Faculty of Engineering and Information Technology

University of Technology Sydney

Learning Complex Relations for Session-based Recommendations

A thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

by

Shoujin Wang

January 2019

CERTIFICATE OF AUTHORSHIP/ORIGINALITY

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree except as fully acknowledged within the text.

I also certify that the thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

Signature of Candidate

Dedication

To my parents and elder sisters for their love and support

Acknowledgments

Foremost, I would like to express my sincere gratitude to my supervisor Prof. Longbing Cao for his continuous support of my PhD study and research, for his patience, motivation, enthusiasm, and immense knowledge. His open and inclusive attitude has allowed me to become an interdisciplinary research student in data analytic and to build my own research capability step by step from scratch. His patient help has enlightened and encouraged me to rebuild my confidence when I lost confidence in research. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my PhD study.

I also would like to appreciate my co-supervisor Dr Wei Liu for providing me with continuous support throughout my PhD study and research, especially for his full guidance and unique help in the early stage. Without his professional guidance and persistent help, this thesis would not have been possible.

I thank my fellow labmates in Advanced Analystics Institute: Junfu Yin, Bin Fu, Fangfang Li, Guansong Pang, Chengzhang Zhu, Wei Wang, Thac Do, Xiaoshui Huang, Songlei Jian, Yan Xing, Sheng Luo, Qi Zhang, Ke Liu and Longxiang Shi for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had during my PhD period.

I place on record, my sense of gratitude to the key collaborator, Liang Hu in Advanced Analystics Institute for his expert and sincere help in my research. Without his professional guidance and persistent help, this thesis would not have been possible.

I am also grateful to those who provided help to my PhD study, including but not limited to: Associate Professor Ying Qian from Shanghai University, Professor Wanggen Wan from Shanghai University, Associate Professor Paul Kennedy from University of Technology Sydney, Dr Qinxue Meng from University of Technology Sydney, Associate Professor Defu Lian from University of Electronic Science and Technology of China, Associate Professor Wenpeng Lu, Associate Professor Quangui Zhang, Dr Jia Wu from Macquarie University, Dr Zhigang Zheng, Dr Peng Zhang, Dr Yonggang Huang and Dr Ting Zhang.

Last but not the least, I would like to thank my family for their unconditional support, both financially and emotionally throughout the whole PhD studying.

Shoujin Wang
August 2018 @ Analytics Cathedral,
Advanced Analytics Institute,
University of Technology Sydney, Sydney

Contents

List of Figures

List of Tables

List of Publications

Papers Published

- **Shoujin Wang**, Wei Liu, Jia Wu, Longbing Cao, Qinxue Meng and Paul J. Kennedy (2016), Training Deep Neural Networks on Imbalanced Data Sets. *in 'Proceedings of the 29th International Joint Conference on Neural Networks* (**IJCNN2016**)' , pp. 4368-4374.

- Liang Hu, Longbing Cao, **Shoujin Wang** (2017), Diversifying Personalized Recommendation with User-session Context. *in 'Proceedings of the 26th International Joint Conference on Artificial Intelligence* (**IJCAI2017**)', pp. 1858-1864.

- **Shoujin Wang**, Longbing Cao (2017), Inferring Implicit Rules by Learning Explicit and Hidden Item Dependency. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 10.1109/TSMC.2017.2768547.

- **Shoujin Wang**, Liang Hu, Longbing Cao (2017), Perceiving the Next Choice with Comprehensive Transaction Embeddings for Online Recommendation. *in 'Proceedings of the 15th Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (**ECML-PKDD2017**)', pp.285-302.

- **Shoujin Wang**, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian and Wei Liu (2018), Attention-based Transactional Context Embedding for Next-Item Recommendation. *in 'Proceedings of the 32nd*

AAAI Conference on Artificial Intelligence (**AAAI2018**)', pp.2532-2539.

Papers to be Submitted/Under Review

- **Shoujin Wang**, Longbing Cao, and Liang Hu (2018), HATE: Jointly Modeling Intra- and Inter-transaction Dependencies with Hierarchical Attentive Transaction Embeddings. *IEEE Transactions on Knowledge and Data Engineering.*
- **Shoujin Wang**, Longbing Cao, and Liang Hu (2018), A Survey on Session-based Recommender Systems. *ACM Computing Surveys.*

Abstract

In the era of big data, recommender systems (RSs) are a powerful engine to promote intelligent life by helping humans to make decisions concerning their daily necessities (e.g., food, clothes, and houses) much more efficiently and effectively, selecting from a large number of choices. Of the various types of recommender systems, session-based (SB) ones are of great value and significance, but they are not well studied.

The value of session-based recommender systems comes from two fold. From the research perspective, a SBRS takes a session as the basic unit for data organization and thus keeps the intrinsic nature of the original transaction-like data. As a result, the system effectively retains and models the rich information (e.g., intra-session dependency) embedded in a session structure to produce a more reliable recommendation. This modelling cannot be achieved by other types of recommender systems because they usually break down the original session data into multiple pair-wised user-item interactions to fit the models. From the business perspective, session data for session-based recommender systems is much more readily available than either the rating data or the item attribute data required by other recommender systems including content-based or collaborative filtering ones. This actually makes session-based RSs much more applicable in real-world business.

Though valuable, SBRSs are quite challenging. Generally, a hierarchical architecture consisting of five levels (cf. Figure 1.1) is built from the low-level feature values till to the high-level sessions in session data,as demonstrated in Chapter 1. The challenge arises mainly comes from three considerations: the heterogeneity of the elements in each level (e.g., there are both categorical and numerical features), the complex dependency within each level (e.g., the implicit inter-item relations), and the interactions between different levels (e.g., the inter-session dependency may affect the item occurrence). From my observation, the existing works mainly focus on the general item-level dependency modelling for session-based recommendations while ignoring other level relations as demonstrated in Chapter 3.

To bridge the huge gaps between the existing works and great challenges mentioned above, I build a systematic framework consisting of dependency modelling from the three core levels, i.e., feature-level, item-level and session-level (cf. Figure 1.1), for session-based recommendations. To the best of my knowledge, this is the first framework to systematically address various levels of challenges in session-based recommendations. Particularly, due to the limitations regarding space, I address one or two critical challenges in each level, as shown below.

In Chapter 4, to capture the implicit inter-item relations ignored by existing rule-based approaches, I proposed an implicit rule-based RS that first infers implicit rules and then applies the resultant rules for reliable rule-based recommendations, a basic approach for session-based recommendations.

In Chapter 5, I continue to work on item-level dependency modelling and focus on the issue of item heterogeneity, referring to different items with different levels of relevance to the next choice of an item. To this end, I build an attentive transaction-embedding model to discriminatively integrate multiple items in a transaction context into a unified context-embedding for next-item recommendations.

In Chapter 6, the feature-level dependency and feature-item interactions are modelled by a shallow neural network which takes both contextual items in a session and their corresponding features as the input. Accordingly, the cold-start issue in SBRSs has been well addressed.

In Chapter 7, the session-level (i.e.,transaction-level) dependency and session-item interactions are modelled. A hierarchical attentive transaction embedding model is built to jointly model the intra-session (item-level) and inter-session (session-level) dependency. Accordingly, the influence from previous sessions on a current session is incorporated for more accurate next-item recommendations.

All these models are applied to real-world transaction data, like Tmall and Tafang and they clearly outperform other representative SBRSs. More importantly, this thesis proposes a systematic framework to explore the driving force behind SBRSs, which provides some insights into both the researchers and engineers in this domain.

Chapter 1

Introduction

1.1 Research Background

1.1.1 An Overview of Recommender Systems

In the era of big data, people's daily living activities from eating, and deciding on clothes to housing and traveling–are becoming more and more convenient and intelligent. A prominent driver behind this trend is recommender systems (RSs): software tools and techniques that provide suggestions for products or services most likely to be of interest to a particular user (Resnick & Varian 1997, Burke 2007). The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read (Ricci, Rokach & Shapira 2015). For instance, thanks to the suggestions from recommender systems implanted in E-commerce websites and smart phone apps, people now use much less time and effort browsing and deciding upone what to eat or buy from the array of available outlets (Hu, Cao, Wang, Xu, Cao & Gu 2017). In practice, RSs accumulate massive profile and behaviour data from the end users to understand their habitats (Shoujin, Ying & Kun 2013, Xiaoling & Qiaofeng 2012) and preferences, so as to provide valuable and applicable suggestions to them.

There are various types of RSs, based on different work mechanisms; they work on different kinds of datasets and are applicable to different scenarios. For example, content-based recommender systems (CBRSs) work on item and user description (content) data to build item representations and user profiles, suggesting items similar to those a target user has liked in past (de Gemmis, Lops, Musto, Narducci & Semeraro 2015). On the contrary, collaborative filtering recommender systems (CFRSs) work mainly on user-item interaction data, usually the users' preference matrix on items, to predict those missing preference values without need for exogenous information about either items or users (Koren & Bell 2015). The hybrid recommender systems are the combination of these two while context-aware recommender systems (CARSs) incorporate the contextual information such as time, and place into a recommender system to make recommendations under certain circumstance (Adomavicius & Tuzhilin 2015). All these recommender systems have achieved great success and have been applied in a large variety of areas, such as item recommendations, music recommendations, movie recommendations, point of interest (POI) recommendations, and news recommendations. Some successful real-world applications include item-to-item collaborative filtering for product recommendations in Amazon (Linden, Smith & York 2003) and the various approaches for video recommendations that make up the Netflix recommender system (Gomez-Uribe & Hunt 2016).

1.1.2 Values and Significance of Session-based Recommender Systems

However, the success of the above RSs does not mean they are flawless. Some issues still have not been well addressed in existing RSs. One of the most critical is how to model the intrinsic session structure of transaction data, which is usually ignored in existing works. This blind spot exists because these RSs, including collaborative filtering, usually first break down the inherent session structure of the original transaction data into pair wised user-item interactions and then put the interaction pairs from all sessions together to form a global interaction pool. Note that in the shopping basket-based transaction data, *a session* refers to a transaction consisting of multiple items purchased in one supermarket visit (I will formally define the concept *session* in Chapter 2). In practice, a session is a basic unit in transaction-like data, one of the most essential characteristics of such data. The breaking down of sessions may easily lead to local information loss and thus cause problems. For example, the item co-occurrence information can be lost without session structures (e.g., the well known combination of bread and milk can be only observed in a session) , and the users' short-term preferences, usually indicated by the items occurrence in a session, are lost as well. Such local transaction information is quite critical for reliable recommendations in a specific transaction event. The reasons this information is important are varied: (a) Without session information, it is easy to generate duplicated recommendations, namely to recommend items similar or identical to those ones already in hand. (b) Without session information, the users' shopping behaviour patterns (e.g., some users may like to buy milk when they have got bread, while others may like to pick up juice instead) are lost and thus it is hard to generate personalized recommendations that match a user's specific behaviours well. (c) Without session information, the users' preference shifts are lost, and thus it is impossible to capture a users' current preferences to make timely recommendations; users' preferences are often dynamic rather than static. (d) Without session information, it is difficult to capture users' local and short-term preferences, which is shown only in very recent sessions rather than in all sessions, for reliable recommendations. In practice, the aforementioned recommender systems usually capture only the users' global and long-term preferences.

To address these issues, session-based recommender systems (SBRSs) are proposed. Different from other RSs, SBRSs remain the session structure of the transaction data and take it as a basic data unit. In this case, all the local information is kept that is ignored by other RSs, such as content-based filtering and collaborative filtering approaches. As a result, SBRSs can easily fix the defects of these RSs and thus provide more reliable recommendations by focusing more on the dynamic and local aspects. To be specific, a SBRS usually takes a part of session as the context for the prediction of the unknown part of the session, for example, next-item(s) recommendations (Hu, Cao, Wang, Xu, Cao & Gu 2017), or takes several recent sessions as the context to predict the next session, for example, next-basket recommendations (Rendle, Freudenthaler & Schmidt-Thieme 2010). As the items already chosen in the current or recent sessions are considered, SBRSs more easily avoid recommending items that are similar or identical to them; thus, the duplicated recommendations are avoided. Also, as each user's purchased items are organized into sessions, it becomes easy to know his or her transaction behaviour patterns within and between sessions by analysing item distributions in and across sessions. Furthermore, only the current or recent sessions, rather than the whole item pool from all sessions, are inputted into a SBRS, which enables the possibility of paying attention to users' short-term and local preferences at specific time points rather than the long-term and general preference. This approach actually makes the recommendations more timely, focused and reliable. More importantly, it makes easier the observation of a user's preference shifts by analysing the item changes in a sequence of sessions in SBRSs. In practice, this treats a user's preferences as dynamic rather than static, more closely matching real-world cases.

From the business perspective, SBRSs are even more important than other RSs. Essentially, the availability of session data is much stronger than that of data (e.g., item features or ratings) required by other RSs. This provides SBRSs a much wider application range. In practice, SBRSs can be applied anywhere as long as transactions are recorded.

Noticing the necessity and significance of SBRSs in both the academy and industry, some researchers have started to work in this area on various of specific tasks, including next-song recommendations, next-item recommendations and next-basket recommendations. The approaches are built on different techniques including pattern/rule mining, Markov-chain models, factorization machines and neural networks. Although great progress has been achieved, some obvious defects remain, promoting the work of this thesis. A comprehensive review of these existing session-based recommender systems will be given in Chapter 3. Next, I offer a brief summary and a gap analysis, which triggered my thinking in this topic and motivate my ideas in this thesis.

1.2 Research Motivations

To motivate my research, I first present an overview of the key challenges and complexities in this area and then briefly summarize existing works.

1.2.1 An Overview of Challenges and Issue Complexity

Though extremely valuable and significant, session-based recommender systems are yet quite challenging and complex as shown in Figure 1.1. Clearly, there is a hierarchical structure consisting of five main levels in session data, from the feature value level till to the domain level. Out of these five, the three levels in the middle are the core in most SBRSs. The logical relations between different components of this structure are as follows: items are the core concept and element in this structure because they are not only the basic data unit in the source transaction dataset but are also the focus of the final goal of the systems (e.g., to recommend the next item or items). Each item may have multiple features, like category and price. Each feature usually contains multiple values, and some values may be more frequent, while others may not. Furthermore, a collection of items typically forms a session (e.g., a transaction consisting of multiple items). A dataset collected from one domain (e.g., food) in a certain period often contains multiple sessions.

Generally, the challenges in such a complex structure are varied: (a) the heterogeneity issue within each level (e.g., usually an item has both categorical and numerical features, which are heterogeneous and cannot be modelled in the same way); (b) the dependency or coupling issue within each level (e.g., some items are dependent upon each other; for instance, bread and milk are usually bought together); (c) other complexities within each level (e.g., imbalance issue of items in a session data); and (d) the interactions between different levels (e.g., items belonging to the same category are more likely to occur together in transactions). To be more specific, a list of critical challenges is presented for each level on the right side of Figure 1.1.

1.2.2 A Summary of Research Progress and Gaps

Certain existing works have focused on SBRSs, such as pattern/rule-based recommendations and next-item recommendations; these works have achieved

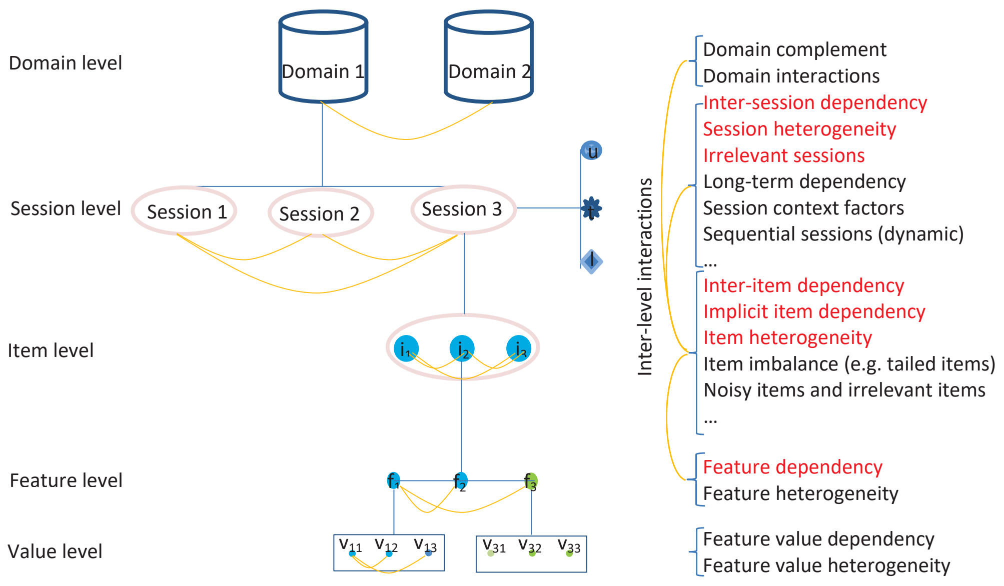


Figure 1.1: Key challenges in session-based recommender systems

some progress. However, from my observations, most of these works fell into the same branch and focused on one typical issue in SBRSs: only on inter-item dependency modelling at the item level for the recommendations. In another words, these works are limited to solving only one critical challenge of the more than 10 listed in Figure 1.1. To be specific, addressing the challenge of inter-item dependency on item-level, they not only ignore other issues, including implicit dependency (Wang & Cao 2017) and item heterogeneity on the same level, but also ignore challenges from other levels (e.g., feature dependency on feature level and inter-session dependency on session level), and the interactions between different levels. Here, item heterogeneity particularly refers to some items being relevant to subsequent items, while others are not. For example, in a session {*bread, orange, apple, milk*}, *bread* is relevant to the subsequent item *milk* while *orange, apple* are not. Therefore, *bread* and *orange (apple)* are two heterogeneous items to *milk*. Such ignorance in the modelling leads to information loss and thus degrades recommendation performance in most cases. For instance, the missing implicit relations between items in rule-based RSs leads to duplicated and unreliable recommendations (Wang & Cao 2017). Furthermore, either feature-level or session-level dependency can affect item occurrence significantly (Weng & Liu 2004). A typical example is that items from the same category, sharing similar features, often have the same function, and thus they tend not to occur in one transaction event to avoid duplicated items during shopping. I detail the gaps of related works in the following paragraphs.

The existing works focusing on item-level dependency modelling for SBRSs can be divided into two branches: the pattern/rule-based type and modeling-based type. Both branches focus mainly on the inter-item dependency modelling and thus recommending next items given the session context, (e.g., those items chosen in hand). To be specific, pattern/rule-based recommender systems discover association rules, correlation rules or sequential patterns (Yap, Li & Philip 2012) from transaction data based on item co-occurrence to guide recommendations (Wang & Cao 2017, Adda, Missaoui, Valtchev & Djeraba 2005). Such approaches, on the one hand, treat all items equally and thus ignore the item heterogeneity, on other hand, focus only on frequent and explicitly co-occurring items while missing implicit relations over items. Modelling-based approaches build models to learn the complex inter-item dependencies and then employ these models for recommendations without explicit rules or patterns (Hidasi, Karatzoglou, Baltrunas & Tikk 2015). The Markov chain(MC) (Rendle et al. 2010) offers a straightforward way to model transitions between items in sequential data. However, it captures only first-order transitions while ignoring higher-order ones. A model based on matrix factorization (MF) factorizes the matrix of transition probability into latent factors (Chou, Yang, Jang & Lin 2016), but it commonly suffers from sparsity issues. (Hidasi et al. 2015) have applied deep recurrent neural networks (RNN) to model sequential data but the high computational cost caused by the complex structures prevents its application to large data. Moreover, MC, MF and RNN were originally designed for time-series data with a rigid natural order, hence they do not fit unordered session data (e.g., it makes no difference whether milk or bread is put into the cart first (Wang, Hu, Cao, Huang, Lian & Liu 2018)). More importantly, all these models do not take into account item heterogeneity.

The significance of incorporating item features to improve recommendation performance has been substantially demonstrated by other recommender systems like content-based filtering (Han & Karypis 2005, Karypis 2001) and collaborative-filtering (Menon, Chitrapura, Garg, Agarwal & Kota 2011). However, only quite few works in SBRSs involve item features and model the feature level dependence and its interactions with item-level dependence. One typical method is to calculate the similarity between items in terms of their features and then connect new items or rarely occurring items to popular ones to drive new rules or patterns, guiding cold start item recommendations in pattern/rule-based RSs (Weng & Liu 2004). Such a method is too adhoc and relies on a strong assumption that similar items with respect to features have the alternative functions and thus appear similar. As a result, the recommendation results may not be so solid where the assumption does not hold.

Studies have been done to model the session-level dependency. The inter-transaction association rule (Tung, Lu, Han & Feng 2003, Berberidis, Angelis & Vlahavas 2004) or pattern mining (Lee & Wang 2007) is a basic approach to capture inter-transaction dependency. Such frequent co-occurrence-based methods are limited to frequent items only and ignore infrequent ones or the implicit relations. In addition, they have not been well applied to recommender systems to the best of my knowledge. Another typical approach is next-basket recommendations, in which the RSs try to predict what would be possibly bought in the next transaction via modelling the dependency between different transactions (Rendle et al. 2010, Yu, Liu, Wu, Wang & Tan 2016). The success of such works has demonstrated the great impact of session-level dependency on item choice. However, these works are limited to next-basket recommendation task, different from my specific focus (next-item recommendations). From my observations, nearly all existing works on next-item recommendations are limited to intra-transaction (item-level) dependency modelling.

1.3 Research Issues

In this thesis, I aim to build a systematic and original framework for session-based recommendation task based on my own understanding of the comprehensive challenges in this area and my observation of the current research progress in the community. Due to limitations of space for this work, I build a hierarchical framework consisting of the three core levels rather than all five levels presented in Figure 1.1 for SBRSs. Specifically, I would like to build a framework which contains the feature level, item level and session level and will address one or two critical challenges in each level which have not yet been well-studied in existing works. Particularly, I focus on the challenges in red in Figure 1.1. Next, I demonstrate them one by one in the following four subsections.

1.3.1 Implicit Rule-Based Recommender Systems

I aim to address implicit inter-item dependency at the item level in this section. Specifically, the duplicated and unreliable recommendation issues in existing pattern/rule-based recommender systems are fixed by inferring implicit rules with much more implicit relations over items embedded.

Pattern/rule-based recommender systems are an intuitive and basic solution to session-based recommendations and have been widely used due to their simplicity (Lin, Alvarez & Ruiz 2002). However, one of the most significant drawbacks of such methods is that they are based only on the explicit co-occurrence-based relations, such as association or correlation, while ignoring the implicit relations between items. This leads to information loss and thus results in unreliable recommendations. For instance, an existing rule-based RS usually recommends coke to a user if he/she just bought pizza, as these two are frequently bought together. However, if the user has just bought a sprite, the coke should not be recommended, as it likely would be a duplicated recommendation. In this case, the implicit relation between coke and sprite plays an important role to form a more reliable recommendation. This example actually demonstrates the significance of incorporating implicit relations into rule-based RSs. To this end, I would like to explore how to effectively and efficiently discover implicit rules that are built on implicit relations – more importantly, how to employ these implicit rules to help increase recommendation reliability of rule-based recommender systems? Both issues are comprehensively explored in Chapter 4.

1.3.2 Attention-Based Transaction Embedding for Heterogeneous Items

In this part, I focus on another critical challenge at the item level: the item heterogeneity issue in the inter-item dependency modelling, as shown in Figure 1.1. In practice, different items in a session usually have different relevance scales and thus contribute differently to the occurrence of the successive items; namely, the items are heterogeneous. Therefore, to identify and emphasize those more relevant items in a session context is quite critical to build an informative and discriminative session context. This will lead to more reliable and accurate recommendations of the following items. However, such a challenge is ignored by most existing session-based recommender systems. Motivated by this observation, I particularly focus on how to learn the relevance scales of different items and how to integrate them to form a unified session context for the later recommendations in Chapter 5.

1.3.3 Integrating Item Features for Cold Start Item Recommendations

In this section, I switch to the feature level from the item level. Particularly, I focus on feature level dependency and the interactions with item level dependency modelling to address the cold start issue in session-based recommender systems. Cold start item is a common issue faced by recommender system community. Although many existing works focus on this issue in other recommender systems including content-based and collaborative filtering ones, it is not yet well studied in session-based recommender systems. To this end, in Chapter 6, I explore how to incorporate item features when modelling item dependency effectively to jointly model item level dependency, feature level dependency and the interactions between them in an session-based recommender system.

1.3.4 Jointly Modelling Intra- and Inter-Session Dependency for Next-item Recommendations

In this section, I focus on the session level dependency (i.e., inter-session dependency) and the interactions with item level dependency modelling to improve the recommendation performance of session-based recommender systems. A session is a basic data unit in session-based recommender systems. However, most existing session-based RSs recommend next items by modeling only the intra-session dependency at the item level within a single session while ignoring the effect from other sessions. In Chapter 7, I explore how to incorporate inter-session dependency into session-based RSs and how to jointly model both intra- and inter-session dependency for next-item recommendations.

1.4 Research Contributions

The contributions of this thesis come from multiple folds and are summarized as below:

- A comprehensive and systemic framework (Figure 1.1) is built to reveal the complexities in session data and to reveal the challenges in session-based recommendations. It enables the in-depth understanding of the working mechanism behind session-based recommender systems from different levels and perspectives. This provides a good reference to the recommender system community (Chapters 1 and 2).

- An implicit rule discovery framework combing a corresponding implementation algorithm is proposed. The algorithm can effectively and efficiently discover implicit rules on items based on the implicit relation analysis, which goes far beyond the traditional rule mining work that focuses only on the explicit rule mining (Chapter 4).
- A basic session-based recommender system: an implicit rule-based receommneder system is built on the mined implicit rules; empirical analysis results show that it can provide more reliable recommendations compared to existing rule-based recommender systems which are based only on explicit rules (Chapter 4).
- I argue that items in a session are heterogeneous in terms of the contributions to the next-item occurrence, namely different items contribute differently to the next-item recommendations. Accordingly, a framework is proposed to address the item heterogeneity issues in session-based recommender systems (Chapter 5).
- An algorithm based on shallow neural network and attention mechanism is proposed to implement the proposed framework. The experimental results show its superiority over state-of-the-art session-based recommender systems (Chapter 5).
- A framework to address the cold-start issue in session-based recommender systems is proposed, in which, item IDs and their features are mapped into a latent vectors simultaneously and are jointly learned. In such a case, the feature level dependency and the interactions between item features and item occurrence are embedded into their latent vectors for the following recommendation tasks (Chapter 6).
- A corresponding implementation algorithm is designed for cold-start item recommendations. Empirical studies on real-word transaction

data sets demonstrate the effectiveness of the designed algorithm (Chapter 6).

- I argue that in addition to the item level intra-session dependency, the session level inter-session dependency also contributes greatly to the choices on items. In such a case, a framework is proposed for next-item recommendations by jointly modelling intra- and inter-session dependencies (Chapter 7).
- An algorithm is designed to implement the proposed framework and the corresponding empirical studies demonstrate the significance of considering inter-session relations and the superiority of the designed algorithm (Chapter 7).

To summarize, a unified hierarchical framework consisting three levels is proposed for session-based recommender systems. On each level, one or two critical challenges are addressed by a corresponding original and well-designed framework together with a corresponding algorithm. Particularly, the addressed issues include implicit relation modelling and the item heterogeneity issues at the item level, feature level dependency modelling for the cold-start issue, and the inter-session dependency modelling issue at the session level.

1.5 Thesis Structure

This thesis is structured as below:

Chapter 2 first formalizes the session-based recommendation problem with a series of relevant definitions and notations, followed by a brief introduction of some preliminaries which are used in this thesis. Finally, the commonly used evaluation metrics and baseline methods are introduced.

Chapter 3 provides a comprehensive survey of session-based recommender systems. I first give an introduction to demonstrate the necessity of session-based recommender systems and its difference from other kinds of recommender systems like collaborative filtering. Then I compare different scenario settings of session-based recommender systems, followed by a categorization of session-based recommender system approaches. Following the categorization, I review all categories one by one, from the pattern/rule-based methods and sequential pattern-based methods to Markov chain-based ones, factorization machine-based approaches, shallow neural network-based methods, and deep neural network-based ones.

Chapter 4 emphasizes the significance of implicit relations over items in pattern/rule-based approaches for session-based RSs. An implicit rule discovery framework is built on the basis of implicit relation analysis over items in a session or transaction. Accordingly, an effective and efficient implicit rule mining algorithm is designed and implemented. An implicit rule-based recommender system for session-based recommendations is built on the mined implicit rules. Empirical studies of the implicit-rule based recommender system on two real-world transaction datasets show the significance of considering implicit relations over items and the superiority of my implicit rule-based recommender systems over other rule-based recommender systems.

Chapter 5 identifies the item heterogeneity issue in session-based recommender systems (SBRSs). The significance of taking item heterogeneity into account is illustrated and a framework to solve this issue is proposed. A corresponding algorithm based on shallow neural network is designed to implement it. Experimental analysis on real-world transaction datasets demonstrates my work's advantage over state-of-art works in this area.

Chapter 6 focuses on another significant issue in session-based recommender systems: the cold-start issue. A framework combing a corresponding algorithm to embed both items and their corresponding features into latent representations simultaneously for the following session recommendation tasks is designed. Experimental analysis is conducted to show the algorithm's merit.

Chapter 7 targets another significant issue in session-based recommendations: inter-session modelling. A framework together with a corresponding algorithm is proposed to consider both intra-session dependency and inter-session dependency simultaneously for one of session-based recommendation tasks: next-item recommendation. Empirical analysis on real-world datasets is done to show the algorithms advantages and the significance of considering inter-session dependency.

Chapter 8 concludes this thesis by summarizing the aforementioned works. Meanwhile, some possible future directions related to this thesis are given.

A thesis profile to show the overview of this thesis is given in Figure 1.2 below.

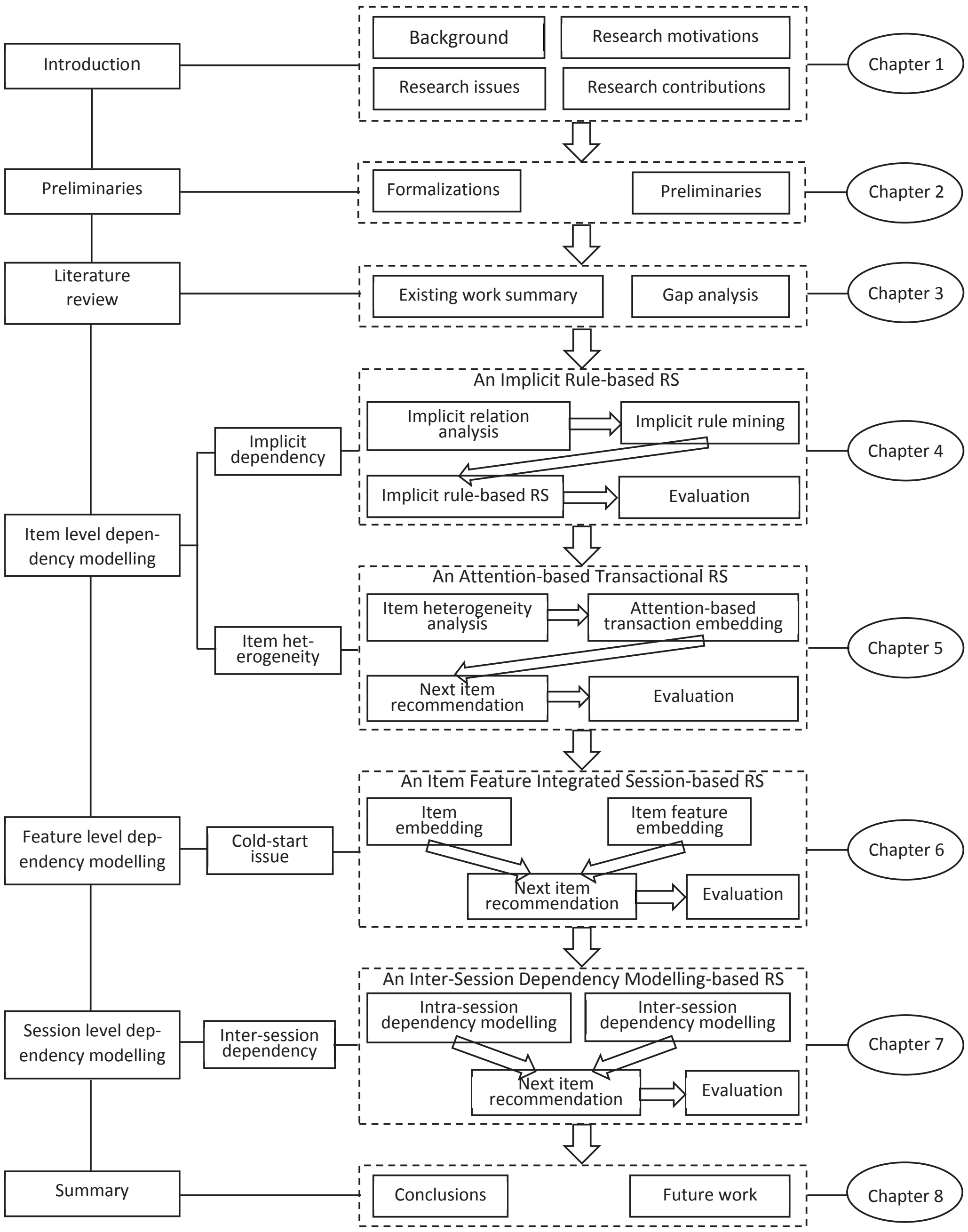


Figure 1.2: The profile of work in this thesis

Chapter 2

Preliminaries and Foundations

2.1 Formalisations and Definitions

In this section, I firstly define the notion of a session and session-based recommender systems, which are the core concepts in this thesis. Then, I formalise the session-based recommender system task, and accordingly, give some vital relevant definitions and notations that are commonly used in this thesis.

Session: The Oxford Dictionary defines this as ‘ A meeting of an official body, especially a legislature, council, or court of law, to conduct its business’. In this thesis, I have expanded the concept of session to garner a more general meaning.

Definition 2.1 (Session)**.** *A session can be a set of objects that are collected or consumed during one event or over a certain period of time, or a session can be a collection of actions, or events that happen over a period of time.*

For instance, both a set of items purchased in one transaction and a list a songs that are listened to by a user in one hour can be interpreted as being ’sessions’. In addition, a user’s successive clicks on different web pages in one hour can also be regarded as a ’session’. In this thesis, as I mainly work on shopping basket-based transaction data, a ’session’ refers particularly to a transaction consisting of all the items purchased in it.

Session-based Recommender Systems (SBRSs): As the name implies, an SBRS is a recommender system that is built on the basis of sessions and that takes a session as the basic organisational unit of the data. It is the session which actually differentiates an SBRS from other representative recommender system models such as content-based ones and collaborative filtering ones. They usually split a session structure into a smaller granularity, for example, user-item (song, or movie) interaction pairs, which actually destroys the intrinsic structure of a session and thus leads to information loss. Accordingly, I have formally defined the Session-based Recommender Systems.

Definition 2.2 (Session-based Recommender Systems (SBRSs))**.** *Recommender systems that built on session data; usually given a part of known* *session information, such as a part of a transaction consisting of multiple items, a SBRS then tries to predict those unknowns such as the left part of that transaction, as the recommendation target.*

Specifically, according to whether one recommends a part of the current session or the whole next session, SBRSs are usually divided into two branches: 'next-item(s) recommendations' and 'next-session recommendations' (which are also called 'next-basket recommendations' in some literatures'), which will be formally defined later. In this thesis, I only focus on the first branch.

In following the general definitions of these two key concepts in this thesis, I next specify and formalise the research issues in this work, and I then give the specific definitions of some relevant concepts which are used in the following chapters. It should be noted, however, that session-based recommender systems are widely applied in various scenarios including shopping basket-based transaction recommendations, next-song recommendations, and next-POI recommendations, of which, the first of these is the most representative and has attracted the most critical attention. To narrow the research scope and achieve a more in-depth research, this thesis only works on shopping basket-based transaction data for next-item recommendations. Other scenarios can be transferred into this framework to some degree by manipulating the source data.

Generally, in recommender systems, especially in shopping basket-based transaction data, a user u and item i are two basic concepts, and all the users and items constitute, the user set $U = \{u_1, u_2 ... u_{|U|}\}$ and the item set $I = \{i_1, i_2 ... i_{|I|}\}$ respectively. The interactions between users and items, such as 'click' or 'buy', form the other impotent component in recommender systems, and they are usually presented in the form of a session. For instance, the clicks of one user on all items in one online shopping create a 'click session'. More generally, the items purchased by one user during a certain shopping event form a transaction session, which is simplified as a 'transaction' in this thesis. Accordingly, a transaction of user $u (u \in U)$ is formalised as

$t_u = \{i_1, i_2 ... i_{tk} ... i_{|t_u|}\}(i_{tk} \in I, t_u \subset I)$, and all transactions in a transaction dataset constitute the transaction set $T = \{t_1, t_2 ... t_{|T|}\}$. In the next-item recommendations, a session s often refers particularly to a transaction $t(t \in T)$. It should be noted that a user u may have multiple sessions to form a session set $S_u = \{s_1, s_2 ... s_{uk} ... s_{|S_u|}\}$ since a user can often have multiple transaction events over a certain period of time, such as a week or a month. Here a session $s_{uk} = \{i_1, i_2 ... i_{sk} ... i_{|s_{uk}|}\}(i_{sk} \in I, s_{uk} \subset I)$ consists of multiple items purchased in user u's k^{th} transaction.

Generally, a session-based recommender system tries to make predictions on unknown session information as the recommendation target $\boldsymbol{t}$ such as next item(s) or the next session by taking the prior session information as the context and condition. The prior session information is then unified as a session context $\boldsymbol{c}$ in this work. In this thesis, I mainly focus on making recommendations on the next item(s) on the basis of a certain session context.

According to whether the session context only comes from one session or whether it crosses multiple sessions, the session context is usually divided into the intra-session context $\boldsymbol{c}^{Ia}$ and the inter-session context $\boldsymbol{c}^{Ie}$. They are defined respectively as below.

Definition 2.3 (Intra-Session Context)**.** *In taking the session s_n as the current session for recommendations (recommend unknown items in s_n), the intra-session context $\mathbf{c}^{Ia}$ is the set of items that are already known in s_n: namely, $\mathbf{c}^{Ia} = \{i | i \in s_n, i \neq i_t\}$.*

Definition 2.4 (Inter-Session Context)**.** *In taking session the s_n as the current session for recommendations, the inter-session context $\mathbf{c}^{Ie}$ is the set of recent sessions that have happened before s_n: namely, $\mathbf{c}^{Ie} = \{s_{n-1}, s_{n-2} ... s_{|c^{Ie}|}\}$.*

I wanted to differentiate between two kinds of session contexts because they actually deliver different kinds of relations for the recommendation task: intra-session contexts embed intra-session dependency, while inter-session contexts convey inter-session dependency for item recommendations.

Now I can specify and and formulate a session-based recommendation task as below:

Definition 2.5 (Session-based Recommendation Task). *Given a session context* $\mathbf{c}$*, to learn a function* f *which maps* $\mathbf{c}$ *to the recommendation target* $\mathbf{t}$*:* $\mathbf{t} \Leftarrow f(\mathbf{c})$*. Note that, the session context is the main but not the only information for session-based recommendations; sometimes, other information such as items' and users' attribute information can also be incorporated into session-based recommendations as a complement.*

According to the task difference, session-based recommender systems are generally categorized into two main branches: next-item(s) recommender systems and next-session (basket) recommender systems. I have defined these separately.

Definition 2.6 (Next-Item Recommendations). *Given a session context* $\mathbf{c}$ *from the current session* s_n*, a next-item recommendation tries to predict the next item* i_t *in* s_n*. It should be noted that most next-item(s) recommendations only take the current session (Hidasi et al. 2015, Chou et al. 2016) to form an intra-session context; minority ones incorporate recent previous sessions into the context (Quadrana, Karatzoglou, Hidasi & Cremonesi 2017). In such a case, the context* $\mathbf{c}$ *comes from multiple sessions, and it is actually a mixture of intra- and inter-session contexts.*

Definition 2.7 (Next-Session (Basket) Recommendations). *Given a context* $\mathbf{c}$*, a next-session (basket) recommendation tries to predict the items possibly occurring in the next session* s_{n+1}*. In this case, the context* $\mathbf{c}$ *is a collection of recent sessions* $\{s_n, s_{n-1}, ...\}$ *prior to the next one, namely, the inter-session context.*

As I mentioned above, this thesis only focuses on session-based next-item(s) recommendation tasks.

To help the readers to better understand the contents in this thesis, especially the notations in each chapter, I list the commonly used notations below.

Table 2.1: Notation list

Notation	Meaning
f	a feature value of a feature (e.g., food)
F	a feature of an item (e.g., category of items)
i	an item (e.g., bread)
I	an itemset consisting of multiple items
t	a transaction consisting of all items purchased in one transaction event
T	a transaction set
s	a session consisting of multiple items consumed in a duration
S	a session set
q	a sequence consisting of a set of ordered itemsets
Q	a sequence set
u	an user
U	an user set
$\mathbf{c}$	a context consisting of one or multiple sessions
p	a pattern
P	a pattern set
X	a antecedent (i.e., an item or itemset)
Y	a consequent (i.e., an item or itemset)
r	a rule
$\mathbf{R}$	a rule set
R	a recommendation list
h	a hidden state
$\mathbf{h}$	a latent vector
$\mathbf{e}$	a latent vector
$\mathbf{E}$	a latent vector
$\mathbf{W}$	a weight matrix in a neural network

2.2 Preliminaries

In this thesis, various data mining and machine learning techniques are employed to address the challenges in session-based recommender systems and, therefore, to build my own mansion for session-based RSs. To help readers to have a better understanding of the following chapters, I introduce some preliminaries of data mining and machine learning models in this section. Particularly, I present association rule mining methods, embedding models and attention mechanism.

2.2.1 The Association Rule Mining

An association rule is an implication in the form of $X \Rightarrow Y, X \cap Y \neq \emptyset$, where X and Y are itemsets. X is called the antecedent, while Y is called consequent; the rule means that X implies Y (Zhao & Bhowmick 2003). *Support* and *confidence* are two important basic measures for association rules. Accordingly, two thresholds 'minimal support' and 'minimal confidence' are two commonly used constraints in association rule mining to filter out those uninteresting rules from a large number of candidates.

The support of an association rule $X \Rightarrow Y$ is defined as the fraction of transactions containing X and Y to all the transactions in the database. The confidence is defined as the percentage of the transactions that contain both X and Y with respect to the transactions that contain just X, and it can be calculated from the support values.

$$Support(X \Rightarrow Y) = \frac{|\{t|X \in t \cap Y \in t\}|}{|T|} \tag{2.1}$$

$$Confidence(X \Rightarrow Y) = \frac{Support(X \cap Y)}{Support(X)} \tag{2.2}$$

In the above two equations, t is a transaction consisting of all items purchased in the corresponding transaction event, and T is the collection of all transactions in a dataset.

‘Association rule mining’ aims to discover association rules which satisfy the predefined constraints of ‘minimum support’ and ‘minimum confidence’ from a given dataset. The process is usually divided into two stages (Agrawal, Srikant et al. 1994). The first stage is to find frequent itemsets whose support exceeds the predefined threshold, while the second stage is to generate rules from the resultant frequent itemsets by using minimum confidence as the constraint.

2.2.2 Embedding Models

Embedding models were originally used in natural language process domain so as to embed words to latent vectors which have richer semantic meaning (Mnih & Kavukcuoglu 2013). Later, they were applied to other domains including node embedding, network embedding, and item embedding. Here I introduce two representative word-embedding models: Skip-gram and Continuous Bag-of-Words (CBOW).

Skip-gram predicts surrounding words given the current word, while the CBOW model predicts the current word based on the surrounding context. To be specific, Skip-gram builds a log-linear classifier by ussing each current word as its input to predict words within a certain range before and after the current word; CBOW takes the identify number of both future and past words as the input to the log-linear classifier so as to correctly classify the current (middle) word (Mikolov, Chen, Corrado & Dean 2013). Each of the input words in both models should be first projected onto a continuous valued vector. The architectures of both models are shown in Figure 2.1.

