[96] Aditya G Parameswaran, Georgia Koutrika, Benjamin Bercovitz, and Hector Garcia-Molina. Recsplorer: recommendation algorithms based on precedence mining. In *SIGMOD*, pages 87–98, 2010.

[97] Rajiv Pasricha and Julian McAuley. Translation-based factorization machines for sequential recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 2018.

[98] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. Generating and personalizing bundle recommendations on steam. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2017.

[99] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

[100] Bruno Pradel, Savaneary Sean, Julien Delporte, Sébastien Guérif, Céline Rouveirol, Nicolas Usunier, Françoise Fogelman-Soulié, and Frédéric Dufau-Joel. A case study in a recommender system based on purchase data. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–385, 2011.

[101] Sanjay Purushotham, Yan Liu, and C-C Jay Kuo. Collaborative topic regression with social matrix factorization for recommendation systems. 2012.

[102] Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Recommending packages to groups. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 449–458. IEEE, 2016.

[103] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *arXiv preprint arXiv:1802.08452*, 2018.

[104] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 130–137. ACM, 2017.

[105] LR Rabiner. An introduction to hidden markov models. *IEEE ASSP MAGAZINE*, 1986.

[106] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. Repeatnet: A repeat aware neural recommendation machine for session-based recommendation. In *Proceedings of The 33rd AAAI Conference on Artificial Intelligence*, 2019.

[107] Steffen Rendle. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology*, 3(3):57, 2012.

[108] Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 273–282. ACM, 2014.

[109] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[110] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820, 2010.

[111] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *WWW*, pages 811–820. ACM, 2010.

[112] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

[113] Cynthia Rudin, Benjamin Letham, Ansaf Salleb-Aouissi, Eugene Kogan, and David Madigan. Sequential event prediction with association rules. In *Proceedings of the 24th annual conference on learning theory*, pages 615–634, 2011.

[114] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. A hidden markov model for collaborative filtering. *Mis Quarterly*, pages 1329–1356, 2012.

[115] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

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

[117] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[118] J Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166. ACM, 1999.

[119] Dimitris Serbos, Shuyao Qi, Nikos Mamoulis, Evaggelia Pitoura, and Panayiotis Tsaparas. Fairness in package-to-group recommendations. In *Proceedings of the 26th International Conference on World Wide Web*, pages 371–379, 2017.

[120] H Shan, Jens Kattge, P Reich, A Banerjee, F Schrodt, and Markus Reichstein. Gap filling in the plant kingdom-trait prediction using hierarchical probabilistic matrix factorization. In *ICML*, 2012.

[121] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.

[122] Yelong Shen, Ruoming Jin, Jianshu Chen, Xiaodong He, Jianfeng Gao, and Li Deng. A deep embedding model for co-occurrence learning. *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, pages 631–638, 2015.

[123] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: learning to maximize reciprocal rank with

collaborative less-is-more filtering. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 139–146. ACM, 2012.

[124] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, Nuria Oliver, and Alan Hanjalic. Climf: collaborative less-is-more filtering. In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, pages 3077–3081. AAAI Press, 2013.

[125] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the 5th ACM Conference on Recommender Systems*, pages 269–272. ACM, 2010.

[126] Brent Smith and Greg Linden. Two decades of recommender systems at amazon. com. *IEEE Internet Computing*, 21(3):12–18, 2017.

[127] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the 12th ACM international conference on Web search and data mining*, 2019.

[128] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015.

[129] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[130] Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. Modeling user activities on the web using paragraph vector. In *Proceedings of the 24th International Conference on World Wide Web*, pages 125–126. ACM, 2015.

[131] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM international conference on Web search and data mining*, 2018.

[132] Maryam Tavakol and Ulf Brefeld. Factored mdps for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 33–40. ACM, 2014.

[133] Nguyen ThaiBinh and Takasu Atsuhiro. Npe: Neural personalized embedding for collaborative filtering. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2018.

[134] Tran The Truyen, Dinh Q Phung, and Svetha Venkatesh. Ordinal boltzmann machines for collaborative filtering. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 548–556. AUAI Press, 2009.

[135] Trinh Xuan Tuan and Tu Minh Phuong. 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the ACM Conference on Recommender Systems*, pages 138–146, 2017.

[136] Flavian Vasile, Elena Smirnova, and Alexis Conneau. Meta-prod2vec: Product embeddings using side-information for recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 225–232. ACM, 2016.

[137] Andreas Veit, Balazs Kovacs, Sean Bell, Julian McAuley, Kavita Bala, and Serge Belongie. Learning visual clothing style with heterogeneous dyadic co-occurrences. In *Computer Vision (ICCV), 2015 IEEE International Conference on*, pages 4642–4650. IEEE, 2015.

