Explainable and Unexpectable Recommendations Using Relational Learning on Multiple Domains

Submitted to

Graduate School of Information Science and Technology

Osaka University

January 2020

Sirawit SOPCHOKE

Abstract

In this thesis, relational learning was combined with multiple domains to form a new framework for recommendation systems. The design of the proposed framework aims at: (i) providing suggested items with clear and understandable explanations, and (ii) delivering a broad range of recommendations including novel and unexpected items.

Relational learning is used to form an explanation which focuses on why a user would potentially like the item recommended. The explanation is directly transformed from the recommendation rules which learned from the data and probabilistic setting using Inductive Logic Programming (ILP). The data used is prepared in prolog format. The predicates and all related information can be defined in the setting. The output learned rules, in *if-then* logic format, will be ranked based on their probabilities and be used to compute the recommendation items. The top-N items in the list accompanied with the explanations will be recommended to a particular user.

In addition to relational learning, various sources of information and knowledge are required to produce the unseen or unexpected items. The proposed framework shows how relational learning, with its ability in finding all possible relations especially the novel relations on multiple domains, can broaden the search space to significantly increase a chance to discover items which are previously hidden from a user.

This study is highly original in the following aspects: the proposed framework provides an explainable recommendation on multiple domains which is not available in the literatures. The content of each explanation varies, depending on the actual causal relationship found between objects, not limited to the type of the underlying algorithm as commonly found in typical recommender systems. The explanation form presented is simple and readable which is scarce in other explanation recommendation systems. The framework also provides more coverage of the new and unseen items. Furthermore, the generality is expressed via the recommendation rule which is in first-order logic format. The extensibility is governed by the use of relational learning requiring no tedious work e.g. re-training.

To illustrate that the framework design is indeed feasible and applicable, an experiment was conducted and evaluated. The experiment results confirm that the proposed framework is very promising as it does produce interesting recommendations not found in the primitive recommender systems and accompanied with readable and concise explanations.

Acknowledgement

I would like to acknowledge the following people who, at different times, have helped me with this thesis. My deepest gratitude and appreciation goes to my advisor, Professor Masayuki Numao, who has given me a great deal of freedom to pursue ideas and research topics and who is always there when I need help and guidance. His unfailing support, his patience, his constructive criticism and his vast experience and invaluable advice have made the writing of this thesis possible and enjoyable.

I am also thankful to Associate Professor Ken-ichi Fukui for his support, helpful comments and enlightenment with regards to my research. My gratitude also goes to Professor Hideyuki Suzuki and Professor Jun Tanida, the members of my thesis committee, for the remarkable feedback received during my viva.

I would like to express my sincere appreciation to Professor Luc De Raedt for his guidance and invaluable advice during my short programs at the Department of Computer Science, KU Leuven in Belgium.

I would like to acknowledge the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan for the Japanese Government (MEXT) scholarship I received and the JSPS Core-to-Core Program for the generous support for the overseas study program during my research.

I am grateful to Professor Boonserm Kijsirikul for supporting and encouraging me to pursue my doctorate study at the Osaka University. A special gratitude also goes to Associate Professor Saranya Maneeroj who introduced me to the field of Recommender Systems.

I thank my friends over the years, Dr. Wasin Kalintha, Chonlathep Kitsinthopchai, Noppayut Sriwatanasakdi, Ekasit Phermphoonphiphat, Juan Lorenzo Hagad, and all my friends in Japan. We shared so many great experiences and thoughts that will always stay with me.

I am also thankful to former and current students of Numao laboratory for the sharing, the support & friendship and the fun moments together. I am also grateful to the laboratory secretaries for all the support they have given me during my study at the university.

I am deeply grateful to my grandma, Somjit Naruedomkul, for her unconditional love and support and for always being there for me in good and difficult times and to my whole family for their love, understanding and support throughout my study in Japan.

Lastly, I dedicate this thesis in memory of my late grandpa, Kiat Naruedomkul, who taught me the value of education and prepared me for my future.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Research Motivation

"A lot of times, people don't know what they want until you show it to them" [64], is one of the most famous opinions from a highly opinionated man - Steve Jobs. Regardless of other views (especially Forbes) on this statement, I totally agree with Jobs, at least in regard to developing the recommender system (RS). Recommender system tries to present people with suggestions that they are most likely to buy or use.