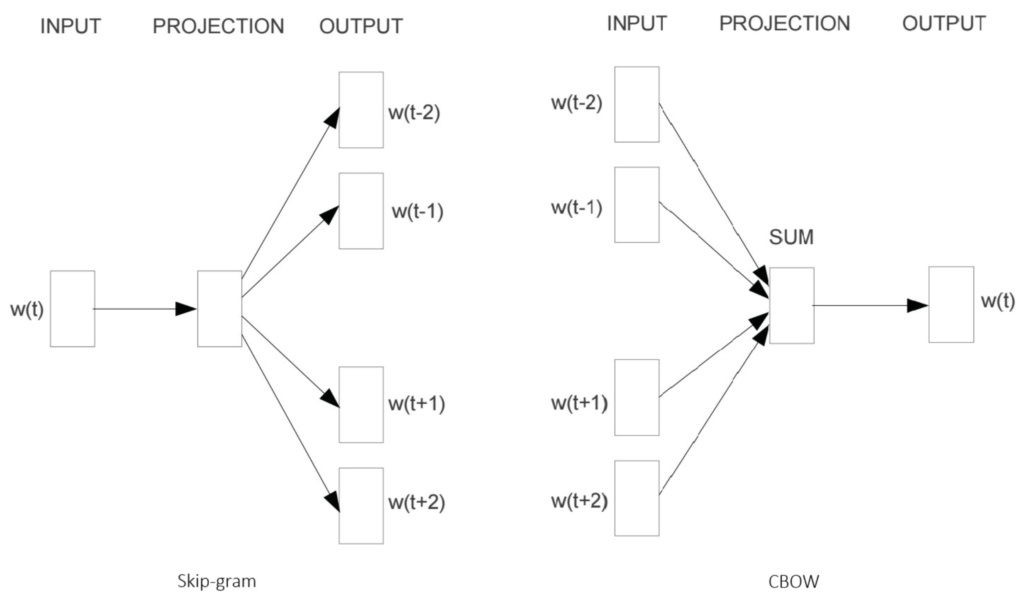


Figure 2.1: The architectures of Skip-gram and CBOW models [1]

The training objective of the Skip-gram model is to learn word representations for predicting the surrounding words in a sentence or a document. More formally, given a sequence of training words $w_1, w_2, w_3, ..., w_L$, the objective of the model is to maximise the average log probability (Mikolov, Sutskever, Chen, Corrado & Dean 2013)

$$\frac{1}{L}\sum_{l=1}^{L}\sum_{-c\leq j\leq c,\ j\neq 0} logp(w_{l+j}|w_l) \tag{2.3}$$

where c is the size of the training context (which can be a function of the centre word w_t). The basic Skip-gram formulation defines $p(w_{l+j}|w_l)$ by using a softmax function as below:

$$p(w_O|w_I) = \frac{exp({v'_{w_O}}^{\mathsf{T}} v_{w_I})}{\sum_{w=1}^{W} exp({v'_w}^{\mathsf{T}} v_{w_I})} \tag{2.4}$$

where w_i and w_o are the input words and output words, respectively. v_w and v'_w are the input and output vector representations of the word w, and W is the total number of words.

[1]from (Mikolov, Chen, Corrado & Dean 2013)

In contrast to Skip-gram, CBOW reverses the training process. Accordingly, its training objective is to learn word representations for predicting the current word by using the surrounding words as the input. Therefore, the objective is changed to maximise the following average log probability:

$$\frac{1}{L}\sum_{l=1}^{L}\sum_{-c\leq j\leq c,\ j\neq 0} logp(w_l|\{w_{l+j}\}) \tag{2.5}$$

2.2.3 Attention Mechanism

An attention mechanism is designed to model one of the basic work mechanisms of a human brain, especially our visual mechanism. For instance, when we look at a picture, we may see the whole picture at the beginning, however, when we would like to have an in-depth observation, our eyes must focus on a small region within the whole picture. Accordingly, our brains need to pay much more attention to this small region while less to the left. This indicates that our brains do not treat all regions in a picture equally; instead, they weight them with respect to their importance. An attention mechanism is designed to model this process and thus to capture more discriminating information from the input data so as to better cater for the subsequent tasks. Initially, they were widely applied in computer vision (Lu, Xiong, Parikh & Socher 2017, Pedersoli, Lucas, Schmid & Verbeek 2016) and natural language processing (NLP) (Luong, Pham & Manning 2015, Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser & Polosukhin 2017) areas and have achieved great success. Later, they were more broadly introduced into other domains including sequence modelling and recommender systems.

In this thesis, I use a representative model (Bahdanau, Cho & Bengio 2014) which was built for machine translation in NLP as an example to introduce attention mechanism. Suppose $\mathbf{x} = \{x_1, ..., x_{T_x}\}$ denotes the source language while $\mathbf{y} = \{y_1, ..., y_{T_y}\}$ denotes the target language in a translation scenario. Usually, a machine translator predicts the next word in the target language by maximising the conditional probability based on the current word and the input from the source language.

$$p(y_i|y_1, ..., y_{i-1}, \mathbf{x}) = g(y_{i-1}, h_i, c_i) \tag{2.6}$$

where h_i is a hidden state for time point i:

$$h_i = f(h_{i-1}, y_{i-1}, c_i) \tag{2.7}$$

where c_i is a context vector which has been built on the input words from source language for a target word y_i. Particularly, c_i is based on a sequence of annotations $(\mathbf{e}_1, ..., \mathbf{e}_{T_x})$ onto which the input words are mapped.

c_i is then computed as a weighted sum of these annotations:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij}\mathbf{e}_j \tag{2.8}$$

The weight α_{ij} of e_j is computed by the following formula:

$$\alpha_{ij} = \frac{exp(a_{ij})}{\sum_{k=1}^{T_x} exp(a_{ik})}, \tag{2.9}$$

where

$$a_{ij} = b(h_{i-1}, \mathbf{e}_j) \tag{2.10}$$

where b is parametrized as the weights to connect two layers in a feed-forward neural network and is jointly trained.

The attention model here learns the weights of the input vectors so as to indicate their relevance scales w.r.t the current word.

2.3 Evaluation Metrics

For the implicit rule-based recommender system in Chapter 4, I focus on the improvement of recommendation reliability, therefore I employ the reliability defined in section 4.4 as the metrics to evaluate the proposed methods. In addition, I evaluate the efficiency of the proposed algorithm, as most rule mining works do.

For the session-based recommender systems that I built in Chapters 5, 6, and 7, I focus on the improvement of recommendation accuracy and novelty, just as other SBRSs do. Therefore, I evaluate the performance of these three recommender systems in terms of shared accuracy and novelty metrics, which will be introduced below.

2.3.1 Accuracy Metrics

The most common way to assess the performance of a session-based recommender system is to measure whether the truly relevant items are ranked at the top in the resultant recommendation list. Therefore, information retrieval metrics are often used to evaluate the ranking performance of a session-based RS. In this thesis, I denote $rel(k) = 1$ if the item at position k is relevant, and $rel(k) = 0$ if it is not. I have introduced and defined the following two measures to evaluate the ranking performance.

- ***REC@K***: This measures the recall of the top-K ranked items in the recommendation list over all the N testing instances (Yuan, Cong, Ma, Sun & Thalmann 2013).

$$REC@K = \frac{\sum_{k=1}^{K} rel(k)}{N} \tag{2.11}$$

- ***MRR***: This measures the mean reciprocal rank of the predictive position of the true target item in the top-K recommendation list on all the N testing instances (Chou et al. 2016).

$$REC@K = \frac{1}{N}\sum_{k=1}^{K} \frac{1}{k} rel(k) \tag{2.12}$$

It should be recalled that in the real world most customers are only interested in the items recommended on the first few web pages, thus I choose $K \in \{10, 50\}$ in this thesis. In practice, it is a significant challenge to find exactly the one true item from thousands of candidates.

2.3.2 Novelty Metrics

Since more and more users not only care about the recommendation accuracy but also the novelty of the recommended items, in practice, novel items can bring users unexpected exciting experiences and, thus, can leverage their satisfaction. In this thesis, novelty metrics were designed to evaluate the recommendation performance as well.

• ***Global novelty***: Intuitively, the novelty of an item from the global perspective can be defined as the opposite of the item's popularity with respect to the whole population. The item is novel if it only occurs over a few transactions, in another words, the item is far in the long tail of the popularity distribution (Park & Tuzhilin 2008). Having drawn inspiration from the inverse user frequency (IUF) which has been proposed in (Breese, Heckerman & Kadie 1998) and (Francesco Ricci 2015), I have defined the concept of inverse transaction frequency (ITF) as $ITF = -log_2|T_i|/|T|$, where $T_i = \{t \in T | i \in t\}$ denotes the set of transactions containing item i. Similar to the novelty metric MIUF defined in (Park & Tuzhilin 2008), I here use the average of the ITF (MITF) of the recommended items to measure the global novelty of a recommendation:

$$MITF = -\frac{1}{|R|}\sum_{i\in R} log_2\frac{|T_i|}{|T|} \tag{2.13}$$

where R is the set of recommended items in one recommendation and T is the whole set of transactions. For all the N recommendations on a certain testing dataset, the global recommendation novelty is defined as the mean of $MITF$ for each recommendation:

$$M^2ITF = \frac{1}{N}\sum MITF \tag{2.14}$$

• ***Local novelty***: Local novelty refers to the difference of recommended items with respect to the previous experience of the users. Generally, if the recommended items are more different from the already-bought items, the more novel these items are. In my model, the already-bought items correspond to the context **c** used for recommendation R. Given a recommendation list R, the more items there are in R having been seen in the context **c**, the less novel the recommendation is. Based on this observation, the context-aware novelty (CAN) of recommendation R can be defined as:

$$CAN = 1 - \frac{|R \cap \mathbf{c}|}{|R|} \tag{2.15}$$

Similarly, for all the N recommendations on a certain testing set, the local recommendation novelty is defined as the mean of CAN ($MCAN$) of all recommendations:

$$MCAN = \frac{1}{N} \sum CAN \tag{2.16}$$

2.4 Baseline Methods for Experiments

Several existing representative methods for session-based recommendations were used as the baselines to compare with my proposed methods in the experiment part in most of the following chapters. I list them in this section as below.

- ***PBRS***: This is a typical pattern-based recommender system which uses mined frequent patterns to guide the recommendations (Li, Wang, Zhang, Zhang & Chang 2008).
- ***FPMC***: This is a model that combines matrix factorisation and first-order Markov chains for next-basket recommendations. The model factorizes the personalized transition matrix between items with a pairwise interaction model (Rendle et al. 2010).
- ***PRME***: A personalized ranking metric embedding method (PRME) to model personalized check-in sequences in a Markov chain framework. The learned PRME is used to recommend the next POI of users (Feng, Li, Zeng, Cong, Chee & Yuan 2015).

- ***GRU4Rec***: An RNN-based approach for session-based recommendations by modelling the session through using a deep RNN which consists of GRU units (Hidasi et al. 2015).

Chapter 3

Literature Review

3.1 Introduction

Recommender systems (RSs) have evolved into a fundamental tool for helping users to make informed decisions and choices, especially in the big data era in which customers have to make choices from a large number of products and services. A good deal of recommender system techniques and models have been proposed and most of them have achieved good performance. Out of these, the content-based recommender systems (Aggarwal 2016, Pazzani & Billsus 2007) and the collaborative filtering recommender systems (Schafer, Frankowski, Herlocker & Sen 2007, Ekstrand, Riedl, Konstan et al. 2011) are two representative recommender systems. Their effectiveness has been demonstrated in both the research and industry community.

However, these aforementioned conventional recommender systems still have had some drawbacks. One critical disadvantage is that they only focus on the long-term static preference of users, while they ignore the short-term transaction behaviour patterns and users' preference shifts over time. In this case, the users' intent at one point may be easily submerged by his or her historical behaviours, and this leads to unreliable recommendations. This is usually because these recommender systems have mixed together all the transactions of a user and have, thus, broken down the intrinsic transaction structure. For example, in the matrix factorization (Koren, Bell & Volinsky 2009, He, Zhang, Kan & Chua 2016), which is a representative model in collaborative filtering recommender systems, the items a user buys in all transactions are put into one row of a matrix as shown in the top half of Figure 3.1. In another case, the users' IDs are not always available due to privacy issues, and the conventional recommender systems which require user information are not applicable. To this end, the recommendations can be only based on the transaction events.

All these issues have triggered the necessity to take transaction structure into account when developing recommender systems. In other words, it is necessary to learn users' transaction behaviour patterns and preference shifts according to the relationship between one transaction and another. To

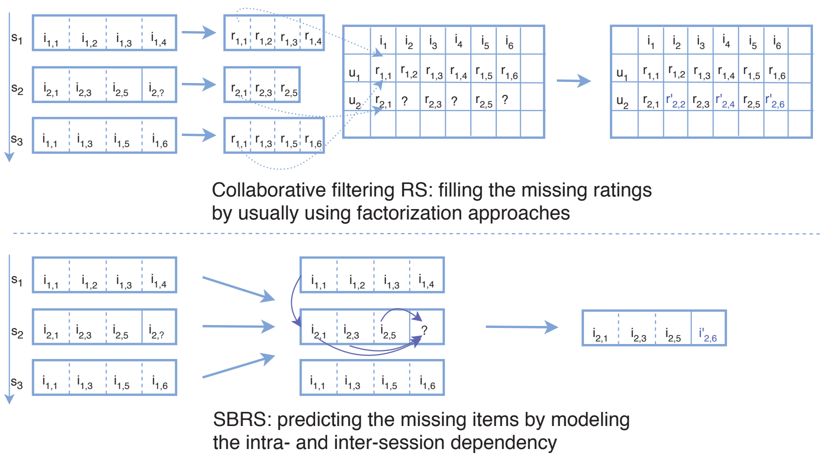

Collaborative filtering RS: filling the missing ratings by usually using factorization approaches
SBRS: predicting the missing items by modeling the intra- and inter-session dependency

Figure 3.1: A comparison between collaborative filtering RS and SBRS

this end, session-based recommender systems have been proposed in recent years. In this research, a session can be regarded as a transaction with multiple purchased items in one shopping event. Different from content-based and collaborative-filtering recommender systems, session-based recommender systems comprehensively consider the information embedded from one session to another, and they take a session as the basic unit for recommendations as shown in the bottom half of Figure 3.1. Therefore, session-based recommender systems can prevent the information loss caused by breaking down the session structures in other approaches to the maximum extent.

Besides transaction domain, session-based recommendations are widely applied in other domains such as the next web page recommendations, the next POI recommendations, the tourism recommendations, the next song recommendations, and the next movie recommendations. To cover these various domains, the concept of 'session' is not limited to a transaction in this work; instead, a session refers to a collection of consumed or visited elements at one time or over a certain period of time. For instance, the web pages visited by a user during one Internet surf can be gathered as a session, and the songs listened by a user over an hour or a day can form a session as well.

Although session-based recommendations have been conducted in different domains, the sense of 'session' may not be identical for these domains. In some domains, the sense of a session is stronger as one session is performed to combine a coherent collection of items that work together to cater for a certain purpose. For example, bread, egg and milk are usually bought together to form a session for the purpose of breakfast. In other domains, like watching movies, the watching actions may be somehow random in most cases, so the movies watched in a session may not be some dependent and thus the sense of session may not be so strong. The sense of session in different domains is a critical issues and need to be further explored, which goes out of the scope of this thesis.

Except for the key difference illustrated in Paragraphs 2 and 3, session-based recommender systems are different from other ones in multiple ways. To have a substantial understanding of such differences, I conducted a comprehensive comparison between session-based recommender systems and other typical ones in Table 3.1.

Table 3.1: Comparisons between session-based recommender systems and other ones

Recommender systems	Input	Task	Time span	Status	Core assumption	Work mechanism	Pros	Cons
Content-based filtering (CBF) Rss (Tang & Wang 2018)	User, Item content information	Predict users preference on items	The whole shopping history	Staic, does not consider preference-shift along time	User likes what he/she used to likes	Matching up users profile against item content	Simple and straight-forward, can handle cold-start issues	The assumption may not fit real cases well
Collaborative filtering (Schafer et al. 2007) (CF) RSs	Uer-item (U-I) interaction data	Predict users preference on items	The whole shopping history	Static, does not consider preference-shift along time	User likes what he/she used to likes	Model user-item interactions	Effective and relative simple	Easy to face sparsity issues and cold-start issues
Context-aware RSs (Adomavicius & Tuzhilin 2015)	User, items, context and user-item interaction data	Predict users preference under particular context	The whole shopping history	Static	A user may have different preferences under different contexts	Model user-item-context interactions	Incorporate more information and fit the real-world cases better	less data availability and sparsity issues
Session-based RSs (Tang & Wang 2018)	Users transaction records (item-item interaction)	Predict following items directly	One transaction or a short period	Dynamic, consider the preference-shift along time	Users preference is changed according to different session-context	Recommend items that have occurred in a similar context	Consider the users' preference changes, which fit the real cases better	Ignore users' general and long-term preference

In this literature review, I give a comprehensive and systematic overview of the session-based recommendation scenario that regards a session as the basic unit for recommendations, and the corresponding techniques: session-based recommender systems. This is actually a relative novel recommendation paradigm which has been proposed in recent years. By doing this review, I hope to provide a comprehensive vision and a basic foundation for the research topic in this thesis as well as a useful resource for the recommender system community.

The rest of the survey is organised as follows. In the next four subsections, I first provide an overview of session-based recommender systems; particularly, I summarize the evolution of session-based RSs, give a comparison for some typical scenarios in session-based recommendations, and I categorize of the existing works into two main folds in terms of their techniques. Then for each main fold, I review and compare different representative approaches in the following two subsections respectively. Finally, I give a brief summary for this chapter.

3.2 Overview of Session-Based Recommender Systems

In this section, a comprehensive overview of session-based recommender systems is given. Specifically, an evolutionary history of session-based RSs, a comparison between different scenarios in session-based RSs, and a categorization on various session-based recommendation approaches from the technique perspective are provided.

3.2.1 A Brief History of Session-Based RSs

Traditional recommender systems have usually been divided into three categories: content-based recommender systems, collaborative-filtering recommender systems, and hybrid approaches (Adomavicius & Tuzhilin 2005, Basilico & Hofmann 2004). Content-based recommendation systems try to recommend items similar to those that a given user has liked in the past in terms of item attributes (Lops, De Gemmis & Semeraro 2011). Collaborative-filtering is the process of filtering or evaluating items by using the opinions of other people (Schafer et al. 2007). The hybrid approach combines the previous two to pluck out the advantages of both. Although different mechanisms have been behind these different approaches, essentially their settings and tasks are the same, that is, to predict users' unknown preferences based on their known preferences. These approaches usually break all transaction data down and then put all user-item pairs into a unified user-item interaction matrix and, finally, the models work on such matrix. As a result, they do not have the transaction concept, and the transaction structure is destroyed. Such a process would lose the essential information for shopping basket-based transaction data. Subsequently, this information loss leads to unreliable recommendation results. One of the obvious deficiencies is that it is easy to recommend duplicate items that are identical or similar to what a user has bought in recent transactions or has chosen in the current transaction without considering the items in a transaction as a whole (Hu, Cao, Wang, Xu, Cao & Gu 2017). Session-based, also sometimes called 'transaction-based recommender systems' (Wang, Hu & Cao 2017), in contrast, retain the transaction data structure well and treat all the items in a transaction as a set. More importantly, they emphasize the co-occurence -based dependency over items for recommendations. In this case, session-based RSs are much closer to the real-world transaction scenes. On the other hand, either the item(user) attribute data for content-based RSs or user-item rating data for collaborative-filtering-based RSs may not always be available. However, the shopping-basket-based transaction data for session-based RSs is always there as long as the basic transactions are recorded. Therefore, session-based RSs are much more applicable than traditional ones.

Research on session-based recommender systems has attracted much attention since the 1990s (Ahmad Wasfi 1998), though under different names:

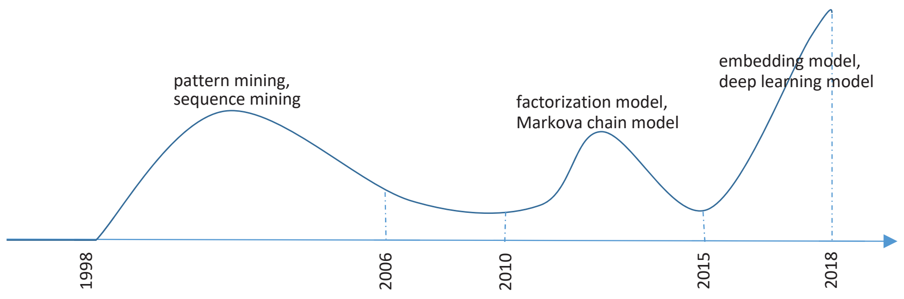


Figure 3.2: The development roadmap of session-based RSs

for example, pattern-based recommender systems, using patterns for recommendations, rule-based recommender systems, sequence-based recommender systems, transaction-based recommender systems, session-aware recommender systems, next-item recommendations, and next-basket recommendations.

Generally speaking, works on session-based recommender system can be divided into two clearly different stages: the model-free stage from the late 1990s to the early 2010s and the model-based stage from the early 2010s up until now. The first stage was driven by the development of data mining techniques, especially pattern mining, association rule discovery, and sequence mining, and, subsequently, it was dominated by pattern/rule-based and sequence-based recommender systems. According to the literature review, I ascertained that the middle of 2000s witnessed the peak of this stage, and many relevant works were published during this period. The second stage has been driven by the development of machine learning techniques, especially some time series-related models such as Markov chain models, and RNN models. On account of the fast developments of deep learning techniques in recent years, model-based RSs have reached their peak since 2017. Many researchers have rushed into this area and have developed various neural models for next-item or next-basket recommendations over the past two years. A development road-map of session-based recommender systems is given in Figure 3.2.

Currently, works related to session-based RSs have appeared in several top conferences in data mining such as KDD, CIKM, machine learning like ECML, AAAI and IJCAI and their applications such as Recsys, WWW, and SIGIR.

3.2.2 Comparisons of Session-Based RSs

There are different scenario settings in session-based recommendations. In this section, I discuss some typical scenario settings in session recommender systems.

Next-Item Recommendations versus Next-Basket Recommendations

According to whether one recommends items for the current transaction or for the next transaction, session-based recommender systems are divided into next-item recommendations and next-basket recommendations. Next-item recommender systems recommend items that are likely to be bought in a current transaction based on those chosen items in that very transaction, while next-basket recommender systems recommend items that are possibly to be bought in the next basket. Obviously, the former one is mainly based on intra-session relation modelling, while the latter is mainly based on inter-session relation modelling.

There are quite a few works focusing on next-item recommendations, while just a few focus on next-basket recommendations. Except for these two scenarios in the shopping area where the explicit sessions naturally exist (one transaction or one shopping basket naturally form a session), there are some other cases in which it is hard to divide the recommender systems into next-item or next-basket recommendations since there are no explicit natural sessions. Such cases include next-song recommendations, next-movie recommendation, and next-POI reocmmendation. In these areas, the transaction data is actually different from the shopping basket data seeing that no explicit natural session exists. For example, for a movie, which is different from shopping in which a user usually puts multiple items into a cart, usually a user only watches one each time. In cases such as movie data, the session is usually constructed manually by put several movies watched in a certain period together.

Unorderly Modelling versus Orderly Modelling

Based on whether one assumes an order over items within sessions, session-based recommender systems are divided into order assumption-based and unorder assumption-based ones. From my observations, most existing works have been order assumption-based. However, in real-word session data, especially the transaction data, there is usually no meaningful order for items within sessions. Taking the transaction data as an example, the reason for this comes from two folds: on the one hand the time-stamp for each item bought in a transaction are usually not recorded in the real world; on the other hand, even if the order of items in a transaction is available, it may not be so meaningful since a user may pick up some items randomly. Therefore, the order over items cannot really reflect the relations between them. However, in other session data, such as the song and movie data, the order over songs and movies usually follows some kind of pattern and can reflect the dependency between songs or movies to some degree.

Intra-Session Modelling versus Inter-Session Modelling

Intra-session modelling means one models the relations between items within sessions for the recommendation tasks, while inter-session modelling means that one models the relations between sessions. Most existing next-item recommendations are based on intra-session modelling without considering for inter-session relations. Quite few next-item recommendation works (Quadrana et al. 2017) take both intra- and inter-session relations into account to predict the next item in current sessions. On the contrary, next-basket recommendations mainly focus on inter-session modelling so as to capture cross-basket dependency; it should be noted, however, that here each basket is actually treated as a session.

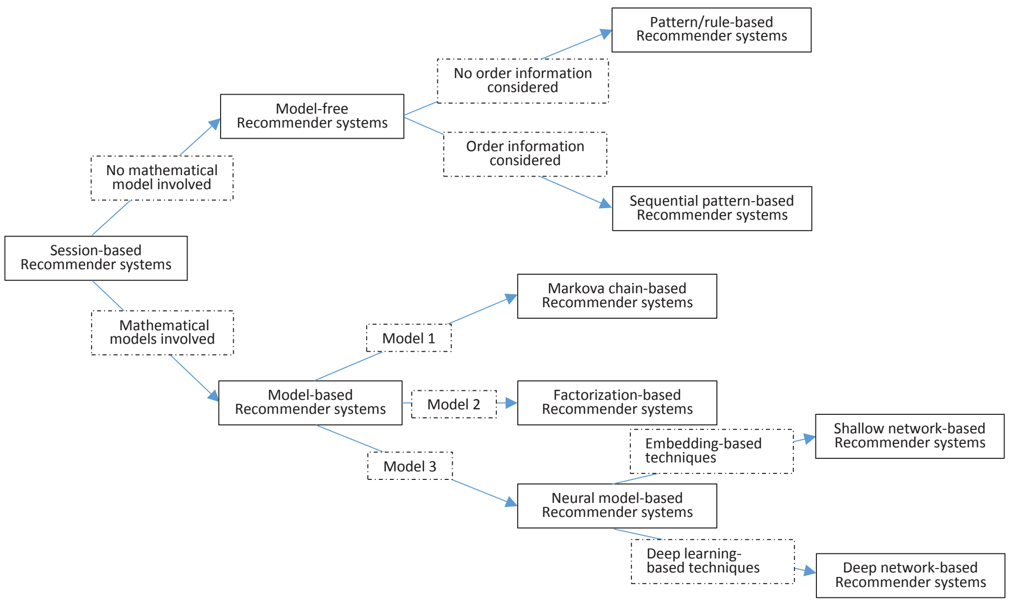

Pattern/rule-based Recommender systems
No order information considered
Model-free Recommender systems
Order information considered
No mathematical model involved
Sequential pattern-based Recommender systems
Session-based Recommender systems
Markova chain-based Recommender systems
Mathematical models involved
Model 1
Model-based Recommender systems
Model 2
Factorization-based Recommender systems
Shallow network-based Recommender systems
Embedding-based techniques
Model 3
Neural model-based Recommender systems
Deep learning-based techniques
Deep network-based Recommender systems

Figure 3.3: The categorisation of session-based recommender systems

3.2.3 Categorisation of Session-Based RSs

In this section, I categorise session-based recommender systems from the perspective of technique. Specifically, the existing works are divided into two branches: model-free approaches and model-based approaches. Each branch contains several types of approaches.

Model-Free Approaches

Model-free approaches have been mainly built on data mining techniques, and they usually do not involve mathematical models. Two typical approaches in this branch are pattern/rule-based RSs for unordinary data and sequential pattern-based RSs for ordinary data.

•***Pattern/Rule-Based Recommender Systems***: Pattern/rule-based recommender systems firstly mine frequent patterns or association rules and then use these patterns and rules to guide the subsequent recommendations. This is based on the assumption that most customers would follow the common shopping patterns (Mobasher, Dai, Luo & Nakagawa 2001, Lin et al. 2002, Abel, Bittencourt, Henze, Krause & Vassileva 2008). For instance, customers usually buy milk and bread together when they go shopping. Thus $\{milk, bread\}$ can be identified as a frequent pattern and once a user has bought milk, bread can be recommended to him or her. It should be noticed that pattern/rule-based Recommender systems are applied in unordinary data, like the items are usually picked up into a basket without strict order.

• ***Sequential Pattern-Based Recommender Systems***: For handling data having strict order over items or involving time factor-based effects, sequential pattern-based recommender systems have been proposed. Similar to pattern-based recommender systems, they firstly mine a collection of sequential patterns and then recommend the following items based on prior items and these patterns (Morales, Pérez, Soto, Martınez & Zafra 2006, Niranjan, Subramanyam & Khanaa 2010, Yap et al. 2012).

Model-Based Approaches

Different from model-free approaches, model-based recommenders are usually built on strict assumptions: for example, like order over items and solid models such as Markov chain models. To the best of my knowledge, existing model-based approaches can be mainly categorized into three branches according to the model they involve: Markov chain-based approaches, factorization-based approaches, and neural model-based approaches.

• ***Markov Chain-Based Approaches***: Markov chain-based recommender systems model the first-order or higher-order dependency over a sequence of items by using transitional probabilities, and they then generate the recommendations for the following items guided by such dependency (Shani, Heckerman & Brafman 2005, Eirinaki, Vazirgiannis & Kapogiannis 2005, Rendle et al. 2010). Different from sequential pattern-based approaches which are easy to filter out those infrequent items and to patterns and only take the frequent ones into account, Markov chain-based recommender systems take all items into consideration and, thus, they decrease the information loss greatly.

- ***Factorization-Based Approaches***: These approaches firstly factorize the item co-occurrence matrix or item-to-item transitional matrix into a latent representation vector for each item, and they then predict the following items by using these latent representations (Liang, Altosaar, Charlin & Blei 2016, Rendle et al. 2010). This approach should be distinguished from the commonly used factorization machine (such as the matrix factorization) in collaborative filtering-based recommender systems, which usually factorize the user-item interaction matrix (like a rating matrix) into user- and item-latent factors (Linden et al. 2003, Su & Khoshgoftaar 2009).

- ***Neural Model-Based Approaches***: Neural model-based approaches take advantage of the neural network to learn the complex relationships and interactions over items within or between sessions and then to generate recommendations based on such interactions. From the perspective of neural model structures, these approaches can be divided into a shallow neural model like shallow wide-in-wide-out network (Krishnamurthy, Puri & Goel 2016, Greenstein-Messica, Rokach & Friedman 2017, Wang et al. 2018) and a deep neural model such as a recurrent neural network (Hidasi et al. 2015, Song, Elkahky & He 2016).

3.3 Model-Free Approaches

Model-free recommender systems are mainly based on data mining, especially pattern mining techniques. The general idea of these is to find out the common and explicit regularities related to transactions by mining the patterns from transaction data and then to generate recommendations guided by these regularities. Two typical approaches are pattern/rule-based ones and sequential pattern-based ones.

3.3.1 Pattern/Rule-Based Approaches

Pattern/rule-based recommender systems mainly contain two stages: frequent pattern mining, session matching and item recommendation. To be specific, given a transaction set $T = \{t_1, t_2, ...\}$ and a whole item set $I = \{i_1, i_2, ...\}$ over T, a set of frequent itemsets (or patterns) $P = \{p_1, p_2, ...\}$ are mined by using pattern mining algorithms like Apriori (Aggarwal & Han 2014) and FP-Tree (Han, Pei & Yin 2000). For a given active partial session s (a collection of chosen items in one transaction), if an item i' exists so that $s \cup i'(i' \in I \setminus s)$ is a frequent pattern, namely $\{s \cup i'\} \in P$, then item i' is a candidate for recommendation. Furthermore, if the conditional probability $p(i'|s)$ is greater than a confidence threshold β, then i' is added into the recommendation list. It should be noted that $p(i'|s)$ is actually the confidence of association rule $s \Rightarrow i'$, and it is also used as the recommendation score for item i' (Mobasher et al. 2001).

I use an example derived from (Mobasher et al. 2001) to illustrate the whole process of pattern/rule-based recommender systems. Given the transaction data illustrated in Table 3.2, where each row indicates a transaction, frequent itemsets of different sizes are mined with the Aprioi algorithm with a minimum support value of 4 as shown in Table 3.3. For a user-active partial session $< B, E >$, the system matches it with two frequent itemsets of Size 3, namely $\{A, B, E\}$ and $\{B, C, E\}$, accordingly, the recommendation generation process find items A and C as the recommendation candidates. Their recommendation scores were 1 and 0.8, which correspond to the confidences of association rules $\{B, E\} \Rightarrow A$ and $\{B, E\} \Rightarrow C$, respectively.

Besides the basic pattern/rule-based framework, there are many different variants. Lin etc. (Lin et al. 2002) have utilised association rule mining techniques for collaborative filtering. Specifically, they did not require a minimum support in advance; instead, a range was given for the number of rules and algorithm adjust the minimum support for each user to obtain the desired number of rules. In this way, the algorithm can be more efficient to avoid mining too many rules that are irrelevant to a specific end user,

Table 3.2: Sample transactions involving items A, B, C, D and E (Mobasher et al. 2001)

T_1	A, B, D, E
T_2	A, B, E, C, D
T_3	A, B, E, C
T_4	B, E, B, A, C
T_5	D, A, B, E, C

Table 3.3: Frequent patterns mined from Table 3.2 and their corresponding frequency (Mobasher et al. 2001)

Size 1	Size 2	Size 3	Size 4
{A}(5)	{A,B}(5)	{A,B,C}(4)	{A,B,C,E}(4)
{B}(6)	{A,C}(4)	{A,B,E}(5)	
{C}(4)	{A,E}(5)	{A,C,E}(4)	
{E}(5)	{B,C}(4)	{B,C,E}(4)	
	{B,E}(5)		
	{C,E}(4)		

which is quite different from traditional association rule mining, in which the minimum support is given in advance without considering any user. To considering the different significance of different items and thus to recommend more useful items, (Yan & Li 2006) and (Forsati, Meybodi & Neiat 2009) have used page-view duration to weight the significance of each page, and they have then incorporated such weights into the association rule model so as to develop weighted association rule-based web recommender systems. By mining users' behaviour patterns, e.g., web navigation patterns, the association rule mining technique is applied to capture the needs or preference of a specific user or a group of users and thus to help to generate personalized recommendations (Forsati et al. 2009, Lee, Kim & Rhee 2001, Adomavicius & Tuzhilin 2001, Zhang 2007). Some other works have combined pattern mining into traditional collaborative filtering methods so as to help address some issues such as sparsity, robustness, and personalizition (Lee et al. 2001, Huang, Chen & Zeng 2004, Sandvig, Mobasher & Burke 2007). For the application domains, except for the traditional shopping basket-based recommendations, pattern-based recommender systems have been commonly applied in web recommendations (Moreno, García, Polo & López 2004) and music recommendations (Shao, Wang, Li & Ogihara 2009).

3.3.2 Sequential Pattern-Based Approaches

Sequential pattern mining (SPM) (Han, Pei, Mortazavi-Asl, Chen, Dayal & Hsu 2000) is different from the aforementioned frequent pattern mining (FPM) or association rule mining in two regards: firstly, SPM takes the order or time series factor into account; in other words, it requires a strict order over itemsets. Secondly, SPM mainly captures the inter-session dependency, namely, the association between items from different itemsets while FPM mainly focuses on the intra-session dependency, namely the association over items within transactions. In this way, the mined sequential patterns incorporate more factors and thus are more informative compared to frequent patterns. Therefore, the sequential pattern-based recommender systems are more suitable for time-aware session-based RSs or inter-session-based RSs, like next-basket recommendations. Similar to the aforementioned pattern/rule based recommender systems, sequential pattern-based recommender systems also contain three main stages: sequential pattern mining, sequence matching and recommendation generation. In the next paragraph, I give a formulation of sequential pattern-based recommender systems.

Given a sequence database Q, which is a collection of sequences, i.e., $Q = \{q_1, q_2, ..., q_n\}$, where $n = |Q|$ denotes the number of sequence in Q. A sequence $q \in Q$ is a collection of ordered itemsets, i.e., $q = \{I_1, I_2, ..., I_m\}$ associated with a user, namely the collection of transactions collected from one user over a certain period, where $m = |q|$ is the number of itemsets or transactions. It should be noted that each transaction happens in a unique timestamp, and all the itemsets in one sequence are ordered in terms of their happening time. An itemset $I \in q$ is a set of multiple items transacted in one transaction, i.e., $I = \{i_1, i_2, ..., i_k\}$ where $k = |I|$ is the size of I. A sequence $q_x = \{q_{x1}, q_{x2}, ..., q_{x|q_x|}\}$ contains another sequence $q_y = \{q_{y1}, q_{y2}, ..., q_{y|q_y|}\}$, or q_y is a subsequence of q_x if there exists a series of integers $d_1 < d_2 < ...d_{|q_y|}$ such that $q_{y1} \subset q_{xd_1}, q_{y2} \subset q_{xd_2}, ..., q_{y|q_y|} \subset q_{xd_{|q_x|}}$ (Agrawal & Srikant 1995). For any subsequence q', its frequency is the number of sequences containing q' in the sequence database Q, i.e., $freq(q') = |Q'|, q' = \{q''|q' \subset q'', q'' \in Q\}$. The support of subsequent q' is defined as $supp(q') = freq(q')/|Q|$. A sequence is defined as a sequential pattern p if its support is not less than a minimum threshold δ.

Given a user u's past sequence q_u and the sequence database Q, the sequential pattern-based recommendation task is to predict the items that the user is most probably to buy in the near future by matching q_u to the mined sequential patterns from Q (Yap et al. 2012). Specifically, let $q_u = \{I_1, I_2, ..., I_h\}$ where I_{uj} is the set of items bought by user u at timestamp j and $QP = \{p_1, p_2, ..., p_{|QP|}\}$ is the set of mined sequential patterns on Q. For any pattern $p \in QP$, if the last itemset $I_h \in p$, namely $p = \{I_1, I_2, ..., I_h, I_l...\}$, then the pattern p is a relevant pattern for this specific recommendation and the items in itemsets after I_h in p like I_l are candidate items for recommendations. For each candidate item i_c, the support is the sum of the support of all mined relevant patterns. This can be formally represented as follows:

$$supp(i_c) = \sum_{I_h \in q_u, I_h \in p, i_c \in I_l, I_l \in p, p \in QP} supp(p) \tag{3.1}$$

Finally, those candidate items with top support values are recommended to user u.

The above definition and formulation illustrate the basic framework of sequential pattern-based recommender systems. Actually, there are various variants and extensions to make a more reliable recommender system. A typical extension is to utilize user-related weighted sequential pattern mining for personalized recommendations. To be specific, each sequence is no longer treated equally contributed to the target user's recommendations, as the aforementioned basic framework, ; instead, all the sequences are assigned a weight based on the similarity between them and the past sequence of the target user (Song & Yang 2014, Zhang & Cao 2013, Yap et al. 2012). In this way, the recommender system can generate more reliable and precise recommendations for different users by incorporating more user-relevant information. Another extention is to build a hybrid recommender system by combining sequential pattern-based recommender systems and traditional collaborative filtering methods (Zhang & Cao 2013, Choi, Yoo, Kim & Suh 2012, Huang & Huang 2009, Liu, Lai & Lee 2009). Because of this combination, both the dynamic individual behaviour pattern reflected by the sequential patterns and the general preference modelled by the collaborative filtering approaches are considered. As a result, the recommendation would be more accurate. From the application perspective, item recommendations in shopping-basket-based transaction data and web page recommendations in web access log data are two typical application domains, which are built on customer shopping behaviour sequential patterns and user access sequential patterns (Zhou, Hui & Fong 2006, Niranjan et al. 2010) respectively.

3.4 Model-Based Approaches

Different from model-free approaches, model-based ones are mainly based on special mathematical models such as the Markov chain and matrix factorization etc. In this section, the three representative types of model-based approaches are introduced and discussed, i.e, Markov chain-based ones, matrix factorization-based ones, and neural model-based ones.

3.4.1 Markov Chain Model-Based Approaches

Markov chain Model-based recommender systems adopt the Markov chain model to model the transition of items on the transaction sequence data so as to predict the probable next item given a sequence of prior items. According to the number of states taken into account when computing the transition probability, the model can be either a first-order or higher-order Markov chain model. To decrease the model's complexity, most recommender systems have been built on first-order Markov chains, like (Shani et al. 2005, Wu, Liu, Chen, He, Lv, Cao & Hu 2013, Feng et al. 2015, Bonchi, Perego, Silvestri, Vahabi & Venturini 2011). In the next section, I give the formulation for a basic Markov chain-based recommender system as well as some variants and extensions.

Basic Markov chain-based RSs

Generally speaking, the process of a basic Markov chain-based recommender system is simple: firstly to calculate the transition probability over a sequence of items from the training data and then to match a user's shopping sequence to the sequence with calculated transition probability for prediction and recommendations; subsequently, those candidate items with high probability are put into the recommendation list (Eirinaki et al. 2005).

Usually, a set of user transaction sessions $S = \{s_1, s_2, ..., s_{|S|}\}$ are given, where each session $s = \{i_1, i_2, ..., i_{|s|}\}(s \in S)$ corresponds to a sequence of items bought consecutively in a transaction. It should be noted that the strict order assumed over items in a session, which is quite different from the case in pattern/rule -based RSs as illustrated in Section 3.3.1. A Markov chain synopsis was built to encode all the transaction sessions into a directed graph G. Each item in S corresponds to a node in the graph; in addition, an external start node and an end node were added into the graph. Therefore, each session is described by a path in the graph from the start to the end node. The frequency of each item and the co-occurrence times of each pair of successive items correspond to the node weight and edge weight respectively.

The Markov chain model is defined as a set of tuples $\{S, P_t, P_0\}$, where S is state space including all all distinct nodes in G, P_t is the $m * m$ one-step transition probability matrix between m distinct items and P_0 is the initial probability of each state in S. The first-order transitional probability from item i_j to i_k is defined as follows:

$$P_t(j,k) = P(i_j \to i_k) = \frac{freq(i_j \to i_k)}{\sum_{i_t \in I} freq(i_j \to i_t)} \tag{3.2}$$

where $I = \{i_1, i_2, ..., i_{|I|}\}$ is the whole set of all items.In the case of higher-order Markov model, the transition probability of an item should be computed given the past multiple items. For the n^{th}-order model, the transition probability is calculated given the past n items and result in a $m^n * m$ transition probability matrix.

For the initial probability of each state, there are mainly two basic calculation methods: The first one is to assign equal probabilities to all nodes, namely use uniform distribution. The second one is to estimate the initial probability of a node proportionally to its frequency. (the ratio of its frequency to the total frequency of all nodes). In addition, there are some more calculation methods that can be related to specific scenarios, like treating the first visited page more important than others in web page recommendations or taking the link structure into account (Eirinaki et al. 2005).

After the transition and initial probability is ascertained, the chain rule is applied to compute the probability $P(i_1 \to i_2 \to ... \to i_l)$ of each path indicating a sequence of items $\{i_1, i_2, ..., i_l\}$. For instance, if we want to build an n^{th}-order Markov model, the path probability can be ascertained as follows:

$$P(i_1 \to i_2 \to ... \to i_l) = P(i_1) * \prod_{j=2}^{l} P(i_j|i_{j-n}...i_{j-1}) \tag{3.3}$$

For example, if we want to estimate the probability of a shopping path $\{i_1, i_2, i_3\}$ using the first-order Markov Chain, the above equation can be reduced to the following equation:

$$P(i_1 \to i_2 \to i_3) = P(i_1) * P(i_2|i_1) * P(i_3|i_2) \tag{3.4}$$

With regard to the recommendation task, given a sequence of chosen or bought items, I chose the shopping paths in this research with high probabilities, and I took the given sequence as the prefix as the guidance. Those items occurred in these paths and after the given sequence they were put into the recommendation list.