[138] Kiewan Villatel, Elena Smirnova, Jérémie Mary, and Philippe Preux. Recurrent neural networks for long and short-term sequential recommendation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, 2018.

[139] Mengting Wan and Julian McAuley. Item recommendation on monotonic behavior chains. In *Proceedings of the 13th ACM Conference on Recommender Systems*, 2018.

[140] Mengting Wan, Di Wang, Matt Goldman, Matt Taddy, Justin Rao, Jie Liu, Dimitrios Lymberopoulos, and Julian McAuley. Modeling consumer preferences and price sensitivities from large-scale grocery shopping transaction logs. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1103–1112, 2017.

[141] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. Representing and recommending shopping baskets with complementarity, compatibility and loyalty. In *Proceedings of the 27th ACM international conference on Information & Knowledge Management*, pages 1133–1142. ACM, 2018.

[142] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 448–456, 2011.

[143] Jian Wang, Yi Zhang, Christian Posse, and Anmol Bhasin. Is it time for a career switch? In *Proceedings of the 22nd international conference on World Wide Web*, pages 1377–1388. ACM, 2013.

[144] Jianling Wang and James Caverlee. Recurrent recommendation with local coherence. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, 2019.

[145] Jun Wang, Arjen P De Vries, and Marcel JT Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508. ACM, 2006.

[146] Pengfei Wang, Jiafeng Guo, and Yanyan Lan. Modeling retail transaction data for personalized shopping recommendation. In *Proceedings of the 23rd ACM international conference on Information & Knowledge Management*, pages 1979–1982, 2014.

[147] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 403–412, 2015.

[148] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. Attention-based transactional context embedding for next-item recommendation. In *Proceedings of The 32nd AAAI Conference on Artificial Intelligence*, 2018.

[149] Yu-Ting Wen, Pei-Wen Yeh, Tzu-Hao Tsai, Wen-Chih Peng, and Hong-Han Shuai. Customer purchase behavior prediction from payment datasets. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 628–636. ACM, 2018.

[150] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *Proceedings of International Conference on Learning Representations*, 2015.

[151] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. Recurrent recommender networks. In *Proceedings of the 10th ACM international conference on Web search and data mining*, pages 495–503, 2017.

[152] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based recommendation with graph neural networks. In *Proceedings of The 33rd AAAI Conference on Artificial Intelligence*, 2019.

[153] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. Personalized next-song recommendation in online karaokes.

In *Proceedings of the 7th ACM conference on Recommender systems*, pages 137–140. ACM, 2013.

[154] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long-and short-term preference fusion. In *Proceedings of the ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 723–732, 2010.

[155] Chen Xu, Xu Hongteng, Zhang Yongfeng, Tang Jiaxi, Cao Yixin, Qin Zheng, and Zha Hongyuan. Sequential recommendation with user memory networks. In *Proceedings of the 11th ACM international conference on Web search and data mining*, 2018.

[156] Jaewon Yang, Julian McAuley, Jure Leskovec, Paea LePendu, and Nigam Shah. Finding progression stages in time-evolving event sequences. In *Proceedings of the 23rd international conference on World wide web*, pages 783–794. ACM, 2014.

[157] Ghim-Eng Yap, Xiao-Li Li, and S Yu Philip. Effective next-items recommendation via personalized sequential pattern mining. In *International Conference on Database Systems for Advanced Applications*, pages 48–64. Springer, 2012.

[158] Ankush Yewatkar, Faiz Inamdar, Raj Singh, Amol Bandal, et al. Smart cart with automatic billing, product information, product recommendation using rfid & zigbee with anti-theft. *Elsevier Computer Science*, 79:793–800, 2016.

[159] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. Lcars: a location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 221–229. ACM, 2013.

[160] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based

on hierarchical attention networks. In *the 27th International Joint Conference on Artificial Intelligence*, 2018.

[161] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. A dynamic recurrent model for next basket recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 729–732, 2016.

[162] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. Multi-order attentive ranking model for sequential recommendation. In *Proceedings of The 33rd AAAI Conference on Artificial Intelligence*, 2019.

[163] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, 2019.

[164] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 363–372. ACM, 2013.

[165] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. Sequential click prediction for sponsored search with recurrent neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1369–1375, 2014.

[166] Gang Zhao, Mong Li Lee, Wynne Hsu, and Wei Chen. Increasing temporal diversity with purchase intervals. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 165–174. ACM, 2012.

[167] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. Bundle recommendation in ecommerce. In *Proceedings of the International ACM SIGIR Confer-*

ence on Research and Development in Information Retrieval, pages 657–666, 2014.

[168] Yu Zhu, Hao Li, Yikang Liao, Beidou Wang, Ziyu Guan, Haifeng Liu, and Deng Cai. What to do next: modeling user behaviors by time-lstm. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3602–3608. AAAI Press, 2017.