During the last few decades, recommendation system has been almost everywhere from Amazon[1] to Netflix[2], from Facebook[3] to LinkedIn[4]. RS has taken more and more places in people's lives, influencing them on, for example, the music they listen to, the movies they watch, the books they read, the persons they make friend with. It has changed the way businesses communicate with users and strengthened the interactivity among them. RS has, thus, become one of the most basic supportive techniques in an online world and it has proven to be a major source of enhanced functionality, user satisfaction, and revenue improvement.

The most common critical issues found with RS include but not limited to maximizing prediction accuracy, solving cold-start problem, reducing sparsity, and providing novelty, diversity and serendipity [1]. However, solving one problem may create another problem, or a trade-off. All issues have not been perfectly solved since many current recommender algorithms seem to

[1] https://www.amazon.com/
[2] https://www.netflix.com/
[3] https://www.facebook.com/
[4] https://www.linkedin.com/

be locked away inside a black box [27]. Once an algorithm is processed, it is quite difficult to understand why it gives a particular recommendation to a set of data. If the reason behind the recommendation is understood, I believe it is possible to find the way to handle the problems mentioned above more effectively. For example, the cold-start and sparsity problems can be managed using such rationale to form the preference for new users. I also believe that the rationale can draw users' attention and influence their decision to try out or to purchase eventually since that rationale indicates how the suggested items relate to their preference items either directly or indirectly.

Moreover, the traditional recommendation approaches (i.e. Content-Based Filtering [5] and Collaborative Filtering [50]) assume that users want to see the content similar to what they (or their friends) already rate highly. Thus, users are locked into clusters of similarity and may find recommendations provided practically useless for possibly a number of reasons, for example, the user has all items on the list or the user feels bored by having too many similar items or the user is no longer interested in the similar items. This is the long tail problem in RS [12], where the recommendations are centered around very small popular items in the head. There may be surprised items from the long tail that users may be interested in but they do not know about or are not aware that those items exist. To overcome this problem, RS should suggest broadly the novel and unexpected items from the long tail which is a challenging task.

Novel recommendations refer to recommendations of those items that the user does not know (previously unknown) [33]. Unexpected recommendations are those recommendations that significantly depart from the user's expectations [1], either previously known or unknown. In fact, there is possibly an overlap between novel and unexpected recommendations. To achieve novel or unexpected recommendations, I need to look beyond a single domain that contains only a user's preferences or ratings. I believe that the discovery of surprising correlations in the repositories coming from diverse domains can lead to novel and unexpected recommendations. In addition, the recommendation should be able to be made in any domain depending on the generated relational rule, not limited to a particular domain specified by the user. In other words, there is no need for a user to define the target domain at the beginning of the process.

The other key supporting idea in this study is the success of cross-domain recommender

systems which suggest items in one domain using information from another domain. User's knowledge acquired in one domain can be transferred to and exploited in the other domain instead of treating each domain separately. Cross-domain approach improves prediction accuracy by reducing data sparsity and offers added values to recommendations by providing diversity, novelty and serendipity predictions [10]. These benefits thus encourage the development of the proposed framework on multi-domain. However, unlike the typical cross-domain approach, a setting for specific domains (source and target domain) should not be required prior to constructing the recommendation rules and knowledge transfer across domains should not be needed either.

Relational learning has already shown its use in RS. The evidences from the researches by Kouki et al. [35] and Catherine and Cohen [11] indicate that relational learning provides better recommendations by incorporating additional information, compared to traditional methods with a single dyadic relationship between the objects i.e. users and items, The most important capabilities of relational learning are that: it produces simple and understandable rules, it does not need to predefine the role of the domain and it constructs the general rule for recommendation. Consequently, the relational learning captures my interest to model and provide a potential solution for explainable multi-domain recommendations.