Except for the basic process of Markov chain-based RSs, as defined above, there are many variants and extensions like (He & McAuley 2016*a*) . For example, Zhang etc. combined first and second-order Markov model together to make more accurate web recommendations(Zhang & Nasraoui 2007) while Le etc. developed a hidden Markov model-based probabilistic model for sequence modelling and next item recommendations. Besides the sequences of the items themselves, they have also incorporated other factors such as context features which may dynamically influence the state transition into the model leverage the recommendation accuracy (Le, Fang & Lauw 2016). Another important variant has been to adopt factorization method over the transition probability of the Markov model so as to estimate those unobserved transitions (Rendle et al. 2010).

Latent Markov Embedding (LME)-based RSs

Different from the basic Markov chain-based recommender systems which calculate the transition probability based on the explicit observations directly, latent Markov Embedding (LME)-based RSs first embed the Markov chains into a Euclidean space and then calculate the transition probability between items based on their Euclidean distance in this latent space (Chen, Moore, Turnbull & Joachims 2012). In this way, it can drive the unobserved transitions and thus solve the sparsity issue in the case of limited observed data. Formally, each item i is represented as a vector V_i in a d-dimensional Euclidean space M, and the transition probability $p(i_{(j-1)} \rightarrow i_j)$ is assumed to be negatively related to the Euclidean distance $||V_{i_j} - V_{i_{j-1}}||_2$ between items i_j and i_{j-1} via the following logistic function:

$$P(i_{j-1} \rightarrow i_j) \propto e^{-||V_{i_j} - V_{i_{j-1}}||_2^2} \tag{3.5}$$

It should be noted $\sum_{i_K} P(i_{j-1} \rightarrow i_k) = 1$, thus the transition probability is normalized as:

$$P(i_{j-1} \rightarrow i_j) = \frac{e^{-||V_{i_j} - V_{i_{j-1}}||_2^2}}{\sum_{i_k \in I} e^{-||V_{i_k} - V_{i_{j-1}}||_2^2}} \tag{3.6}$$

where I is the whole set of all items.

As a result, given this transition probability, the probability of an shopping sequence $p = \{i_1 \rightarrow i_2 \rightarrow, ..., i_l\}$ can be defined as follows, based on the Markov model:

$$P(p) = \prod_{j=2}^{l} P(i_{j-1} \rightarrow i_j) = \prod_{j=2}^{l} \frac{e^{-||V_{i_j} - V_{i_{j-1}}||_2^2}}{\sum_{i_k \in I} e^{-||V_{i_k} - V_{i_{j-1}}||_2^2}} \tag{3.7}$$

As this definition only focuses on the transitions over items, while it ignores the user factors, it cannot generate personalized recommendations. Wu etc. (Wu et al. 2013) proposed personalized Markov embedding (PME) which maps both users and items into a Euclidean space where the distance between user and item reflects their relationship, while the distance between a pair of items also reflects the relation of them as (Chen et al. 2012). In this way, a personalized transition probability $P(i_{j-1} \rightarrow i_j, u)$ from item i_{j-1} to i_j by user u is negatively related to both $||V_{i_j} - V_{i_{j-1}}||_2$ and $||V_u - V_{i_{j-1}}||_2$. This can be also explained as both the short-term sequential behaviours modelled by the item pairwise relations and the long-term general preference modelled by the user-item relations are both taken into account. Therefore the Equation (3.6) is changed into the following:

$$P(i_{j-1} \rightarrow i_j, u) = \frac{e^{-||V_{i_j} - V_{i_{j-1}}||_2^2 - ||V_u - V_{i_{j-1}}||_2^2}}{\sum_{i_k \in I} e^{-||V_{i_k} - V_{i_{j-1}}||_2^2 - ||V_u - V_{i_{j-1}}||_2^2}} \tag{3.8}$$

Both the above approaches learn the latent embeddings of items and users by only exploiting the observed data, however, such data is usually quite sparse and leads to poor performance. Personalized ranking metric embedding has been proposed in (Feng et al. 2015) to learn the embeddings by fitting the ranking of the POI transition. In such a case, the unobserved data can also be used for the learning task. Specifically, it assumes the observed next POI is more related to the current one than those unobserved. If the transition $i_j \rightarrow i_k$ is observed while $i_j \rightarrow i_t$ is not, i_k should be ranked higher than i_t in terms of i_j, this is modeled as a ranking $>$ over POIs. Instead of utilizing the transition probability, it models the ranking of it. In this regard, one should recall that transition probability is related to the Euclidean distance in the item embedding space, so the whole model is transferred into the following formula:

$$P(i_j \rightarrow i_k) > P(i_j \rightarrow i_t) \Rightarrow d_{i_j,i_k} - d_{i_j,i_t} > 0 \tag{3.9}$$

where d_{i_j,i_k} is the Euclidean distance between items i_j and i_t in the embedding space.

3.4.2 Factorization Machine-Based Approaches

It is well known that factorization machine is commonly used in collaborative filtering recommender systems to factorize the user-item preference matrix (e.g., the rating matrix) into user and item latent factor vectors, and it has achieved great success. Inspired by this, researchers have applied it into session-based recommender systems recently so as to model short-term sequential shopping patterns (Shani et al. 2005, Lian, Zheng & Xie 2013, Hidasi & Tikk 2016). Specifically, factorization machines have been adopted to factorize the observed preference or item transitions from the current time point or transaction to the next one into latent representations of items or users. Subsequently, the resultant latent representations have been used to estimate the unobserved transitions for prediction and recommendation tasks. In the next section, I formalize the basic process of factorization machine-based session-based recommender systems, and I then review some variants and extensions.

For a next-basket recommendation issue as demonstrated in (Rendle et al. 2010), the transitions exist between items bought in two successive transactions (basket). Given a set of successive transactions, the recommender system aims to predict what may be bought in the next transaction. Similarly, let $U = \{u_1, u_2, ..., u_{|U|}\}$ and $I = \{i_1, i_2, ..., i_{|I|}\}$ be a set of users and items respectively. For each user u, a collection of baskets recording his purchase history is known as $B^u = \{B^u_1, B^u_2 ... B^u_t\}$. Therefore an unpersonalized first-order Markov chain for the next-basket recommendation problem is as follows:

$$P(B_t|B_{t-1}) \tag{3.10}$$

To reduce the model complexity of Markov chain for set/basket, the model can be transferred to model the transitions over $|I|$ binary variables that describe a basket:

$$a_{i_j,i_k} = P(i_k \in B_t | i_j \in B_{t-1}) \tag{3.11}$$

Given the historical transaction data, the transition probability can be estimated by the frequency:

$$\begin{aligned} a_{i_j,i_k} = P(i_k \in B_t | i_j \in B_{t-1}) &= \frac{P(i_j \in B_{t-1} \wedge i_k \in B_t)}{P(i_j \in B_{t-1})} \\ &= \frac{|(B_{t-1}, B_t) : i_j \in B_{t-1} \wedge i_k \in B_t|}{|(B_{t-1}, B_t) : i_j \in B_{t-1}|} \end{aligned} \tag{3.12}$$

Further more, the user u can be taken into account for estimating personalized transitions, as the equation below shows:

$$a_{u,i_j,i_k} = P(i_k \in B^u_t | i_j \in B^u_{t-1}) = \frac{P(i_j \in B^u_{t-1} \wedge i_k \in B^u_t)}{P(i_j \in B^u_{t-1})} \tag{3.13}$$

Once the personalized transition probability is achieved from the observed data, a transition matrix A^u is built for each user u. Therefore, for all users, a transition tensor A is built like $\mathcal{A}^{|U|\times|I|\times|I|}$. A general linear factorization model, the Tucker decomposition is used to factorize the transition cube and to estimate the unobserved transitions:

$$\hat{A} = C \times V^U \times V^{I_j} \times V^{I_k} \tag{3.14}$$

where C is a core tensor, V^U is the feature matrix for users while V^{I_j} and V^{I_k} are the feature matrix for last items and the following items respectively. The Tucker Decomposition subsumes factorization models like the Canonical Decomposition (Bandelt & Dress 1992), also known as the parallel factor analysis (PARAFAC). The parallel factor model assumes a diagonal core tensor with equal factorization dimensionality. As the observed transitions for $\mathcal{A}$ are extremely sparse, a special case of Canonical Decomposition is used to model the pairwise interactions:

$$\hat{a}_{u,i_j,i_k} =< v_u^{u,i_j}, v_{i_j}^{u,i_j} > + < v_{i_j}^{i_j,i_k}, v_{i_k}^{i_j,i_k} > + < v_u^{u,i_k}, v_{i_k}^{u,i_k} > \tag{3.15}$$

where v_u^{u,i_j} and v_i^{u,i_j} are the latent factor vector of user u and the current item i_j respectively. To this end, a factorizing personalized Markov chain (FPMC) model was built for next-basket recommendations.

Cheng etc. (Cheng, Yang, Lyu & King 2013) have applied FPMC into next-POI recommendations and added the user's movement constraints to limit the users' movements into a localized region, which is claimed to be more consistent with the real-world tourism cases. In this case, the FPMC is extended into FPMC-LR model.

The factorization models which are only built on item co-occurrence transition matrix actually only captures the sequential shopping patterns over items, while they ignore the users' individual preferences. These preferences can be captured by the traditional reocmmender systems like collaborative filtering. In (Liang et al. 2016), the authors combined the traditional matrix factorization used in collaborative filtering and that used in session-based recommender systems together to capture both the individual preferences and the item transition patterns. Specifically, a co-factorization model, CoFactor, was proposed to jointly decompose the user-item interaction matrix and the item-item co-occurrence matrix with shared item latent factors. Based on the unordinary assumption over items within transactions, they used point wise mutual information (PMI) (Bouma 2009) to measure the relationship between a pair of items, and then integrated all PMI values into a PMI matrix and then transferred it into a shifted positive PMI (SPPMI) matrix:

$$SPPMI(i_j, i_k) = max\{PMI(i_j, i_k) - \log(K), 0\} \tag{3.16}$$

where K is a hyper-parameter to control the sparsity of the SPPMI matrix.

Once the item co-occurrence matrix SPPMI matrix is achieved and, combined with the user-item interaction matrix, the CoFactor model is formalized as:

$$\begin{aligned}\mathcal{L} = \sum_{u,i_j} \alpha_{u,i_j}(r_{u,i_j} - \theta_u^T \beta_{i_j})^2 + \sum_{i_j,i_k}(pm_{i_j,i_k} - \beta_{i_j}^T \gamma_{i_k})^2 \\ +\lambda_\theta \sum_u ||\theta_u||_2^2 + \lambda_\beta \sum_{i_j} ||\beta_{i_j}||_2^2 + \lambda_\gamma \sum_{i_k} ||\gamma_{i_k}||_2^2\end{aligned} \tag{3.17}$$

where θ_u, β_{i_j} and γ_{i_k} are the latent factor vectors of user u, current item i_j and the next item i_K respectively. r_{u,i_j} and pm_{i_j,i_k} are the interaction value (e.g., the rating) between user u and item i_j and SPPMI value between items i_j and i_k respectively. α_{u,i_j} is a sacling parameter to balance the observed interactions and unobserved interactions between users and items.This objective can also be seen as a means of regularizing the traditional matrix factorization in collaborative filtering with the item co-occurrence term.

Other similar works include (Liu, Liu, Aberer & Miao 2013) which utilized the matrix factorization model to learn the preference transitions from one location category to another and thus to provide location recommendations.

3.4.3 Neural Model-Based Approaches

Having been derived from the powerful generative and representative capability of neural networks, a series of neural model-based approaches have been developed to model the comprehensive relations between item features, items and transactions and thus to generate recommendations. Generally, session-based recommender systems based on neural models can be divided into two groups: shallow neural models and deep neural models, according to the number of layers incorporated into the neural networks. In the next section, I introduce these two kinds of recommendation models respectively.

Shallow Neural Network-Based Models

Shallow neural models usually contain a shallow network structure (Goth 2016) which maps the items or sessions into a latent space and then conducts operations in this latent space more easily. The mechanism behind this is that, when mapping items onto a multi-dimensional latent space, their positions in such a space reflect their relations, for instance, the items within a short distance have a higher similarity and are more relevant to each other. To the end, the latent numerical vector representation of each item contains much more information than the original item ID. As a result, the operations conducted on these latent representations are much more effective than the original representations, such as the ID.

Generally speaking, these types of recommender systems have been mainly developed in the past three to four years, and they have been inspired by the great success of word-embedding techniques, which were proposed in 2013 for natural language processing (Mikolov, Chen, Corrado & Dean 2013, Mikolov, Sutskever, Chen, Corrado & Dean 2013). Word embedding learns a latent vector representation of each word in a sentence by considering the interaction between the given word and its corresponding context in the sentence. Two typical neural word embedding models are Skip-gram (Pennington, Socher & Manning 2014) and CBOW model (Mikolov, Le & Sutskever 2013).

A representative shallow neural model for session-based recommendations has been given by (Hu, Cao, Wang, Xu, Cao & Gu 2017). It has designed a shallow network with wide-in-wide-out structure so as to first map user ID and its corresponding item IDs into latent vector representations and then combine them together as the given context representation, which is finally fed into the output layer (a softmax layer) to predict the corresponding next item. In the next section, I define the main process of this model, which is a basic framework for shallow neural model for session-based recommender systems.

Similarly, $U = \{u_1, u_2, ..., u_{|U|}\}$ and $I = \{i_1, i_2, ..., i_I\}$ denote a set of users and items, respectively. $S = \{s_1, s_2, ..., s_{|S|}\}$ denotes the collection of all the observed shopping sessions (transactions), while each session contains a subset of items $s \subset I$. The shallow network-based recommender system is trained as a probabilistic classifier which learns a conditional probability $P(i_t|\mathbf{c})$ w.r.t the target item i_t to predict where $\mathbf{c} = s \setminus i_t$ is the session context for item i_t, similar to the context in word embedding models. To generate personalized recommendations, the corresponding user u is added as user context. Accordingly, the the shallow network is refined and trained as a classifier over the conditional distribution $P(i_t|u, \mathbf{c})$, while the session-based recommendation problem is reduced to generate rankings over all candidate items according to the conditional probability $P(i_t^{'}|u', \mathbf{c}')$, given a user-session context $< u', \mathbf{c}' >$.

The shallow network first embeds a user u and an item i into a latent vector by using the following equations:

$$e_u = \delta(\mathbf{W}^1 :, u) \tag{3.18}$$

$$e_i = \delta(\mathbf{W}^2 :, i) \tag{3.19}$$

where $\mathbf{W}^1 \in \mathbb{R}^{K \times |U|}$ is the weight matrix to connect the user input to the hidden layer while $\mathbf{W}^2 \in \mathbb{R}^{L \times |I|}$ is the weight matrix to connect session input to the hidden layer. In this way, each user and item are represented by a numerical vector $e_u \in [0, 1]^K$ and $e_i \in [0, 1]^L$ respectively, which is similar to the vector representation of each word in word2vec model (Goldberg & Levy 2014), just as the aforementioned Sikp-gram or CBOW.

$$e_c = \sum_{i \in \mathbf{c}} \omega_i e_i \tag{3.20}$$

where ω_i is a weight to measure the importance scale of contextual item i and $\sum_{i \in \mathbf{c}} \omega_i = 1$.

From the hidden layer to the output layer, two weight matrices $\mathbf{W}^3 \in \mathbb{R}^{|I| \times K}$ and $\mathbf{W}^4 \in \mathbb{R}^{|I| \times L}$ are used to fully connect the user embedding layer and the item embedding layer to the output layer, in turn. Then the score S_{i_t} of a target item i_t w.r.t the user-session context $< u, \mathbf{c} >$ in terms of the embeddings $< e_u, e_c >$:

$$S_{i_t}(u, \mathbf{c}) = \mathbf{W}^3_{t,:} e_u + \mathbf{W}^4_{t,:} e_c \tag{3.21}$$

Finally, the conditional distribution is defined in terms of the softmax function:

$$P_\theta(i_t|(u, \mathbf{c})) = \frac{exp(S_{i_t}(u, \mathbf{c}))}{Z(u, \mathbf{c})} \tag{3.22}$$

where $Z(u, \mathbf{c}) = \sum_{i \in I} exp(S_i(u, \mathbf{c}))$ is the normalisation constant, and $\theta = \{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3, \mathbf{W}^4\}$ defines the model parameter set.

With regard to further improvement, one might incorporate item features into the network to embed an item's ID and its feature simultaneously to tackle cold-start recommendation issues (Wang, Hu & Cao 2017);(Vasile, Smirnova & Conneau 2016, Wang, Deng, Zhang & Xu 2016) have also incorporated side information to help with building a more informative item embedding. To attentively learn the real relevance scale of different contextual items to the target item, the attention mechanism has been adapted into the embedding process of items (Wang et al. 2018). (Wang, Guo, Lan, Xu, Wan & Cheng 2015) have learned hierarchical representation for the next-basket representation (Wan, Lan, Wang, Guo, Xu & Cheng 2015). In addition, there are more similar session-based recommender systems that have been built on item embedding inspired by word2vec models, such as that of (Greenstein-Messica et al. 2017, Wang, Deng & Xu 2017, Barkan, Brumer & Koenigstein 2016, Krishnamurthy et al. 2016, Li, Chen & Yan 2017, Zhao, Huang & Wen 2016, Ozsoy 2016).

Deep Neural Network-Based Models

Deep neural networks have become popular since 2006 when Hinton proposed an unsupervised layer-wise pre-training approach to train a deep neural network (Hinton, Osindero & Teh 2006). Deep neural networks have been widely employed into recommender systems in recent years; for instance, some researchers have incorporated them into collaborative-filtering recommender systems to better model the user-item interactions, like (Salakhutdinov, Mnih & Hinton 2007, Wang, Wang & Yeung 2015) and (Li, Kawale & Fu 2015). In 2016, a workshop focused on deep learning for recommender systems (Karatzoglou, Hidasi, Tikk, Sar-Shalom, Roitman, Shapira & Rokach 2016) was held in the prestigious flagship conference for recommender systems - Recs2016, followed by another workshop (Hidasi, Karatzoglou, Sar-Shalom, Dieleman, Shapira & Tikk 2017) and a tutorial(Karatzoglou & Hidasi 2017) on the same topic at the same conference in 2017. It is generally believed that the deep neural network-based models for session-based recommendations started in 2016 when a GRU-based recurrent neural network (GRU4Rec) was designed for session-based recommender systems by (Karatzoglou et al. 2016). Following this work, a series of deep neural models have been proposed for session-based recommendations, like (Tan, Xu & Liu 2016, Hidasi, Quadrana, Karatzoglou & Tikk 2016, Quadrana et al. 2017, Hidasi & Karatzoglou 2017). Out of these, recurrent neural network (RNN) -based models have dominated this area due to their intrinsic natures and advantages for modelling sequential data in a session since the elements such as items, songs in a session are usually regarded as ordered data points. In addition, to explore the other characteristics of session-based data, naive deep neural networks (DNN) and convolutional neural networks (CNN) have also been employed in this area -but much less when compared to RNN. In the next section, I introduce the representative deep neural session-based recommender systems based on different network architectures.

•***RNN-Based Models***: As mentioned before, the first representative deep neural model for session-based recommendations was the GRU-based RNN model (GRU4Rec) proposed in 2016 (Hidasi et al. 2015). In addition, this model is also the most representative RNN-based session-based recommender system. In the following section, I first introduce this model, then give some improved versions of this model, and finally show some variants of RNN-based models for session-based recommendations.

Similar to the scenario setting in Markov chain-based recommender systems, usually a set of user transaction sessions $S = \{s_1, s_2, ..., s_{|S|}\}$ are given, where each session $s = \{i_1, i_2, ..., i_{|s|}\}(s \in S)$ corresponds to a sequence of items bought consecutively in a transaction. GRU4Rec models each session as a sequence, particularly, it predicts a probability distribution over the next element of the sequence when given the current hidden state.

Usually, a standard RNN update the hidden state h using the following update function:

$$h_t = f(\mathbf{W}x_t + Uh_{t-1}) \tag{3.23}$$

Where f is a activation function such as a logistic sigmoid function and x_t is the input of the unit at time t, a probability distribution over the next element of the sequence is outputted given the current hidden state h_t (Hidasi et al. 2015). Essentially, RNN uses all the given items in a sequence as the condition to predict the next possible item when it is employed into session-based recommendations.

A gated recurrent unit (GRU) is a more elaborate RNN unit and aims at handling gradient vanishing problems by learning when and by how much to update the hidden state of the unit (Cho, Van Merriënboer, Bahdanau & Bengio 2014). To this end, the hidden state h_t is updated in the following form:

$$h_t = (1 - z_t)h_{t-1} + z_t\hat{h_t} \tag{3.24}$$

where the update gate z_t and the candidate hidden state $\hat{h_t}$ are computed by the following equations respectively. z_t actually decides how much the unit updates its hidden state from the last state.

$$z_t = \sigma(\mathbf{W}_z x_t + U_z h_{t-1}) \tag{3.25}$$

$$\hat{h_t} = tanh(\mathbf{W}x_t + U(r_t \odot h_{t-1})) \tag{3.26}$$

where σ is the logistic sigmoid function and the reset gate r_t is given by:

$$r_t = \sigma(\mathbf{W}_r x_t + U_r h_{t-1}) \tag{3.27}$$

Each GRU unit stands for one hidden state and thus a GRU layer consisting of a sequence of connected GRU units forms a hidden layer of an RNN. In this case, a l-length sequence can be modeled by a hidden layer with l GRU units. The whole model built in (Hidasi et al. 2015) tarts from the input layer to input the one-hot item encoding into the model and then followed by an embedding layer to map each ont-hot encoding into a K-dimensional numerical vector representation. This vector representation is the input of subsequent multiple GRU layers. Finally, a feed-forward layer is used to connect the last GRU layer and the output layer.

Now it is necessary to go back to my session-based recommender system. In (Hidasi et al. 2015) a session-parallel mini-batches technique was adopted to feed the session data into the GRU-based RNN model. At each state t, the input x_t is the embedding of item i_t in a session s. In this way, given the past t items in session s, GRU4Rec is built to model the intra-session dependency over sequential items and, thus, to predict the probability distribution on the next item.

To further improve the recommendation performance, Tan et al. have exploited and adopted some techniques into the training and test processes of the GRU4Rec model to generate an improved version of the GRU4Rec (Tan et al. 2016). To be specific, data augmentation via sequence preprocessing and embedding dropout were applied to enhance training and to reduce overfitting of the model. In addition, a model pre-training approach was proposed to consider the possible temporal shifts in the input data distribution. The recommendation accuracy was claimed to be improved by over 10% due to the application of these techniques. To build a personalized user profile based on a user's shopping history and, thus, to generate personalized recommendations, Quadrana et al.(Quadrana et al. 2017) have further improved the GRU4Rec by proposing a hierarchical recurrent neural networks model to incorporate cross-session information for recommendation tasks. Specifically, a two-level GRU-based RNN was designed: the session-level GRU models the user shopping activity within sessions and generates recommendations for the next items while the user-level GRU models the cross-session information transfer and provides personalization to the session-level GRU by initializing its hidden state. To this end, both the intra-session dependency and the inter-session dependency were captured to generate more reliable next item recommendations compared to those session-only methods in which only intra-session dependency was captured. Another quite similar work to apply two-level RNN structure to model both intra- and inter-session dependency for next item recommendations was II-RNN (Inter-Intra RNN) proposed in (Ruocco, Skrede & Langseth 2017). In (Donkers, Loepp & Ziegler 2017), the authors have designed a unique user-based GRU model to incorporate users' characteristics into the model for personalized next-item recommendations.

Except for the main representative GRU like the RNN-based recommender systems mentioned above, there are many other variants and extensions for RNN-based next-item recommendation models. An interesting one is the Dynamic RRcurrent bAsket Model (DREAM) proposed by Yu et al. in 2016 (Yu et al. 2016). DREAM both learns a dynamic representation of a user and captures global sequential features among baskets by modeling all sequential historical shopping baskets with a recurrent architecture. Specifically, DREAM utilizes the hidden state h_t^u of user u at time point t of the RNN as the representation of the corresponding user at time t while the input for each state is the embedding of each basket aggregated from the embeddings of all the items contained in it. In this case, the user representation is different at each time point as new baskets are added into the model as time progresses. Formally, this can be formulated as follows:

$$h_t^u = f(Xb_t^u + Rh_{t-1}^u) \tag{3.28}$$

where b_t^u is a latent vector representation of the user u's basket purchased at time t, and h_{t-1}^u is the dynamic representation of u's last time point $t-1$. f is an activation function and sigmoid function (Yu et al. 2016).

Other extensions include the incorporation of variational inference into the RNN architecture to handle the uncertainty in sparse transaction data and to simultaneously leverage the model's scalability with large real-world datasets (Chatzis, Christodoulou & Andreou 2017, Christodoulou, Chatzis & Andreou 2017), incorporating other additional information like the item features and other contextual factors like time, location, and interfaces into RNN-based models to enhance the recommendations (Hidasi et al. 2016, Beutel, Covington, Jain, Xu, Li, Gatto & Chi 2018), introducing time decay or attention mechanisms into RNN-based models to discriminate the intra-session dependency over items and, thus, to achieve more precise recommendations (Bogina & Kuflik 2017, Pei, Yang, Sun, Zhang, Bozzon & Tax 2017) and combining RNN with transitional approaches like factorization machines or neighbourhood-based methods to make up the drawbacks of RNN-only models (Twardowski 2016, Jannach & Ludewig 2017*b*). Other similar RNN-based models include (Jing & Smola 2017, Hidasi & Karatzoglou 2017, Wu, Ahmed, Beutel, Smola & Jing 2017).

•***DNN-Based Models***: Except for the RNN, the deep neural network (DNN) is an alternative solution for session-based recommendation tasks, especially when no strict order exist over items within sessions. Here DNN actually refers to the naive deep neural network structure consisting of multi-layer perceptron (MLP). In the next section, I introduce a typical DNN-based session recommender system in detail, and I then talk about some other variants.

In (Wu & Yan 2017), a DNN has been applied to learn a session's representation for recommendations. Specifically, a user u's online shopping session $s_u = \{s_c, s_v, i_t\}$ contains a set of click items $s_c = \{i_{c1}, i_{c2}, ..., i_{c|s_c|}\}$ and a set of view items $s_v = \{i_{v1}, i_{v2}, ..., i_{v|s_v|}\}$ and then the target item i_t. The proposed list-wise DNN-based model firstly maped each item i contained in a session s to a numerical-valued embedding vector e_i and then applied max pooling on the corresponding items to obtain the click sub-session embedding vector e_{s_c} and the view sub-session embedding vector e_{s_v}. Once the sub-session embedding vector, target item and corresponding user's embedding vectors are ready, a DNN was applied to learn an optimized combination of all these embeddings and, finally, export a representation e_{s_u} of the whole session s_u as the input of the final classification layer. Formally, this can be represented as follows:

$$e_{s_c} = max(e_{i_{c1}}, e_{i_{c2}}, ..., e_{i_{c|s_c|}}) \tag{3.29}$$

where $e_{i_{c1}}$ is the embedding of item i_{c1} in s_c learned from the lower embedding layer. max is to get the maximum value of each dimension from the embeddings of all items in e_{s_c}.

Similarly, the sub-session embedding e_{s_v} is achieved as below:

$$e_{s_v} = max(e_{i_{v1}}, e_{i_{v2}}, ..., e_{i_{v|s_v|}}) \tag{3.30}$$

The output of the first hidden layer of DNN is calculated as below:

$$h_1 = \sigma(\mathbf{W}_c e_{s_c} + \mathbf{W}_v e_{s_v} + \mathbf{W}_t e_{i_t} + \mathbf{W}_u e_u) \tag{3.31}$$

where e_{i_t} and e_u are the embedding of target item and of the corresponding user, respectively; $\mathbf{W}_c$, $\mathbf{W}_v$, $\mathbf{W}_t$ and $\mathbf{W}_u$ are the corresponding weight matrixes to fully connect each of the embedding vector to the first hidden layer of the DNN.

Then, the output of the n^{th} hidden layer h_n is calculated from the previous layer's output:

$$h_n = \sigma(\mathbf{W}^n_{n-1} h_{n-1}) \tag{3.32}$$

where $\mathbf{W}^n_{n-1}$ is the weight matrix to fully connect the $(n-1)^{th}$ hidden layer to the n^{th} hidden layer in the DNN.

The output of the last hidden layer is taken as the latent representation of the original session s_u and it is then inputted into the output layer for classification tasks.

Similar to the above approach, Jannach et al. have applied DNN to learn the optimized combination of different factors like 'reminders', 'item popularity' and 'discount' as compound session-based features for next item predictions (Jannach, Ludewig & Lerche 2017). Another example is the implementation of DNN to transfer and generalize the sparse user-item interactions to dense and informative session features for prediction. A wide liner model combined with a DNN was applied to a mobile phone app store -'Goole play', to improve the app's recommendation performance (Cheng, Koc, Harmsen, Shaked, Chandra, Aradhye, Anderson, Corrado, Chai, Ispir et al. 2016). In (Song et al. 2016), a DNN-based architecture has been proposed to model the combination of long-term static and short-term temporal user preferences to improve the recommendation performance.

•***CNN-Based Models***: To relax the rigid order assumption over items within sessions, a convolutional neural network (CNN) is another choice for session-based recommendations. It is quite common that not all adjacent items in a session have dependent relationships since a given person may buy an irrelevant item (e.g., *pizza*) between two relevant items (e.g. *bread* and *milk*) to form a sub-session ($\langle bread, pizza, milk\rangle$). Such a kind of dependency cannot be well modelled by RNN-based approaches as they only capture adjacent-item-based sequential dependency. In addition, RNN takes each single item as a state, and, thus, it can only capture the item-level dependency, while it cannot capture the union-level dependency. This may ignore the case where several previous items may co-influence the occurrence of the next item. For instance, buying both *milk* and *butter* together leads to a higher probability of buying $flour$. Due to the relaxed assumption on data order and the high capacity in learning local features from a certain area and the relationships between different areas, CNN can effectively mitigate the aforementioned drawbacks of RNN in session-based recommender systems. In the next section, I specifically introduce a representative CNN-based session recommender system (Tang & Wang 2018), and I then demonstrate some other similar works.

Similarly, let a user u's session $S_u = \{i_1, i_2, ...i_l, i_{l+1}, ..., i_{l+m}\}$ contains $l+m$ items, in which the first l items are used as the input to predict the following m items. The model proposed in (Tang & Wang 2018) mainly contains embedding layers, convolutional layers and fully-connected layers. Once the training instances are ready, first the embedding layer maps each input item i to an embedding vector $e_i \in \mathbb{R}^d$. Therefore, an embedding matrix $E^{(u,t)}$ is achieved by putting together of the embeddings of all l input items purchased until time step t:

$$E^{(u,t)} = \begin{bmatrix} e_1 \\ e_2 \\ ... \\ e_l \end{bmatrix} \tag{3.33}$$

The convolutional layers include a horizontal convolutional layer and a vertical convolutional layer, respectively. Specifically, a filtering and a pooling operation are conducted in the horizontal convolutional layer in succession.

Suppose the horizontal convolutional layer has n horizontal filters $F^k \in \mathbb{R}^{h \times d}, 1 \leq k \leq n$. $h \in 1, ..., l$ is the height of a filter. F^k slides from top to bottom on E and interacts with all the horizontal dimensions of the item i, $1 \leq i \leq l+h-1$. Therefore, the i_{th} convolution value is computed as follows:

$$c_i^k = \phi_c(E_{i:i+h-1} \odot F^k) \tag{3.34}$$

where ϕ_c is the activation function for the convolutional layer. The final convolution result of F^k is a vector:

$$c^k = [c_1^k \; c_2^k ... c_{l-h+1}^k] \tag{3.35}$$

After a max pooling is applied on c^k, the final output value from the n filters in this layer is a vector $o \in \mathbb{R}^n$, which captures the most significant features extracted by the filters.

$$o = max\{max(c^1), max(c^2), ..., max(c^n)\} \tag{3.36}$$

A significant signal is picked up w.r.t an area by sliding filters of various heights. Therefore, horizontal filters can be trained to capture union-level patterns with multiple union sizes.

Similarly, suppose the vertical convolutional layer has $\tilde{n}$ filters $\tilde{F}^k \in \mathbb{R}^l, 1 \leq k \leq \tilde{n}$. Each filter interacts with columns of E by sliding d times from left to right on E to get the vertical convolutional output:

$$\tilde{c}^k = \sum_{v=1}^{l} \tilde{F}_v^k E_{v,:} \tag{3.37}$$

Essentially, vertical filter is trained to learn to aggregate the embeddings of the l previous items. Therefore, it can capture the point(item)-level relations. Finally, the output of $\tilde{n}$ vertical filters is a matrix $\tilde{o} \in \mathbb{R}^{d \times \tilde{n}}$:

$$\tilde{o} = [\tilde{c}^1 \; \tilde{c}^2 ... \tilde{c}^{\tilde{n}}] \tag{3.38}$$

In the fully-connected layers, the outputs of the two convolutional filters are concatenated together as the input to get higher-level features through the fully-connected layers.

$$z = \phi_a(\mathbf{W} \begin{bmatrix} o \\ \tilde{o} \end{bmatrix} + b) \tag{3.39}$$

where $\mathbf{W}$ is the weight matrix to project the concatenated layer to a hidden layer, b is the bias term while ϕ_a is the activation function for the fully-connected layers.

Finally, the embedding e_u of user u is concatenated with the output z' of the final fully-connected layer as the input of the final output layer for prediction:

$$y^{(u,t)} = \phi_o(\mathbf{W}' \begin{bmatrix} z' \\ e_u \end{bmatrix} + b') \tag{3.40}$$

where ϕ_o is the softmax function and y is the probability of user u at time step t to interact with each of the candidate items.

Until recently, the session-based recommendation tasks have been addressed by this CNN-based models, which actually learn a comprehensive representation of a session with the complex relations over items encoded. Another similar work is a 3D convolutional network which has been built for session-based recommendations (Tuan & Phuong 2017). Specifically, a 3D CNN model was built to jointly model the sequential patterns in session click data and the item characteristics from item content features for the prediction and recommendation tasks. Furthermore, in (Park, Lee & Choi 2017), a convolutional neural network (CNN) model has been proposed to capture user preferences and to personalise recommendation results.

In summary, different deep network architectures have different strengths and, thus, usually focus on different aspects of session-based recommendations. RNN-based models dominate session recommender systems by capturing the sequential patterns within sessions through utilizing their strong capability for sequence modeling. While DNN-based models are usually applied to learn optimized combinations of the representations of different aspects, and they result in a comprehensive representation of the session-context for the ensuing prediction and recommendation tasks. CNN-based models are usually adopted to extract more informative feature representations from the sessions and other contents for better predictions and recommendations.

3.5 Summary

In this chapter, I have conducted a systematic review on the session-based recommender system. The main trends and approaches were discussed: from the quite naive pattern/rule-based recommender systems used in the very early stages to the currently popular and advanced approaches such as the deep learning-based ones. All these session-based recommender systems can be classified into two main branches: model-free approaches and model-based approaches. Furthermore, the model-free approaches include pattern/rule-based and sequential pattern-based recommender systems, while the model-based ones include Markov chain-based ones, factorization-based ones, and neural model-based ones. In recent years, neural model-based session recommender systems have attracted a substantial amount of attention from the research community, especially the deep learning-based approaches.

In addition, I discussed some different scenario settings in session-based recommender systems in Section 3.2.2. Particularly, three typical scenario pairs were discussed: next-item versus next basket recommendations, ordinary versus unordinary assumptions, and intra- versus inter-session dependency modelling. In practice, I found the works focusing on next-item recommendations to be much more than the next-basket ones. In addition, most works have been based on the ordinary assumption for intra-session dependency modelling. Quite few works have been unorder-based or have taken into account the inter-session dependency for next-item recommendations, which may need more explorations in the future work.

Moreover, most existing session-based recommender systems from various categories have only focused on those most typical issues, while quite few scholars have worked on some more specific issues in this area, such as the implicit relation learning in pattern/rule-based recommender systems, the item heterogeneity issue, the cold start item issue in session-based RSs, and the inter-session dependency issue in next-item recommendations. These gaps certainly motivated me to take a step forward to solve some more specific issues in session-based recommender systems in this thesis.

Chapter 4

Inferring Implicit Rules to Enhance Rule-based Recommendations

4.1 Introduction

In this chapter, I focus on one of the critical challenges on item level in session-based recommender systems: the inter-item dependency, especially the implicit dependency as shown in Figure 1.1 in Chapter 1. This is also the first challenge I addressed in this thesis as demonstrated in the thesis structure Figure 1.2 in the first chapter.

4.1.1 Target Problem and Motivation

As discussed in Chapter 3, especially in Section 3.3.1, rule-based recommender systems are one of the most basic and straightforward approaches for session-based recommendation tasks. Generally, rule-based RSs first mine a set of rules like correlation rules or association rules from the shopping-basket transaction data and then take them as the guidance for the following recommendations. For example, if an association rule $pizza \Rightarrow coke$ indicating that customers who bought a pizza always like to buy a coke is mined from the historical transaction record data of a shopping center, once a customers bought a pizza, the shopping center can recommend a coke to him or her for cross-sale promotion purpose. Note that a rule-based recommender system belongs to the session-based recommender systems because it takes a transaction as the basic unit for data organization in the rule mining process and thus it remains the session structure of the original data in the recommender systems. This is the essential feature of a session-based recommender system as discussed in Section 2.1. Rule-based recommender systems have been proposed for decades and have achieved great success in various application domains including product recommendations(Kim & Yum 2011), discussion forum recommendations (Abel et al. 2008), Web information recommendations (Moreno et al. 2004), music recommendtaions (Shao et al. 2009) and so on.

Although simple and effective, rule-based recommender systems are not flawless, instead, their drawbacks are obvious. For instance, many association rule-based recommender systems usually just recommend those co-occurred items according to their explicit pairwise-like co-occurrence relations while ignoring more complex implicit relations between items like the influence from the third-party items (items except the co-occurred items themselves) (Lin et al. 2002, Kumar & Kumar 2013). The accordingly resultant recommendations based on such simple and straightforward relations may not be so reliable. In addition, without the consideration of effect from the other third-party items, the existing rule-based recommender systems are built on the pairwise-like relations between items or itemsets. As a result, they are easy to recommend similar items duplicately to a user who just bought a certain item as such item may have multiple co-occurred but similar items. For instance, suppose both coke and sprite are highly associated or correlated to pizza according to the transaction data, for a user who bought pizza, both coke and sprite may be recommended to him or her simultaneously. But coke and sprite actually are quite similar and share the same function.

If we go one step deeper, it is clear that all the above mentioned drawbacks originate from the foundation of rule-based recommender systems: the rule they used for recommendations. If a rule-based recommender systems are based on more complex rules with more complicate relations embedded, they are essentially can generate more reliable recommendations. To this end, I first analyze the defects of the existing rule mining works from the rule-based recommender system perspective and then accordingly propose an implicit rule mining framework with the purpose of making up these defects and thus build more reliable implicit rule-based recommender systems.

Rule mining is the foundation and the core challenge of rule-based recommender systems. Classic rule mining methods, e.g., association rule mining (Sahoo, Das & Goswami 2015) and causal rule mining (Sokolova, Groot, Claassen, von Rhein, Buitelaar & Heskes 2015), are essentially based on explicit co-occurrences only, and focus on explicit and dependent relations (e.g., associations, causal relationships) while ignoring more implicit relations (Singh & Jain 2005, Beg & Butt 2009, Cao 2013). Such information loss usually reduces the rules' actionability and the forthcoming recommendation reliability. Next I briefly review and summarize some representative rule mining algorithms which are broadly applied in rule-based recommender systems, like association rule mining, indirect association rule mining, correlation rule mining and casual rule mining.

Well-known algorithms focusing on association rule mining include AIS (Agrawal, Imieliński & Swami 1993), Apriori (Agrawal et al. 1994), FP-Tree (Han, Pei & Yin 2000) and the linear prefix tree-based algorithm (Pyun, Yun & Ryu 2014). All these methods focus on the improvement of algorithm efficiency and much progress has been achieved by utilizing more effective candidate generation methods and pruning strategies. However, they are all based on the *support-confidence* framework and target explicit 'co-occurrence' based associations. As a result, they only capture the explicit and straightforward relations while ignoring implicit relations. They filter out infrequent items which may be of significance and simply focus on the main aspects (antecedent and consequent) while ignoring the influence of other related aspects (e.g., link items). Affected by these defects, the corresponding association rule-based recommender systems are easy to only recommend those frequently co-occurred items while ignoring other ones. Moreover, the neglect of effects from other items reduces the recommendation reliability.

Correlation rule mining is another important branch in the rule mining area. It tries to mine those statistically correlated items driven by frameworks different from the 'support-confidence' one. Specifically, some measures like *lift*, χ^2 *correlation* (Brin, Motwani & Silverstein 1997), *all confidence* and *bond* (Omiecinski 2003) to describe the correlations between distinct items are used as the selection criteria. The progress achieved in correlation rule mining can leverage some of the drawbacks of association rule mining illustrated above by including infrequent items. However, correlation rules still only focus on explicit relations while ignoring implicit ones. In addition, similar to association rules, correlation rules do not take the influence of related aspects (e.g. link items) into account when capturing the relations between the main aspects. Therefore, the corresponding correlation rule-based recommender systems suffer from these drawbacks.

Identifying implicit rules has rarely been explored to the best of my knowledge. The most related work is indirect association rule mining. The concept of indirect association rules infers the relation between two items which are not associated with each other directly but both of them are associated with identical third-party itemsets, called mediators. Hamano and Sato (Hamano & Sato 2004) proposed a framework to mine indirect association rules to analyze targeting consumers and competitors. Specifically, given an item pair which co-occurs infrequently and a mediator itemset, a dependency constraint μ is used to ensure the strong direct associations between each item from the item pair and the mediator itemset, and then the indirect association between the pair of items is derived based on the strong direct associations. Other similar works include the IPMA (Herawan, Noraziah, Abdullah, Deris & Abawajy 2013) framework, mining indirect association mining in web data (Kazienko 2009). All of these algorithms take a step forward in capturing implicit relations; however, they basically extend the association rule mining framework by simply putting two association rules together to derive an indirect one (Wan & An 2003), which limits them to frequent items while filtering out infrequent ones. As a result, they can only discover the indirect associations between two frequent items. However, implicit relations may also exist between several infrequent items and between some infrequent items and frequent ones. Such two kinds of implicit relations involving infrequent items are ignored by all the existing methods. Another drawback of existing broadly used frameworks is that they only focus on the relations between *two* items (e.g., i_1, i_2), which makes them inapplicable for itemsets with more than two items (e.g.,i_1, i_2, i_3...). In addition, most of the existing work in this area only focus on rule mining algorithm development while ignoring the application mechanisms (Wu, Zhu, Wu & Ding 2014) of the resultant rules.

However, it is not trivial to capture *implicit relations* (Yang, Tang, Dai, Yang & Jiang 2013, Peska & Vojtas 2016) by analyzing rule relations (also called pattern relation analysis) (Cao 2013) to make the identified rules actionable (Cao 2012) and informative for better recommendations. In this paper, *implicit relations* (Cao 2013, Chen, Hu, Xu, Liu & Cao 2015) refer to the connections between several items which do not co-occur frequently but have a high probability of co-occurring with the third-party identical items. Here the third-party items are called *link itemset* as they serve as bridges to connect those rarely or never co-occurring items. Such implicit relations cannot be identified by association rule mining or causal rule discovery without pattern relation analysis. In some cases, implicit relations are even more valuable for discovering novel and unexpected rules to support business events like product recommendations, compared to straightforward associations or causal relations. By taking the third-party items (i.e., link items) into account, implicit rules are also more informative than explicit ones, which only focus on their main aspects (e.g., antecedent and consequent).

Researchers have realized the significance of implicit relations between items and have proposed indirect association mining (Hamano & Sato 2004, Herawan et al. 2013). However, it is built on association rule mining (Wan & An 2003), which only makes it applicable for frequent items while ignoring infrequent ones. Furthermore, existing indirect association mining only focuses on pairwise relations (e.g., the relation between sprite and coke) while ignoring the complex relations among multiple items (e.g., the relation among sprite, coke and pepsi). More significantly, they have not been applied in recommender systems and thus there is no implicit or indirect rule-based recommender systems till now to the best of my knowledge.