In this research, a new framework for RS is proposed with the following key components: (i) the use of relational learning to learn the relations, possibly new, between users and items, (ii) the incorporation of user information and knowledge from various domains. With the advantages of relational learning, the proposed framework is able to construct the probabilistic general rules, the rules with associating probability, for recommendations. These rules are used to create the recommendations for the users with transparency by describing how the recommendation was chosen for a particular user. Moreover, the recommendation can be made in any domain, no need to pre-specify the source-target domain pair. With the information available in various domains, a wide range of recommendations including novel and unexpected items can also be delivered. Additionally, new domains, new data (e.g. features), and context (e.g. location, time, and mood) can simply be incorporated to the framework.

1.2 Motivating Examples

In this section, the motivating examples from *music* and *movie* domains and a scenario are provided to illustrate my ideas as shown below.

Example 1: Imagine Kelly likes music "A" and enjoys movie "B" which has three music C, D, E. With the single-domain based RS, she will be recommended only the music which is similar to music "A". Unfortunately, the music C, D, E will not be recommended to her since they are not considered as similar to "A". Now imagine that if the RS can somehow relate the two different domains, music and movie, it can then recommend the items besides the music similar to A to Kelly. For example, Kelly can be recommended with the music C, D, E or their similarities or the music which somehow relates to music A via its properties or attributes (i.e. music theme, movie genre, music artist). At any rate, no matter what the recommendation is, it will not be what Kelly expects.

Example 2: Suppose Kelly likes movie "A" and John likes movie "B". Both "A" and "B" share the same genre "thriller". John likes music "C" which is not the item that Kelly is familiar with (not found in Kelly's preference). With the links or the relations between this information, RS can possibly recommend music "C" to Kelly or its similarity or the music which somehow relates to music "C" via its properties or attributes. Again, the recommendation will be a surprise to Kelly.

1.3 Contributions

The main contributions of this thesis can be summarized as follows:

(i) **New RS framework to construct the general recommendation rules**

A new RS framework is designed to construct the general rules for recommendations. To my best knowledge, it is the first time that the recommendation rule, not the recommendation item, is generated. To attain this goal, relational learning is used on multiple domains to find all possible relations, including novel relations, to form the rules. Each rule is represented in relational logic, a formal language, associating with probability. The main advantage of this framework is that the rules can be used to suggest the items in any do-

main to the user whose preferences or other properties satisfy the conditions of the rule. The associated probability can drive the user to make a decision easier and faster with more confidence. This work was accepted to be appeared in *Intelligent Data Analysis, an international journal* (impact factor 2019 - 0.612) [59]. The details of the framework can be found in Chapter 3.

(ii) **Inductive explanation**

In this thesis, the main focus is on the explanation that serves *transparency* purpose which means that the generated explanation must clearly present why such recommendation is made for a particular user. In this framework, the explanation is inducted from user preferences and all the provided knowledge. This study contributes to an explanation which is based on the actual relationship found between the recommended items and the users' preference items either directly or indirectly (via other features, other users), not constrained by the type of the underlying recommendation model. The explanation is formed at the same time during the recommendation rule construction process. It does not need any additional complicated process.

(iii) **Clear and understandable explanation form**

The proposed framework provides suggested items along with the clear and easy to understand explanations. The explanation is presented in *if-then* form which is unambiguous, less redundant and more concise compared to a natural language, take for example, the English used in other explanation recommendation systems. It is also readable with no additional knowledge required as opposed to other forms of explanation, e.g., visual image, tag cloud.

(iv) **Novel and unexpected items coverage**

With the information available in various domains, the proposed framework is able to deliver a broad range of recommendations including novel and unexpected items. The items which are normally not familiar to the users or are hidden from them will be brought out. The recommendation no longer focusses around very small popular items in the head and the long tail problem can possibly be dealt with.

(v) **Extendibility**

The proposed framework is extendible in that it allows the new domains (e.g. TV show, game, sport, and beauty) to be simply added to the framework and only predicate setting is simply specified. New data (e.g., user/item feature and feature from review text) and different contexts (e.g., location, time, and mood) can also be included as different domains.

1.4 List of Publications

Journal

- Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. “Explainable and Unexpectable Recommendations using Relational Learning on Multiple Domains”, Intelligent Data Analysis, 24(6). 2020. (in Press)

Peer-Reviewed International Conference Proceedings

- Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. "ILP Recommender System: Explainable and More", Proc. the 29th International Conference on Inductive Logic Programming (ILP 2019), Plovdiv, Bulgaria, Sep. 2019.

- Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. “Explainable Cross-Domain Recommendations Through Relational Learning”, Proc. the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), New Orleans, USA, Feb. 2018.

- Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. “Bisociative Serendipity Music Recommendation”, Workshop on Computation: Theory and Practice (WCTP-2017), Osaka, Japan, Sep. 2017.

Non-Peer-Reviewed International Conference Proceedings

- Sirawit Sopchoke. “Cross-domain Music Recommender System”, Spring workshop on Mining and Learning 2017, Oostende, Belgium, May. 2017.

- Sirawit Sopchoke, Luc De Raedt, Ken-ichi Fukui, and Masayuki Numao. “Cross-domain

Recommendation in Heterogeneous Settings", Proc. the 20th SANKEN International & the 15th SANKEN Nanotechnology Symposium, Osaka, Japan, Dec. 2016.

Non-Peer-Reviewed Domestic Conference Proceedings

- Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. "Feature-Based Cross-Domain Recommendation Using Inductive Learning", 大阪大学産業科学研究所 第73回学術講演会, Osaka, Nov. 2017.

- Sirawit Sopchoke, Ken-ichi Fukui, and Masayuki Numao. "Beyond Similarity: Serendipitous Music Recommenders", Proc. the 30th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI-2016), Kitakyushu, Jun. 2015.

1.5 Organization of the Thesis

In Chapter 2, a brief introduction to RS including a discussion of the methodologies used in developing RS is provided. Following this introduction, an overview of the published studies in explanation in RS and relational learning in RS are summarized. The chapter ends with the background information required for the thesis.

Chapter 3 begins with the two research questions addressed in this thesis. The answer to each question results in the design of the framework. The key features of the framework which make the framework differ from the typical explainable and cross domain RS are presented. The probabilistic first order rule learner (ProbFOIL) which is adopted to learn the recommendation rule is also introduced.

In Chapter 4, The experiment results on Amazon standard dataset and an evaluation on the framework performance are presented.

In Chapter 5, the extendibility feature of the framework is further discussed.

Chapter 6, the last chapter of the thesis summarizes key findings, draws the conclusions along with interesting future research directions.

Chapter 2

Literature Review

2.1 What is Recommender System?

People are increasingly overwhelmed by information available on online channels which seem to provide ample choices. It has become more challenging for people to find the choices that will perfectly satisfy their needs and some may have insufficient experience to make right decisions. Hence, a tool which can scope or screen choices for users or suggest other alternatives that users do not even know that they exist becomes necessary. With this tool, they (users) will have more chances to make a better decision or select a better choice. Such tool is known as "Recommender System".

The recommender system (RS) problem is about how to predict the unknown rating of user u on target item i. It is not the same problem as that of the information retrieval or the search engine because in the recommender system, we do not know what we are looking for at the very beginning.

The main objective of the service providers or sellers is revenue (including customer satisfaction and loyalty). For example, 35% of what consumers purchased on Amazon and 75% of what they watched on Netflix came from recommendation systems [39]. As for the user, his/her objective is to find the needed items.

2.2 Recommender System Methodologies - Traditional Methods

In the past decade, a number of approaches were used in developing RS. Among them, the traditional techniques: content-based and collaborative filtering were the most widely used.

Content-based filtering recommends items that are similar in content to items that you liked in the past while collaborative filtering learns user's preferences based on user community's previous behaviors.

2.2.1 Content-Based Filtering Methods

Content-Based Filtering (CBF) recommends items that are similar in content to items that you liked, or positively rated, or purchased in the past. The basic concept of CBF is depicted in Fig. 2.1. The similarity of items is calculated based on the features of the items associated with a user's previous behaviors. For example, if a user has a 5-star rating on a movie that belongs to the action genre, then, the system can learn to recommend other movies from this genre. The updated progress of CBF methods can be found in details at [38].