Taking the ERD data[1] as an example (Table 4.1), the values 0 and 1 in the first and second rows of Column 1 indicate that *pizza* is not bought in transaction t_1 but in t_2. It is easy to infer the implicit rule $coke \oplus sprite | pizza$, which indicates that either *coke* or *sprite*, but not both, is quite likely to be bought with *pizza*. This rule reveals the shopping preferences that *coke* and

[1]An electronic retail transaction dataset from a Chinese E-commerce platform.

Table 4.1: An instance of the ERD dataset

Transactions \ Items	*pizza*	*napkins*	*coke*	*sprite*
t_1	0	1	0	1
t_2	1	1	1	0
t_3	0	1	0	0
t_4	0	1	1	0
t_5	1	1	0	1
t_6	1	1	1	1

sprite may not be usually bought together since they share the same function, whereas they are quite likely to be bought together with the same third-party goods like *pizza*. This kind of implicit connection between *coke* and *sprite* is conditional on the link itemset *pizza*. It can not only help with increasing profit through competitive product analysis (Fleisher & Bensoussan 2015) (*coke* and *sprite* are competitive products) but can also contribute to reliable recommendations by reducing redundant items (*coke* and *sprite* are likely to be redundant if recommended to one consumer at the same time). Such implicit relations cannot be identified by traditional explicit co-occurrence-based rule mining approaches like association rule mining (Agrawal et al. 1994) or causal rule discovery (Pearl, Glymour & Jewell 2016) because of their extremely low co-occurrences and hidden relations between the items involved.

In fact, although some items are implicitly related, it is possible to identify such relationships. For example, a person may buy *pizza* and *coke* for a lunch, but try *pizza* and *sprite* next time. In reality, such partial replacement in product combinations is quite popular in areas such as commerce and medical services. Capturing such implicit and complex relations and then inferring implicit rules helps businesses to deeply understand customer consuming behaviors, which provides more solid support for business optimization and product recommendations (Garcia-Nunes, Souza & da Silva 2017, Leng

& Jiang 2017). Therefore, the rule-based recommender systems could be enhanced by taking those more informative implicit rules rather than those simple and explicit rules as the basis.

4.1.2 My Design and Main Contributions

The above observation shows the importance of counting implicit relations between items in rule-based recommendations and the feasibility of identifying implicit rules composed of infrequently or even never co-occurring items. Here I propose a novel framework to mine implicit rules and then derive implicit rule-based recommender systems.

A 'three-step' framework is proposed to mine implicit rule $x \oplus y|Z$ (meaning items x and y are implicitly related with Z as the link itemset), which is illustrated in Figure 4.1 and explained as follows:

(1) Identify all dependent itemsets of each item in the transactional dataset.

(2) For given items x and y, if they share at least one identical dependent itemset Z, itemset $\{x, y\}$ is chosen as a hidden dependent itemset.

(3) Compute the implicit relation strength (IRS) between x and y; if it is larger than a predefined threshold, itemset $\{x, y\}$ is selected as an implicitly related itemset. Based on this, an implicit rule $x \oplus y|Z$ is inferred.

Once an implicit rule $x \oplus y|Z$ is achieved, it is immediately to be used to derive an implicit rule-based recommendation rule which is indicated in the form of $Z \wedge \neg y \rightarrow x$ ($Z \wedge y \not\rightarrow x$). It means based on the complex information revealed by $x \oplus y|Z$, we follow the following recommendation strategy: item x will be recommended to those customers who bought itemset Z but not y due to the strong relevance between Z and x and the implicit duplicate relation between x and y. Simultaneously, if the customers have already bought Z and y, item x will not be recommended to them. It should be noticed that in order to differentiate our implicit rules from the existing association rules, we use '$\rightarrow$' as the inference symbol instead of using '$\Rightarrow$'.

The main contributions of this work are as follows:

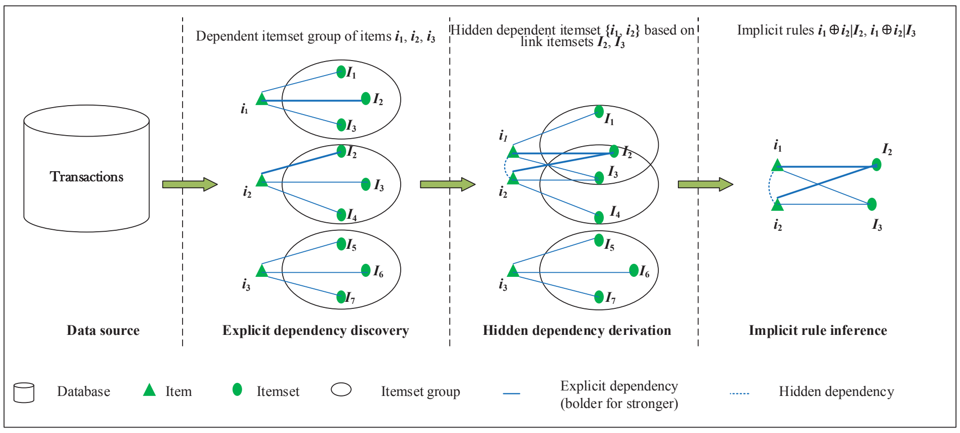

Dependent itemset group of items i_1, i_2, i_3
Hidden dependent itemset $\{i_1, i_2\}$ based on link itemsets I_2, I_3
Implicit rules $i_1 \oplus i_2|I_2$, $i_1 \oplus i_2|I_3$
Transactions
i_1
i_2
i_3
I_1
I_2
I_3
I_4
I_5
I_6
I_7
Data source
Explicit dependency discovery
Hidden dependency derivation
Implicit rule inference
Database
Item
Itemset
Itemset group
Explicit dependency (bolder for stronger)
Hidden dependency

Figure 4.1: The implicit rule inference framework which combines both explicit and hidden dependency

(1) I emphasize the importance of implicit relations in rule-based recommender systems and thus proposed a implicit rule-based recommender system which first inferring implicit rules and then generate recommendations based on them.

(2) A novel *implicit rule inference* framework is proposed to infer implicit relation-based rules (IRR), which follows a three-step strategy. I call the implicit relation-based rules *implicit rules* for simplification in this chapter.

(3) An *implicit rule inference* algorithm, *IRRMiner*, is proposed, by which those items which rarely or never co-occur but are implicitly closely related to each other are detected.

(4) An implicit rule-based recommender system is built on IRRMiner, which effectively enhances the existing rule-based recommender systems by enabling the resultant recommendations to be more reliable and precise in real-world business.

4.2 A Framework for Implicit Rule Inference

I first give the precondition and then illustrate the ‘three-step’ framework discussed in the introduction. The three steps refer to explicit dependency discovery, hidden dependency derivation, and implicit rule inference by integrating both explicit and hidden item dependency.

4.2.1 Precondition

To identify the implicit rules in which items infrequently or never co-occur, the first step is to remove those item combinations of frequently co-occurring items like $\{pizza, coke\}$, i.e., itemsets with high frequency (e.g., $\{pizza, coke\}$). This is because these items are explicitly associated and can be easily and efficiently mined using frequent pattern mining techniques like Apriori, which is out of the scope of this work. Note that frequent items are still kept to form implicit rules. For example, both *coke* and *sprite* are frequent, but they rarely co-occur within one transaction, i.e., $Sup(coke, sprite)$ is low, and they constitute a typical implicit rule. The precondition for a given itemset to be implicitly related is that its support is not larger than a minimum threshold. This precondition greatly benefits my proposed algorithm by pruning those frequently co-occurring items thus reducing the search space.
Precondition 1: It is possible for a given itemset I=$\{i_1 \cdots i_j\}$ to be an implicitly related itemset only if it meets the following precondition:

$$Sup(I) \leq minsup \tag{4.1}$$

where $Sup(I)$ is the support of itemset I and $minsup$ is the predefined minimal support.

4.2.2 Explicit Item Dependency Discovery

Given a transactional dataset as shown in Table 4.1, each row indicates a transaction, such as t_1, t_2, and all the transactions constitute the transactional set T, $T = \{t_1, t_2 \cdots t_{|T|}\}$. Each column indicates an item like pizza and coke while the value 1 means that an item occurs in the corresponding transaction, otherwise the value is 0. All the items in the transactional table constitute the full itemset $U, U = \{i_1, i_2 \cdots i_{|U|}\}$. Each transaction t is a subset of $U, t \subseteq U$. For example, the first transaction t_1 includes two distinct items: napkins and sprite.

With the transactional information, the explicit dependency between an item and itemsets is identified using point-wise mutual information (PMI) (Church & Hanks 1990) for its strong ability to capture both non-linear and linear dependencies (Bouma 2009, Role & Nadif 2011). The point-wise mutual information between item i and itemset I is calculated as:

$$PMI(i, I) = log\frac{p(i, I)}{p(i)p(I)} \tag{4.2}$$

where $p(i)$ and $p(I)$ are the marginal probabilities of i and I respectively, while $p(i, I)$ is their joint probability.

Definition 4.1 (Dependent itemset)**.** *An itemset I is defined as a dependent itemset of a given item $i (i \notin I)$ (denoted as $S_i = I$) if the PMI between them is positive, that is, $PMI(i, I) > 0$.*

It is denoted as $S_{i1}, S_{i2}, \cdots$ if item i has more than one dependent itemset. Note that generally $PMI(i, I) \in (-\infty, min[-logp(i), -logp(I)]]$, however $PMI(i, I) \in (0, min[-logp(i), -logp(I)]]$ in my algorithm to ensure the positive dependency between i and I.

Based on the dependent itemset concept, the dependent itemset group is defined as follows.

Definition 4.2 (Dependent itemset group)**.** *For a given item i, all its dependent itemsets $(S_{i1}, S_{i2} \ldots)$ constitute its dependent itemset group, denoted as $\mathbf{A}_i$:*

$$\mathbf{A}_i = \{S_{i1}, S_{i2} \cdots\} \tag{4.3}$$

Example 4.1. *Taking item coke in Table 4.1 as an example, its dependent itemset group $\mathbf{A}_{coke} = \{S_{coke1}, S_{coke2}\} = \{\{pizza\}, \{pizza, napkins\}\}$.*

4.2.3 Hidden Item Dependency Derivation

Given an itemset $I = \{i_1, i_2 \cdots i_j\}$, the dependent itemset group $\boldsymbol{A}_{i_1}, \boldsymbol{A}_{i_2} \cdots \boldsymbol{A}_{i_j}$ of each item from I is respectively identified. $\boldsymbol{A}_{i_j} = \{S_{i_j1}, S_{i_j2} \cdots S_{i_jk}\}$, where S_{i_jk} is the k^{th} dependent itemset of i_j.

Definition 4.3 (Link itemset and link itemset group)**.** *Given an itemset $I = \{i_1, i_2 \dots i_j\}$ and the dependent itemset group $\mathbf{A}_{i_j}$ of each item i_j from I, the link itemset group of I is defined as the intersection set of all dependent itemset groups of items within I, denoted as $\mathbf{G}_I$. Each element of $\mathbf{G}_I$ is defined as a link itemset of I, denoted as H_I. It is denoted as $H_{I1}, H_{I2} \dots$ when itemset I has more than one link itemset. Formally,*

$$\mathbf{G}_I = \mathbf{A}_{i_1} \cap \mathbf{A}_{i_2} \dots \mathbf{A}_{i_j} = \{H_{I1}, H_{I2} \dots H_{I|\mathbf{G}_I|}\} \tag{4.4}$$

Definition 4.4 (Hidden dependent itemset)**.** *Given an itemset I, it is defined as hidden dependent if its link itemset group is not empty. Formally,*

$$\mathbf{G}_I \neq \emptyset \tag{4.5}$$

Example 4.2. *Let us take itemset $\{coke, sprite\}$ from Table 4.1 as an example. The dependent itemset groups of items coke and sprite are $\mathbf{A}_{coke} = \{\{pizza\}, \{pizza, napkins\}\}$ and $\mathbf{A}_{sprite} = \{\{pizza\}, \{pizza, napkins\}\}$, respectively. Hence, the link itemset group of $\{coke, sprite\}$ is $\mathbf{G}_{\{coke,sprite\}} = \mathbf{A}_{coke} \bigcap \mathbf{A}_{sprite} = \{\{pizza\}, \{pizza, napkins\}\} \neq \emptyset$. Accordingly, $\{coke, sprite\}$ is a hidden dependent itemset with two link itemsets $\{pizza\}$ and $\{pizza, napkins\}$.*

4.2.4 Implicit Rule Inference

Given a hidden dependent itemset, I first compute its *IRS* and then select those itemsets whose *IRS* is larger than a minimum threshold as implicitly related itemsets. Lastly, I infer implicit rules based on these itemsets.

Given a hidden dependent itemset $I = \{i_1, i_2 \cdots i_j\}$ together with its link itemset group $\boldsymbol{G}_I (\boldsymbol{G}_I \neq \emptyset)$, its *IRS* is calculated under the intuition that if I has more link itemsets and the items $(i_1, i_2 \cdots i_j)$ within I have stronger dependencies on them, these items are more strongly implicitly connected. As a result, the IRS of I is larger.

Definition 4.5 (Conditional implicit relation strength (CIRS)). *Given a hidden dependent itemset $I = \{i_1, i_2 \cdots i_j\}$ and a link itemset H_I, its implicit relation strength conditional on H_I is computed as:*

$$CIRS(I|H_I) = min(PMI(i_1, H_I) \cdots PMI(i_j, H_I)) \tag{4.6}$$

where $CIRS(I|H_I) \in (0, min(-logp(i_1), ... - logp(i_j), -log(H_I))$.

Definition 4.6 (Implicit relation strength (IRS)). *Given an hidden dependent itemset $I = \{i_1, i_2 \cdots i_j\}$ and its link itemset group $\mathbf{G}_I$, its implicit relation strength is computed by summing its CIRS on all link itemsets. Formally,*

$$IRS(I) = \sum_{H_I \in \mathbf{G}_I} CIRS(I|H_I) \tag{4.7}$$

where $IRS(I) \in (0, \sum_{H_I \in \mathbf{G}_I} min(-logp(i_1), ... - logp(i_j), -log(H_I))$. The larger $IRS(I)$ is, the stronger the implicit relation that exists between the items within I.

Definition 4.7 (Implicitly related itemset). *An implicitly related itemset candidate is implicitly related if its IRS is larger than the minimum threshold. Formally,*

$$IRS(I) \geq minIRS \tag{4.8}$$

where $minIRS$ is a predefined threshold to ensure that strong enough implicit relations exist between the items within I.

Definition 4.8 (Implicit rules). *Given an implicitly related itemset $I = \{i_1, i_2 \cdots i_j\}$ and its link itemset group $\mathbf{G}_I = \{H_{I1} \cdots H_{In}\} (n = |\mathbf{G}_I|)$, n implicit rules are inferred, which constitute an implicit rule cluster $\mathbf{R}_0$.*

$$\mathbf{R}_0 \begin{cases} r_{01} : i_1 \oplus i_2 \oplus \cdots i_j | H_{I1} \\ r_{02} : i_1 \oplus i_2 \oplus \cdots i_j | H_{I2} \\ \cdots \\ r_{0n} : i_1 \oplus i_2 \oplus \cdots i_j | H_{In} \end{cases} \tag{4.9}$$

All the rules from $\boldsymbol{R}_0$ share the same implicitly related itemset ($\{i_1, i_2...i_j\}$) but take different link itemsets (e.g., H_{I1} or H_{I2}) as their conditions. The first rule $i_1 \oplus i_2 \oplus \cdots \oplus i_j|H_{I1}$ implies that once H_{I1} has been bought, there is great probability that one out of $i_1, i_2 \cdots i_j$ will be bought.

Example 4.3. *Following the implicitly related itemset candidate $\{coke, sprite\}$ in Example 4.2, its conditional implicit relation strength (CIRS) conditional on its link itemset $\{pizza\}$ is $CIRS(\{coke, sprite\}|\{pizza\}) = min\ (PMI(coke, \{pizza\}), PMI(sprite, \{pizza\})) = 0.12$. Similarly, $CIRS(\{coke, sprite\}|\{pizza, napkins\}) = 0.12$. Accordingly, $IRS(\{coke, sprite\}) = 0.24$. If we set $minIRS = 0.1$, $\{coke, sprite\}$ is an implicitly related itemset. Based on this, two implicit rules $coke{\oplus}sprite|pizza$ and $coke{\oplus}sprite|\{pizza, napkins\}$ are derived. In reality, such rules indicate that coke and sprite are rarely bought together whereas they are much more likely to be bought together with other itemsets $\{pizza\}$ or $\{pizza, napkins\}$. This can be seen from the transactions in Table 4.1. These observations are consistent with customer shopping behaviors whereby one may prefer to buy a basket of products with different functions rather than the same function.*

4.3 The IRRMiner Algorithm

Following the framework illustrated in the previous section, the *IRRMiner* algorithm is developed to mine implicit rules as shown in Algorithms 4.1 and 4.2. The following anti-monotonous Property 4.1 is used to generate size-L implicit itemset candidates from size-(L-1) ones directly to reduce the search space. The size of an implicit rule is defined in Equation (4.10) in Section 4.4. Applying this property from Lines 2 to 5 in Algorithm 4.2 guarantees to find all rules satisfying the given constraints efficiently. Next, I first give and prove such a property theoretically and then describe the implicit rule inference algorithm below.

Property 4.1. *Given a candidate itemset $I = \{i_1 \cdots i_j\}$ and its implicit relation strength ($IRS(I)$), any subset $I'(I' \subseteq I, |I'| \geq 2)$ must not have a* *lower implicit relation strength, namely $IRS(I^{'}) \geq IRS(I)$.*

Proof. Given a candidate itemset $I = \{i_1, i_2 \cdots i_l \cdots i_j\}(|I| \geq 3)$ and one of its subset $I^{'} = \{i_1, i_2 \cdots i_l\}(l<j, |I^{'}| \geq 2)$, according to Definition 4.3, their link itemset groups are $\boldsymbol{G}_I = \boldsymbol{A}_{i_1} \cap \boldsymbol{A}_{i_2} \ldots \boldsymbol{A}_{i_l} \ldots \boldsymbol{A}_{i_j}$ and $\boldsymbol{G}_I^{'} = \boldsymbol{A}_{i_1} \cap \boldsymbol{A}_{i_2} \ldots \boldsymbol{A}_{i_l}$ respectively. So $\boldsymbol{G}_I = \boldsymbol{G}_I^{'} \cap \boldsymbol{A}_{i_{l+1}} \ldots \boldsymbol{A}_{i_j} \subseteq \boldsymbol{G}_I^{'}$. Assume $\boldsymbol{G}_I = \{H_1, H_2 ... H_k\}$ while $\boldsymbol{G}_I^{'} = \{H_1, H_2 ... H_k ... H_m\}(m>k)$, based on Definitions 4.5 and 4.8, $IRS(I) = CIRS(I|H_1) + CIRS(I|H_2) + ...CIRS(I|H_k)$ while $IRS(I^{'}) = CIRS(I^{'}|H_1)+CIRS(I^{'}|H_2)+...CIRS(I^{'}|H_k)+...+CIRS(I^{'}|H_m)$. Now we compare $CIRS(I|H_1)$ and $CIRS(I^{'}|H_1)$, $CIRS(I|H_1) = min\{PMI (i_1, H_1), PMI(i_2, H_1)...PMI(i_l, H_1), PMI(i_{l+1}, H_1)...PMI(i_j, H_1)\}$ and $CIRS(I^{'}|H_1) = min\{PMI(i_1, H_1), PMI(i_2, H_1)...PMI(i_l, H_1)\}$, it's clear that $CIRS(I|H_1) = min\{min\{PMI(i_1, H_1), PMI(i_2, H_1)...PMI(i_l, H_1)\}, min\{PMI(i_{1+1}, H_1)...PMI(i_j, H_1)\}\} = min\{CIRS(I^{'}|H_1), min\{PMI(i_{1+1}, H_1)...PMI(i_j, H_1)\}\} \leq CIRS(I^{'}|H_1)$. Similarly, $CIRS(I|H_2) \leq CIRS(I^{'}| H_2), ...CIRS(I|H_k) \leq CIRS(I^{'}|H_k)$, hence $IRS(I) \leq CIRS(I^{'}|H_1)+CIRS (I^{'}|H_2)+...CIRS(I^{'}|H_k)$. Recall that $CIRS(I|H_I)>0$ as illustrated in Definition 4.5, it is easy to conclude that $IRS(I) \leq CIRS(I^{'}|H_1)+CIRS(I^{'}|H_2)+ ...CIRS(I^{'}|H_k) + ... + CIRS(I^{'}|H_m) = IRS(I^{'})$. Hence, Property 4.1 is proved. □

Combining Property 4.1 and Definition 4.7, it is easy to conclude that any subset $I^{'}(I^{'} \subseteq I, |I^{'}| \geq 2)$ of an implicitly related itemset I is also an implicitly related itemset if $I^{'}$ meets the precondition $Sup(I^{'}) \leq minsup$. Such conclusion not only reduces the search space in the candidate generation process but also helps to identify whether a size-L candidate itemset is implicitly related or not by checking all its size-(L-1) subsets. This contributes a lot to the space and time efficiency of the whole *IRRMiner* algorithm.

Overall, the implicit rule inference algorithm is divided into two stages, which are described in Algorithms 4.1 and 4.2 respectively. Algorithm 4.1 is used to mine Size-2 implicitly related rules and has been divided into two parts: (1) Discover dependent itemset groups (Line 1); (2) Mine size-2 implicit rules and prepare to mine implicit rules of larger sizes (Lines 2 to

Algorithm 4.1 Mine Size-2 Implicit Rules

```
Require: $T$ : binary transaction matrix; $minsup$ : support threshold; $minIRS$ : IRS
    threshold;
Ensure: $P$ : implicitly related itemsets; $H$ : link itemsets;
 1: Mine dependent itemset group $A_i$ of each item $i$ according to Definition 4.2 and store
    items with non-empty $A_i$ in D;
 2: while L=2 do
 3:    Generate all possible size-L itemsets on D and store them in $Q\{L\}$;
 4:    for each itemset $I \in Q\{L\}$ do
 5:       if $\boldsymbol{G_I} \neq \emptyset$ then
 6:          Select $I$ as size-L hidden dependent itemset and store it in $C\{L\}$;
 7:       end if
 8:    end for
 9:    for each itemset $I_c$ in $C\{L\}$ do
10:       if $IRS(I_c) \geq minIRS$ then
11:          Store $I_c$ in $CC\{L\}$;
12:          if $Sup(I_c) \leq minsup$ then
13:             Select $I_c$ as implicitly related itemset and store it in $P\{L\}$, and store all its
                link itemsets in $H\{L\}$;
14:          end if
15:       end if
16:    end for
17: end while
```

17). Specifically, all dependent itemsets are identified and possible size-2 itemsets are generated on those items whose dependent itemset group is not empty (Lines 1 to 3). Then the link itemset group, implicit relation strength (IRS) and support of these generated itemsets are checked one by one to filter out those non-potential implicitly related itemsets step by step while keeping implicitly related itemsets together with their corresponding link itemsets as the output (Lines 4 to 17). When the size of an itemset grows larger than 2, Algorithm 4.2 is utilized. The anti-monotonous Property 4.1 is used to generate larger candidate itemsets more efficiently based on the pattern growth method (Lines 2 to 5) and to conduct pre-filtering on these itemsets (Lines 9 to 10). Finally, the implicitly related itemsets and their

Algorithm 4.2 Mine Implicit Rules Larger Than 2

```
Require: $CC2$ : Size-2 implicitly related itemset candidates; $minsup$ : support threshold; $minIRS$ : IRS threshold; $MaxSize$: the maximum size of implicitly related itemsets;
Ensure: $P$ : implicitly related itemsets; $H$ : link itemsets;
1: while L=3:$MaxSize$ do
2:   for $m = 1 : Size(CC\{L-1\})$ do
3:     for $n = m+1 : Size(CC\{L-1\})$ do
4:       if $CC\{L-1\}(m)([1 : L-2]) == CC\{L-1\}(n)([1 : L-2])$ and $CC\{L-1\}(m)(L-1) \neq CCL-1(n)(L-1)$ then
5:         Store $[CC\{L-1\}(m), CC\{L-1\}(n)(L-1)]$ into $Q\{L\}$ as size-L itemset ;
6:       end if
7:     end for
8:   end for
9:   for each size-L itemset $I$ in $Q\{L\}$ do
10:    if all size-(L-1) sub-itemsets of $I$ are in $CC\{L-1\}$ then
11:      Execute the same operations from Lines 4 to 16 in Algorithm 4.1 ;
12:    end if
13:  end for
14: end while
```

link itemsets are achieved by undertaking the same filtering operations (Line 11) as those used in the mining of size-2 implicit rules.

4.4 Experimental Evaluation of IRRMiner

4.4.1 Experiment Set Up

No existing work can exactly mine my proposed implicit rules, to the best of my knowledge, and only a typical indirect association rule mining algorithm (IARMiner) (Hamano & Sato 2004) can partially discover rules similar to mine. The rules mined by IARMiner are in the form of $(M; \{x, y\})$ where itemset M is the mediator itemset for connecting items x and y. These rules can be transferred to my implicit rules, such as $x \oplus y|M$. To evaluate my proposed algorithm, I compare my proposed IRRMiner with the representa-

Table 4.2: Statistics of experimental datasets

Statistics	ERD	Bookcross	MovieLens_1	MovieLens_2
No. of Transactions	4626	10938	69878	69878
No. of Items	72	162	200	200
Avg. Items per Transaction	12.78	3.58	34.05	3.43
Avg. Frequency per Item	820.81	241.3	11895	1199
Density	20.56%	2.21%	17.02%	1.72%

tive indirect association rule mining algorithm IARMiner on four real-world transactional datasets: ERD, Bookcross[2], MovieLens_1 and MovieLens_2. Such comparison has some limitations due to the non-exactly identical goals of the compared algorithms. Specifically, I can only make a comparison on the capability of mining size-2 rules instead of larger ones (Rules of size-3 and size-4) between IRRMiner and IARMiner as IARMiner can only mine size-2 rules; the mined rules may not always be completely identical as the constraints used in IARMiner and IRRMiner are not completely the same. However, empirical results show that most of the resultant size-2 rules from both algorithms are the same. MovieLens_1 and MovieLens_2 are extracted from the MovieLens 10M[3] dataset by including different parts of transactions. A detailed description of these datasets is given in Table 4.2. Items, books and movies in the experimental datasets are called items uniformly in this work to simplify the terms. Note that all the four transactional datasets are transferred into 0-1 encodings as Table 4.1 and the density shown in Table 4.2 is quantified by $Density = \frac{\#entries\ valued\ 1}{\#entries}$, for instance, the density of transaction Table 4.1 is $\frac{15}{24}$=62.5%.

Both IRRMiner and IARMiner have two key parameters: *minsup* (called t_r in (Hamano & Sato 2004)) is shared by the two algorithms while $minIRS$ and t_μ are used in IRRMiner and IARMiner respectively. To be specific, *minsup* is a frequency-constraint to ensure the implicitly related or indi-

[2]Available on http://grouplens.org/datasets/book-crossing/

[3]Available on http://grouplens.org/datasets/movielens/

rect associated items do not co-occur frequently (e.g., $sup(x, y) \leq minsup$). $minIRS$ is used to guarantee strong implicit relation strength (IRS) between implicitly related items (e.g., $IRS(x, y) \geq minIRS$) in IRRMiner while t_μ is to ensure strong dependency between each of the indirectly associated items and the corresponding mediate itemset M (e.g., $\mu(x, M) \geq t\_\mu$). In addition, two extra parameters t_f and t_m are also used in IARMiner, where t_f is to make sure each item in an indirectly associated item pair is frequent (e.g., $sup(x) \geq t\_f$) and t_m is to guarantee that it co-occurs with the mediate itemset frequently (e.g., $sup(x, M) \geq t\_m$). In all the experiments, I keep the common parameter $minsup$(t_r) identical for both algorithms to ensure fair comparisons while empirically tuning other non-common parameters.

In order to show the capacity of IRRMiner to cover infrequent items and to mine implicit rules larger than size-2, I conduct comparisons between IARMiner and IRRMiner in terms of rule coverage, rule size and rule number in the following Sections 4.4.2 and 4.4.3, respectively. To test the efficiency of my proposed IRRMiner, I compare the run time of IRRMiner and that of IARMiner in the following Section 4.4.4. A data factor test is conducted in Section 4.4.5 to test the outcome difference of my proposed IRAMiner on datasets with different characteristics.

4.4.2 Rule Coverage Comparison

Nearly all the indirect rule mining approaches including IARMiner can only mine size-2 rules, to make a fair comparison, I also limit the size of rules from IRRMiner to 2 when comparing rule coverage. The coverage of the size-2 rules resulting from IARMiner and IRRMiner together with the average frequency of their covered items are given in the two sub-figures in Figure 4.2 respectively. Here *coverage* is defined as the ratio of items covered by all the mined rules w.r.t the whole item population, while the frequency of a certain item is its occurrence times divided by the total number of transactions in a dataset. On one hand, the left hand side sub-figure in Figure 4.2 shows that the coverage of IRRMiner is obviously larger than

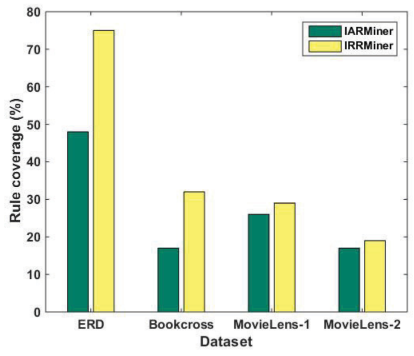
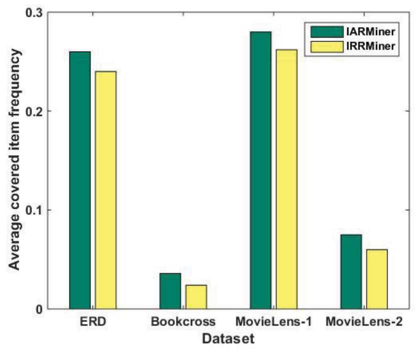



Figure 4.2: Rule coverage and average covered item frequency on different datasets

IARMiner on all the four datasets, which means my proposed algorithm can discover implicit relations between more items than IARMiner. On the other hand, the right hand side sub-figure in Figure 4.2 illustrates that the average frequency of the covered items by IRRMiner is clearly lower than that by IARMiner, which means my algorithm can discover implicit relations between more infrequent items. Combining these two figures, it is easy to conclude that IRRMiner discovers not only the implicit rules between frequent items as the existing indirect association mining approaches do, but also implicit rules between infrequent items. The reason behind this is easy to find by looking at the algorithm design, which is different from most indirect association mining algorithms, which are limited to frequent items only, due to their base (frequent association mining). However, my algorithm goes beyond such a base, and it is not necessary for the items to be frequent.

4.4.3 Rule Size and Number Comparison

To make a fair comparison, the corresponding constraints in IARMiner and IRRMiner are set to be equivalent to each other. Specifically, the common parameter $minsup(t\_r)$ is set to 10%, 1%, 15% and 1.5% empirically on ERD, Bookcross, MovieLens_1 and MovieLens_2 respectively both in IARMiner and IRRMiner algorithms. Both $minIRS$ and t_μ are set to 0 as it is only in this case that these two constraints are essentially equivalent to each other as proved below.

Proof. Suppose implicitly related itemset $I = \{x, y\}$, itemset M is a link itemset to connect x and y. According to Equations (4.2), (4.6), (4.7) and (4.8), $minIRS = 0$ indicates $IRS(x, y) \geq 0$, which means there exists at least one link itemset H_I which makes $CIRS(I|H_I) \geq 0$. Suppose $CIRS(I|M) \geq 0$, according to Equation (4.6), $PMI(x, M) \geq 0$ and $PMI(y, M) \geq 0$, namely $p(x, M) \geq p(x)p(M)$ and $p(y, M) \geq p(y)p(M)$. On the other hand, according to the definition of $\mu(x, M)$ in (Hamano & Sato 2004), $t\_\mu = 0$ means $\mu(x, M) = \frac{p(Mx)\ -\ p(M)p(x)}{p(Mx)(1-p(x))} \geq 0$, which also indicates $p(x, M) \geq p(x)p(M)$, for the same reason, $p(y, M) \geq p(y)p(M)$. So the key constraints $IRS(x, y) \geq minIRS$ in IRRMiner and $\mu(x, M) \geq t\_\mu$ in IARMiner are actually the same when their thresholds are set to 0. $\square$

The size of an implicit rule 'r: $i_1 \oplus i_2 \oplus \cdots i_j | H_I(I = \{i_1, i_2 \cdots i_j\})$' is defined as the size of itemset I, which measures how many items are implicitly related conditioned on H_I. Formally,

$$Size(r) = |\{i_1, i_2 \cdots i_j\}| \tag{4.10}$$

Table 4.3 shows the number of indirect association rules mined by IARMiner (IAR. for short) and implicit rules mined by IRRMiner (IRR.). Two main conclusions can be drawn from this: (a) IRRMiner can mine more size-2 rules than IARMiner (i.e., 4,243 vs. 2,226 on the ERD dataset). By checking the rules more deeply, I find the rules mined by IARMiner are a subset of rules from IRRMiner. This is because IRRMiner targets not only the implicit rules between frequent items but also the infrequent ones, as stated in the introduction. Therefore, IRRMiner can cover more items and generate more rules, which is consistent with the results of the coverage comparison in Section 4.4.2. (b) IRRMiner can mine implicit rules with a size larger than 2 while IARMiner cannot. Theoretically, IRRMiner can output rules as large as the number of items in the transaction as long as the dataset supports such rules. Note that rules with a size larger than 4 are not shown due to space limitations. Thanks to the design behind IRRMiner, which considers the implicit relations among all possible items rather than just the indirect association between a pair of frequent items as IARMiner does, the complex implicit relations among multiple items discovered by IRRMiner are more general and more consistent with the real-world cases compared to the indirect association rules of size-2. In summary, IRRMiner goes far beyond IARMiner by returning more rules of larger sizes. Implicit rules of a large size reveal much more hidden information between multiple items than size-2 rules which only reflect pairwise relations between every two items. Taking a sample from the ERD dataset as an example, the triad implicit relation among three books 'An Introduction to Secondary Data Analysis', 'Python Data Science Handbook' and 'Microsoft Excel 2013 Data Analysis' is richer than the pair-wised indirect association between any two of them.

Essentially, the indirect association rule mining framework is a special case of my proposed framework. When I only focus on the implicit relations between frequent items and limit the rule size to 2, my framework is simplified to the existing indirect association mining one, and can mine the same rules as indirect association mining does.

Table 4.3: Number of mined rules by IARMiner (IAR.) and IRRMiner (IAR.)

		Size 2	Size 3	Size 4
ERD	IAR.	2226	-	-
	IRR.	4243	50405	336216
Bookcross	IAR.	82	-	-
	IRR.	156	221	342
MovieLens_1	IAR.	8495	-	-
	IRR.	15219	203656	1926018
MovieLens_2	IAR.	908	-	-
	IRR.	1689	20792	109800

4.4.4 Run Time Comparison

To evaluate the efficiency of the developed IRRMiner algorithm, two sets of experiments are organized. One is to compare the run time of IARMiner and IRRMiner to mine the same size-2 rules, in which both the rules and rule numbers resultant from both algorithms are exactly the same. The other is to compare the run time of both algorithms to mine the same number of size-2 rules, where the rules may not be completely identical.

Recall that IRRMiner can be simplified to indirect association mining algorithms and can mine the same rules as them, as discussed in the last paragraph in Section 4.4.3. I add an extra frequency constraint on IRRMiner and limit the rule size to 2 to make sure it only discovers the same rules or the same number of rules as IARMiner. I keep the values of corresponding parameters the same in both algorithms for a fair comparison, namely $minIRS$ and $minsup$ in IRRMiner are equal to t_μ and $minsup$ in IARMiner respectively. In addition, I keep both $minIRS$ and t_μ unchanged ($minIRS = t\_\mu = 0$, according to the proof in Section 4.4.3) while change $minsup$ in two algorithms synchronously in the first set of experiments; in the second set of experiments, the $minsup$ in IARMiner and IRRMiner are kept equal and unchanged (empirically 10% in ERD, 1% in Bookcross, 15% in MovieLens_1 and 1.5% in MovieLens_2 for both algorithms) while $minIRS$ in IRRMiner and t_μ in IARMiner are adjusted accordingly to mine the same number of rules. The results of these two sets of experiments are given in Tables 4.4 and 4.5 respectively, in which the symbol '*' represents the time spent by IARMiner under certain t_μ values. Note that in the second set of experiments, to achieve the identical number of rules, t_μ in IARMiner does not necessarily need to be equal to $minIRS$ in IRRMiner.

It is clear that, in mining either the same implicit size-2 rules in Table 4.4 or the identical number of size-2 implicit rules in Table 4.5, my proposed IRRMiner is much more efficient than IARMiner. The run time is reduced by around 80% on the ERD dataset and around 90% on the other three datasets by IRRMiner, compared to IARMiner.

Table 4.4: Run time (in second) under different $minsup(m.s.)$

ERD			Bookcross		
	IAR.	IRR.		IAR.	IRR.
$m.s.$=6%	9.4	**2.4**	$m.s.$=0.6%	474.5	**5.31**
$m.s.$=10%	16.6	**2.6**	$m.s.$=1%	511.6	**5.21**
$m.s.$=14%	20.8	**2.8**	$m.s.$=1.4%	518.6	**5.18**
MovieLens_1			MovieLens_2		
	IAR.	IRR.		IAR.	IRR.
$m.s.$=13%	1130	**24.5**	$m.s.$=1.3%	600	**5.74**
$m.s.$=15%	1162	**24.6**	$m.s.$=1.5%	641	**5.75**
$m.s.$=17%	1192	**25**	$m.s.$=1.7%	700	**5.79**

Table 4.5: Run time (in second) under different $minIRS(m.I.)$

	ERD		Bookcross		MovieLens_1		MovieLens_2	
	IAR.	IRR.	IAR.	IRR.	IAR.	IRR.	IAR.	IRR.
$m.I.$=0	16.6*	**2.6**	511.6*	**5.21**	1162*	**24.6**	641*	**5.75**
$m.I.$=0.8	15.5*	**2.4**	510.2*	**5.2**	1080*	**24**	552*	**5.66**
$m.I.$=1.6	13.8*	**2.3**	481.7*	**5.17**	964*	**23.3**	492*	**5.63**

One main contribution to efficiency improvement is the first step (explicit dependency discovery) in my proposed IRRMiner, which only exists in my algorithm. It checks whether an item has dependent itemsets; items without dependent itemsets are not considered in the subsequent steps. Many items without dependent itemsets but are frequent are filtered out in an early stage. However, such items cannot be removed in the beginning of IARMiner which uses a support threshold to filter out non-frequent items in its first step. This partly explains why IRRMiner is clearly more efficient than IARMiner.

The time complexity analysis of the proposed Algorithm 4.1 is detailed in this paragraph. Assume the total number of items in the transactional matrix is N. Given an implicit rule mining task, the total process of the IRRMiner algorithm is divided into two parts: the preparation stage illustrated by Line 1 and the implicit rule mining stage described from Line 2 to the end of the algorithm. Please note that the preparation stage is a pre-processing operation, once it is ready, various sizes of implicit rules under different $minIRS$ values can be mined without the need to conduct the preparation stage again, which means it is not necessary to run Line 1 every time when we mine implicit rules. Hence, the run time of IRRMiner mainly depends on the second stage. Specifically, suppose a percentage of α of all the N items have dependent itemsets, so $C^2_{\alpha N}$ possible size-2 itemsets will be generated in Line 2, which result in $C^2_{\alpha N} = \frac{(\alpha N)*(\alpha N-1)}{2}$ times of computation from Lines 3 to 7, accordingly, the time complexity of these lines is $O((\alpha N)^2)$. Meanwhile, at most $C^2_{\alpha N}$ hidden dependent itemsets will be selected in Line 5, which results in the maximum computational times being also $C^2_{\alpha N}$ from Lines 8 to 15. The time complexity of Lines 8 to 15 is $O((\alpha N)^2)$ too. Overall, the time complexity of the implicit rule mining stage is $O((\alpha N)^2)$ in mining size-2 implicit rules. Similarly, the time complexity of mining size-3, size-4, ..., etc. implicit rules in Algorithm 4.2 are $O((\alpha N)^4)$, $O((\alpha N)^8)$, ..., etc. For the preparation stage, when I set the maximum length of link itemset to 1, 2, 3, etc., the time complexity is $O((\alpha N))$, $O((\alpha N)^2)$, $O((\alpha N)^4)$, etc.. In IARMiner, the time complexity is $O((\beta N)^3)$, $O((\beta N)^4)$, $O((\beta N)^6)$, etc., when the maximum length of the mediate itemset is set to 1, 2, 3, etc., where parameter β is the percentage of frequent items w.r.t all items.

To summarize, when mining size-2 rules, the time complexity of IRRMiner is $O((\alpha N)^2)$, compared to $O((\beta N)^3)$, $O((\beta N)^4)$, $O((\beta N)^6)$, etc., of IARMiner if I do not consider the preparation stage of IRRMiner. Even if I take the preparation into consideration, the time complexity of IRRMiner is $O((\alpha N)^2)$, $O((\alpha N)^2)$, $O((\alpha N)^4)$ compared to $O((\beta N)^3)$, $O((\beta N)^4)$, $O((\beta N)^6)$ of IARMiner respectively by setting the maximum length of the dependent itemsets in IRRMiner to the same as that of the mediate itemset in IARMiner. Accordingly, IRRMiner reduces the time complexity of $O((N))$ and $O(N^2)$ respectively when I set the maximum length of link itemset to 1 and larger than 1 with the consideration of preparation. This explains why IRRMiner is always much more efficient than IARMiner, especially in a dataset with a large number of items. This is consistent with the empirical results in Tables 4.4 and 4.5.

4.4.5 Data Factor Test

To test the performance of the proposed algorithm on datasets of different characteristics, I conduct data factor test. Specifically, two data factors: density (D) and the total number of items (N) are selected. Recall the number of possible size-2, size-3, etc., itemsets are $C^2_{\alpha N}$, $C^3_{\alpha N}$, etc., respectively, as illustrated in the fourth paragraph of Section 4.4.4, it is obvious that α and N can substantially affect the number of mined implicit rules by firstly deciding the number of possible generated itemsets and candidate itemsets. Furthermore, α is greatly affected by data density, because the items in a dense dataset are more likely to be dependent on each other than those in a sparse one. In other words, data density closely relates to the number of resultant rules via α.

To make a fair comparison, when one data factor is tested, I ensure the other data factor is identical on all the datasets by conducting necessary processing on them. For example, when testing the effect of density, the number of items in the three datasets used in the following density test is kept the same.

Density Test

I test the effect of dataset density on the number of implicit rules mined by IRRMiner by running it on three real-world datasets with various density degrees but with the same number of items. The number of obtained rules is shown in the left hand-side sub-figure of Figure 4.3.

It is quite obvious that under the same experimental settings, the number of rules is significantly influenced by the data density. The left hand-side sub-figure of Figure 4.3 shows that the denser a dataset is, the more rules are obtained. The number of rules of all sizes for the dataset with a density of 20.6% is much larger than that for the dataset with density of 10.3%, the latter is also much larger than that on the dataset with density of 5.2%. In addition, it is much more likely that larger rules will be obtained for denser datasets, for instance, only the densest dataset ($D = 20.6\%$) results in rules of size larger than 9. This is consistent with the statement in the first paragraph of this subsection that items in a dense dataset are more likely to depend on each other and lead to a higher α for the dataset, and produce more rules.