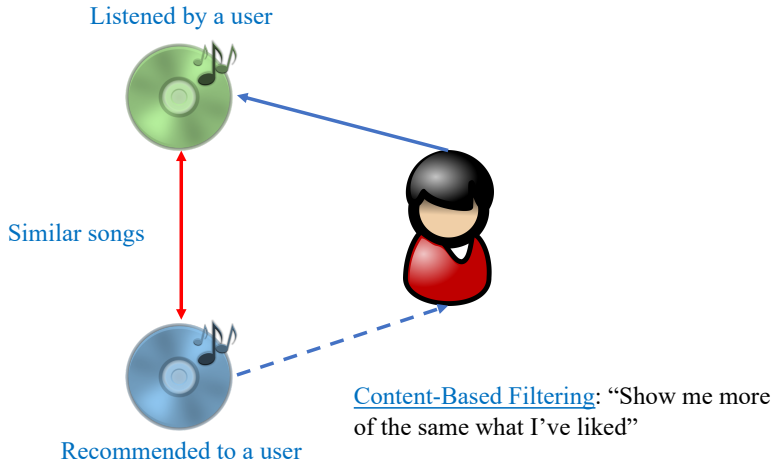

Figure 2.1: Content-Based Filtering

The advantages of CBF are that no community is required, and comparison between items is possible. However, the disadvantages are that content-descriptions are needed, new users will face a cold start problem, and recommendations are no surprise to the users.

2.2.2 Collaborative Filtering Methods

The most popular approach to recommender systems is Collaborative Filtering (CF). CF strategies have been successfully applied to several real world systems including Amazon.com and Netflix. CF learns user's preferences based on user community's previous behaviors. The basic concept of CF is depicted in Fig. 2.2. CF strategies do not require domain knowledge and can

predict user's interests, while the CBF strategies cannot.

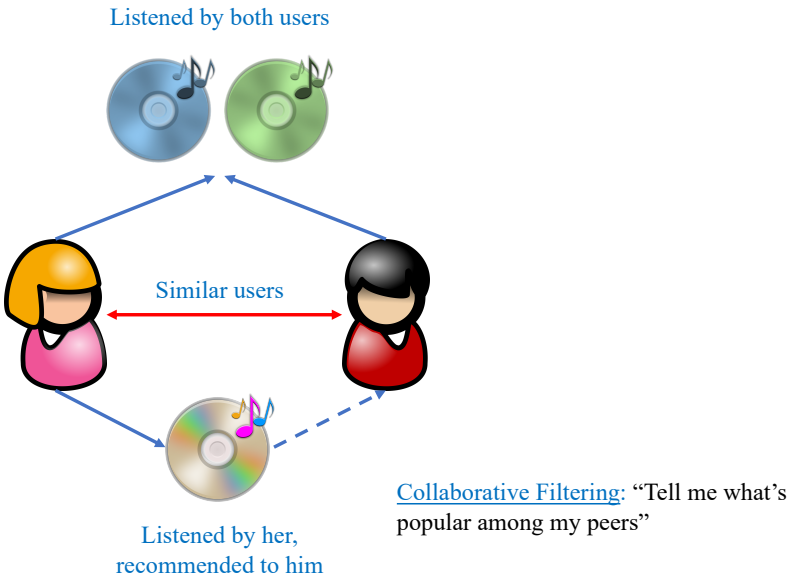

Figure 2.2: Collaborative Filtering

CF algorithms can be classified into two distinct categories: memory-based and model-based. Memory-based algorithm uses users' previous preferences to make new predictions. In a typical memory-based algorithm, an unknown rating of an item given by a user is simply an aggregation of the ratings given by other users, who have similar interests to that individual, for that item. Unlike memory-based algorithm, model-based CF algorithm uses user's preference database to learn a model after which it uses this model to make predictions and recommendations. The state-of-the-art techniques to model-based CF which provide accurate and efficient recommendations are latent factor models based on matrix factorization.

The advantages of CF are that it requires nearly no ramp-up effort, recommendations may be serendipitious to the users and the system is able to learn the market segments. However, the disadvantages are that it requires some form of rating feedback, and new users and/or new items will face a cold start problem.

Matrix Factorization

Matrix Factorization (MF) for model-based CF was proposed by Koren et al. [34]. The technique attempts to learn the latent factors to predict the missing ratings in a user-item rating matrix. The basic idea of matrix factorization is that the user-item interaction matrix **R** can

be approximated by decomposing **R** into two low rank matrices **X** and **Y**. The result of dot product, $\mathbf{X}_u \cdot \mathbf{Y}_i^T$, approximates the interaction between user u and item i. If there are m users and n items, the dimension of matrix **X** and matrix **Y** is m by k and n by k respectively. The k-dimension represents the number of latent factors inferred from user-interest information patterns (Fig. 2.3).

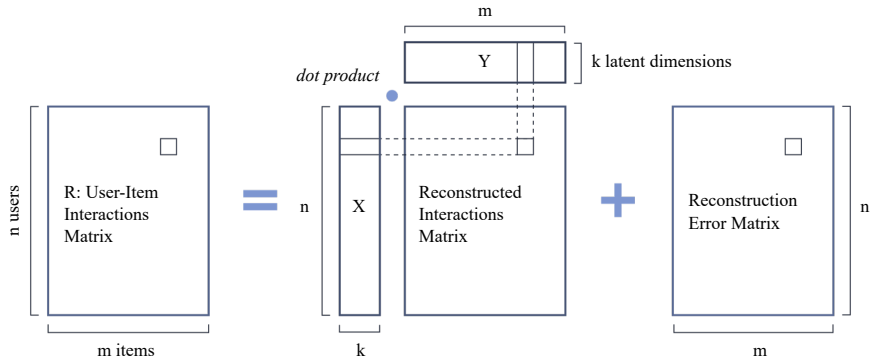

Figure 2.3: The basic idea of matrix factorization

The representatives of MF approaches include Singular Value Decomposition (SVD) [34], Non-negative MF [36], Max-Margin MF [61], Probabilistic MF [51], Localized MF [77]. MF approaches are also referred to as point-wise prediction approaches and frequently used for prediction with explicit feedback such as numerical ratings.

Even though the latent factor models based on MF are successful in recommendation performance, the "latent factors" do not possess intuitive meanings which make it difficult to understand why an item receiving positive predictions, and hence, is recommended out of other candidates. Some users may not accept the recommendation simply because it gets a higher prediction rating by the RS model. Recently, some of MF approaches have been used for providing an explainable recommendation which will be introduced in the next section.

2.2.3 Knowledge-Based Methods

Knowledge-based methods recommend items based on specific domain knowledge on how particular item features meet users' preferences or are useful to the users. Case-based approach (e.g. [9]) and Constraint-based approach (e.g. [22, 72]) are the two types of knowledge-based methods and the difference between the two approaches lies in how the recommendation result

is calculated. Case-based approach determines the recommendations based on similarity metrics while constraint-based approach exploits predefined knowledge which contains explicit rules on the relationships between user requirements and item features.

2.2.4 Other Types of Recommendation Methods

Group Recommendation

Group recommendation has attracted significant research efforts since it benefits a group of users. PolyLens [47] is the first RS to recommend movies to groups of users. Aggregation is a main strategy used in most researches in group RS with a small variation to suit each purpose [41]. Recommending to groups is clearly more complicated than recommending to individuals since each user may have different preferences.

Context-Aware Recommender System

Context-aware recommender system (CARS) is RS which incorporates contextual information. The importance of using context data in the RS can be found in [2]. The challenges in CARS research include how to actually capture and exploit context in producing proper recommendations.

2.2.5 Problems in Recommender System

In developing recommender systems, a few problems arise either from the algorithms used or from the input datasets. Most algorithms used in RS work in a mysterious way causing a black box problem. The accuracy-oriented algorithms may lead to the lack of diversity problem and long tail problem. The data sparsity problem and cold-start problem of the available data are commonly found in RS. In this section, black box, long tail and cold-start problems are briefly discussed.

Current recommender systems are black boxes

An important problem that has recently caught RS researchers' attention is referred to as the "black box" problem. Under many current recommender algorithms, it is rather difficult to understand why they give particular recommendations to a set of data because their algorithms

seem to be locked up inside a black box [27] (Fig. 2.4). This lack of transparency may cause, for example, an ineffective system development in that it is difficult to filter out the problems, failure or inability to gain the customers' attention to try or to purchase a particular suggested object which then results in no sale.