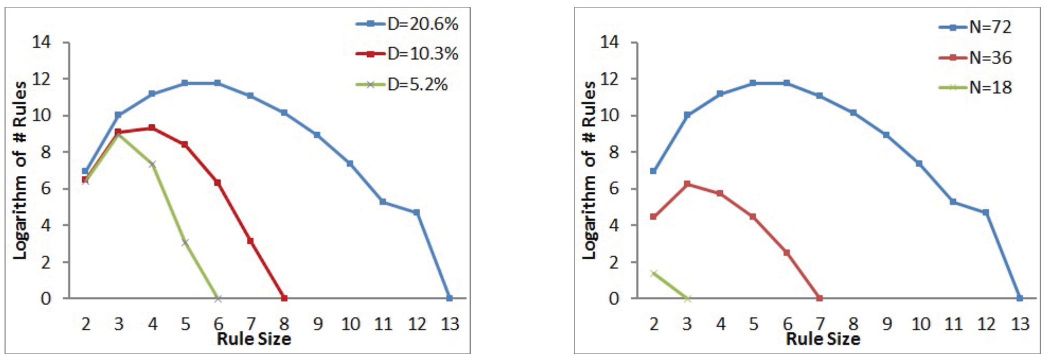


Figure 4.3: Rule number comparisons w.r.t different data density degrees (D) or item numbers (N)

Item number Test

I test the effect of item number (N) on the number of implicit rules mined by IRRMiner by running it on three real-world datasets with different number of items but with the same density. The number of rules obtained is shown in the right hand-side sub-figure of Figure 4.3. The results are also consistent with the analysis in the first paragraph in this subsection. When the data density is unchanged, larger N implies more candidate itemsets generated and also more rules mined.

4.5 Implicit Rule-based Recommendations

IRRMiner can be applied to different cases, such as product promotion and cross-saling by exploring the implicit relations hidden behind various products. Here, I show how the mined implicit rules assist in pattern-based recommendation (Hiltz-Laforge, Nonez, Pourshahid & Watts 2013) to increase recommendation reliability. I first analyze the theoretical benefits and then justify them with real-world case studies. Note that the implicit rules used for recommendations are mined on transactional data, hence we can only make recommendations based on transactional information, which is the typical scenario where pattern-based recommendation is applicable. This is quite different from the well-known content-based or collaborative filtering (Sun, Wang, Gao & Ma 2012)-based recommendations which are built on the rating data.

One of the most important applications of association rules or correlation rules is to increase product sales by recommending some items associated with the items that a customer has just bought. To apply these rules to the recommendation domain, I introduce the concept of recommendation rules in the form of $X \rightarrow Y$ to describe the recommendation strategy whereby itemset Y is recommended to those consumers who have just bought itemset X. In this case, a direct method to evaluate recommendation quality is to check whether the recommended items have actually been bought by the customers or not. The higher possibility of Y to be bought together with X, the more reliable the recommendation rule $X \rightarrow Y$. Based on such observation, the recommendation reliability of a typical recommendation rule $X \rightarrow Y$ is defined as the percentage of transactions with X and Y included w.r.t those including X. Formally,

$$Reliability(X \rightarrow Y) = \frac{\#transactions\ including\ X\ and\ Y}{\#transactions\ including\ X} \tag{4.11}$$

Usually, different patterns lead to different recommendation strategies and recommendation rules in pattern-based recommendation (Choi et al. 2012), which result in different levels of recommendation reliability. Note that here, *pattern* is a general concept; implicit rules, association rules and correlation rules (Nijssen, Guns & De Raedt 2009) are all specific forms of patterns. To be specific, association rule-based recommendation suggests those explicitly associated items; correlation rule-based recommendation suggests explicitly correlated items; while implicit rule-based recommendation makes recommendations by considering not only explicit dependency but also implicit dependency between items. Since my mined implicit rules are built on the basis of dependency between items, which is similar to correlation rules, I compare implicit relation-based recommendation with correlation rule-based recommendation in terms of reliability. Next, I analyze the recommendation strategies in the form of recommendation rules based on implicit rules and correlation rules (Brin et al. 1997) respectively and then compare these two kinds of rule-based recommendations in terms of reliability.

Suppose x and y are two distinct items while Z is an itemset and an implicit rule $x \oplus y|Z$ is mined among them. Accordingly, a recommendation rule $Z \wedge \neg y \rightarrow x$ ($Z \wedge y \not\rightarrow x$) is derived for recommending a related item x to those who have just bought itemset Z based on the explicit dependency between x and Z, and at the same time the effect of the implicitly related item y of x is considered. Specifically, when a business plan to promote or recommend item x to a customer, if the customer has bought Z but not y (and x), it can be proceeded. However, if the customer has already bought Z and y, the planed recommendation action should be terminated. In this way, the conditions (antecedents of recommendation rules) to recommend an item are more precise, and accordingly, recommendation reliability is improved. On the other hand, for items x and y and itemset Z, two correlation rules $[x, Z]$ and $[y, Z]$ can be easily obtained due to the strong explicit dependency between $x(y)$ and Z. Here $[x, Z]$ infers that x and Z are positively correlated. Based on these correlation rules, two recommendation rules $Z \rightarrow x$ and $Z \rightarrow y$ are derived for recommending correlated item x and y to those who have just bought Z. Technically, such recommendations are less reliable than implicit rule-based recommendations due to the lack of consideration of the implicit relations between items (e.g., the implicit relation between x and y). Unfortunately, both the existing association rule-based recommendation and correlation rule-based recommendation do not take such kind of implicit relations into account.

Taking the ERD data as an example, let us consider under which conditions recommending item *coke* is more reliable. Two recommendation rules r_1: $pizza \rightarrow coke$ and r_2: $pizza \wedge \neg sprite \rightarrow coke$ can be derived based on correlation rules and implicit rules respectively. The left-hand side of each rule indicates the conditions in which to recommend the right-hand side items. The first rule r_1 infers that once customers buy *pizza* we can recommend *coke* to them while the second rule means when customers have bought *pizza*, we need to check if the other items implicitly related to *coke* have already been bought, if they have not been bought, we would recommend *coke*. In practice, when we go back to the transactions in Table 4.1, it is obvious that the conditions in which to recommend *coke* described by r_2 are more precise and reliable than that of r_1, which can also be illustrated by a higher reliability of 100% of r_2 than a lower reliability of 66.67% of r_1. This shows that taking into account implicit relations between items contributes to reliable recommendations.

In addition to the above theoretical benefits of implicit rules in increasing recommendation reliability, I also calculate the reliability of both implicit rule-based recommendations and correlation rule-based recommendations on two real-world datasets: ERD and MovieLens_1. The results are given in Tables 4.6 and 4.7, where the mean values of the corresponding recommendation rules based on implicit rules and correlation rules respectively are given. Please note that here the parameter $minIRS$ only exists in the implicit rule-based recommendation algorithm (IRRMiner), and under each $minIRS$ value setting, the identical number of implicit rules and correlation rules are selected to compare their average reliability. It is obvious that, under all the $minIRS$ value settings, implicit rule-based recommendation always has higher reliability than the correlation rule-based one, as shown in the first column in Table 4.6 (35.54% for the Im_Rule compared to 33.7% for the Core_Rule), and in the first column in Table 4.7 (49.41% for the Im_Rule compared to 43.98% for Core_Rule). It is clear that the reliability of all implicit rules (shown in the first column) increases with an increase in the $minIRS$. This is because, the implicit rules with weak implicit relations are filtered out during the increase of $minIRS$, and fewer and stronger implicit rules are selected. Such strong rules have higher reliability. Also less strong correlation rules are selected to guarantee the identical number of rules as implicit rules, so the reliability of the correlation rules also increases with an increase in $minIRS$. Please note that the top 10, top 5 and top 3 rules are rarely affected by the increase of $minIRS$ as the implicit relation strength (IRS) of these rules is usually much higher than $minIRS$.

Table 4.6: Mean recommendation reliability using implicit rules (Im_Rule) and correlation rule (Core_Rule) w.r.t different $minIRS$

		$minIRS$=0	$minIRS$ = 0.5	$minIRS$=1	$minIRS$ = 1.5
ERD	Im_Rule	35.54%	37.5%	38.52%	39.17%
	Core_Rule	33.7%*	35.6%*	36.3%*	36.8%*
Movie Lens_1	Im_Rule	49.41%	49.97%	51.8%	54.11%
	Core_Rule	43.98%*	44.45%*	45.97%*	47.72%*

In addition, some specific recommendation rules based on implicit rules

Table 4.7: Mean recommendation reliability of top-K rules when $minIRS = 1$

		All rules	$K = 10$	$K = 5$	$K = 3$
ERD	Im_Rule	38.52%	40.58%	42%	43.2%
	Core_Rule	36.3%*	38.33%*	39.7%*	40.9%*
MovieLens_1	Im_Rule	51.8%	62.89%	63.79%	64.21%
	Core_Rule	45.97%*	53.87%*	55.32%*	55.32%*

and correlation rules are selected below in Equations (4.12) and (4.13). To differentiate these from recommendation rules, I use $p_1^{'}$ and $p_2^{'}$ to represent two implicit rules mined from the ERD dataset. Based on the relations between the items included in these implicit rules, some recommendation rules based on correlation rules (e.g., $r_{11}^{'}, r_{21}^{'}$) and implicit rules (e.g., $r_{13}^{'}, r_{23}^{'}$) are derived. In addition, to show the significant effect of the implicitly related items on recommendation reliability, I also add another rules ($r_{12}^{'}, r_{22}^{'}$). The name of each item is given below.

$$\boldsymbol{R}_1^{'} \begin{cases} p_1^{'}: i_{11}^{'} \oplus i_{48}^{'} | i_{12}^{'} & \\ r_{11}^{'}: i_{11}^{'} \rightarrow i_{12}^{'} & reliability(r_{11}^{'}) = 42.4\% \\ r_{12}^{'}: i_{11}^{'} \wedge i_{48}^{'} \rightarrow i_{12}^{'} & reliability(r_{12}^{'}) = 36.3\% \\ r_{13}^{'}: i_{11}^{'} \wedge \neg i_{48}^{'} \rightarrow i_{12}^{'} & reliability(r_{13}^{'}) = 45.1\% \end{cases} \tag{4.12}$$

$$\boldsymbol{R}_2^{'} \begin{cases} p_2^{'}: i_{19}^{'} \oplus i_{48}^{'} | i_{12}^{'} & \\ r_{21}^{'}: i_{19}^{'} \rightarrow i_{12}^{'} & reliability(r_{21}^{'}) = 41.5\% \\ r_{22}^{'}: i_{19}^{'} \wedge i_{48}^{'} \rightarrow i_{12}^{'} & reliability(r_{22}^{'}) = 38.69\% \\ r_{23}^{'}: i_{19}^{'} \wedge \neg i_{48}^{'} \rightarrow i_{12}^{'} & reliability(r_{23}^{'}) = 43.5\% \end{cases} \tag{4.13}$$

$i_{11}^{'}$: A book titled 'An Introduction to Secondary Data Analysis with IBM SPSS Statistics' (Book 1)

$i_{12}^{'}$: Philips Peripherals SWR2122/27 Retractable USB Cable

$i_{16}^{'}$: Tableau software for data analysis

$i_{19}^{'}$: A book titled 'Python Data Science Handbook: Essential Tools for Working with Data' (Book 2)

i'_{48} : A book titled 'Microsoft Excel 2013 Data Analysis and Business Modeling (Introducing)' (Book 3)
i'_{61} : San-Disk Memory Card

Please note that I target the scenario that, for a given item, in what conditions (described by the antecedent of the recommendation rules) it should be recommended to achieve greater reliability. In practice, a given item can be recommended in all the different conditions described by different rules (e.g., $r'_{11}, r'_{12}, r'_{13}$), but usually the market wants to make a recommendation as reliable as possible. It is quite clear, given the same item (e.g., Retractable USB Cable), more reliable recommendation rules (r'_{13}, r'_{23}) can be achieved if more implicit relations (the relations between Books 3 and 1 and 2 respectively) are taken into consideration. This reflects the common shopping behavior that customers do not prefer to buy two similar items within one transaction. For the other dataset MovieLens_1, the movie name is not given in the source data MovieLens10M, so I do not show the specific rules mined on it.

4.6 Summary

Rule-based recommender systems are one of the most simple and straightforward solution to session-based recommender systems and they are still valuable in some real-world business organization and optimization due to their simplicity and easy practicality. However, most existing rule-based recommender systems are only based on explicit and straightforward relations embedded in association rules, correlation rules and so on while ignoring more comprehensive relations like implicit relations between items, which usually reduces the recommendation reliability. To bridge this gap, in this chapter, I have proposed a new approach for rule-based recommendations by taking more complex relations into account: implicit rule-based recommender system. The proposed implicit rule-based recommender system basically contains two stages: implicit rule inference and recommendation generation. Accordingly, I have proposed a new approach which first captures the dependency between items and then links those items that share the same dependent items (or itemsets) to infer implicitly related rules. Thanks to the special new structures of implicit rules, the complex relations between multiple items are comprehensively revealed. Experimental results on real-world datasets show that my proposed implicit rule mining algorithm is very promising and can generate implicit rules which cannot be discovered by existing algorithms. The resultant implicit rules are then used for implicit rule-based recommendations and the results on real-world datasets demonstrate that they greatly benefit recommendations by increasing their reliability. In the future, I will explore the possibility of incorporating item features into my rule inference framework to reveal low-level intrinsic inter-item relations (e.g., similarity) for recommendations. Hopefully, more informative implicit rules can be achieved to support more powerful implicit rule-based recommender systems.

Chapter 5

Attention-based Transactional Context Embedding for Next-Item Recommendation

5.1 Introduction

In this chapter, I focus on another critical challenge on item level in session-based recommender systems: the item heterogeneity issue as shown in Figure 1.1 in Chapter 1. This is also the second challenge I addressed in this thesis as demonstrated in the thesis structure Figure 1.2 in the first chapter. *Item heterogeneity* refers to that different items in a transaction are often relevant to the following items in different scales, some are more relevant to next items and others may not. Note that, to make the concept more specific, the session-based recommender systems are specialized to transaction-based recommender systems in this chapter as I conduct recommendations only on transaction data, which is one out of the multiple scenarios of session data as discussed in Section 2.1.

5.1.1 Target Problem and Motivation

Nowadays, recommender systems (RSs) play an important role in real-world business especially in the e-commerce domain. However, most existing RS theories face various issues (Cao 2016) such as tending to repeat items that are similar to what users may have already chosen (Deshpande & Karypis 2004). In reality, users may prefer items that are novel and different from that already in hand. To address this aspect, new recommendation paradigm (Cao 2016) needs to be made on a transactional context, i.e., what has already been chosen in a transaction. On one hand, transaction-based RSs (TBRSs) (Huang & Zeng 2011) incorporate previous transactions, i.e., inter-transactions, to generate more sensible and reliable new transactional recommendations, such as next-basket and next-item recommendations (Wang, Guo, Lan, Xu, Wan & Cheng 2015) through analyzing inter-transaction coupling relationships (Cao 2015). These are quite different from the typical RS approaches built on user preferences and item property. On the other, however, it is still unclear what next-item should be recommended when a collection of items has been placed into a transaction. This generates the need to recommend the next item under a transactional context by analyzing intra-transaction dependency. Here, the *context* for recommending the next item refers to the corresponding item-related transaction, e.g., a shopping-basket record consisting of multiple chosen items.

Let us illustrate the above problem with an example. A user first puts three items $\{milk, apple, orange\}$ into a cart and then adds *bread* to the same cart. Subsequently, the transaction is finalized as $\{milk, apple, orange, bread\}$. If I take the first three items as the context and the last one as the target to recommend, existing methods may suggest vegetables like *green salad* due to the nearest contextual items (*orange* and *apple*). However, the choice of the target item *bread* may depend on the first item (*milk*). In this case, a TBRS should pay more attention to *milk* than to *orange* and *apple*, because *milk* may be more related to the next choice *bread*. This example shows the importance of next-item recommendation which can be misled by irrelevant items in a transaction. Moreover, real-world transactional data often only indicates those items co-appear in a transaction with the order (e.g., the item timestamps) between items. Therefore, it may not be possible and realistic to recommend transactional items with a rigid order.

It is quite challenging to learn the relevance and transition between items in a transactional context. In TBRSs, a general challenge is to build an attentive context which outputs the real next choice with a high probability (Verbert, Manouselis, Ochoa, Wolpers, Drachsler, Bosnic & Duval 2012). Some existing approaches aim to generate recommendations by taking a transaction as the context. However, most existing TBRSs utilize a partial context with an ordering assumption. Sequential pattern mining (Yap et al. 2012) is used to predict the next item using associations between items with a rigid order assumption. Although simple and effective, such kind of approaches usually lose those infrequent items (Hu, Cao, Cao, Gu, Xu & Wang 2017) due to the minimum support constraint. In addition, the dynamic context containing arbitrary items may fail to match any mined frequent patterns (Wang, Bao & Zhou 2017). Markov chain (MC) (Rendle et al. 2010, Cao, Ou & Yu 2012) is another way to model sequential data. (Wu et al. 2013) proposed Personalized Markov Embedding (PME) to first embed users and songs into a Euclidean space by modeling sequential singing behaviours and then generate recommendations based on the embeddings. Recently, a personalized ranking metric embedding method (PRME) was proposed to precisely model personalized check-in sequences for next POI recommendation (Feng et al. 2015). Both PME and PRME are first-order MC models built on rigid ordered data to model the transition between sequential items from the same transaction. They may lose higher-order dependencies and the assumed rigid ordered data may not always be real-world cases. Recently, a matrix factorization (MF) based approach (Chou et al. 2016) factorizes the matrix of transition probability from the current item to the next one into the latent factors. However, MF easily suffers from sparsity issues due to the power-law distributed data in the real world (Hu, Cao, Cao, Gu, Xu & Yang 2016). Factorized Personalized Markov Chains (FPMC) (Rendle et al. 2010) combines the power of MF and MC to factorize the transition matrix over underlying MC to model personalized sequential behaviours for next-basket recommendation. Similar to MC and MF, FPMC also suffers from the unrealistic rigid order assumption and data sparsity issue. Inspired by great success of deep networks, (Hidasi et al. 2015) applied deep recurrent neural networks (RNN) to model the transaction of sequential data but the high computational cost caused by the complex structures prevents its application to large data. Compared to deep architectures (Wang, Liu, Wu, Cao, Meng & Kennedy 2016), shallow networks are more efficient in dealing with such kinds of issues, especially on large datasets. But currently quite few literature works on shallow network-based recommmender systems to the best of my knowledge.

Moreover, MC, MF and RNN were originally designed for time-series data with a rigid natural order, hence they do not fit unordered transactions. For example, it makes no difference whether *milk* or *bread* is put into the cart first. In addition, existing methods do not effectively weight the items within a context, namely paying more attention to those relevant items. Such attention distinction is quite important especially for long transactions which often contain many items irrelevant to the next choice.

5.1.2 My Design and Main Contributions

This chapter addresses the above issues by proposing an attention-based transaction embedding model (ATEM). ATEM builds an attentive context embedding over the embeddings (Jian, Cao, Pang, Lu & Gao 2017) of all the observed items in a transaction by identifying the contextual items with high relevance to the next choice. Considering the large number of items, usually over 10^5, in real-world business, I build a shallow wide-in-wide-out network (Goth 2016) to reduce the time and space cost. Specifically, I incorporate the attention mechanism (Shaonan, Jiajun & Chengqing 2017) into the shallow network to build an attentive context over all the observed items in a transaction without the rigid ordering assumption. Thanks to the attention mechanism, the proposed model is able to pay greater attention to more relevant items and less attention to less relevant ones. As a result, ATEM is more effective and robust to predict the next item in a transaction with less constraints. The main contributions of this chapter are as follows:

- I figure out and highlight the item heterogeneity issue in session-based recommender systems, which could be of great significance to satisfied recommendation outputs.

- An attention-based model learns an attentive context embedding that intensifies relevant items but downplays those irrelevant to the next choice. My method does not involve a rigid ordering assumption over items in a transaction.

- A shallow wide-in-wide-out network implements ATEM, which is more effective and efficient for learning and prediction over a large number of items.

- My empirical study shows that (1) ATEM significantly outperforms the state-of-art TBRSs on two real-world datasets in both accuracy and novelty; and (2) the attention mechanism makes a significant difference to TBRSs by comparing the methods with and without the attention mechanism.

5.2 Problem Statement

Before going into the details of my proposed model, I first define the problem and define basic concepts.

Generally, transaction-based recommendations are built on shopping basket-based transaction data. For a given transactional dataset, let $T = \{t_1, t_2 ... t_{|T|}\}$ be the set of all transactions, and each transaction $t = \{i_1, i_2 ... i_{|t|}\}$ consists of a subset of items, where $|T|$ denotes the number of elements in set T. All the items occurring in all transactions constitute the whole item set $I = \{i_1, i_2 ... i_{|I|}\}$. Note that the items in a transaction t may not have a rigid order. For a given target item $i_s \in t$, all the items in t except i_s are picked up as its corresponding context $\mathbf{c}$, namely $\mathbf{c} = t \backslash i_s$. Particularly, an attentive context means items within the context contribute differently to the context embedding for next-item recommendation. Given the context $\mathbf{c}$, my ATEM is constructed and trained as a probabilistic classifier that learns to predict a conditional probability distribution $P(i_s|\mathbf{c})$. A total of $|t|$ training instances are built for each transaction t by picking up each item as the target one each time.

Therefore, TBRS is boiled down to rank all candidate items in terms of their conditional probability over the given context. Note that in the prediction stage, the conditional probability is computed based on the attentive embedding of the context $\mathbf{c}$. Such embedding is built on all the contextual items included in $\mathbf{c}$ by utilizing the attention mechanism to learn the weight of each contextual item.

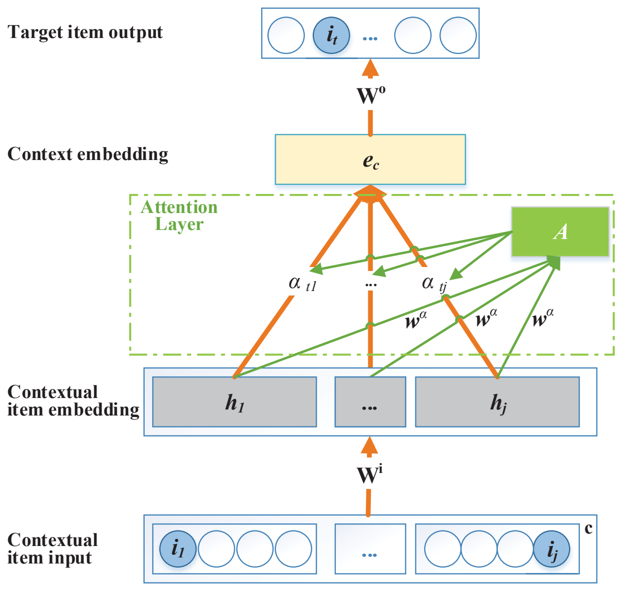


Figure 5.1: The ATEM architecture, which first learns item embeddings and then integrates them into the context embedding for target item prediction, where 'A' represents the attention model.

5.3 Modeling and Learning

In this section, I first demonstrate the architecture of the proposed ATEM model, and then discuss how to train the model and learn the parameters. Finally, I show how to make predictions and accordingly generate recommendations using the trained model.

5.3.1 Attention-based Transaction Embedding Model

Overall, from bottom to top, the proposed ATEM model consists of an input layer, an item embedding layer, a context embedding layer, an output layer, plus an attention layer between the item and context embedding layers, as shown in Figure 5.1. Next, I explain the working mechanism of the model layer by layer from the input to the output.

Item Embedding

Giving a contextual itemset $\mathbf{c}$ to the input layer, the input units in the bottom of Figure 5.1 constitute a one-hot encoding vector where only the unit at position i_j ($i_j \in \mathbf{c}$) is set to 1 and all others are set to 0. For each $i \in \mathbf{c}$, I encode it in the same way as i_j. Therefore, a vector with length $|I|$ is achieved to represent each item in the context and a total of $|\mathbf{c}|$ vectors can be achieved for a given context $\mathbf{c}$.

The information delivered by the sparse one-hot vectors is limited. In ATEM, I create an embedding mechanism to map these vectors to an informative and lower-dimensional vector representation in the item embedding layer, where a K-dimension real-valued vector $\mathbf{h}_j \in \mathbb{R}^K$ is used to represent the embedding of item i_j. The input weight matrix $\mathbf{W}^i \in \mathbb{R}^{K \times |I|}$ is used to fully connect the input-layer and item embedding-layer. Note that the j^{th} column of the weight matrix $\mathbf{W}^i_{:,j}$ actually encodes the one-hot vector of item i_j to the real-valued embedding $\mathbf{h}_j$, namely:

$$\mathbf{h}_j = \mathbf{W}^i_{:,j} \tag{5.1}$$

Transactional Context Embedding with Attention

When the embeddings of all items in context $\mathbf{c}$ are ready, I can obtain the embedding $\mathbf{e}_c \in \mathbb{R}^K$ of context $\mathbf{c}$ by integrating the embeddings of all items in $\mathbf{c}$. Specifically, the attentive context embedding is built as a weighted sum of $\mathbf{h}_j$:

$$\mathbf{e}_c = \sum_{i_j \in \boldsymbol{c}} \alpha_{tj} \mathbf{h}_j, \quad s.t. \sum_{i_j \in \boldsymbol{c}} \alpha_{tj} = 1 \tag{5.2}$$

where α_{tj} is the integration weight of contextual item i_j w.r.t the target item i_t, which indicates the contribution scale of i_j to the occurrence of i_t. In my model, to better capture the different contribution scale of various contextual items, I develop an attention layer to learn the integration weights automatically and effectively. Compared to assigning the weights manually under certain assumptions, or directly learning the weights without the attention mechanism, my method not only works more flexibly without such assumptions but also focuses more on the key items and reduces the interference from irrelevant items in a long context. Next, I demonstrate how the attention model achieves the integration weights.

Specifically, I use a softmax layer to determine the weights of different contextual items. In this case, those contextual items more relevant to the target item are given larger weights, while the input of the softmax is a transformation of each item embedding.

$$\alpha_{tj} = \frac{exp(e(\mathbf{h}_j))}{\sum_{s \in \mathbf{c}_t} exp(e(\mathbf{h}_s))} \tag{5.3}$$

$$e(\mathbf{h}_j) = \mathbf{w}^{\alpha} \mathbf{h}_j^T \tag{5.4}$$

where $\mathbf{w}^{\alpha}$ is an item-level context vector shared by all contextual items as shown in Figure 5.1. The shared context vector $\mathbf{w}^{\alpha}$ can be seen as a high level representation of a fixed query 'which is the informative item' over the contextual items like that used in memory networks (Kumar, Irsoy, Ondruska, Iyyer, Bradbury, Gulrajani, Zhong, Paulus & Socher 2016). It is randomly initialized and jointly learned during the training stage. As $\mathbf{w}^{\alpha}$ serves as a weight vector to connect the item embedding layer to the attention model, I also refer to it as attention weight in the following section to keep it consistent with input and output weights.

Essentially, I measure the importance of each item i_j as the similarity of its embedding $\mathbf{h}_j$ with the item level context vector $\mathbf{w}^{\alpha}$ and get a normalized importance weight α_{tj} of item i_j w.r.t the target item i_t through a softmax function (Yang, Yang, Dyer, He, Smola & Hovy 2016). Consequently, the attentive context representation vector can be computed using Equation (5.2).

Target Item Prediction

After getting the representation of context $\mathbf{c}$, we feed it into the output layer for the prediction task, which is shown in the top of Figure 5.1. Here the output weight matrix $\mathbf{W}^o \in \mathbb{R}^{|I| \times K}$ is used to fully connect the context embedding layer and output layer. With the embedding of the given context $\mathbf{c}$ plus the weight matrix $\mathbf{W}^o$, the score S_t of a target item i_t w.r.t the given context $\mathbf{c}$ is computed as:

$$S_t(\mathbf{c}) = \mathbf{W}^o_{t,:}\mathbf{e}_\mathbf{c} \tag{5.5}$$

where $\mathbf{W}^o_{t,:}$ denotes the t^{th} row of $\mathbf{W}^o$. The resultant score $S_t(\mathbf{c})$ quantifies the relevance of the target item i_t w.r.t the given context $\mathbf{c}$. As a result, the conditional probability distribution $P_\Theta(i_t|\mathbf{c})$ can be defined in terms of the softmax function, which is commonly used in neural network or regression model.

$$P_\Theta(i_t|\mathbf{c}) = \frac{exp(S_t(\mathbf{c}))}{Z(\mathbf{c})} \tag{5.6}$$

where $Z(\mathbf{c}) = \sum_{i \in I} exp(S_i(\mathbf{c}))$ is the normalization constant and $\Theta = \{\mathbf{W}^i, \mathbf{w}^\alpha, \mathbf{W}^o\}$ is the model parameters. Therefore, a probabilistic classifier modeled by my proposed ATEM is obtained to predict the target item.

5.3.2 Learning and Prediction

In the previous subsection, I have described the construction of a probability classifier over the transaction data $d = \langle \mathbf{c}, i_c \rangle$, where $\mathbf{c}$ is the input, namely the context constructed on the items within a transaction, and i_c is the observed output, namely the corresponding relevant item conditional on this context. Given a training dataset $D = \{\langle \mathbf{c}, i_c \rangle\}$, the joint probability distribution can be obtained as:

$$P_\Theta(D) \propto \prod_{d \in D} P_\Theta(i_c|\mathbf{c}) \tag{5.7}$$

Therefore, the model parameters Θ can be learned by maximizing the conditional log-likelihood (cf. Equation (5.6)):

$$L_\Theta = \sum_{d \in D} logP_\Theta(i_c|\mathbf{c}) = \sum_{d \in D} S_{i_c}(\mathbf{c}) - logZ(\mathbf{c}) \tag{5.8}$$

Note that, both the evaluation of L_Θ and the computation of its corresponding log-likelihood gradient involve the normalization term $Z(\mathbf{c})$, which needs to sum $exp(S_{i_c}(\mathbf{c}))$ over the entire item set for each training instance. This means, it takes $O(|I| \times |D|)$ time of computation to get the normalization constant for each iteration to train this model. Unfortunately, $|I|$ and $|D|$ are usually quite large in real-world business. For example, the Amazon dataset contains millions of transactions for more than ten thousand of products. Such a high computation cost makes the training process intractable.

Noise Contrastive Estimation

To tackle the aforementioned issue, I adopt a subsampling approach to deal with the softmax layer, namely noise-contrastive estimation (NCE) (Gutmann & Hyvärinen 2012) which was proposed for training unnormalized probabilistic models and has been broadly used to handle similar issues in NLP etc. NCE does not directly compute the normalization constant of the softmax to avoid the high computation cost, instead it works with the other approximate objective which is much cheaper to compute.

The main idea of NCE is to use a binary classifier to distinguish samples from the data distribution from those with a known noise distribution Q. In my case, given a training example $\langle \mathbf{c}, i_c \rangle$, the probability of sampling from either a positive example or K noise examples is represented as a mixture of these two distributions (Mnih & Teh 2012):

$$P_\Theta(y, i_c|\mathbf{c}) = \frac{1}{K+1} P_\Theta(i_c|\mathbf{c}) + \frac{K}{K+1} Q(i_c) \tag{5.9}$$

Then the posterior probability of a sample i_c coming from the data distribution, namely the probability of a positive example, is calculated as:

$$P_\Theta(y = 1|i_c, \mathbf{c}) = \frac{P_\Theta(i_c|\mathbf{c})}{P_\Theta(i_c|\mathbf{c}) + KQ(i_c)} \approx \frac{exp(S_{i_c}(\mathbf{c}))}{exp(S_{i_c}(\mathbf{c})) + KQ(i_c)} \tag{5.10}$$

where the normalization term $Z(\mathbf{c})$ is dropped from $P_\Theta(i_c|\mathbf{c})$. This is because the NCE is a normalized estimator where the objective encourages $P_\Theta(i_c|\mathbf{c})$ to be approximately self-normalized (Gutmann & Hyvärinen 2012). Hence, the probability of i_c coming from the noise samples is $P_\Theta(y = 0|i_c, \mathbf{c}) = 1 - P_\Theta(y = 1|i_c, \mathbf{c})$. Subsequently, instead of maximizing the original log-likelihood in Equation (5.8), we can maximize the likelihood of the training samples against K noise samples as (Mnih & Kavukcuoglu 2013):

$$\begin{aligned} J_\Theta(i_c, \mathbf{c}) &= logP_\Theta(y = 1|i_c, \mathbf{c}) + K\mathbb{E}_{i_k \sim Q}[logP_\Theta(y = 0|i_c, \mathbf{c})] \\ &\approx logP_\Theta(y = 1|i_c, \mathbf{c}) + \sum_{k=1}^{K} logP_\Theta(y = 0|i_k, \mathbf{c}) \end{aligned} \tag{4.11}$$

Substituting Equation (5.10) into Equation (4.11), the gradient of $J_\Theta(i_c, \mathbf{c})$ can be immediately obtained. It approaches the original maximum likelihood (Equation (5.8)) gradient when K increases (Mnih & Teh 2012). K is empirically set to 8 in my experiments.

$$\nabla J_\Theta(i_c, \mathbf{c}) = \frac{KQ(i_c)}{exp(S_{i_c}(\mathbf{c})) + KQ(i_c)} \nabla S_{i_c}(\mathbf{c}) - \sum_{k=1}^{K} \frac{exp(S_{i_k}(\mathbf{c}))}{exp(S_{i_k}(\mathbf{c})) + KQ(i_k)} \nabla S_{i_k}(\mathbf{c}) \tag{4.12}$$

Learning and Ranking

After we get the gradient of $J_\Theta(i_c, \mathbf{c})$ as illustrated in Equation (4.12), all the parameters Θ are learned by back-propagation. Algorithm 5.1 briefly summarizes the learning process. Note that $\nabla_{e(\mathbf{h}_j)} \alpha_{tj}$ is the gradient of a softmax function (cf. Equation (5.3)) and it can be well computed (Buntine & Weigend 1994).

In Algorithm 5.1, $\odot$ denotes the element-wise product. Index t corresponds to the output item i_t which includes both the positive sample i_c and all noise ones $\{i_k\}$ and $i_j \in \mathbf{c}$ is one of the input items from the context. $w_{:,j}$ is the corresponding input weight (cf. Equation (5.1)). A specific gradient-based update process for the parameter is achieved after we substitute the gradients demonstrated in Algorithm 5.1 (cf. Steps 3-5) into Equation (4.12). To reduce the high computation cost caused by the large number of training samples, we adopt a mini-batch scheme to train the model, where each batch contains 50 training instances. The details to build the instances are given in the experimental part. My experimental results are achieved by using Adam (Kingma & Ba 2014) for the specific gradient descent operation.

Algorithm 5.1 ATEM Parameter Learning Using SGD

```
1: $l \leftarrow 0$
2: while not converged do
3:     Compute output weight $w^o_{t,:}$-gradient (Equation (5.5)): $g_{w^o_{t,:}} \leftarrow \mathbf{e}_c$
4:     Compute attention weight $w^\alpha_{tj}$-gradient (Equations (5.2)-(5.5)): $g_{w^\alpha_{tj}} \leftarrow$
       $\mathbf{W}^o_{t,:} \odot \mathbf{h}^2_j \odot \nabla_{e(\mathbf{h}_j)}\alpha_{tj}$
5:     Compute input weight $w^i_{:,j}$-gradient (Equations (5.1)-(5.5)):
       $g_{w^i_{:,j}} \leftarrow \mathbf{W}^{o\top}_{t,:} \odot (\alpha_{tj} + \nabla_{e(\mathbf{h}_j)}\alpha_{tj} \odot \mathbf{w}^\alpha \odot \mathbf{h}_j)$
6:     Perform SGD-updates for $w^o_{t,:}$, $w^\alpha_{tj}$ and $w^i_{:,j}$ :
       $w^o_{t,:} \leftarrow w^o_{t,:} + S^l_t(g)g_{w^o_{t,:}}$ (output weight update),
       $w^\alpha_{tj} \leftarrow w^\alpha_{tj} + S^l_{tj}(g)g_{w^\alpha_{tj}}$ (attention weight update),
       $w^i_{:,j} \leftarrow w^i_{:,j} + S^l_j(g)g_{w^i_{:,j}}$ (input weight update)
7:     $l \leftarrow l + 1$
8: end while
```

After all the parameters have been learned by the training process, the model can be used as a transaction-based recommender system, which is ready to make predictions and accordingly generate recommendations. To be specific, given an arbitrary transaction-based context $\mathbf{c}$ containing all the items chosen in a certain transaction, the probabilities of choosing each next candidate item can be calculated according to Equation (5.6) immediately, and then the ranking over all of them can be achieved accordingly.

5.4 Experiments and Evaluation

The empirical study of the proposed ATEM is given in this section. Specifically, I first setup the experiments by preparing the experimental datasets and introducing the comparison methods, and then evaluate the performance in terms of recommendation accuracy and novelty.

5.4.1 Experimental Setup

Data Preparation

I evaluate my method on two real-world transaction data sets: IJCAI-15 [1] and Tafang [2].

First, a shopping-basket-based transaction table is extracted from each of the original datasets. The transaction table contains multiple transactions and each transaction consists of multiple items. Note that those transactions containing only one item are removed as they do not fit my model as I use at least one item as context and another as the target. Second, the transaction table is split into training and testing sets. Specifically, I randomly choose 20% from the transactions happened in last 30 days as the testing set, while the remainder is for training. Finally, to build the training and testing instances of format $d = \langle \mathbf{c}, i_c \rangle$ as illustrated in last section, for a transaction t, each time one out of which is picked up as the target item i_c and all the remaining ones are used as the corresponding context $\mathbf{c}$. Subsequently, for a transaction containing $|t|$ items, $|t|$ instances are built in total. The characteristics of the datasets are shown in Table 5.1.

During the training stage, transactions in the training set are imported into the model in batches to learn the context embeddings. In the testing process, the learned embeddings are used to predict the target item. The true target item is used as the ground truth. I calculate the accuracy measures Recall@K and MRR (Chou et al. 2016) by comparing the predicted results to the ground truth. However, it is not enough to evaluate a recommender system only using accuracy metrics (Ge, Delgado-Battenfeld & Jannach 2010). Considering the fact that an increasing number of customers prefer to enjoy a more surprising experience by discovering novel products which they have not chosen before, I also measure the recommendation novelty by comparing the recommendation list to the corresponding contextual itemset.

[1] https://tianchi.aliyun.com/datalab/dataSet.htm?id=1

[2] http://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset

Table 5.1: Statistics of experimental datasets

Statistics	**IJCAI-15**	**Tafang**
#Transactions	144,936	19,538
#Items	27,863	5,263
Avg. Transaction Length	2.91	7.41
#Training Transactions	141,840	18,840
#Training Instances	412,679	141,768
#Testing Transactions	3,096	698
#Testing Instances	9,030	3,150

Comparison Methods

I use the four representative start-of-the-art methods, i.e., *PBRS*, *FPMC*, *PRME* and *GRU4Rec* introduced in Section 2.4, as the baselines for the experiments. In addition, I built another model *TEM* as below to test the efficacy of attention mechanism.

- ***TEM***: A model similar to ATEM except that it utilizes distance-based exponential decay (Hu, Cao, Wang, Xu, Cao & Gu 2017) to replace the attention mechanism to assign the weights manually. The contextual items near to the target one are given larger weights.

5.4.2 Performance Evaluation

In this section, the accuracy evaluation is first given, followed by the novelty evaluation.

Accuracy Evaluation

Two commonly used accuracy metrics for session-based RSs are used for the evaluations. Particularly, recall ($REC@K$) and mean reciprocal recall ($MRR@K$) are used as the evaluation measures, their definitions are given in Section 2.3.1. Note that rating-based RS evaluation metrics, e.g., MSE,

Table 5.2: Accuracy comparisons on IJCAI-15

Model	**REC@10**	**REC@50**	**MRR**
PBRS	0.0780	0.0998	0.0245
FPMC	0.0211	0.0602	0.0232
PRME	0.0555	0.0612	0.0405
GRU4Rec	0.2283	0.3021	0.1586
TEM	0.3177	0.3796	0.1918
ATEM	**0.3542**	**0.5134**	**0.2041**
Improve (%)	11.49	35.25	5.01

Table 5.3: Accuracy comparisons on Tafang

Model	**REC@10**	**REC@50**	**MRR**
PBRS	0.0307	0.0307	0.0133
FPMC	0.0191	0.0263	0.0190
PRME	0.0212	0.0305	0.0102
GRU4Rec	0.0628	0.0907	0.0271
TEM	0.0789	0.1716	0.0231
ATEM	**0.1089**	**0.2016**	**0.0347**
Improve (%)	38.02	17.48	28.04

are not applicable in my work as I do not work on rating data to predict the ratings.

Table 5.2 and Table 5.3 demonstrate the results of REC@10, REC@50 and MRR over the testing sets on two real-world datasets, respectively. For PBRS, I empirically set the minimum support to 0.01 and 0.008 on IJCAI-15 and Tafang dataset respectively. As it only focuses on those frequent items and filters out infrequent ones, the performance is not so good. The number of factors is set to 10 for training the FPMC to achieve the best performance.

However, the accuracy performance of FPMC on both datasets is quite poor. This is due to the fact that both datasets are extremely sparse and thus a very large but quite sparse item transition matrix is constructed on each dataset to train the MF model. For example, in IJCAI-15 dataset, each transaction only contains an average of 2.91 items from over 27,000 ones (cf. Table 5.1). This indicates each row of the built matrix contains less than two items. In practice, the non-empty entries account for less than 0.01%. I set the embedding dimensions to 60 as suggested in (Feng et al. 2015) when training the PRME model. Compared to FPMC, the performance of PRME is a little better, but it is still poor. This is because PRME is a first-order MC model, which learns the transition probability over the successive item rather than the whole context. This may lead to information loss. Furthermore, in the real word, the purchase of goods does not always follow a rigid sequence assumed by such kind of models. Benefiting from the deep structure, GRU4Rec achieves much better performance compared to FPMC and PRME.

For my ATEM model, the batch size is empirically set to 50 and the number of hidden units for item embeddings is set to 128 and 40 on IJCAI-15 and Tafang dataset respectively. I run 20 epochs to train the model. It clearly achieves a better performance than GRU4Rec, where the REC@10 and REC@50 exceed 35% and 50% respectively on IJCAI-15 dataset. The highest MRR also proves that my model can effectively put the users' desired items in the front of the recommendation list, the reason being that, different from the previous models which either capture only first-order dependency between items or capture the dependency between each contextual item and target one respectively, ATEM builds an embedding of the context by treating all the contextual items as a whole. Therefore, the complex dependency relations (e.g., intra-context dependency, context-target dependency) can be better captured. More importantly, the attention mechanism is applied here to discriminate the contributions of different contextual items to the prediction of a certain target item. This actually helps greatly to build a more informative context embedding for different target items. From the point of view of real-world applications, my model has a very shallow and concise structure for easy training, which makes it more efficient to recompute the scores for ranking all candidate items when contexts keep updating in online recommendations, compared to those complex and deep models (e.g., GRU4Rec).

The settings of TEM are kept the same as ATEM and the parameter λ in exponential decay is set to 0.75 to obtain the best performance. The performance of TEM is obviously weaker than ATEM. This is caused by the distance assumption used in TEM, which essentially still has an order assumption over items within a transaction. This may not be consistent with real-word cases, as previously stated.

The Effect of Context Length

Although longer transactional contexts consisting of more items may be more informative, they may be more fragile and contain irrelevant items, resulting in reduced recommendation accuracy when these are not identified. To show the advantages of my model under varies lengths of contexts, I test the effect of context length. Figure 5.2 illustrates that longer contexts benefit accuracy, and my method clearly outperforms the others under longer contexts, such as a context consisting of four items (denoted as Len-4). Note that PBRS, FPMC and PRME are mainly first-order dependency based and thus are not sensitive to context length. Hence, they are not included in this test.

The Effect of Item Order

To test the effect of the order of items within a transaction on recommendation accuracy, I randomize the default item order in the IJCAI-15 data to build a disordered dataset. Table 5.4 shows the accuracy of different methods on this new data. Compared to the results in Table 5.2, other approaches experience much more performance degradation than ATEM. This indicates the stronger ability of ATEM compared to the other methods in handling

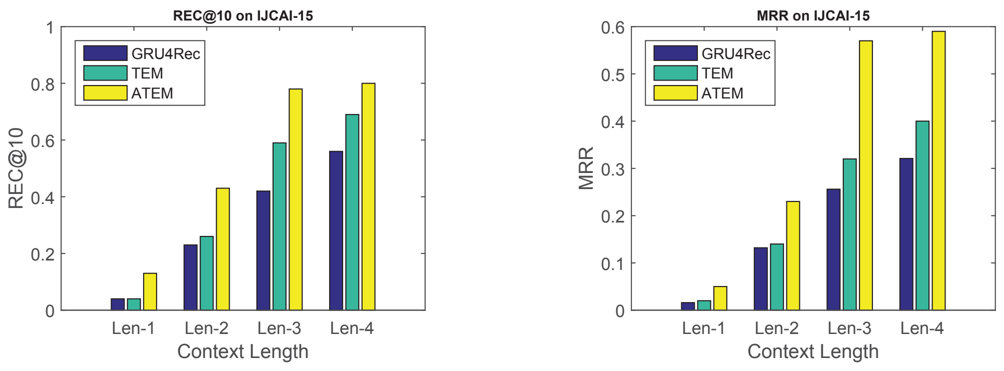


Figure 5.2: ATEM achieves higher REC@10 and MRR than the other approaches, especially under long contexts.

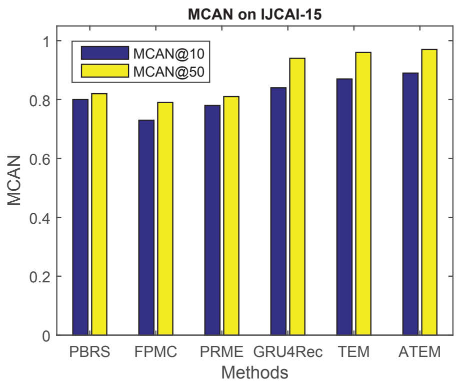
Table 5.4: Accuracy on disordered IJCAI-15

Model	**REC@10**	**REC@50**	**MRR**
PBRS	0.0500	0.0559	0.0185
FPMC	0.0151	0.0412	0.0183
PRME	0.0346	0.0389	0.0351
GRU4Rec	0.1636	0.2121	0.1022
TEM	0.2660	0.3012	0.1431
ATEM	**0.3423**	**0.4981**	**0.1960**
Improve (%)	28.68	65.37	36.97

disordered data.

Novelty Evaluation

Except for accuracy, novelty is another important quality which should be considered in real-world RS (Wang, Hu & Cao 2017). Recall that by considering what a customer has already chosen in a transaction (transactional context), my proposed TBRS can effectively avoid recommending duplicate items and suggest some novel items which can result in a surprising experience. Therefore, I employ the local novelty measure $MCAN$ defined in Section 2.3.2 to quantify the difference between the given context and the



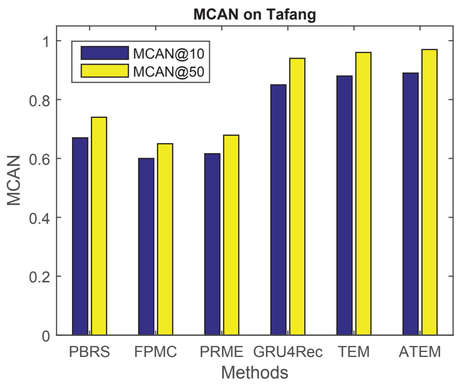


Figure 5.3: ATEM achieves higher novelty than the other approaches.

recommendation list. The larger the difference, the higher the novelty. It should be noted that the aforementioned accuracy guarantees the relevance of recommended items, so highly novel items are also of high relevance.

In my model, the items which have already been chosen correspond to the context $\mathbf{c}$ used for recommendation R. Subsequently, $MCAN$ (cf. Section 2.3.2) measures the mean non-overlap ratio between each context-recommendation pair $\langle \mathbf{c}_i, R_i \rangle$ over all N top-K recommendations.

Figure 5.3 illustrates the results of novelty comparison on the two datasets. PBRS only results in frequent patterns, hence it is difficult to match the whole context exactly, especially for long contexts. Subsequently, low novelty is achieved without considering all the contextual items. FPMC does not learn its parameters well on such sparse datasets, so it outputs relatively random recommendations. Accordingly, FPMC obtain low novelty. PRME is a first-order MC model which makes recommendations by only considering the exact prior item while ignoring other contextual items, so it may recommend duplicate items and thus lead to low novelty. GRU4Rec can accumulate the influence of all the sequential items from the context to make relatively reliable and novel recommendations. Compared to the aforementioned methods, my model not only considers the whole context but also tries to build an attentive context embedding utilizing attention mechanism. Consequently, it is easier for me to generate novel and relevant recommendations.

In summary, the higher accuracy and novelty of the next-item recommended by ATEM than the baselines verifies the significance of weighting all the contextual items in building an attentive context embedding for transaction-based recommender systems. In addition, the power of the attention mechanism is further justified by the comparison between ATEM and TEM.

5.5 Summary

To effectively recommend the next item within a transactional context, which cannot be addressed by existing next-basket and next-items recommender systems, this chapter proposed an attention-based transaction embedding model ATEM. ATEM is a shallow wide-in-wide-out neural network. It learns an attentive context embedding that is expected to be the most relevant to the next choice over all the observed items in a transaction. The empirical evaluation on the real-world transactional data shows its significant superiority in addressing gaps in state-of-the-art approaches. This also demonstrates the effectiveness of my solution to the widely existed item heterogeneity issue in session-based recommender systems. I will explore the application of ATEM to other session-based recommendation scenarios like next-song or next-web page recommendations. We will also explore its application to other problems such as the author-topic relation learning (Rosen-Zvi, Chemudugunta, Griffiths, Smyth & Steyvers 2010).

Chapter 6

Perceiving the Next Choice with Comprehensive Transaction Embeddings by Integrating Item Features

6.1 Introduction

In this chapter, I focus on a quite common but critical challenge in recommender system area: cold start issue, which is less-studied in session-based recommender systems. This is also the third challenge we addressed in this thesis as demonstrated in the thesis structure Figure 1.2 in the first chapter. To address this challenge, I go down to the feature level and focus more the feature dependency and the joint modeling of feature dependency, item dependency and the interactions between them, as shown in Figure 1.1 in Chapter 1. *Cold start items* means those items rarely occurred or totally new. Similar to Chapter 5, I specify session-based recommender systems to transaction-based recommender systems in this chapter.

6.1.1 Target Problem and Motivation

Nowadays, recommender systems (RS) play an important role in real-world business especially in the e-commerce domain. For example, the RS behind thousands of websites (e.g., Amazon) provide magic power to help end users to discover and make choices from a huge number of items. As a result, RS can not only improve customers' shopping experience but also increase the business profits. Although lots of work has been done to produce high quality recommendations, some issues are still challenging and need more efforts. One of them is to enable RS to dynamically perceive customers' next choice on thousands of candidate items online according to what they have just put in the shopping carts. In this chapter, I call the next item to choose as the target item while those items having been added to cart are treated as its context. The challenges come from two sides: on one hand, the context is dynamic along with the shopping transaction; on the other hand, the RS needs to keep updating the recommendations when the context was changed. For instance, suppose a customer Robin starts an online shopping transaction on Amazon: first, he bought a cellphone and then a protective film may be his next choice, so good RS should be capable of perceiving the protective film according to the context of cellphone. When Robin has bought the cellphone and protective film, his next choice is probably an earphone rather than buying another type of protective film again. Therefore, the recommended item should be changed from the protective film to the earphone accordingly. This dynamic recommendation process keeps updating until the transaction is finished.

This kind of recommender systems can be easily formalized as the session-based recommender systems as demonstrated in Section 2.1. Similar to Chapter 5, in this chapter, they are called transaction-based RS (TBRS) (Huang & Zeng 2011) instead as they work on transactional data, which are different from the rating-based RS (RBRS) (Adomavicius & Tuzhilin 2015) working on rating data. Although existing TBRS can recommend next items given the context, most of them treat the context as static rather than dynamic and can only work on static transactional data, an example is the pattern-based RS (Yap et al. 2012). Moreover, it is quite hard for them to keep updating recommendations due to the long computational time, like deep network-based RS (Hidasi et al. 2015, Wang, Liu, Wu, Cao, Meng & Kennedy 2016). As a result, they do not work efficiently for online recommendations. RBRS has been well studied but it cannot tackle my problems here due to the lack of consideration of context, existing TBRS cannot perform well either for online recommendations due to the aforementioned reasons. In this chapter, I focus on TBRS and target at handling the dynamic context and producing online recommendations.

Due to the lack of effective approach to model the context of a transaction event, most current TBRS cannot capture the intra-session relevance over items perfectly and thus cannot produce high quality recommendations. They tend to recommend those popular and long-released items while ignore those less popular or newly-released ones. In practice, customers may not always need popular or similar items to form immutable shopping behaviors, instead, they want to explore something novel or unpopular. For example, when the first smart-phone *iPhone* appeared in 2010, most customers prefer to it rather than *Nokia*, a popular function-phone (not smart-phone) brand lasted for more than a decade at that time. Therefore, more sensible RS which can bring surprising experience to users by recommending novel but relevant items is increasingly important. In this chapter, novel items refer to unpopular or newly-released items while 'relevant' infers the recommended items are strongly relevant to the context. It's clear that an effective approach to model the dynamic context in real-time is necessary to produce high quality online recommendations.

Content-based filtering (CBF) (Melville, Mooney & Nagarajan 2002) and collaborative filtering (CF) (Koren 2008) are two approaches that are most commonly used in RS. They are not applicable to TBRS directly though they perform well in RBRS. This is because these methods are actually designed to work on rating matrix in RBRS, which is quite different from the shopping-basket data in TBRS, thus it is hard for them to capture the relevance between items embedded in transactional data.

Pattern or rule-based recommendation (Lin et al. 2002, Adda et al. 2005) is a intuitive and straightforward solution to transaction-based recommendation issues. It first captures associations between items and then recommends items associated to the context items. Although simple and effective sometimes, patterns are extracted from those frequent items due to the 'support' measure, whereas those infrequent ones are missed. In addition, they can not be applied to online recommendation directly as the dynamic context may contain arbitrary items, it probably fail to match any mined pattern. As a result, the discovered patterns can neither capture the relevance between all items nor achieve the goal in this chapter. Considering the order between items, some sequential pattern mining (SPM) based recommendation methods are proposed, such as (Yap et al. 2012). They assume a rigid order over items within transactions (Han, Pei, Mortazavi-Asl, Chen, Dayal & Hsu 2000), which may not always be the case. For instance, it makes no difference whether milk or bread is put into the cart firstly.

Markov Chain (MC) (Rendle et al. 2010) is another straightforward way to model sequential data and thus can be used for TBRS on sequence data. (Rendle et al. 2010) used Markov Chain to estimate the transition probability from current item to the next item and thus make prediction based on this probability. (Wu et al. 2013) proposed Personalized Markov Embedding (PME) to map the users and songs to an Euclidean space by modeling the sequential singing behaviours, the prediction and recommendation are conducted on the base of the embeddings. Although sequential behaviour prediction based on Markov Chain is effective for capturing the transition preferences of certain users and thus make good recommendations, it is essentially based on item order within a transaction, which is not always available in real-world business and such method only captures the first-order dependency between items. Recently, matrix factorization (MF) (Chou et al. 2016) is used to factorize the transition probability from current item to the next one to the latent factors of each item and user. (Rendle et al. 2010) combined MF and MC for next-item recommendation, the latent user and item representations from the {user, item, last item} triplets are learned for next-item recommendation. Similar to sequential patterns, both MC and MF were originally designed for time-series data with rigid natural order, which limit their applications in TBRS, where the order between items within a transaction usually makes no difference. Moreover, they cannot handle those novel and cold start items well as the transitions between them and other items tend to be weak due to their low frequencies. To the best of my knowledge, there is no existing literature particularly works on the cold start issues in session-based recommender systems so far.

The above illustrations reveal the difficulty of modeling the context especially for dynamic context. In practice, the next choice is not only affected by one item or part of items in front of the target item, but by all items bought in the transaction event (i.e., the whole context). It is important to model the whole context and learn the relevance between the whole context and the target item. Furthermore, the intra-transaction relevance over items are greatly driven by their intrinsic nature, in another words, there are complex coupling relations (Cao 2015) between item features and item relevance. Such relations are particularly critical for those novel items with quite a few transaction records. For example, milk and bread are always bought together probably due to their different but closely related categories 'drink' and 'food'. This indicates that not only the context items but also the features of these items can affect the choice of the target item. To capture the indicators on the next choice as more as possible, I propose a neural-network-based comprehensive transaction embedding model (NTEM) to learn the embeddings of both items and their features when modeling the relevance between the context and the known choice. The model is comprehensive for several reasons: it models the relations between the target item and the whole context rather than part of it. It learns the embedding of the two important aspects (e.g., items and their features) of a transaction at the same time. During model training, the coupling relations between item relevance and item features are learnt and encoded into feature embeddings, which is useful for novel item discovery and recommendation. Though comprehensive, the embedding model has a shallow and wide network structure containing only one hidden layer, which guarantees its efficiency to find the best next choices over thousands of candidate items when the given context changes over time. This is suitable for online recommendation.

6.1.2 My Design and Main Contributions

Inspired by the great success of modern word embedding models, such as Word2Vec (Mikolov, Sutskever, Chen, Corrado & Dean 2013), in natural language processing (NLP) domain, I propose a shallow and wide network-based transaction embedding model (NTEM) to learn the relevance between different items efficiently on a large number of items with its wide-in-wide-out structure. Such relevance is learnt by capturing both the explicit relevance between items from the shopping-basket data and the implicit relevance from item features together with the coupling relations between them and the item relevance. It is noted that a deep structure is not efficient for online recommendation due to the long computational time needed to deal with thousands of items in real time. The Word2Vec cannot be directly applied to RS for two reasons: on one hand, it lacks of necessary element to incorporate the item features. On the other hand, the words in NLP often have a strict sequence, which is different from my case.

My model, NTEM has a three-layer network structure consisting of input layer, embedding layer and output layer as shown in Figure 6.1. The input layer contains double wide-in data vectors, the contextual itemset is collected from one while their corresponding features are acquired from the other. The embedding layer learns the item embeddings and feature embeddings respectively. The target item is then predicted by the output layer taking the embeddings of contextual items and features as the input. The NTEM learns the relevance between items with comprehensive transaction embeddings using a concise network structure. The main contributions of this work are summarized below.

(1). I model the whole context using a comprehensive network-based transaction embedding model for the next choice prediction.

(2). A TBRS model is proposed, which does not require the strict order over items within one transaction. This is more consistent with the real-world case.

(3). I incorporate item features into the model and encode the feature-item relevance coupling relations into feature embeddings, which makes my model also work well on cold start cases.

(4). I propose a shallow and wide network, which recommends the next item efficiently on large number of items under dynamic context.

6.2 Transaction Embeddings for Online Recommendation

In this section, I start with the problem formulation, then I talk about the proposed NTEM model including the network architecture in the model and

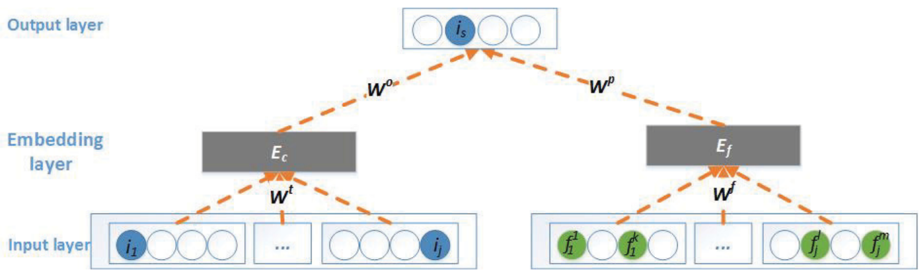


Figure 6.1: The NTEM architecture, which learns item embeddings and feature embeddings for target item prediction based on contextual items and their features

the model construction, finally I illustrate how to train the model and how to make prediction and thus produce recommendations for online shopping using the trained model.

6.2.1 Problem Formalization

Let $T = \{t_1, t_2...t_{|T|}\}$ be a set of transactions, each transaction $t = \{i_1, i_2...i_{|t|}\}$ contains a set of items, where $|T|$ denotes the number of elements in set T. All the items occurred in all transactions constitute the whole item set $I = \{i_1, i_2...i_{|I|}\}$. Let $F = \{f^1, f^2...f^{|F|}\}$ be a set a features which describe the items from I. Each item i is described by a set of feature values $F_i = \{f_i^1, f_i^2...f_i^{|F|}\}$. Note that the items in one transaction t may not have a rigid order, which is consistent with the real-world cases. Given the set of context itemset $\mathbf{c}$, my NTEM is constructed and trained as a probabilistic classifier that learns to predict a conditional probability distribution $P(i_s|\mathbf{c})$, where $\mathbf{c} \subseteq t \backslash i_s$ is the context from a transaction $t \in T$ w.r.t the target item i_s. This is similar to the bag of word (BOW) model in natural language processing, which trains a classifier to learn a conditional probability distribution $P(w_j|w_I)$, where w_I is the context consisting of several words of the target word w_j (Rong 2014). Similar to BOW, for each target item $i_s \in t$, the transaction context is $\mathbf{c} = t \backslash i_s$, namely all the items except the target one in the transaction are picked up as the context. Totally $|t|$ training instances are built for each transaction t by picking up one item as the target one each time.

Since I want to capture more information from the context for prediction, the features of items are added to the model as part of context, which result in $\mathbf{c}_f =< \mathbf{c}, F_c >$, where $F_c = \{F_i | i \in \mathbf{c}\}$ is the corresponding features of the items from $\mathbf{c}$. Thus my NTEM model is refined to predict the conditional probability distribution $P(i_s|\mathbf{c}, F_c)$ when the transaction-feature context $< \mathbf{c}, F_c >$ is given. We call $\mathbf{c}$ and F_c as transaction context and feature context respectively in this chapter. Thus, the TBRS is reduced to ranking all candidate items in terms of their conditional probability over the given transaction-feature context. Note that in the prediction stage, the conditional probability is computed based on the embeddings of item set $\mathbf{c}$ and its corresponding features F_c learned in the training stage.

Particularly, the incorporation of features contributes greatly to the recommendation of novel items. Due to the low frequencies of novel items in training set, the embeddings of these items may not be learned well during the model training process and it leads to poor prediction. Thanks to the feature embeddings synchronously learned with the item embeddings, the intra-transaction item relevance can be partly encoded into feature value embeddings of novel items. In addition, part of the feature values of novel items may already be embedded when encoding those of frequent items as some feature values may be shared between frequent items and infrequent ones.

6.2.2 Neural-Network-based Transaction Embedding Model (NTEM)

In this section, I mainly talk about the details of constructing NTEM and learning its parameters.

Giving a context $< \mathbf{c}, F_c >$ to the input layer, the input units in the bottom left corner of Figure 6.1 constitute a one-hot encoding vector where only the units at position i_j ($i_j \in \mathbf{c}$) is set to 1 and all other units are set to 0. For each $i \in \mathbf{c}$, I encode it in the same way as i_j. Note that items may have both numerical and categorical features in real-world business. In this work, I only consider those categorical features. For a value from a categorical feature f ($f \in F$) with m different values, I transform it to a $1 \times m$ vector using one-hot encoding. Suppose $V = \sum n_k$ is the total number of distinct values of all features, where n_k is the number of distinct values in feature f^k. For the features of a given item, a $1 \times V$ vector is achieved by doing the transformation of all feature values first and then concatenate all the transformed vectors together. Thus the input layer for each training example consists of $|\mathbf{c}|$ item vector with length $|I|$ and $|\mathbf{c}|$ item feature vectors with length V.

In the input layer, items and item features are represented by sparse one-hot item and feature vectors. In NTEM, I create an embedding mechanism to map these vectors to an informative and lower-dimensional vector representation in the embedding layer, where a K-dimension vector $\mathbf{E}_i \in [0,1]^K$ is used to represent the item embedding. The transaction context weight matrix $\mathbf{W}^t \in \mathbb{R}^{K \times |I|}$ is used to fully connect between input-layer and embedding-layer. Where the i^{th} column $\mathbf{W}^t_{:,i}$ encodes the one-hot vector of item i to the embedding $\mathbf{E}_i$ using the commonly used logistic function $\sigma(\cdot)$.

$$\mathbf{E}_i = \sigma(\mathbf{W}^t_{:,i}) \tag{6.1}$$

To make the training and prediction more stable, here I use the nonlinear embeddings as they are bounded in [0,1] compared to the linear embeddings. Furthermore, the nonlinear embeddings are more expressive than linear one though they may involve a little more computation cost. After embedding all items in $\mathbf{c}$, I can obtain the embedding $\mathbf{E}_c \in [0,1]^L$ of transaction context $\mathbf{c}$ by combining all embeddings of items in such context. As illustrated in the following equation, the transaction context embedding is built as a combination of $\mathbf{E}_i, i \in \mathbf{c}$.

$$\mathbf{E}_c = \sum_{i \in \boldsymbol{c}} \omega_i \mathbf{E}_i \tag{6.2}$$

where $\sum_{i \in \boldsymbol{c}} \omega_i = 1$. The combination weight ω_i for each item i in context $\mathbf{c}$ can be assigned to different values according to specific applications. For instance, in sequential data, the weights decay with time span to the target item. As illustrated in the introduction part, I treat the items within a transaction as unordered, so uniform weights are used in this chapter, i.e., the items in context are equally important for the prediction of the target item.

Similarly, I use the feature context weight matrix $\mathbf{W}^f \in \mathbb{R}^{L \times V}$ to encode the one-hot item feature vector F_i of item i to the embedding $\mathbf{E}_{F_i} \in [0,1]^L$.

$$\mathbf{E}_{F_i} = \sigma(\mathbf{W}^f_{:,F_i}) \tag{6.3}$$

Similar to transaction context, I combine the embeddings of features of all items from $\mathbf{c}$ to construct the whole feature embedding $\mathbf{E}_{F_c}$ as below.

$$\mathbf{E}_{F_c} = \sum_{i \in \mathbf{c}} \omega_{F_i} \mathbf{E}_{F_i} \tag{6.4}$$

where $\sum_{i \in \mathbf{c}} \omega_{F_i} = 1$. Uniform weights are assigned to the feature embedding of each item for the same reason as ω_i illustrated above.

The output weight matrices $\mathbf{W}^o \in \mathbb{R}^{|I| \times K}$ and $\mathbf{W}^p \in \mathbb{R}^{|I| \times L}$ is used to fully connect the embedding-layger and output-layer as depicted in the top of Figure 6.1. With the embeddings of given contextual itemset $\mathbf{c}$ and its features F_c, plus the weight metrics $\mathbf{W}^o$ and $\mathbf{W}^p$, the score S_{i_s} of a target item i_s w.r.t the given context $< \mathbf{c}, F_c >$ is computed as:

$$S_{i_s}(\mathbf{c}, F_c) = \mathbf{W}^o_{s,:} \mathbf{E_c} + \mathbf{W}^p_{s,:} \mathbf{E}_{F_c} \tag{6.5}$$

where $W^o_{s,:}$ denotes the s^{th} row of W^o. This score quantifies the relevance of the target item i_s w.r.t the given context $< \mathbf{c}, F_c >$. As a result, the conditional probability distribution $P_\Theta(i_s|\mathbf{c}, F_c)$ can be defined in terms of softmax function, which is commonly used in neural network or regression model.

$$P_\Theta(i_s|\mathbf{c}, F_c) = \frac{exp(S_{i_s}(\mathbf{c}, F_c))}{Z(\mathbf{c}, F_c)} \tag{6.6}$$

where $Z(\mathbf{c}, F_c) = \sum_{i \in I} exp(S_i(\mathbf{c}, F_c))$ is the normalization constant and $\Theta = \{\mathbf{W}^t, \mathbf{W}^f, \mathbf{W}^o, \mathbf{W}^p\}$ is the model parameters. Thus a probabilistic classifier modeled by my NTEM is obtained.

6.2.3 Learning and Prediction

In the above subsection, I have built a probabilistic classifier based on the transaction and item feature information data $b =< \mathbf{g}, i_g >$, where $\mathbf{g} =< \mathbf{c}, F_c >$ is the input data, namely the transaction-feature context, and i_g is the corresponding observed output, namely an item bought together with the given transaction context $\mathbf{c}$. Given a training dataset $D = \{< \mathbf{g}, i_g >\}$, the joint probability distribution over it is obtained:

$$P_\Theta(D) \propto \prod_{b \in D} P_\Theta(i_g|\mathbf{c}, F_c) \tag{6.7}$$

As a result, the model parameters Θ can be learned by maximizing the conditional log-likelihood:

$$L_\Theta = \sum_{b \in D} logP_\Theta(i_g|\mathbf{c}, F_c) = \sum_{b \in D} S_{i_g}(\mathbf{c}, F_c) - logZ(\mathbf{c}, F_c) \tag{6.8}$$

Evaluating L_Θ and evaluating the corresponding log-likelihood gradient involve the normalization term $Z(\mathbf{c}, F_c)$, which needs to sum $exp(S_{i_g}(\mathbf{c}, F_c))$ over the whole itemset for each training instance. That is to say, training this model take $|I| \times |D|$ times of computation to get the normalization constant for each iteration, which makes the training process intractable. To tackle this problem, I adopt a sub-sampling approach, namely noise-constrictive estimation (NCE) (Gutmann & Hyvärinen 2012) to deal with the normalization calculation of softmax function in the training process. I sample 50 negative items each time in the experiment.

All the parameters Θ are learned by back propagation. Algorithm 6.1 summarizes the learning process briefly. In Algorithm 6.1, $\odot$ denotes element-wise product operation, i_o is the output item which includes both positive example and noise examples. $i_j \in \mathbf{c}$ is the input item from the context and ω_{i_j} is the corresponding combination weight used in Equation (6.2). Similarly, F_{i_j} is the corresponding input feature of item i_j and $\omega_{F_{i_j}}$ is its corresponding combination weight used in Equation (6.4).

Algorithm 6.1 NTEM Parameter Learning Using Gradient Descent

```
1: $l \leftarrow 0$
2: **while** not converged **do**
3:     Compute $w^o_{i_o}$-gradient (Equation (6.1)): $g_{w^o_{i_o}} \leftarrow \mathbf{E}_c^\top$
4:     Compute $w^p_{i_o}$-gradient (Equation (6.3)): $g_{w^p_{i_o}} \leftarrow \mathbf{E}_{F_c}^\top$
5:     Compute $w^t_{:,i_j}$-gradient (Equation (6.5)):
       $g_{w^t_{:,i_j}} \leftarrow \omega_{i_j} \mathbf{W}^{o\top}_{i_o,:} \odot \mathbf{E}_i \odot (1 - \mathbf{E}_i)$
6:     Compute $w^f_{:,F_{i_j}}$-gradient (Equation (6.5)):
       $g_{w^f_{:,F_{i_j}}} \leftarrow \omega_{F_{i_j}} \mathbf{W}^{p\top}_{F_{i_o},:} \odot \mathbf{E}_{F_i} \odot (1 - \mathbf{E}_{F_i})$
7:     Perform SGD-updates for $w^o_{i_o}$, $w^p_{i_o}$, $w^t_{:,i_j}$ and $w^t_{:,i_j}$:
       $w^o_{i_o} \leftarrow w^o_{i_o} + S^l_{i_o}(g) g_{w^o_{i_o}}$, $w^p_{i_o} \leftarrow w^p_{i_o} + S^l_{i_o}(g) g_{w^p_{i_o}}$, $w^t_{:,i_j} \leftarrow w^t_{:,i_j} + S^l_{i_o}(g) g_{w^t_{:,i_j}}$, $w^f_{:,F_{i_j}} \leftarrow w^f_{:,F_{i_j}} + S^l_{i_o}(g) g_{w^f_{:,F_{i_j}}}$
8:     $l \leftarrow l + 1$
9: **end while**
```

Due to the large computation cost and the strength in matrix calculation of GPUs, a GPU-based adaptive stochastic gradient descent (SGD) optimizer is designed to speed up the training process.

6.3 Experiments and Evaluation

6.3.1 Experimental Setup

Data Preparation

I evaluate my method on two real-world transaction data sets: IJCAI-15 [1] and Tafang [2]. First, a shopping-basket-based transaction table and an item information table are extracted from each of the original data. The transaction table contains multiple transactions and each transaction consists of multiple items. Note that those transactions containing only one item are removed as they can not fit my model. This is because, I use at least one item as context for constructing the embeddings. The item information table contains the feature values of each item occurred in the transaction table, note that I only focus on those categorical features in this chapter. Secondly, the transaction table is splitted into training and testing set. Specifically, I randomly choose 20% from the transactions happened in last 30 days as the testing set, while others are used for training. The item information table is used in both training and testing processes. Finally, to test the performance of my proposed model under different cold-start levels, part of the transactions in training set are removed following certain rules. To be specific, I construct 4 different training sets with a drop rate of $0, 40\%, 80\%, 95\%$ respectively. Taking the one with drop rate of 40% as an example, for each target item selected in the testing set, 40% of all the transactions containing it in the training set are dropped. The characteristics of the datasets are shown in Table 6.1.

During the training, the transactions in the training set together with the corresponding item features are imported into the model in batches to learn embeddings of each item and each feature value. In the testing process, the learned embeddings are used to predict the target item given the $(n-1)$ ones.

[1]https://tianchi.aliyun.com/datalab/dataSet.htm?id=1

[2]http://stackoverflow.com/questions/25014904/download-link-for-ta-feng-grocery-dataset

Table 6.1: Statistics of experimental datasets

Statistics	**IJCAI-15**	**Tafang**
#Transactions	144,936	19,538
#Items	27,863	5,263
#Features	3	1
Avg. Transaction Length	2.91	7.41
#Training Transactions	141,840	18,840
#Training Instances	412,679	141,768
#Testing Transactions	3,096	698
#Testing Instances	9,030	3,150

The real target item in the testing set is used as the ground truth. I calculate accuracy measures Recall@K (Yuan et al. 2013) and MRR (Chou et al. 2016) by comparing the predicted results and the ground truth. I also calculate the recommendation novelty measures: global novelty and local novelty by comparing the recommendation list and the whole item population and the given context item set respectively. Finally, the performance of my proposed method is evaluated by comparing it and other related ones in terms of recommendation accuracy and novelty.

Evaluated Methods

In the experiments, I compare my proposed NTEM with the three commonly used baselines: *FPMC*, *PRME* and *GRU4Rec*. All of them have been introduced in Section 2.4 in Chapter 2.

6.3.2 Performance Evaluation

Except for the traditional recommendation accuracy, I also evaluate the recommendation novelty to test the capability of the proposed method to recommend novel items.

Table 6.2: Accuracy comparisons between different recommendation models

		IJCAI-15			Tafang		
Scenario	Model	REC@10	REC@50	MRR	REC@10	REC@50	MRR
drop 0	FPMC	0.0016	0.0025	0.0031	0.0189	0.0216	0.0089
	PRME	0.0555	0.0612	0.0405	0.0212	0.0305	0.0102
	GRU4Rec	0.1182	0.1566	0.0965	0.0428	0.0887	0.0221
	NTEM	**0.2026**	**0.3224**	**0.1125**	**0.0689**	**0.1716**	**0.0231**
drop 40%	FPMC	0.0012	0.0021	0.0026	0.0008	0.0010	0.0058
	PRME	0.0327	0.0411	0.0312	0.0102	0.0205	0.0095
	GRU4Rec	0.1108	0.1356	0.0868	0.0330	0.0659	0.0196
	NTEM	**0.1928**	**0.2794**	**0.1117**	**0.0575**	**0.1049**	**0.0377**
drop 80%	FPMC	0.0009	0.0017	0.0021	0.0005	0.0008	0.0020
	PRME	0.0212	0.0287	0.0215	0.0084	0.0125	0.0056
	GRU4Rec	0.0493	0.0611	0.0398	0.0110	0.0244	0.0054
	NTEM	**0.1098**	**0.1450**	**0.0686**	**0.0254**	**0.0494**	**0.0072**
drop 95%	FPMC	0.0003	0.0008	0.0012	0.0002	0.0004	0.0008
	PRME	0.0089	0.0113	0.0105	0.0071	0.0096	0.0043
	GRU4Rec	0.0233	0.0337	0.0173	0.0101	0.0172	0.0042
	NTEM	**0.0318**	**0.0639**	**0.0173**	**0.0215**	**0.0305**	**0.0068**

Accuracy Evaluation

I use the widely used accuracy metrics (i.e., *REC@K* and *MRR*) for transaction-based recommendation defined in Section 2.3.1 to evaluate all the comparison approaches.The result of each approach is given in Table 6.2.

Table 6.2 demonstrates the results of REC@10, REC@50 and MRR over the testing sets of different cold-start levels. The number of factor is set to 10 for training FPMC as the best performance is achieved in such setting. The performance of FPMC on both data sets is quite poor, even in the warm start situation (e.g., REC@50=0.0025 when drop rate=0 on IJCAI-15). The main reason is that both data sets are extremely sparse and thus a very large but quite sparse matrix is constructed to train the MF model for each data set. For instance, in IJCAI-15 data set, each user only has an average of 3.6 transactions and each transaction only contains an average of 2.91 items of over 27,000 ones. This leads to that each row of the constructed matrix contains less than two items (cf. the avg. transaction length in Table 6.1, note that one out of the items need to be taken out as the output) and all other entries are empty. By calculation, I found the non-empty entries in the constructed matrix of IJCAI-15 account for less than 0.01%. I set the embedding dimensions to 60 as suggested in (Feng et al. 2015) when training PRME model. The performance of PRME is a little better than FPMC, but it's still poor especially in those cold start cases. This is because PRME is a fist-order MC model, which learns the transition probability over successive item instead of the whole context. Namely, this model only predict the target item based on its previous one while ignore all those bought before in the same transaction, which lost some information. Furthermore, in the real-word, the choice of items does not follow a rigid sequence assumed by such kind of models. Benefiting from the deep structure, GRU4Rec achieves much better performance compared with FPMC and PRME models. Especially, when I drop no more than 40% transactions, the REC@10 is above 10% on IJCAI-15, which can lead to a relative accurate recommendation in real-world business.

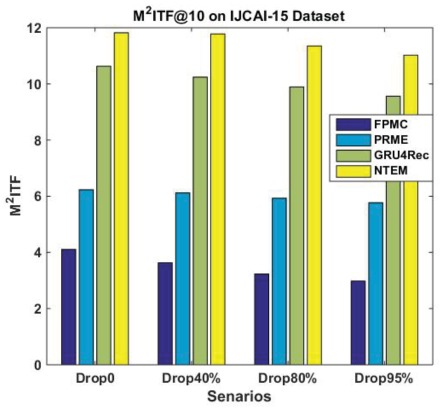
The batch size is empirically set to 200 and number of hidden units for the item and feature value embeddings are set to 50 and 20 respectively. I run 60 epochs to train my NTEM model. It achieves much better performance than GRU4Rec, where the REC@10 and REC@50 exceed 20% and 30% respectively when drop nothing on IJCAI-15. In the extremely cold start case (drop 95%), it also achieves obvious better performance than other methods on both data sets. The REC@10 and REC@50 of my model exceed 3% and 6% respectively, compared to around 2% and 3% of GRU4Rec on IJCAI-15. The higher MRR of my model also shows that we can accurately put the customers' desired items in the front of the recommendation list. The reason is multifaceted. First of all, I use a complete context to predict one target item, and I do not assume a rigid order between the items within one transaction. This makes the input more informative and consistent with the reality compared with those models which only utilize part of the context and assume a rigid order between items. Second, more information is used by my model by incarnating the item features into it. More importantly, the coupling relations between item features and its transactions are learned and encoded into item and its feature embeddings in the back-forward propagation training mechanism. Thanks to the features and the coupling relations, additional information is provided to help with the item prediction and recommendation, especially for those novel items which lack of sufficient transaction information. What's more important, my model has a very concise structure which is easy to train. This shallow structure is efficient enough to retrain and recompute the score for ranking of all candidate items in an incremental dataset for online recommendations. However, GRU4Rec is a deep RNN consisting of GRU layers, which makes it more time consuming when new transaction records are added into the dataset.

Novelty Evaluation

Except for accuracy, novelty is another important quality which should be considered in real-world recommendation scenarios (Vargas & Castells 2011). Recall that in this paper I also try to recommend those infrequent or unpopular items, namely novel items, so the novelty is particularly important to measure the recommendation quality of my model. Specifically, I evaluate the recommendation novelty from both global perspective and local perspective by using different metrics.

Accordingly, a global novelty measure M^2ITF and a local novelty measure $MCAN$ are defined in Section 2.3.2 to measure the recommendation performance from novelty perspective.

Figure 6.2 and Figure 6.3 show the results of global novelty and local novelty comparisons of top-10 recommendations over testing sets using the aforementioned measures M^2ITF@10 and $MCAN$@10 respectively on both data sets. Overally, my proposed NTEM achieves both higher global novelty and local novelty compared to other approaches. FPMC is not trained well



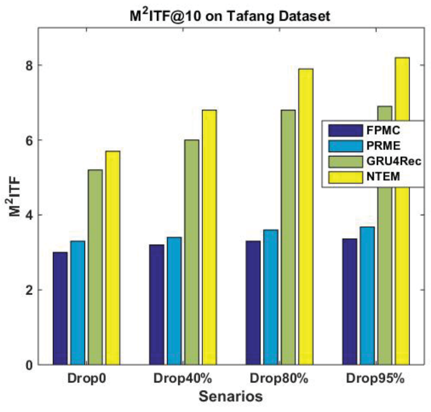


Figure 6.2: M^2ITF comparisons on top-10 items, NTEM achieves higher global novelty than other approaches

on such sparse dataset and can not work well, so its recommendations are not precise and have a lot of randomness. Recall the fact that most of the candidate items in the dataset are frequent and not novel. Accordingly, we get low M^2ITF and $MCAN$ on FPMC. PRME is a first-order Markov Chain model in which a sequential hypothesis is forced between the items within a transaction and it predicts the target item by only utilizing the one before it in the transaction. This is not always the case in the real world and it leads to information loss by ignoring other items in the transaction, so PRME also gets low novelties here. GRU units in GRU4Rec can accumulate the effect of all the sequential items in a transaction, it makes use of all other items in the transaction to predict the target one. Taking the advantage of the deep structure, it achieves relative high novelties. As a result, GRU4Rec is a relative good TBRS to make novel recommendations.

My proposed NTEM do not have the sparse issue as FPMC due to its totally different work mechanism from matrix factorization used in FPMC, so it works well even on a sparse data set as shown in my experiment. Furthermore, NTEM does not assume a sequence between the items within a transaction, which is closer to the actual situation. More importantly, we make full use of all the other items in a transaction to predict the target

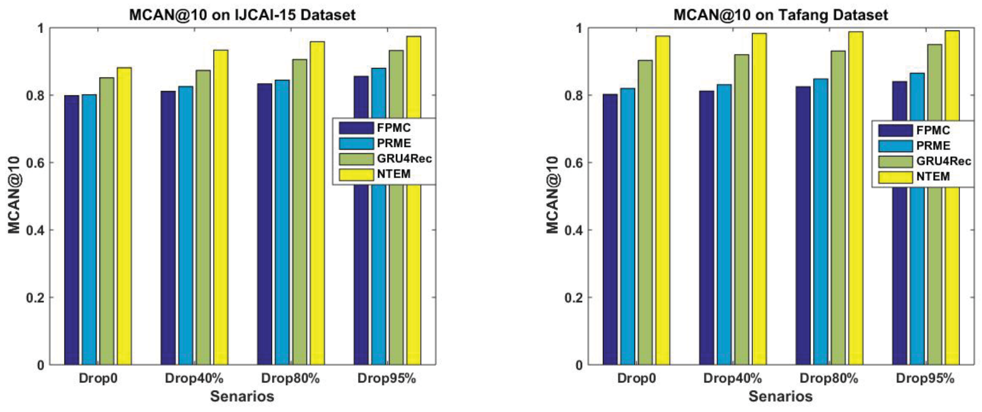


Figure 6.3: $MCAN$ comparisons on top-10 items, NTEM achieves higher local novelty than other approaches

one, which makes the prediction more solid by utilizing richer information. For the cold start issue, we incorporate the item features into the model and take the advantage of the information flow on feature values for prediction when the available transaction information is limited. So NTEM is more easily to provide novel recommendations and to achieve higher global and local novelties.

In practice, both the global and local novelties are somehow related to the recommendation accuracy here, specifically, some kind of positive relations exist between them. In the cold start scenarios, as all the target items are infrequent or novel, higher recommendation accuracy means more chance to get the target items into the recommendation list. Whereas more target items included in the recommendation list lead to higher novelty.

6.4 Discussions

Currently, I have to admit a weakness of my work on dealing with the purely cold items though it can get good results in the extremely high drop rate (i.e. 95%) case. I try to analyze the reasons and provide some suggestions in this section. On one hand, the model parameters are learned on training set, so they tend to reveal the existing patterns in training set naturally. However, the purely cold items never appear in the training set, nothing is learned for them when training. On the other hand, for those purely cold items, only their features are used for predictions. This requires their feature values have occurred frequently in the training set and thus good embeddings can be learned. In addition, the item features also affect the prediction of new items greatly. In general, high-dimensional features have stronger capabilities to deliver information than low-dimensional ones. Unfortunately, the shopping-basket datasets in transaction domain usually have diverse feature values and low dimensions. For instance, the IJCAI-15 dataset only has three features: category, brand and seller. The average frequency of feature values in brand and seller is below five, which is quite infrequent. This leads to the feature values of new items in testing set occur too few times or even never occur, which leads to learning poor embeddings of them and thus poor information delivering capacity. Some potential solutions to improving my model for dealing with purely cold items may be: (1) finding more suitable datasets with high-dimensional features and intensive feature values, like audio data, image data or text data; (2) only using those features having already occurred in training set.

6.5 Summary

Perceiving the next choice in a dynamic context especially for onling recommendation circumstance is demanding but challenging. In this chapter, I propose NTEM to build a more precise and efficient TBRS, it also shows great potential to improve the recommendation novelty. With the integration of both high level transaction information and low level item feature information, more comprehensive two level relations between items are captured for session-based recommendations. Particularly, such integration structure benefits greatly the cold start issue in session-based RSs. The empirical evaluation on real-world e-commerce datasets shows the comprehensive superiority of my methods over other state-of-the-art ones. In the future, I will extend my model to other domains, like relational learning on social networks.

Chapter 7

Jointly Modeling Intra- and Inter-transaction Dependencies for Next-Item Recommendations

7.1 Introduction

In this chapter, I go up to the session level and focus on some critical challenges on this level in session-based recommender systems: inter-session dependency, session heterogeneity and irrelevant sessions as shown in Figure 1.1 in Chapter 1. These challenges are unified to the inter-session dependency modeling issue which is the fourth challenge I addressed in this thesis as shown in the thesis structure Figure 1.2 in the first chapter. Similar to item heterogeneity in Chapter 5, *session heterogeneity* means that out of all recent sessions taken into account, different ones are actually relevant to the current session in different degrees, some are more relevant and others may not or even irrelevant. To keep it consistent with previous chapters, session-based recommender systems are specialized to transaction-based recommender systems in this chapter for the same reason.

7.1.1 Target Problem and Motivation

Given a transactional context, which is a set of recent transactions together with several existing items in the current transaction, a transaction-based recommender system (TBRS) tries to predict the next item a user is likely to choose. It is usually formalized as a transaction-based next-item recommendation problem (Wang, Hu & Cao 2017). The set of recent transactions is called inter-transaction context while the already-chosen items form the intra-transaction context (Yap, Tan & Pang 2007). Generally speaking, the main challenge of next-item recommendations is to comprehensively capture the complex dependency embedded in the transaction data. Such dependency can be categorized into *intra-transaction dependency* between the intra-transaction context and the target items and *inter-transaction dependency* between the inter-transaction context and the current transaction (Chiueh & Bajpai 2008).

In the transactional data example shown in Figure 7.1, a user has two recent transactions t_1 and t_2, and the current transaction t_3. I consider the item

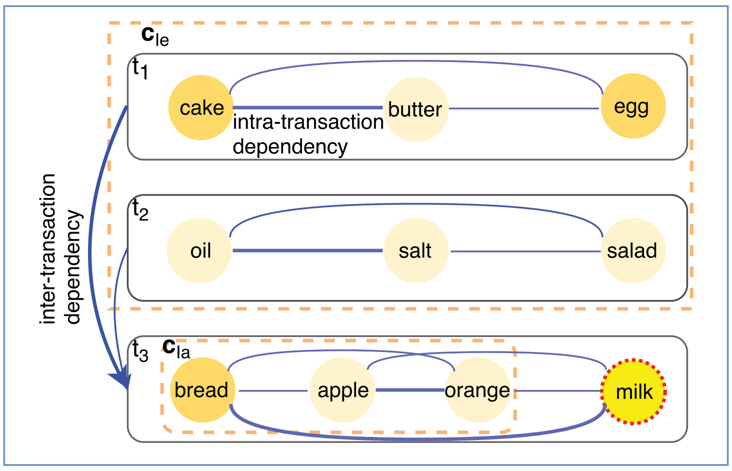


Figure 7.1: An example of user transactions. Thicker lines and darker circles indicate stronger dependencies and items more relevant to *milk*, while $\mathbf{c}_{Ia}(\mathbf{c}_{Ie})$ represents the intra (inter)-transaction context.

milk from t_3 as the target to recommend and all other prior transaction information as the corresponding context. Existing transaction-based next-item recommenders may suggest *salad* by considering only the inter-transaction items *apple* and *orange* in t_3, which may not be accurate as *salad* was just bought in t_2. Moreover, from the intra-transaction perspective, the choice of *milk* may depend much more on *bread* than on *apple* or *orange*. In such a case, a TBRS should be able to pay more attention to *bread* when modelling intra-transaction dependency. From the inter-transaction perspective, *milk* may also be influenced by *cake* and *egg* bought in t_1 but less related to t_2. This indicates that a good TBRS should not only take t_1 and t_2 into account but also concentrate much more on t_1. This example shows the importance of inter-transaction dependency and the significance of discriminating the contribution scales of different items and transactions according to their relevance to the next chosen item. Various approaches have been proposed to model the transaction dependencies for next-item recommendations. Pattern-based RSs predict the next item by using mined frequent patterns (Yap et al. 2012). Although easy to implement, the 'support' constraint filters out many infrequent but interesting items and thus leads to information loss. Rule- and pattern-based RSs are well-studied recommendation approaches, that predict the next item by using mined frequent patterns (Yap et al. 2012): specifically, (Bendakir & Aïmeur 2006) mined rules associating courses and students for course recommendations. To capture the transition between a sequence of songs, (Hariri, Mobasher & Burke 2012) discovered sequential patterns for next-song recommendations. Although simple and effective, these methods often overlook infrequent items (Wang, Hu & Cao 2017). More importantly, they capture only the co-occurrence relationships within transactions while ignoring the available inter-transaction dependency. Markov chain (MC) models (Rendle et al. 2010) offer another way to model inter-item transitions within transactions. Personalized Markov Embedding (PME) generates the embeddings of users and items in a Euclidean space for next-song recommendations (Wu et al. 2013). Recently, to learn users personalized sequential check-in information, a personalized ranking metric embedding method (PRME) was proposed for next POI recommendations (Feng et al. 2015). Both PME and PRME are first-order MC models, where the higher-order dependencies are ignored and where the assumed rigidly ordered data may not always be available. More importantly, they are limited to the intra-transaction relations only, neglecting the inter-transaction dependency, which may lead to unreliable recommendations. To capture higher order dependency in sequential data, recurrent neural networks (RNN) (Hidasi et al. 2015) have been successfully applied in TBRSs. A gated recurrent unit (GRU)-based RNN was proposed to capture long-term dependency within transactions (Hidasi et al. 2015), but the high computational cost caused by the complex structure prevents its application to large data. Compared to deep architectures, shallow networks are usually more efficient in dealing with such issues, especially on large datasets. Particularly, the Word2Vec model has achieved great success in learning the probability distribution over a sequence of words (Goth 2016). However, both networks treat the nearest items as the most important, and this treatment may degrade the influence of relevant items due to their rigid order assumption. More importantly, they are still limited to the intra-transaction dependency modelling and ignore the inter-transaction one. Moreover, both MC and RNN assume a rigid order of items and thus the next choice is assumed to depend more on the recent items. Therefore, those truly relevant contextual items may not be attended to. Such attention is quite important, though, for intra-transaction dependency modelling, especially for long transactions often containing irrelevant items. More importantly, all these approaches capture only intra-transaction dependency, ignoring inter-transaction dependency, which may impact the next item (Chiueh & Bajpai 2008), especially for periodic transactions. However, not all recent transactions relate to the next choice, priority should be given to related transactions.

7.1.2 My Design and Main Contributions

This paper addresses the above issues by proposing a novel *hierarchical attentive transaction embedding* (HATE) model. The HATE first builds an attentive embedding for each transaction by emphasizing the relevant items in it and then builds attentive inter-transaction context embedding by highlighting those recent transactions more related to the current one and the next choice. Simultaneously, an attentive intra-transaction context embedding is built on the items chosen in the current transaction. Finally, a hybrid context representation is achieved by combining both inter- and intra-transaction context embeddings for next-item predictions.

Considering the large number of items in real-world datasets, it is practical to incorporate attention (Vaswani et al. 2017) into a shallow network (Goth 2016) in building a concise but powerful structure for attentive context representation learning. As a result, the proposed model is capable of capturing both intra- and inter-transaction dependency attentively and the resultant context representation is more informative to predict the next item. My validation on two real-world transaction datasets verifies the necessity of combining the inter-transaction dependency with the attention mechanism.

Accordingly, major contributions of this model include the following:

- I notice and highlight the significant influence of inter-transaction dependency on the next-item recommendation tasks. Subsequently, a framework is proposed to incorporate inter-session dependency into such typical session-based recommender sytsems.

- A hierarchical attentive transaction embedding model is proposed to learn the context representation for transaction-based item recommendations by attentively capturing both intra- and inter-transaction dependencies.

- A shallow and wide network is designed for efficiently learning context representations over a large number of items and transactions.

In summary, my model relaxes the rigid order assumption over items within a transaction, which broadens its applications. Empirical evaluation shows that (1) HATE outperforms the state-of-art TBRSs on real-world datasets by around 20%; and (2) the incorporation of an inter-transaction context or attention mechanism achieves at least a 10% improvement in accuracy.

7.2 Problem Statement

Given a transaction dataset, let $T = \{t_1, t_2 ... t_{|T|}\}$ be the set of all transactions, such that each transaction $t = \{i_1, i_2 ... i_{|t|}\}$ consists of a subset of items and is associated with a given user and a specified timestamp, where $|T|$ denotes the number of transactions in T. All the items occurring in all transactions constitute the whole item set $I = \{i_1, i_2 ... i_{|I|}\}$. Note that the items in a transaction t may not have a rigid order.

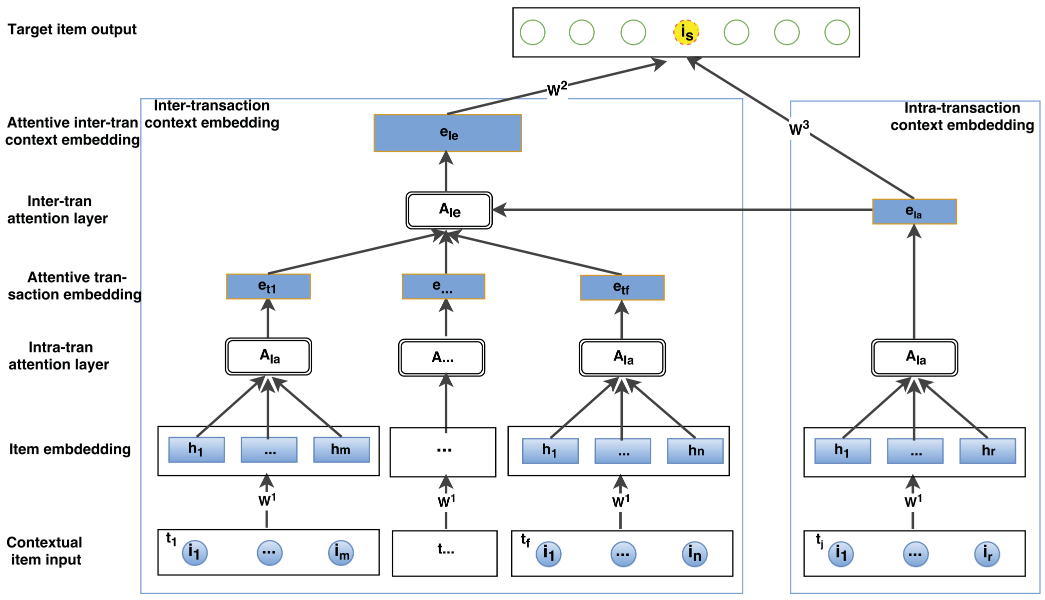

Target item output
i_s
W^2
W^3
Attentive inter-tran context embedding
Inter-transaction context embedding
Intra-transaction context embdedding
e_{Ie}
Inter-tran attention layer
A_{Ie}
e_{Ia}
Attentive tran-saction embedding
e_{t1}
$e_{...}$
e_{tf}
Intra-tran attention layer
A_{Ia}
A...
Item embdedding
h_1
hm
hn
hr
W^1
Contextual item input
t_1
i_1
i_m
t...
t_f
i_n
t_j
i_r

Figure 7.2: The HATE architecture: first learns item embeddings, then integrates them into intra-transaction context embedding or transaction embeddings on which inter-transaction context embedding are built, and finally feeds both intra- and inter-transaction embeddings into the output layer for target item prediction. $A_{Ia}(A_{Ie})$ represents the intra- and inte)-transaction attention model.

Given a target item $i_s \in t_j (j \neq 1)$, all other items in t_j from the intra-transaction context $\mathbf{c_{Ia}} = t_j \backslash i_s$. The recent transactions of the same that happened before t_j form the inter-transaction context $\mathbf{c_{Ie}} = \{t_1, t_2 ... t_{j-1}\}$. $\mathbf{c_{Ia}}$ and $\mathbf{c_{Ie}}$ together constitute the transactional context $\mathbf{c} = \{\mathbf{c_{Ia}}, \mathbf{c_{Ie}}\}$. Given the context $\mathbf{c}$, HATE is trained as a probabilistic classifier that learns to predict a conditional probability distribution $P(i_s|\mathbf{c})$. Therefore, TBRS aims to rank all candidate items in terms of their conditional probability over the given context.

7.3 Modelling and Learning

I first demonstrate the architecture of HATE and then discuss the details of model training, followed by computing predictions and recommendations after training the model.

7.3.1 The Hierarchical Attentive Transaction Embdedding Model

As shown in Figure 7.2, the proposed HATE model consists of two main parts: the transactional context embedding part at the bottom and the prediction part (output layer) at the top. The embedding part contains two modules: inter-transaction context embedding and intra-transaction context embedding. Specifically, the first part is a hierarchical structure which consists of an input layer, the item embedding, intra-transaction attention, transaction embedding, inter-transaction attention and inter-context embedding. The second part is built from the input layer to the intra-transaction context embedding layer. Next, I explain details of the working mechanism of the model from input to output.

Inter-transaction Context Embedding

I present item embedding, then transaction embedding, and lastly attentive inter-transaction context embedding.

Item Embedding For a given contextual item i_l from a transaction t, I create an embedding mechanism to map its ID number to an informative and low-dimensional vector representation in the item embedding layer, where a K-dimensional real-valued vector $\mathbf{h}_l \in \mathbb{R}^K$ is used to represent item i_l. The input weight matrix $\mathbf{W}^1 \in \mathbb{R}^{K \times |I|}$ is used to fully connect the input-layer and item embedding-layer. Note that actually the l^{th} column of $\mathbf{W}^1$ encodes item i_l to a real-valued embedding $\mathbf{h}_l$ as below. Several different mapping approaches including logistic function have been tried to map item ID to its embedding and the following way is found to achieve the best performance in my case.

$$\mathbf{h}_l = \mathbf{W}^1_{:,l} \tag{7.1}$$

Attentive Transaction Embedding When the embeddings of all the items in transaction t are ready, we can obtain the embedding $\mathbf{e}_t \in \mathbb{R}^K$ of contextual transaction t by integrating the embeddings of all items in t using the attention mechanism. Specifically, the attentive transaction embedding is built as a weighted sum of $\mathbf{h}_l$:

$$\mathbf{e}_t = \sum_{i_l \in \boldsymbol{t}} \alpha_{sl}\mathbf{h}_l, \quad s.t. \sum_{i_l \in \boldsymbol{t}} \alpha_{sl} = 1 \tag{7.2}$$

where α_{sl} is the integration weight of contextual item i_l w.r.t the target item i_s, indicating the contribution scale of i_l to the choice of i_s. In my model, to better capture the different contribution scales of contextual items, I develop an attention layer to learn the integration weights automatically and effectively. Compared to assigning the weights manually under certain assumptions, e.g., order assumption, or directly learning the weights without the attention mechanism, my method not only works more flexibly without assumptions but also emphasizes those important items and reduces the interference from irrelevant ones. Next, I demonstrate how the intra-transaction attention model achieves this goal.

Intra-Transaction Attention As with most attention models, I use a softmax layer to learn the weights of different contextual items w.r.t to the target item. In this way, items that are more relevant to the target item are given larger weights, and vice versa. The input of softmax is the transformation of each item's embedding:

$$\alpha_{sl} = \frac{exp(\sigma(\mathbf{h}_l))}{\sum_{i_v \in t} exp(\sigma(\mathbf{h}_v))} \tag{7.3}$$

$$\sigma(\mathbf{h}_l) = \mathbf{w}^{\alpha}\mathbf{h}_l^T \tag{7.4}$$

where $\mathbf{w}^{\alpha}$ is an item-level context vector shared by all contextual items, which can be seen as a high level representation of a fixed query 'which item is relevant to the target item?' over all the contextual items (Kumar et al. 2016).

The vector is randomly initialized and jointly learned during the training stage. As $\mathbf{w}^{\alpha}$ serves as a weight vector connecting the item embedding layer to the intra-transaction attention model, I denote it as an intra-transaction attention weight, to be consistent with input and output weights. Essentially, the importance of each item i_l is achieved by first calculating the similarity between its embedding $\mathbf{h}_l$ and the item level context vector $\mathbf{w}^{\alpha}$ and then normalizing it into an importance weight α_{sl} through a softmax function (Yang et al. 2016).

Attentive Inter-transaction Context Embedding Inter-transaction context embedding is built on top of the embedings of transactions included in the inter-transaction context. Specifically, the inter-transaction context embedding is computed as a weighted sum of transaction embeddings:

$$\mathbf{e}_{Ie} = \sum_{t_x \in \mathbf{c}_{Ie}} \beta_{sx}\mathbf{e}_{t_x}, \quad s.t. \sum_{t_x \in \mathbf{c}_{Ie}} \beta_{sx} = 1 \tag{7.5}$$

where β_{sx} is the integration weight of transaction t_x from the inter-transaction context $\mathbf{c}_{Ie}$ for the target item i_s. It indicates the relevance degree of t_x to the current transaction, i.e., intra-transaction context $\mathbf{c}_{Ia}$, by modelling the interaction between t_x and $\mathbf{c}_{Ia}$ in the inter-transaction attention model A_{Ie}. More relevant to the current transaction, t_x will be more influential in the choice of i_s, therefore β_{sx} essentially implies the contribution scale of transaction t_x to the choice of the target item i_s.

Inter-Transaction Attention Differing from the intra-transaction attention model, except for the transactions from inter-transaction context, I take the intra-transaction context as an additional input to model the interaction between transactions as indicated in Figure 7.2. I first use a matrix to model the interactions between each inter-transaction and the intra-transaction context, and I then import the product of inter-transaction embedding, interaction matrix and intra-transaction context embedding into the attention model.

$$\beta_{sx} = \frac{exp(\varrho(\mathbf{e}_{t_x}))}{\sum_{t_f \in \mathbf{c}_{Ie}} exp(\varrho(\mathbf{e}_{t_f}))} \tag{7.6}$$

$$\varrho(\mathbf{e}_{t_x}) = \mathbf{e}_{t_x}\mathbf{W}^{\beta}\mathbf{e}_{Ia}^{T} \tag{7.7}$$

where $\mathbf{W}^{\beta}$ is a transaction-level interaction matrix shared by all the contextual transactions. It can be regarded as a high level representation of a query 'Which transaction in the inter-transaction context is relevant to the current one?'. This matrix is randomly initialized and jointly learned during the training process. I refer it to as the inter-transaction attention weight. $\mathbf{e}_{Ia}$ is the embedding of intra-transaction context and its calculation will be given shortly.

Intra-Transaction Context Embedding

Intra-transaction context embedding is similar to the aforementioned attentive transaction embedding.

Given an intra-transaction context $\mathbf{c}_{Ia}$ consisting of multiple chosen items in the current transaction, I first get the embedding of each item with the aforementioned item embedding. I then integrate these embeddings attentively to build the intra-transaction context embedding.

$$\mathbf{e}_{Ia} = \sum_{i_z \in \mathbf{c}_{Ia}} \alpha_{sz}\mathbf{h}_z, \quad s.t. \sum_{i_z \in \mathbf{c}_{Ia}} \alpha_{sz} = 1 \tag{7.8}$$

where h_z is the embedding of an intra-transaction context item i_z and is calculated using Equation (7.1) while α_{sz} is the integration weight calculated using Equations (7.3) and (7.4).

Target Item Prediction

Once the embeddings of both intra- and inter-transaction contexts are ready, I feed them into the output layer for the target item prediction, as shown in the upper part of Figure 7.2. Here the output weight matrix $\mathbf{W}^2 \in \mathbb{R}^{|I|\times K}$ and $\mathbf{W}^3 \in \mathbb{R}^{|I|\times K}$ are used to fully connect the intra- and inter-transaction context embeddings to the output layer. Specifically, given the context embeddings and the weights, a score indicating the possibility of the choice of a target item i_s under the context $\mathbf{c}$ is computed using:

$$S_{i_s}(\mathbf{c}) = \mathbf{W}^2_{s,:}\mathbf{e}_{Ie} + \mathbf{W}^3_{s,:}\mathbf{e}_{Ia} \tag{7.9}$$

where $\mathbf{W}^2_{s,:}$ denotes the s^{th} row of $\mathbf{W}^2$ and $S_{i_s}(\mathbf{c})$ quantifies the relevance of the target item i_s w.r.t the given context $\mathbf{c}$. Therefore, the conditional probability distribution $P_\Theta(i_s|\mathbf{c})$ is defined with the commonly used softmax function:

$$P_\Theta(i_s|\mathbf{c}) = \frac{exp(S_{i_s}(\mathbf{c}))}{Z(\mathbf{c})} \tag{7.10}$$

where $Z(\mathbf{c}) = \sum_{i \in I} exp(S_i(\mathbf{c}))$ is the normalization constant and $\Theta = \{\mathbf{W}^1, \mathbf{w}^\alpha, \mathbf{W}^\beta, \mathbf{W}^2, \mathbf{W}^3\}$ includes the model parameters. Therefore, a probabilistic classifier modeled by the proposed HATE model is obtained to predict the target item and accordingly recommend the next item.

7.3.2 Parameter Learning and Item Prediction

I now discuss how to learn the model parameters and predict the next item using the trained model in this section.

A probabilistic classifier is built over the transaction data $d = \langle \mathbf{c}, i_c \rangle$, where $\mathbf{c}$ is the input context and i_c is the observed output conditional on $\mathbf{c}$. Given a training dataset $D = \{\langle \mathbf{c}, i_c \rangle\}$, the joint probability distribution is obtained by:

$$P_\Theta(D) \propto \prod_{d \in D} P_\Theta(i_c|\mathbf{c}) \tag{7.11}$$

Therefore, the model parameters Θ can be learned by maximizing the conditional log-likelihood (cf. Equation (7.10)):

$$L_\Theta = \sum_{d \in D} log P_\Theta(i_c|\mathbf{c}) \tag{7.12}$$

Note that the evaluation of L_Θ and its corresponding gradient computation involve the normalization term $Z(\mathbf{c})$, the computation of which is time consuming as it sums $exp(S_{i_c}(\mathbf{c}))$ over all the items for each training instance. The commonly used Noise contrastive estimation (NCE) technique (Gutmann & Hyvärinen 2012) is adopted here to enhance the training efficiency. Due to the space limitation, I do not provide the details of NCE.

Once the model parameters Θ have been learned, HATE is ready to compute predictions and thus generate next-item recommendations. Specifically, given an arbitrary transactional context which contains both intra- and inter-transaction contexts indicating prior transaction data of a user, the probabilities of choosing next candidate items are calculated according to Equation (7.10), and a ranking reflecting the priority of the candidate items is achieved.

7.4 Experiments and Evaluation

I demonstrate the empirical study of my proposed HATE model in this section. To be specific, I first setup the experiments by preparing the experimental datasets and introducing the comparison methods and evaluation metrics, and then evaluate the performance in terms of recommendation accuracy and novelty.

7.4.1 Experimental Setup

Dataset Preparation

I evaluate my proposed method on two real-world grocery store transaction datasets: a public dataset Dunnhumby[1] and a proprietary ANS dataset (Luo, Li, Koprinska, Berkovsky & Chen 2017). Dunnhumby includes transaction records of around 2,500 households shopping frequently at multiple stores of the same retailer over two years. ANS contains transaction records of about 1,000 customers, collected by an Australian national supermarket chain within a period of one year.

First, a sequence of transactions is extracted for every user and then a sliding window is used to cut each users transactions sequence into multiple triple-transaction units. For each unit, I consider the first two transactions as the inter-transaction context $\mathbf{c}_{Ie}$ and the last one as the current transaction. My selection is data-drive and is explained by the most frequently observed

[1]http://www.dunnhumby.com/sourcefiles

Table 7.1: Statistics of experimental datasets

Statistics	**Dunnhumby**	**ANS**
#Transactions	65,001	99,987
#Items	10,292	11,996
Avg. Transaction Length	12.15	10.81
#Training Sequence of Trans.	149,606	258,561
#Training Instances	402,739	703,062
#Test Sequence of Trans.	7,874	13,608
#Test Instances	21,205	36,933

transaction pattern of three transactions per week in the shopping cycle. Each time one item from the current transaction is picked up as the target item i_s and all others are considered as the intra-transaction context $\mathbf{c}_{Ia}$. I do this because the order information over items within transactions is not provided and thus I relax the rigid order assumption. As a result, the training and test instances are built in the format of $d = \langle \mathbf{c}, i_c \rangle (\mathbf{c} = \{\mathbf{c}_{Ie}, \mathbf{c}_{Ia}\})$ as illustrated in the previous section. Finally, I randomly select 20% of transactions that occurred in the last 30 days as the test set and leave the remainder for training. The characteristics of the datasets are shown in Table 7.1.

Comparison Methods and Metrics

In addition to the four baseline methods, i.e., *PBRS*, *FPMC*, *PRME* and *GRU4Rec*, introduced in Section 2.4, I add other three methods (*SWIWO*, *ATE* and *HTE*) into the baselines for the comparison in this experiment.

- ***SWIWO***: A shallow wide-in-wide-out network embedding model for session-based RSs (Hu, Cao, Wang, Xu, Cao & Gu 2017).
- ***ATE***: A model similar to HATE that only utilizes the intra-transaction context. This assesses the contribution of the inter-transaction context.

Table 7.2: Accuracy comparisons on Dunnhumby

Model	REC@10	REC@50	MRR
PBRS	0.0817	0.0901	0.0421
FPMC	0.0333	0.0711	0.0317
PRME	0.0757	0.0912	0.0613
GRU4Rec	0.2018	0.3002	0.1216
SWOWI	0.2469	0.3379	0.1139
ATE	0.2752	0.3754	0.1250
HTE	0.2752	0.4000	0.1218
HATE	**0.3012**	**0.4513**	**0.1421**
Improve (%)	9.45	12.82	13.68

- ***HTE***: A model similar to HATE that replaces the inter-transaction attention module with a fully-connected layer. This assesses the effect of the inter-transaction attention module.

7.4.2 Performance Evaluation

I evaluate the recommendation performance from both accuracy and novelty perspectives.

Accuracy Evaluation

Two common accuracy metrics $REC@K$ and MRR for ranking issues defined in Section 2.3.1 are used in the accuracy evaluations.

Table 7.2 and Table 7.3 show the obtained REC@10, REC@50 and MRR on two real-world transaction datasets. I empirically set the minimum support to 0.02 on both datasets in PBRS. The information loss caused by filtering out infrequent items leads to poor performance. To achieve the best performance, I set the factor number to 10 for FPMC which performs not good on both datasets, mainly caused by the data sparsity. Due to the

Table 7.3: Accuracy comparisons on ANS

Model	**REC@10**	**REC@50**	**MRR**
PBRS	0.0572	0.0765	0.0410
FPMC	0.0310	0.0555	0.0292
PRME	0.0611	0.0800	0.0522
GRU4Rec	0.1405	0.2951	0.0755
SWOWI	0.1400	0.3015	0.0805
ATE	0.1542	0.2254	0.0805
HTE	0.1756	0.2755	0.0874
HATE	**0.1756**	**0.3515**	**0.0993**
Improve (%)	0	27.59	13.62

large numbers of transactions and items but limited interactions between them, quite large but very sparse item transition matrices are constructed to train this MF model. For instance, the non-empty entries of the transition matrix built on Dunnhumby only account for 0.12%. Following (Feng et al. 2015), the embedding dimension is set to 60 for PRME. Although better than FPMC, it still does not perform good enough. As a first-order MC model, PRME is easy to lose information by learning the transition probability over the successive item instead of the whole context. In addition, the rigid order assumption set by these models may not always match the real world purchasing events. GRU4Rec achieves much better performance compared to the above three methods by benefiting from its deep structure. Building a flexible embedding on the whole context, SWIWO is able to capture the complex intra-transaction dependency for better recommendations. A common drawback of all these models is that they are all limited to the intra-transaction dependency.

For my HATE model, the embedding dimension and the batch size are empirically set to 50 and 30 respectively on both datasets. Adagrad (Duchi, Hazan & Singer 2011) with an initial learning rate of 0.5 is applied to train the model. By attentively incorporating the inter-transaction dependency, HATE outperforms the best baseline SWOWI by approximately 20% in terms of recall and MRR on both datasets, which validates the advantage of my model. In particular, the enhanced 10% performance of HATE compared to ATE and HTE demonstrates the significance of incorporating inter-transaction context and attention mechanism respectively. Different from the baselines, which are based on the intra-transaction dependency only, HATE takes both intra- and inter-transaction dependency into account. Particularly, the hierarchical attention mechanism helps to emphasize those truly relevant items and transactions when modeling dependency. These effectively enhance the prior contextual information and lead to more reliable recommendations. My model has a shallow and concise structure for easy training and prediction, which makes it more practical compared to those deep and complex models like GRU4Rec.

The Effect of Number of Incorporated Inter Transactions

Generally speaking, a long inter-transaction context which contains more recent transactions is more likely to include transactions irrelevant to the current transaction and the next-item choice. As a result, it is harder to identify and emphasize those truly relevant transactions in a long context. To show the advantage of attention mechanism in handling long contexts, I test the effect of the number of incorporated inter-transactions on a subset of Dunnhumby by selecting users with at least 6 transactions. Each time a different number of recent transactions is considered as the inter-transaction context. Figure 7.3 shows that HATE gains larger margins compared to HTE when incorporating more transactions, which demonstrates its ability to emphasize the relevant transactions in longer inter-transaction contexts. I only compare here HATE and HTE because only these two approaches can incorporate inter-transaction context.

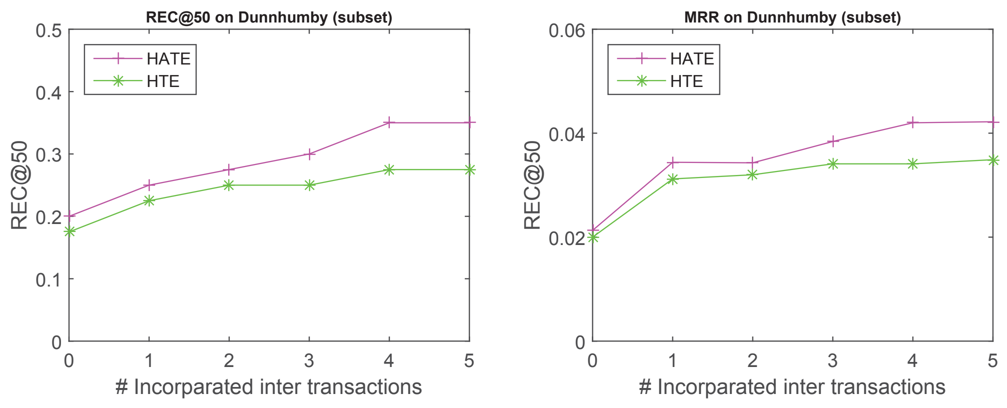


Figure 7.3: HATE gains larger margins when incorporating more inter-transactions.

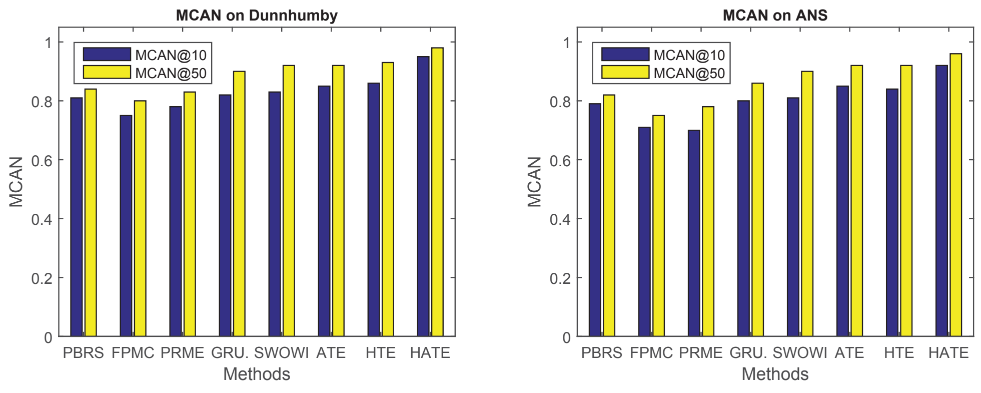


Figure 7.4: HATE achieves higher novelty than compared methods.

Novelty Evaluation

I use the novelty measure $MCAN$ defined in Section 2.3.2 to quantify the difference between the intra-transaction context and the recommendation list. Larger $MCAN$ indicates higher novelty.

Figure 7.4 shows the novelty comparison results on the two experimental datasets. Due to the dropping of infrequent items, PBRS is hard to match the whole intra-transaction context exactly, which results in low novelty. The parameters of FPMC are not learned well on such sparse datasets and thus cannot achieve high novelty. Ignoring higher order dependency, PRME is easy to recommend duplicate items and thus leads to low novelty. Due to the accumulation of effect of all sequential items from the current transaction, GRU4Rec makes relatively novel recommendations. By mapping the whole intra-transaction context into a latent representation, SWIWO is able to capture the complex dependency within a transaction in a more flexible way, which makes it easier to achieve novel recommendations. In comparison, my model not only attentively captures the intra-transaction dependency but also incorporates the inter-transaction dependency attentively. Consequently, it is easier for my model to generate novel and relevant recommendations. In addition, the incorporation of inter-transaction contexts or the attention mechanism contribute clearly to the improvement of novelty by comparing HATE and ATE or HTE.

7.5 Summary

Effectively recommending the next item under a transactional context poses a significant challenge to session-based recommender systems, especially under the transaction context. Accordingly, this work proposes a hierarchical attentive transaction embedding model HATE - a shallow and wide neural network. By incorporating both current transaction and recent transactions, HATE is able to capture both intra- and inter-transaction dependencies to build a more informative context representation. In addition, the incorporation of hierarchical attention models allows us to emphasize items and transactions particularly relevant to the next-item choice when building the attentive representation. An informative and attentive representation leads to better recommendations. Empirical validation on two real-world transaction datasets shows the superiority of HATE over several state-of-the-art approaches. In future, I will explore the applications of HATE to other session-based recommendation contexts like next-song recommendations and other problems e.g., document analysis (Marinai, Gori & Soda 2005) and multimedia recommendations (Chen, Zhang, He, Nie, Liu & Chua 2017).

Chapter 8

Conclusions and Future Work

8.1 Conclusions

In conclusion, this thesis offers a comprehensive and systematic literature review on session-based recommender systems in Chapter 3 and then it presents several techniques to bridge the gaps in session-based recommender systems in the subsequent chapters. To be specific, I address several different challenges in session-based recommender systems – the implicit relation modelling in rule-based recommendations, the item heterogeneity issue, the cold-start item issue, and the inter-session relation modelling issue in next-item recommendations by systematically modeling the dependency of different levels in transaction data (i.e., item-item level, feature-item level and session [transaction]-item level). Each challenge is addressed by one chapter from Chapter 4 to Chapter 7, which will be concluded one by one in the following part.

In Chapter 3, I give a systematic survey on session-based recommender systems to show the existing related works, recent progress and gaps in this particular area. I motivate the necessity of session-based recommender systems by pointing out the gaps in other types of recommender systems, such as collaborative filtering, and by conducting comprehensive comparisons between them and session-based ones. I then systematically formalize the session-based recommendation problem and define some key relevant concepts used in this thesis, followed by the summary of the development history of session-based recommender systems, comparisons between various scenarios in session-based recommender systems and the categorization of existing session-based recommender system techniques. More importantly, I summarize the representative approaches in each category, which includes rule/pattern-based approaches, sequential pattern-based approaches, Markov chain model-based approaches, factorization-based approaches and neural model-based approaches.

In Chapter 4, I focus on the most basic session-based recommender systems, namely rule-based recommender systems and I determine the drawbacks of the existing rule-based approaches. That is they focus only on the explicit co-occurrence relations like associations or correlations while ignoring the complex implicit relations between items, which can often result in unreliable recommendations. To bridge such gaps, I propose an implicit rule-based recommender systems, which first infers implicit rules with more complex implicit relations embedded through rule and pattern relation analysis and then uses the inferred rules for recommendations. Accordingly, an implicit rule inference framework combining with a corresponding implementation algorithm, IRRMiner, is proposed. Empirical evaluations on IRRMiner shows its advantage over other similar indirect rule mining algorithms. The experimental results of my built implicit rule-based recommender system on real-world datasets demonstrate its effectiveness on improving the reliability of recommendations.

In Chapter 5, I focus on the item heterogeneity in session-based recommendations, particularly for next-item recommendations. Item heterogeneity means different items in a transaction context may have different correlation scales to the next items and thus contribute differently to the occurrence of these items. Accordingly, an attention-based transaction embedding model is proposed to build a more precise and informative embedding for the transaction context for item recommendations. Specifically, the attention mechanism is applied to learn the significance weights of different items in the prior transaction context automatically. A shallow neural network model called ATEM is built by incorporating an attention layer to first embed the transaction context and then predict the next item. I conduct empirical evaluations on the recommendation results in terms of recommendation accuracy and novelty. The comparisons with other state-of-the-art session-based recommender systems show the superiority of my model.

In Chapter 6, I focus on the cold start item issue in session-based recommendations, especially in the next-item recommendations. The cold start issue is quite common and typical in recommender systems and has attracted much attention in other types of recommender systems like collaborative filtering, but is less studied in session-based recommender systems. Accordingly, I build a shallow neural network to first map items and their features to latent feature space and then jointly model the feature interactions and item dependencies. During the joint modelling, the interactions between item features and item occurrence are learned, which particularly benefits the recommendations of those cold-start items with quite limited occurrence in the data. Experimental studies on real-world datasets demonstrate the merit of my proposed approaches over other representative ones.

In Chapter 7, I focus on a kind of high-level dependency modelling in session-based recommender systems, which is ignored in most existing methods. Specifically, I figure out and emphasize the importance of inter-session dependency in session-based recommendations, especially in next-item recommendations. To this end, I build a hierarchical attention-based embedding model to jointly model the intra-session dependency and inter-session dependency for next-item recommendations. With the attention mechanism incorporated, the model is able to emphasize those significant items in one session (transaction) and the significant sessions out of multiple contextual sessions. Experiments on real-world transaction datasets demonstrate the importance of inter-session dependency for next-item recommendations and the superiority of my proposed methods over existing state-of-the-art session-based recommender systems.

In summary, in this thesis, I have systematically explored the typical less-studied issues like implicit relations, item heterogeneity, cold-start items, and inter-session dependency in session-based recommender systems. I provide my solutions and demonstrate their efficacy with empirical studies on real-world transaction datasets. Each chapter (i.e., from Chapter 4 to Chapter 7) of this thesis is supported by a research paper[1] listed in list of publications. Therefore, what I have done and propose in this thesis is of great significance to the session-based recommendation research and applications in the area of recommender systems.

[1]The papers of Chapter 4, 5 and 6 have been published, the paper of Chapter 7 is under review.

8.2 Future Work

Although recommender systems (RSs) have been well studied and broadly applied, session-based recommender system is a relative emerging area in RSs. According to my systematic explorations and observations in this area, I have identified a collection of challenges that still face the recommender system researchers; these challenges may represent the future research directions of this community. I divide these challenges into six branches with respect to the information source and scenario settings. Next, I demonstrate each branch in each subsection in the following six subsections by first illustrating its significance and then the specific issues that remain open.

8.2.1 Session-based Recommendations with Users' General Preference

Significance. The key difference between session-based recommender systems and other conventional recommender systems like collaborative filtering is that the former mainly captures the short-term sequential behaviour patterns while the latter mainly learns the long-term general preference and the taste of users. Obviously, even each of the two shows its own advantages over the other, but must sacrifice the other's advantage. In other words, session-based recommeder systems usually ignore the users' general preferences which can be well captured by collaborative-filtering. This may lead to unreliable recommendations as users with different shopping preferences and consumption habits may choose different items next even under the same context (e.g., the same chosen items in hand). In this case, how to learn the users' general preference in transaction data and then incorporate it into the session-based recommender system models is a critical yet challenging task.

Open issues. Here, I discuss two major issues with respect to the general preference learning and its incorporation into session-based recommender systems and I sketch the possibilities for future research.

- How to incorporate explicit user preference into session-based recommender systems? In this case, the users' ratings on their purchased items are available, usually forming a user-item rating matrix by accumulating all users' ratings on the items. An intuitive way is to first predict a users' preference on all candidate items utilizing conventional approaches like matrix factorization and then use the preference data as a constraint to tune the candidate ranking and selection in session-based recommendations. For instance, if some candidate items have similar probabilities to be chosen in the subsequent action, according to the session context, those items with higher preference to particular users can be put to the front of the according recommendation list. Another way is to combine the above two factors together when ranking the candidate items for the reocmmendation list. In (Zhao, Wang, Ye, Gao, Yang, Zhao & Chen 2017), the authors proposed a GAN framework to leverage the MF and RNN hybrid model for movie recommendation, which jointly models long-term preferences and short-term behaviour patterns.

- How to incorporate users' experience into session-based recommeder systems without explicit user preference data? In the real-word cases, the explicit preference data may not be always available as customers may not rate everything they bought. In this case, the shopping-basket based transaction data is usually used as a kind of implicit feedback (He & McAuley 2016*b*, He et al. 2016) to indicate users' preference in some degree. Some existing works (Anyosa, Vinagre & Jorge 2018, Peska & Vojtas 2017) have explored how to learn users' preference from these implicit feedback data in the collaborative filtering framework. In practice, the implicit feedback data is much more readily available in session-based recommendation scenarios like clicks or views of items (Schnabel, Bennett, Dumais & Joachims 2018). Note that in such case, the implicit preference data is actually the same to the session data used in session-based RSs. Therefore how to learn the users' implicit preference from such data and simultaneously avoid the information duplication in session-based reocmcmender systems is a challenge without good existing solutions.

8.2.2 Session-based Recommendations Considering More Contextual Factors

Significance. Recommendation contexts refer to the practical and specific situations in which a user makes his or her next choice of products. Accordingly, contextual factors refer to different aspects from the context that may affect a user's choices, such as weather, season, location, time, or recent popularity trends. Considering these context factors may make a substantial difference in recommendation performance. In fact, a session-based recommender system can be seen as a simplified context-aware system that takes only the chosen items in hand as the context for the choice of next items (Twardowski 2016). The significance of context in recommender systems is also emphasized by other researchers like Gediminas etc. (Adomavicius & Tuzhilin 2005, Adomavicius & Tuzhilin 2015), Shi etc. (Shi, Larson & Hanjalic 2014) and Pagano etc.(Pagano, Cremonesi, Larson, Hidasi, Tikk, Karatzoglou & Quadrana 2016).

Open issues. Although contextual information has been incorporated into conventional recommender systems in some works or even context-aware RSs has been proposed as a new type of RSs(Unger 2015, Adomavicius & Tuzhilin 2015), context is still rarely exploited in session-based recommender systems.

- How to incorporate more contextual factors into session-based recommender systems? Quite limited works have explored on this topic. Contextual Recurrent Neural Networks (CRNNs) for Recommendations are proposed to incorporate contextual factors, such as types of user-item interactions, time gaps between different events in a session, and time of a day, into an RNN-based session recommender system (Smirnova & Vasile 2017). In (Lerche, Jannach & Ludewig 2016, Jannach & Ludewig 2017*a*), the recent popularity trend, users' recently viewed items, and items with discounts in shopping mall are considered as contextual factors in session recommendations. However, these works are just a starting point; more explorations are still needed on how to collect more contextual information and how to develop models to more effectively incorporate such information into session-based recommendation tasks.

8.2.3 Session-Based Recommendations With Noisy and Irrelevant Items

Significance. Currently, sequence modelling-based session-based recommender systems like RNN-based ones and Markov chain-based ones always assume that strong dependency exists over successive items. In other words, the current item has the most influence on the next item. However, this may not be the case in the real-world transaction data because a user may just randomly pick up some items he likes into the cart. These randomly picked items may be irrelevant to both the already-chosen items and the following items to be chosen. If we ignore such cases, the recommendation results may be easily misled by these noisy items, degrading the recommendation performance.

Open issues. How to make reliable recommendations with a noisy session? Although some mechanisms, like attention (Wang et al. 2018) and pooling mechanisms (Tang & Wang 2018), have been applied to session-based recommender systems to emphasize those really relevant and important items for the next choice from the whole session, work in this area remains limited. More efforts are required to develop a more robust and tolerant system for session recommendation tasks for noisy sessions.

8.2.4 Session-based Recommendations for Multi-Step Recommendations

Significance. Usually, a shopping event contains multiple steps rather than just one step. For example, when a user buys a bread, he may buy milk later, followed by cheese. Given the partial session consisting of bread, most current session-based recommender systems only make recommendations one step forward; for instance, they make predictions only on milk rather than predicting milk and then cheese. This tendency may reduce the utility of recommender systems greatly.

Open issues. How to generate multi-step recommendations given a partial session as the context? This issue is practical and critical but extremely challenging. Considering the advantage of multi-step modelling, an encoder-decoder framework (Li et al. 2015, Loyola, Liu & Hirate 2017) may be an intuitive choice for this issue.

8.2.5 Session-based Recommendations with Cross-Session Information

Significance. Actually, a user's choice on the next item may depend not only on the previous items in the current session, but also on items from other sessions. For example, a user buys a cellphone in a shopping event on Monday; he or she may then want to buy a cellphone cover or an accessory in shopping on Wednesday. To this end, taking cross-session information into account may make a difference in session-based recommendations.

Open issues. How to incorporate cross-session information into session-based recommender systems? Exploration of this topic is quite limited. The work in (Quadrana et al. 2017) proposed a hierarchical RNN architecture to model both intra- and inter- session dependency for session recommendations. However, it applied only a very intuitive method to incorporate multiple sessions into an RNN structure. Further exploration of this topic is needed.

8.2.6 Session-based Recommendations With Cross-Domain Information

Significance. Cross-domain means domains different from but relevant to the target domain in which the recommendations are made (Hu, Cao, Xu, Wang, Gu & Cao 2013). Usually, the users' purchased items are not limited to one domain and items from multiple domains are required to meet users' daily necessities. In addition, the choices of items from different domains may not be independent of each other. For instance, a user may see the movie 'Titanic' first and really like it, and then he or she may listen to the movie's primary song, ‘My heart will go on’, and like the singer ‘Celine Dionne’ – the user may listen to others of her songs, such as ‘I’m Your Angel’. In another case, young girls who like the protagonist, ‘Rose’, in *Titanic* would like to buy the same dress as Rose’s. Such examples show that the items from different domains may not only be dependent but can even form a sequence of events with high correlation embedded: {Titanic, ‘My Heart Will Go On’, Celine Dionne, ‘I’m Your Angel’} or {Titanic, Rose, Rose’s style of dress}. The recommendations based on such scenarios are interesting but quite challenging. On one hand, such recommender systems not only cover more aspects of our daily lives but also provide a solution to the data sparsity issue in the case where only one domain is considered. On the other hand, it is hard to collect a user’s consumed products or services from various domains together, and the relations between items from different domains are much more complex than those from one domain. For example, there may be sequential patterns or may be nothing between them.

Open issues. According to whether the products or services from different domains can form a tight session or not, there are two open issues that can be further explored.

- How to borrow knowledge from other domains to help with the session recommendations in the target domain? When no sessions can be built over products or services from different domains, the way to make use of other domains is a main-auxiliary framework. Such a framework takes the target domain where the recommendations are made as the main information source while taking information from other domains as a supplementary source to leverage the recommendation performance in the target domain. An intuitive choice is transfer learning (Pan & Yang 2010, Elkahky, Song & He 2015) which transfers knowledge from other domains to help with the tasks in the target domain. Although transfer learning has been well applied in conventional recommender systems, such as collaborative filtering (Li, Yang & Xue 2009, Pan, Xiang, Liu & Yang 2010, Pan & Yang 2013, Loni, Shi, Larson & Hanjalic 2014), it is still only explored in session-based recommender systems.

- How to make session recommendations over products or services from multiple domains? This case happens when a session can be built on elements from different domains, as in the aforementioned example {Titanic, ‘My Heart Will Go On’, Celine Dionne, ‘I’m Your Angel’}. Different from the case mentioned in the above paragraph, this case treats all elements from different domains equally and every domain can serve as the target domain. It is much more interesting yet challenging than the other case. It can incorporate all of a user’s daily needs into a unified recommender system framework, and thus is able to cover many more aspects of our daily lives. However, the challenge arises from directions: session-building and model development. Different from session recommendations in one domain, where the natural sessions are usually already there for direct usage, usually no obvious sessions on cross-domains are available. Therefore, how to build a reasonable session from multiple heterogeneous data sources is the first challenge. Once the cross-domain sessions are ready, how to develop a model to effectively capture the complex and heterogeneous coupling relations (Cao 2015) between elements from different domains presents another challenge.

Bibliography

Abel, F., Bittencourt, I. I., Henze, N., Krause, D. & Vassileva, J. (2008), A rule-based recommender system for online discussion forums, *in* 'International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems', Springer, pp. 12–21.

Adda, M., Missaoui, R., Valtchev, P. & Djeraba, C. (2005), Recommendation strategy based on relation rule mining, *in* 'IJCAI Workshop on Intelligent Techniques for Web Personalization (ITWP05)', pp. 33–40.

Adomavicius, G. & Tuzhilin, A. (2001), 'Expert-driven validation of rule-based user models in personalization applications', *Data Mining and Knowledge Discovery* **5**(1-2), 33–58.

Adomavicius, G. & Tuzhilin, A. (2005), 'Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions', *IEEE transactions on knowledge and data engineering* **17**(6), 734–749.

Adomavicius, G. & Tuzhilin, A. (2015), Context-aware recommender systems, *in* 'Recommender systems handbook', Springer, pp. 191–226.

Aggarwal, C. C. (2016), Content-based recommender systems, *in* 'Recommender Systems', Springer, pp. 139–166.

Aggarwal, C. C. & Han, J. (2014), *Frequent pattern mining*, Springer.

Agrawal, R., Imieliński, T. & Swami, A. (1993), Mining association rules between sets of items in large databases, *in* 'ACM Sigmod Record', Vol. 22, ACM, pp. 207–216.

Agrawal, R. & Srikant, R. (1995), Mining sequential patterns, *in* 'Data Engineering, 1995. Proceedings of the Eleventh International Conference on', IEEE, pp. 3–14.

Agrawal, R., Srikant, R. et al. (1994), Fast algorithms for mining association rules, *in* 'Proc. 20th int. conf. very large data bases, VLDB', Vol. 1215, pp. 487–499.

Ahmad Wasfi, A. M. (1998), Collecting user access patterns for building user profiles and collaborative filtering, *in* 'Proceedings of the 4th international conference on Intelligent user interfaces', ACM, pp. 57–64.

Anyosa, S. C., Vinagre, J. & Jorge, A. M. (2018), Incremental matrix co-factorization for recommender systems with implicit feedback, *in* 'Companion of the The Web Conference 2018 on The Web Conference 2018', International World Wide Web Conferences Steering Committee, pp. 1413–1418.

Bahdanau, D., Cho, K. & Bengio, Y. (2014), 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473* .

Bandelt, H.-J. & Dress, A. W. (1992), 'A canonical decomposition theory for metrics on a finite set', *Advances in mathematics* **92**(1), 47–105.

Barkan, O., Brumer, Y. & Koenigstein, N. (2016), Modelling session activity with neural embedding., *in* 'RecSys Posters'.

Basilico, J. & Hofmann, T. (2004), Unifying collaborative and content-based filtering, *in* 'Proceedings of the twenty-first international conference on Machine learning', ACM, pp. 65–73.

Beg, I. & Butt, A. R. (2009), 'Fixed point for set-valued mappings satisfying an implicit relation in partially ordered metric spaces', *Nonlinear Analysis: Theory, Methods & Applications* **71**(9), 3699–3704.

Bendakir, N. & Aïmeur, E. (2006), Using association rules for course recommendation, *in* 'Proceedings of the AAAI Workshop on Educational Data Mining', pp. 1–10.

Berberidis, C., Angelis, L. & Vlahavas, I. (2004), Inter-transaction association rules mining for rare events prediction, *in* 'Proc. 3rd Hellenic Conference on Artificial Intellligence', Springer, pp. 1–16.

Beutel, A., Covington, P., Jain, S., Xu, C., Li, J., Gatto, V. & Chi, E. H. (2018), Latent cross: Making use of context in recurrent recommender systems, *in* 'Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining', ACM, pp. 46–54.

Bogina, V. & Kuflik, T. (2017), Incorporating dwell time in session-based recommendations with recurrent neural networks, *in* 'CEUR Workshop Proceedings', Vol. 1922, pp. 57–59.

Bonchi, F., Perego, R., Silvestri, F., Vahabi, H. & Venturini, R. (2011), Recommendations for the long tail by term-query graph, *in* 'Proceedings of the 20th international conference companion on World wide web', ACM, pp. 15–16.

Bouma, G. (2009), 'Normalized (pointwise) mutual information in collocation extraction', *Proceedings of GSCL* pp. 31–40.

Breese, J. S., Heckerman, D. & Kadie, C. (1998), Empirical analysis of predictive algorithms for collaborative filtering, *in* 'Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence', Morgan Kaufmann Publishers Inc., pp. 43–52.

Brin, S., Motwani, R. & Silverstein, C. (1997), Beyond market baskets: Generalizing association rules to correlations, *in* 'ACM SIGMOD Record', Vol. 26, ACM, pp. 265–276.

Buntine, W. L. & Weigend, A. S. (1994), 'Computing second derivatives in feed-forward networks: A review', *IEEE Transactions on Neural Networks* **5**(3), 480–488.

Burke, R. (2007), Hybrid web recommender systems, *in* 'The adaptive web', Springer, pp. 377–408.

Cao, L. (2012), 'Actionable knowledge discovery and delivery', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2**(2), 149–163.

Cao, L. (2013), 'Combined mining: Analyzing object and pattern relations for discovering and constructing complex yet actionable patterns', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **3**(2), 140–155.

Cao, L. (2015), 'Coupling learning of complex interactions', *Information Processing & Management* **51**(2), 167–186.

Cao, L. (2016), 'Non-iid recommender systems: A review and framework of recommendation paradigm shifting', *Engineering* **2**(2), 212–224.

Cao, L., Ou, Y. & Yu, P. S. (2012), 'Coupled behavior analysis with applications', *IEEE Trans. on Knowledge and Data Engineering* **24**(8), 1378–1392.

Chatzis, S., Christodoulou, P. & Andreou, A. S. (2017), 'Recurrent latent variable networks for session-based recommendation', *arXiv preprint arXiv:1706.04026* .

Chen, J., Zhang, H., He, X., Nie, L., Liu, W. & Chua, T.-S. (2017), Attentive collaborative filtering: Multimedia recommendation with item-

and component-level attention, *in* ‘Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval’, pp. 335–344.

Chen, Q., Hu, L., Xu, J., Liu, W. & Cao, L. (2015), Document similarity analysis via involving both explicit and implicit semantic couplings, *in* ‘2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, Campus des Cordeliers, Paris, France, October 19-21, 2015’, pp. 1–10.

Chen, S., Moore, J. L., Turnbull, D. & Joachims, T. (2012), Playlist prediction via metric embedding, *in* ‘Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining’, ACM, pp. 714–722.

Cheng, C., Yang, H., Lyu, M. R. & King, I. (2013), Where you like to go next: Successive point-of-interest recommendation., *in* ‘IJCAI’, Vol. 13, pp. 2605–2611.

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M. et al. (2016), Wide & deep learning for recommender systems, *in* ‘Proceedings of the 1st Workshop on Deep Learning for Recommender Systems’, ACM, pp. 7–10.

Chiueh, T.-c. & Bajpai, S. (2008), Accurate and efficient inter-transaction dependency tracking, *in* ‘Proceedings of the 24th International Conference on Data Engineering’, pp. 1209–1218.

Cho, K., Van Merriënboer, B., Bahdanau, D. & Bengio, Y. (2014), ‘On the properties of neural machine translation: Encoder-decoder approaches’, *arXiv preprint arXiv:1409.1259* .

Choi, K., Yoo, D., Kim, G. & Suh, Y. (2012), ‘A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis', *Electronic Commerce Research and Applications* **11**(4), 309–317.

Chou, S.-Y., Yang, Y.-H., Jang, J.-S. R. & Lin, Y.-C. (2016), Addressing cold start for next-song recommendation, *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 115–118.

Christodoulou, P., Chatzis, S. P. & Andreou, A. S. (2017), 'A variational recurrent neural network for session-based recommendations using bayesian personalized ranking'.

Church, K. W. & Hanks, P. (1990), 'Word association norms, mutual information, and lexicography', *Computational linguistics* **16**(1), 22–29.

de Gemmis, M., Lops, P., Musto, C., Narducci, F. & Semeraro, G. (2015), Semantics-aware content-based recommender systems, *in* 'Recommender Systems Handbook', Springer, pp. 119–159.

Deshpande, M. & Karypis, G. (2004), 'Item-based top-n recommendation algorithms', *ACM Transactions on Information Systems* **22**(1), 143–177.

Donkers, T., Loepp, B. & Ziegler, J. (2017), Sequential user-based recurrent neural network recommendations, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 152–160.

Duchi, J., Hazan, E. & Singer, Y. (2011), 'Adaptive subgradient methods for online learning and stochastic optimization', *Journal of Machine Learning Research* **12**(7), 2121–2159.

Eirinaki, M., Vazirgiannis, M. & Kapogiannis, D. (2005), Web path recommendations based on page ranking and markov models, *in* 'Proceedings of the 7th annual ACM international workshop on Web information and data management', ACM, pp. 2–9.

Ekstrand, M. D., Riedl, J. T., Konstan, J. A. et al. (2011), 'Collaborative filtering recommender systems', *Foundations and Trends® in Human–Computer Interaction* **4**(2), 81–173.

Elkahky, A. M., Song, Y. & He, X. (2015), A multi-view deep learning approach for cross domain user modeling in recommendation systems, *in* 'Proceedings of the 24th International Conference on World Wide Web', International World Wide Web Conferences Steering Committee, pp. 278–288.

Feng, S., Li, X., Zeng, Y., Cong, G., Chee, Y. M. & Yuan, Q. (2015), Personalized ranking metric embedding for next new poi recommendation., *in* 'IJCAI', pp. 2069–2075.

Fleisher, C. S. & Bensoussan, B. E. (2015), *Business and competitive analysis: Effective application of new and classic methods*, 2nd edn. Upper Saddle River, NJ: Pearson.

Forsati, R., Meybodi, M. & Neiat, A. G. (2009), Web page personalization based on weighted association rules, *in* 'Electronic Computer Technology, 2009 International Conference on', IEEE, pp. 130–135.

Francesco Ricci, Lior Rokach, B. S. (2015), *Recommender Systems Handbook (2nd)*, Springer.

Garcia-Nunes, P. I., Souza, R. M. & da Silva, A. E. A. (2017), 'Mental models analysis and comparison based on fuzzy rules: A case study of the protests of June and July 2013 in Brazil', *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **47**(8), 2021–2033.

Ge, M., Delgado-Battenfeld, C. & Jannach, D. (2010), Beyond accuracy: evaluating recommender systems by coverage and serendipity, *in* 'Proceedings of the 4th ACM Conference on Recommender Systems', pp. 257–260.

Goldberg, Y. & Levy, O. (2014), 'word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method', *arXiv preprint arXiv:1402.3722* .

Gomez-Uribe, C. A. & Hunt, N. (2016), 'The netflix recommender system: Algorithms, business value, and innovation', *ACM Transactions on Management Information Systems (TMIS)* **6**(4), 13.

Goth, G. (2016), 'Deep or shallow, nlp is breaking out', *Communications of the ACM* **59**(3), 13–16.

Greenstein-Messica, A., Rokach, L. & Friedman, M. (2017), Session-based recommendations using item embedding, *in* 'Proceedings of the 22nd International Conference on Intelligent User Interfaces', ACM, pp. 629–633.

Gutmann, M. U. & Hyvärinen, A. (2012), 'Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics', *Journal of Machine Learning Research* **13**(2), 307–361.

Hamano, S. & Sato, M. (2004), Mining indirect association rules, *in* 'Industrial Conference on Data Mining', Springer, pp. 106–116.

Han, E.-H. S. & Karypis, G. (2005), Feature-based recommendation system, *in* 'Proceedings of the 14th ACM international conference on Information and knowledge management', ACM, pp. 446–452.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U. & Hsu, M.-C. (2000), Freespan: frequent pattern-projected sequential pattern mining, *in* 'Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 355–359.

Han, J., Pei, J. & Yin, Y. (2000), Mining frequent patterns without candidate generation, *in* 'ACM sigmod record', Vol. 29, ACM, pp. 1–12.

Hariri, N., Mobasher, B. & Burke, R. (2012), Context-aware music recommendation based on latenttopic sequential patterns, *in* 'Proceedings of the sixth ACM conference on Recommender systems', ACM, pp. 131–138.

He, R. & McAuley, J. (2016*a*), Fusing similarity models with markov chains for sparse sequential recommendation, *in* 'Data Mining (ICDM), 2016 IEEE 16th International Conference on', IEEE, pp. 191–200.

He, R. & McAuley, J. (2016*b*), Vbpr: Visual bayesian personalized ranking from implicit feedback., *in* 'AAAI', pp. 144–150.

He, X., Zhang, H., Kan, M.-Y. & Chua, T.-S. (2016), Fast matrix factorization for online recommendation with implicit feedback, *in* 'Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval', ACM, pp. 549–558.

Herawan, T., Noraziah, A., Abdullah, Z., Deris, M. M. & Abawajy, J. H. (2013), IPMA: Indirect patterns mining algorithm, *in* 'Advanced Methods for Computational Collective Intelligence', pp. 187–196.

Hidasi, B. & Karatzoglou, A. (2017), 'Recurrent neural networks with top-k gains for session-based recommendations', *arXiv preprint arXiv:1706.03847* .

Hidasi, B., Karatzoglou, A., Baltrunas, L. & Tikk, D. (2015), 'Session-based recommendations with recurrent neural networks', *arXiv preprint arXiv:1511.06939* .

Hidasi, B., Karatzoglou, A., Sar-Shalom, O., Dieleman, S., Shapira, B. & Tikk, D. (2017), Dlrs 2017: Second workshop on deep learning for recommender systems, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 370–371.

Hidasi, B., Quadrana, M., Karatzoglou, A. & Tikk, D. (2016), Parallel recurrent neural network architectures for feature-rich session-based recommendations, *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 241–248.

Hidasi, B. & Tikk, D. (2016), 'General factorization framework for context-aware recommendations', *Data Mining and Knowledge Discovery* **30**(2), 342–371.

Hiltz-Laforge, J., Nonez, R. Y., Pourshahid, A. & Watts, G. A. (2013), 'Pattern-based analysis recommendation'. US Patent App. 14/144, 456.

Hinton, G. E., Osindero, S. & Teh, Y.-W. (2006), 'A fast learning algorithm for deep belief nets', *Neural computation* **18**(7), 1527–1554.

Hu, L., Cao, J., Xu, G., Wang, J., Gu, Z. & Cao, L. (2013), Cross-domain collaborative filtering via bilinear multilevel analysis, *in* 'International Joint Conference on Artificial Intelligence', IJCAI/AAAI, pp. 1–7.

Hu, L., Cao, L., Cao, J., Gu, Z., Xu, G. & Wang, J. (2017), 'Improving the quality of recommendations for users and items in the tail of distribution', *ACM Transactions on Information Systems (TOIS)* **35**(3), 25.

Hu, L., Cao, L., Cao, J., Gu, Z., Xu, G. & Yang, D. (2016), 'Learning informative priors from heterogeneous domains to improve recommendation in cold-start user domains', *ACM Transactions on Information Systems (TOIS)* **35**(2), 13.

Hu, L., Cao, L., Wang, S., Xu, G., Cao, J. & Gu, Z. (2017), Diversifying personalized recommendation with user-session context, *in* 'Proceedings of the 26th International Joint Conference on Artificial Intelligence', pp. 1937–1943.

Huang, C.-L. & Huang, W.-L. (2009), 'Handling sequential pattern decay: Developing a two-stage collaborative recommender system', *Electronic Commerce Research and Applications* **8**(3), 117–129.

Huang, Z., Chen, H. & Zeng, D. (2004), 'Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering', *ACM Transactions on Information Systems (TOIS)* **22**(1), 116–142.

Huang, Z. & Zeng, D. D. (2011), 'Why does collaborative filtering work? transaction-based recommendation model validation and selection by analyzing bipartite random graphs', *INFORMS Journal on Computing* **23**(1), 138–152.

Jannach, D. & Ludewig, M. (2017*a*), Determining characteristics of successful recommendations from log data: a case study, *in* 'Proceedings of the Symposium on Applied Computing', ACM, pp. 1643–1648.

Jannach, D. & Ludewig, M. (2017*b*), When recurrent neural networks meet the neighborhood for session-based recommendation, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 306–310.

Jannach, D., Ludewig, M. & Lerche, L. (2017), 'Session-based item recommendation in e-commerce: on short-term intents, reminders, trends and discounts', *User Modeling and User-Adapted Interaction* **27**(3-5), 351–392.

Jian, S., Cao, L., Pang, G., Lu, K. & Gao, H. (2017), Embedding-based representation of categorical data by hierarchical value coupling learning., *in* 'Proceedings of the 26th International Joint Conference on Artificial Intelligence', pp. 1937–1943.

Jing, H. & Smola, A. J. (2017), Neural survival recommender, *in* 'Proceedings of the Tenth ACM International Conference on Web Search and Data Mining', ACM, pp. 515–524.

Karatzoglou, A. & Hidasi, B. (2017), Deep learning for recommender systems, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 396–397.

Karatzoglou, A., Hidasi, B., Tikk, D., Sar-Shalom, O., Roitman, H., Shapira, B. & Rokach, L. (2016), Recsys' 16 workshop on deep learning for recommender systems (dlrs), *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 415–416.

Karypis, G. (2001), Evaluation of item-based top-n recommendation algorithms, *in* 'Proceedings of the tenth international conference on Information and knowledge management', ACM, pp. 247–254.

Kazienko, P. (2009), 'Mining indirect association rules for web recommendation', *International Journal of Applied Mathematics and Computer Science* **19**(1), 165–186.

Kim, Y. S. & Yum, B.-J. (2011), 'Recommender system based on click stream data using association rule mining', *Expert Systems with Applications* **38**(10), 13320–13327.

Kingma, D. & Ba, J. (2014), 'Adam: A method for stochastic optimization', *arXiv preprint arXiv:1412.6980* .

Koren, Y. (2008), Factorization meets the neighborhood: a multifaceted collaborative filtering model, *in* 'Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', pp. 426–434.

Koren, Y. & Bell, R. (2015), Advances in collaborative filtering, *in* 'Recommender systems handbook', Springer, pp. 77–118.

Koren, Y., Bell, R. & Volinsky, C. (2009), 'Matrix factorization techniques for recommender systems', *Computer* **42**(8).

Krishnamurthy, B., Puri, N. & Goel, R. (2016), 'Learning vector-space representations of items for recommendations using word embedding models', *Procedia Computer Science* **80**, 2205–2210.

Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., Zhong, V., Paulus, R. & Socher, R. (2016), Ask me anything: dynamic memory networks for natural language processing, *in* 'Proceedings of the 33rd International Conference on Machine Learning', pp. 1378–1387.

Kumar, K. & Kumar, S. (2013), 'A rule-based recommendation system for selection of software development life cycle models', *ACM SIGSOFT Software Engineering Notes* **38**(4), 1–6.

Le, D.-T., Fang, Y. & Lauw, H. W. (2016), Modeling sequential preferences with dynamic user and context factors, *in* 'Joint European Conference on Machine Learning and Knowledge Discovery in Databases', Springer, pp. 145–161.

Lee, A. J. & Wang, C.-S. (2007), 'An efficient algorithm for mining frequent inter-transaction patterns', *Information Sciences* **177**(17), 3453–3476.

Lee, C.-H., Kim, Y.-H. & Rhee, P.-K. (2001), 'Web personalization expert with combining collaborative filtering and association rule mining technique', *Expert Systems with Applications* **21**(3), 131–137.

Leng, J. & Jiang, P. (2017), 'Mining and matching relationships from interaction contexts in a social manufacturing paradigm', *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **47**(2), 276–288.

Lerche, L., Jannach, D. & Ludewig, M. (2016), On the value of reminders within e-commerce recommendations, *in* 'Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization', ACM, pp. 27–35.

Li, B., Yang, Q. & Xue, X. (2009), Transfer learning for collaborative filtering via a rating-matrix generative model, *in* 'Proceedings of the 26th annual international conference on machine learning', ACM, pp. 617–624.

Li, H., Wang, Y., Zhang, D., Zhang, M. & Chang, E. Y. (2008), Pfp: parallel fp-growth for query recommendation, *in* 'Proceedings of the 2nd ACM Conference on Recommender Systems', pp. 107–114.

Li, S., Kawale, J. & Fu, Y. (2015), Deep collaborative filtering via marginalized denoising auto-encoder, *in* 'Proceedings of the 24th ACM International on Conference on Information and Knowledge Management', ACM, pp. 811–820.

Li, Y., Chen, W. & Yan, H. (2017), Learning graph-based embedding for time-aware product recommendation, *in* 'Proceedings of the 2017 ACM on Conference on Information and Knowledge Management', ACM, pp. 2163–2166.

Lian, D., Zheng, V. W. & Xie, X. (2013), Collaborative filtering meets next check-in location prediction, *in* 'Proceedings of the 22nd International Conference on World Wide Web', ACM, pp. 231–232.

Liang, D., Altosaar, J., Charlin, L. & Blei, D. M. (2016), Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence, *in* 'Proceedings of the 10th ACM conference on recommender systems', ACM, pp. 59–66.

Lin, W., Alvarez, S. A. & Ruiz, C. (2002), 'Efficient adaptive-support association rule mining for recommender systems', *Data mining and knowledge discovery* **6**(1), 83–105.

Linden, G., Smith, B. & York, J. (2003), 'Amazon. com recommendations: Item-to-item collaborative filtering', *IEEE Internet computing* **7**(1), 76–80.

Liu, D.-R., Lai, C.-H. & Lee, W.-J. (2009), 'A hybrid of sequential rules and collaborative filtering for product recommendation', *Information Sciences* **179**(20), 3505–3519.

Liu, X., Liu, Y., Aberer, K. & Miao, C. (2013), Personalized point-of-interest recommendation by mining users' preference transition, *in* 'Proceedings of the 22nd ACM international conference on Information & Knowledge Management', ACM, pp. 733–738.

Loni, B., Shi, Y., Larson, M. & Hanjalic, A. (2014), Cross-domain collaborative filtering with factorization machines, *in* 'European conference on information retrieval', Springer, pp. 656–661.

Lops, P., De Gemmis, M. & Semeraro, G. (2011), Content-based recommender systems: State of the art and trends, *in* 'Recommender systems handbook', Springer, pp. 73–105.

Loyola, P., Liu, C. & Hirate, Y. (2017), Modeling user session and intent with an attention-based encoder-decoder architecture, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 147–151.

Lu, J., Xiong, C., Parikh, D. & Socher, R. (2017), Knowing when to look: Adaptive attention via a visual sentinel for image captioning, *in* 'Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)', Vol. 6, p. 2.

Luo, L., Li, B., Koprinska, I., Berkovsky, S. & Chen, F. (2017), Tracking the evolution of customer purchase behavior segmentation via a fragmentation-coagulation process, *in* 'Proceedings of the 26th International Joint Conference on Artificial Intelligence', pp. 2414–2420.

Luong, M.-T., Pham, H. & Manning, C. D. (2015), 'Effective approaches to attention-based neural machine translation', *arXiv preprint arXiv:1508.04025* .

Marinai, S., Gori, M. & Soda, G. (2005), 'Artificial neural networks for document analysis and recognition', *IEEE Transactions on pattern analysis and machine intelligence* **27**(1), 23–35.

Melville, P., Mooney, R. J. & Nagarajan, R. (2002), Content-boosted collaborative filtering for improved recommendations, *in* 'Aaai/iaai', pp. 187–192.

Menon, A. K., Chitrapura, K.-P., Garg, S., Agarwal, D. & Kota, N. (2011), Response prediction using collaborative filtering with hierarchies and side-information, *in* 'Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining', ACM, pp. 141–149.

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013), 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* .

Mikolov, T., Le, Q. V. & Sutskever, I. (2013), 'Exploiting similarities among languages for machine translation', *arXiv preprint arXiv:1309.4168* .

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. (2013), Distributed representations of words and phrases and their compositionality, *in* 'Advances in neural information processing systems', pp. 3111–3119.

Mnih, A. & Kavukcuoglu, K. (2013), Learning word embeddings efficiently with noise-contrastive estimation, *in* 'Advances in neural information processing systems', pp. 2265–2273.

Mnih, A. & Teh, Y. W. (2012), A fast and simple algorithm for training neural probabilistic language models, *in* 'Proceedings of the 29th International Conference on Machine Learning', pp. 419–426.

Mobasher, B., Dai, H., Luo, T. & Nakagawa, M. (2001), Effective personalization based on association rule discovery from web usage data, *in* 'Proceedings of the 3rd international workshop on Web information and data management', ACM, pp. 9–15.

Morales, C. R., Pérez, A. P., Soto, S. V., Martınez, C. H. & Zafra, A. (2006), 'Using sequential pattern mining for links recommendation in adaptive hypermedia educational systems', *Current Developments in Technology-Assisted Education* **2**, 1016–1020.

Moreno, M. N., García, F. J., Polo, M. J. & López, V. F. (2004), Using association analysis of web data in recommender systems, *in* 'International Conference on Electronic Commerce and Web Technologies', Springer, pp. 11–20.

Nijssen, S., Guns, T. & De Raedt, L. (2009), Correlated itemset mining in ROC space: A constraint programming approach, *in* '15th International Conference on Knowledge Discovery and Data Mining', ACM, pp. 647–656.

Niranjan, U., Subramanyam, R. & Khanaa, V. (2010), Developing a web recommendation system based on closed sequential patterns, *in* 'International Conference on Advances in Information and Communication Technologies', Springer, pp. 171–179.

Omiecinski, E. R. (2003), 'Alternative interest measures for mining associations in databases', *IEEE Transactions on Knowledge and Data Engineering* **15**(1), 57–69.

Ozsoy, M. G. (2016), 'From word embeddings to item recommendation', *arXiv preprint arXiv:1601.01356* .

Pagano, R., Cremonesi, P., Larson, M., Hidasi, B., Tikk, D., Karatzoglou, A. & Quadrana, M. (2016), The contextual turn: From context-aware to context-driven recommender systems, *in* 'Proceedings of the 10th ACM conference on recommender systems', ACM, pp. 249–252.

Pan, S. J. & Yang, Q. (2010), 'A survey on transfer learning', *IEEE Transactions on knowledge and data engineering* **22**(10), 1345–1359.

Pan, W., Xiang, E. W., Liu, N. N. & Yang, Q. (2010), Transfer learning in collaborative filtering for sparsity reduction., *in* 'AAAI', Vol. 10, pp. 230–235.

Pan, W. & Yang, Q. (2013), 'Transfer learning in heterogeneous collaborative filtering domains', *Artificial intelligence* **197**, 39–55.

Park, K., Lee, J. & Choi, J. (2017), Deep neural networks for news recommendations, *in* 'Proceedings of the 2017 ACM on Conference on Information and Knowledge Management', ACM, pp. 2255–2258.

Park, Y.-J. & Tuzhilin, A. (2008), The long tail of recommender systems and how to leverage it, *in* 'Proceedings of the 2008 ACM conference on Recommender systems', ACM, pp. 11–18.

Pazzani, M. J. & Billsus, D. (2007), Content-based recommendation systems, *in* 'The adaptive web', Springer, pp. 325–341.

Pearl, J., Glymour, M. & Jewell, N. P. (2016), *Causal Inference in Statistics: A Primer*, New York: John Wiley & Sons.

Pedersoli, M., Lucas, T., Schmid, C. & Verbeek, J. (2016), 'Areas of attention for image captioning', *arXiv preprint arXiv:1612.01033* .

Pei, W., Yang, J., Sun, Z., Zhang, J., Bozzon, A. & Tax, D. M. (2017), Interacting attention-gated recurrent networks for recommendation, *in* 'Proceedings of the 2017 ACM on Conference on Information and Knowledge Management', ACM, pp. 1459–1468.

Pennington, J., Socher, R. & Manning, C. (2014), Glove: Global vectors for word representation, *in* 'Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)', pp. 1532–1543.

Peska, L. & Vojtas, P. (2016), 'Using implicit preference relations to improve recommender systems', *Journal on Data Semantics* pp. 1–16.

Peska, L. & Vojtas, P. (2017), 'Using implicit preference relations to improve recommender systems', *Journal on Data Semantics* **6**(1), 15–30.

Pyun, G., Yun, U. & Ryu, K. H. (2014), 'Efficient frequent pattern mining based on linear prefix tree', *Knowledge-Based Systems* **55**, 125–139.

Quadrana, M., Karatzoglou, A., Hidasi, B. & Cremonesi, P. (2017), Personalizing session-based recommendations with hierarchical recurrent neural networks, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 130–137.

Rendle, S., Freudenthaler, C. & Schmidt-Thieme, L. (2010), Factorizing personalized markov chains for next-basket recommendation, *in* 'Proceedings of the 19th international conference on World wide web', ACM, pp. 811–820.

Resnick, P. & Varian, H. R. (1997), 'Recommender systems', *Communications of the ACM* **40**(3), 56–58.

Ricci, F., Rokach, L. & Shapira, B. (2015), Recommender systems: introduction and challenges, *in* 'Recommender systems handbook', Springer, pp. 1–34.

Role, F. & Nadif, M. (2011), Handling the impact of low frequency events on co-occurrence based measures of word similarity: A case study of pointwise mutual information., *in* '3rd International Conference on Knowledge Discovery and Information Retrieval', pp. 226–231.

Rong, X. (2014), 'word2vec parameter learning explained', *arXiv preprint arXiv:1411.2738* .

Rosen-Zvi, M., Chemudugunta, C., Griffiths, T., Smyth, P. & Steyvers, M. (2010), 'Learning author-topic models from text corpora', *ACM Transactions on Information Systems* **28**(1), 1–38.

Ruocco, M., Skrede, O. S. L. & Langseth, H. (2017), Inter-session modeling for session-based recommendation, *in* 'Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems', ACM, pp. 24–31.

Sahoo, J., Das, A. K. & Goswami, A. (2015), 'An effective association rule mining scheme using a new generic basis', *Knowledge and Information Systems* **43**(1), 127–156.

Salakhutdinov, R., Mnih, A. & Hinton, G. (2007), Restricted boltzmann machines for collaborative filtering, *in* 'Proceedings of the 24th international conference on Machine learning', ACM, pp. 791–798.

Sandvig, J. J., Mobasher, B. & Burke, R. (2007), Robustness of collaborative recommendation based on association rule mining, *in* 'Proceedings of the 2007 ACM conference on Recommender systems', ACM, pp. 105–112.

Schafer, J. B., Frankowski, D., Herlocker, J. & Sen, S. (2007), Collaborative filtering recommender systems, *in* 'The adaptive web', Springer, pp. 291–324.

Schnabel, T., Bennett, P. N., Dumais, S. T. & Joachims, T. (2018), Short-term satisfaction and long-term coverage: Understanding how users tolerate algorithmic exploration, *in* 'Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining', ACM, pp. 513–521.

Shani, G., Heckerman, D. & Brafman, R. I. (2005), 'An mdp-based recommender system', *Journal of Machine Learning Research* **6**(Sep), 1265–1295.

Shao, B., Wang, D., Li, T. & Ogihara, M. (2009), 'Music recommendation based on acoustic features and user access patterns', *IEEE Transactions on Audio, Speech, and Language Processing* **17**(8), 1602–1611.

Shaonan, W., Jiajun, Z. & Chengqing, Z. (2017), Learning sentence representation with guidance of human attention, *in* 'Proceedings of the 26th International Joint Conference on Artificial Intelligence', pp. 4137–4143.

Shi, Y., Larson, M. & Hanjalic, A. (2014), 'Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges', *ACM Computing Surveys (CSUR)* **47**(1), 3.

Shoujin, W., Ying, Q. & Kun, L. W. G. (2013), 'A study of spreading mode and features of microblogging based on topics', *Journal of Intelligence* **6**, 28–33.

Singh, B. & Jain, S. (2005), 'Semicompatibility and fixed point theorems in fuzzy metric space using implicit relation', *International Journal of Mathematics and Mathematical Sciences* **2005**(16), 2617–2629.

Smirnova, E. & Vasile, F. (2017), 'Contextual sequence modeling for recommendation with recurrent neural networks', *arXiv preprint arXiv:1706.07684* .

Sokolova, E., Groot, P., Claassen, T., von Rhein, D., Buitelaar, J. & Heskes, T. (2015), Causal discovery from medical data: Dealing with missing values and a mixture of discrete and continuous data, *in* '18th Conference on Artificial Intelligence in Medicine', pp. 177–181.

Song, W. & Yang, K. (2014), Personalized recommendation based on weighted sequence similarity, *in* 'Practical Applications of Intelligent Systems', Springer, pp. 657–666.

Song, Y., Elkahky, A. M. & He, X. (2016), Multi-rate deep learning for temporal recommendation, *in* 'Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval', ACM, pp. 909–912.

Su, X. & Khoshgoftaar, T. M. (2009), 'A survey of collaborative filtering techniques', *Advances in artificial intelligence* **2009**, 4.

Sun, J., Wang, S., Gao, B. J. & Ma, J. (2012), Learning to rank for hybrid recommendation, *in* 'Proceedings of the 21st ACM International Conference on Information and Knowledge Management', ACM, pp. 2239–2242.

Tan, Y. K., Xu, X. & Liu, Y. (2016), Improved recurrent neural networks for session-based recommendations, *in* 'Proceedings of the 1st Workshop on Deep Learning for Recommender Systems', ACM, pp. 17–22.

Tang, J. & Wang, K. (2018), 'Personalized top-n sequential recommendation via convolutional sequence embedding'.

Tuan, T. X. & Phuong, T. M. (2017), 3d convolutional networks for session-based recommendation with content features, *in* 'Proceedings of the Eleventh ACM Conference on Recommender Systems', ACM, pp. 138–146.

Tung, A. K. H., Lu, H., Han, J. & Feng, L. (2003), 'Efficient mining of intertransaction association rules', *IEEE transactions on knowledge and data engineering* **15**(1), 43–56.

Twardowski, B. (2016), Modelling contextual information in session-aware recommender systems with neural networks, *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 273–276.

Unger, M. (2015), Latent context-aware recommender systems, *in* 'Proceedings of the 9th ACM Conference on Recommender Systems', ACM, pp. 383–386.

Vargas, S. & Castells, P. (2011), Rank and relevance in novelty and diversity metrics for recommender systems, *in* 'Proceedings of the 5th ACM Conference on Recommender Systems', pp. 109–116.

Vasile, F., Smirnova, E. & Conneau, A. (2016), Meta-prod2vec: Product embeddings using side-information for recommendation, *in* 'Proceedings of the 10th ACM Conference on Recommender Systems', ACM, pp. 225–232.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017), Attention is all you need, *in* 'Advances in Neural Information Processing Systems', pp. 5998–6008.

Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I. & Duval, E. (2012), 'Context-aware recommender systems for learning: a survey and future challenges', *IEEE Transactions on Learning Technologies* **5**(4), 318–335.

Wan, Q. & An, A. (2003), Efficient mining of indirect associations using hi-mine, *in* 'Conference of the Canadian Society for Computational Studies of Intelligence', Springer, pp. 206–221.

Wan, S., Lan, Y., Wang, P., Guo, J., Xu, J. & Cheng, X. (2015), Next basket recommendation with neural networks., *in* 'RecSys Posters'.

Wang, D., Deng, S. & Xu, G. (2017), 'Sequence-based context-aware music recommendation', *Information Retrieval Journal* pp. 1–23.

Wang, D., Deng, S., Zhang, X. & Xu, G. (2016), Learning music embedding with metadata for context aware recommendation, *in* 'Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval', ACM, pp. 249–253.

Wang, H., Wang, N. & Yeung, D.-Y. (2015), Collaborative deep learning for recommender systems, *in* 'Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining', ACM, pp. 1235–1244.

Wang, L., Bao, X. & Zhou, L. (2017), 'Redundancy reduction for prevalent co-location patterns', *IEEE Transactions on Knowledge and Data Engineering* **29**, 1–14.

Wang, P., Guo, J., Lan, Y., Xu, J., Wan, S. & Cheng, X. (2015), Learning hierarchical representation model for nextbasket recommendation, *in* 'Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval', ACM, pp. 403–412.

Wang, S. & Cao, L. (2017), 'Inferring implicit rules by learning explicit and hidden item dependency', *IEEE Transactions on Systems, Man, and Cybernetics: Systems* .

Wang, S., Hu, L. & Cao, L. (2017), Perceiving the next choice with comprehensive transaction embeddings for online recommendation, *in* 'Joint European Conference on Machine Learning and Knowledge Discovery in Databases', Springer, pp. 285–302.

Wang, S., Hu, L., Cao, L., Huang, X., Lian, D. & Liu, W. (2018), Attention-based transactional context embedding for next-item recommendation, AAAI, pp. 2532–2539.

Wang, S., Liu, W., Wu, J., Cao, L., Meng, Q. & Kennedy, P. J. (2016), Training deep neural networks on imbalanced data sets, *in* 'Proceedings of the 29th International Joint Conference on Neural Networks (IJCNN)', pp. 4368–4374.

Weng, S.-S. & Liu, M.-J. (2004), 'Feature-based recommendations for one-to-one marketing', *Expert Systems with Applications* **26**(4), 493–508.

Wu, C.-Y., Ahmed, A., Beutel, A., Smola, A. J. & Jing, H. (2017), Recurrent recommender networks, *in* 'Proceedings of the Tenth ACM International Conference on Web Search and Data Mining', ACM, pp. 495–503.

Wu, C. & Yan, M. (2017), Session-aware information embedding for e-commerce product recommendation, *in* 'Proceedings of the 2017 ACM on Conference on Information and Knowledge Management', ACM, pp. 2379–2382.

Wu, X., Liu, Q., Chen, E., He, L., Lv, J., Cao, C. & Hu, G. (2013), Personalized next-song recommendation in online karaokes, *in* 'Proceedings of the 7th ACM conference on Recommender systems', ACM, pp. 137–140.

Wu, X., Zhu, X., Wu, G.-Q. & Ding, W. (2014), 'Data mining with big data', *IEEE Transactions on Knowledge and Data Engineering* **26**(1), 97–107.

Xiaoling, Q. Y. W. S. J. & Qiaofeng, Z. Y. L. (2012), 'Information sharing behavior for different age groups on microblogging [j]', *Journal of Intelligence* **11**, 1–5.

Yan, L. & Li, C. (2006), Incorporating pageview weight into an association-rule-based web recommendation system, *in* 'Australasian Joint Conference on Artificial Intelligence', Springer, pp. 577–586.

Yang, C. C., Tang, X., Dai, Q., Yang, H. & Jiang, L. (2013), 'Identifying implicit and explicit relationships through user activities in social media', *International Journal of Electronic Commerce* **18**(2), 73–96.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A. J. & Hovy, E. H. (2016), Hierarchical attention networks for document classification., *in* 'HLT-NAACL', pp. 1480–1489.

Yap, G.-E., Li, X.-L. & Philip, S. Y. (2012), Effective next-items recommendation via personalized sequential pattern mining, *in* 'International Conference on Database Systems for Advanced Applications', Springer, pp. 48–64.

Yap, G.-E., Tan, A.-H. & Pang, H.-H. (2007), 'Discovering and exploiting causal dependencies for robust mobile context-aware recommenders', *IEEE Transactions on Knowledge and Data Engineering* **19**(7), 977–992.

Yu, F., Liu, Q., Wu, S., Wang, L. & Tan, T. (2016), A dynamic recurrent model for next basket recommendation, *in* 'Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval', ACM, pp. 729–732.

Yuan, Q., Cong, G., Ma, Z., Sun, A. & Thalmann, N. M. (2013), Time-aware point-of-interest recommendation, *in* 'Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval', pp. 363–372.

Zhang, X.-Z. (2007), Building personalized recommendation system in e-commerce using association rule-based mining and classification, *in* 'Machine Learning and Cybernetics, 2007 International Conference on', Vol. 7, IEEE, pp. 4113–4118.

Zhang, Y. & Cao, J. (2013), Personalized recommendation based on behavior sequence similarity measures, *in* 'Behavior and Social Computing', Springer, pp. 165–177.

Zhang, Z. & Nasraoui, O. (2007), Efficient hybrid web recommendations based on markov clickstream models and implicit search, *in* 'Web Intelligence, IEEE/WIC/ACM International Conference on', IEEE, pp. 621–627.

Zhao, Q. & Bhowmick, S. S. (2003), 'Association rule mining: A survey', *Nanyang Technological University, Singapore* .

Zhao, W., Wang, W., Ye, J., Gao, Y., Yang, M., Zhao, Z. & Chen, X. (2017), 'Leveraging long and short-term information in content-aware movie recommendation', *arXiv preprint arXiv:1712.09059* .

Zhao, W. X., Huang, J. & Wen, J.-R. (2016), Learning distributed representations for recommender systems with a network embedding approach, *in* 'Asia Information Retrieval Symposium', Springer, pp. 224–236.

Zhou, B., Hui, S. C. & Fong, A. C. M. (2006), 'Efficient sequential access pattern mining for web recommendations', *International Journal of Knowledge-based and Intelligent Engineering Systems* **10**(2), 155–168.