Transparency allows the system developers and others who interact with RS to understand the system's decision rationale. Understanding the reason behind the recommendation will help in identifying possibly ways to handle some critical issues experienced with RS more effectively. For example, an explanation in the form of a general recommendation rule can be used to manage the cold-start and sparsity problems. An explanation showing how the recommended items link to users' preference items can definitely capture their attention and influence them to buy or to give it a try resulting in the purchase of the item.

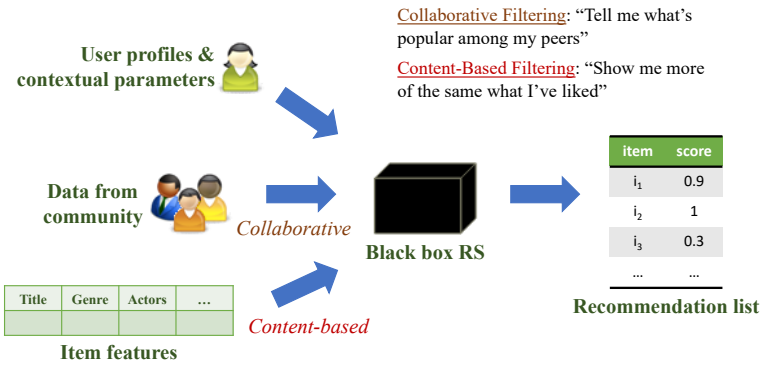

Figure 2.4: Recommender system – traditional approaches

User preference sparsity and the cold start

Recommendation problem is the problem of predicting ratings for the items that have not been seen by the user. As such, most common RS approaches suffer from the user preference sparsity problem in that the number of ratings already obtained is very small compared to the number of ratings needed to be predicted. For example, only a few ratings out of billions of items are available from each user in Amazon. RS has to deal with a very high dimensional data which is usually sparse. In a high dimensional data, finding objects with similar properties seem to be difficult since there is no intuitive notion of density or distance between points as in low dimensional spaces. This is also known as the Curse of Dimensionality [3].

To deal with the sparsity problem, incorporating implicit feedback data is the first approach

to learn the model of user's preferences. Implicit feedback indirectly reflects user's interests and contains only positive examples. This is also known as the one-class collaborative filtering [48, 60].

The second option to overcome the problem is using the most commonly used method i.e. matrix factorization (MF) - which was described in previous section. MF method converts a high dimensional space into a new lower dimensional space to compute and generate recommendations.

The cold start problem is the most common issue faced with RS. It happens when the recommendation has to be made for new users in the system or for new items which have not yet received any rating from the user. Providing the accurate and efficient recommendations in the cold start case is a challenge in CF [37]. The cold start problem can occur in three cases: (i) recommendation on an existing item for a new particular user, (ii) recommendation for an existing user on a new particular item, (iii) recommendation on a new particular item for a new particular user.

To handle a new user case in RS, the system will ask a new user a series of questions and/or ask him/her to provide his/her ratings for different items. This information is used to generate his/her initial profile with preferences stated explicitly. The user profile will be updated when the user rates or purchases more items. However, in this approach, the recommended items are similar to the items that were previously consumed by the user. This can result in a lack of diversity and therefore low satisfaction with recommendations.

In this research, the user preference sparsity and cold-start problem can simply be handled by generating the preferences for new users using the learned recommendation rules from the proposed framework. The recommendations generated are also diverse, interesting, and not boring to the users since they come from the unexpectable rules on multiple domains.

The Long Tail problem

A common challenge in recommender systems is the Long Tail problem. This is an item distribution that affects unpopular or new items. Most of the existing recommender systems cannot recommend tail items because they assume that users want to see the content which is similar

to what they already rated highly. The recommendation outputs, therefore, cannot go beyond the clusters of similarities. Fig. 2.5 [30] shows that due to the similarity bubble problems, the recommendations are made around a very small part of all artists, say 3% only while the rest of the 97% has never been at all taken into consideration; hence, these artists are forever hidden from the (particular) users [13].

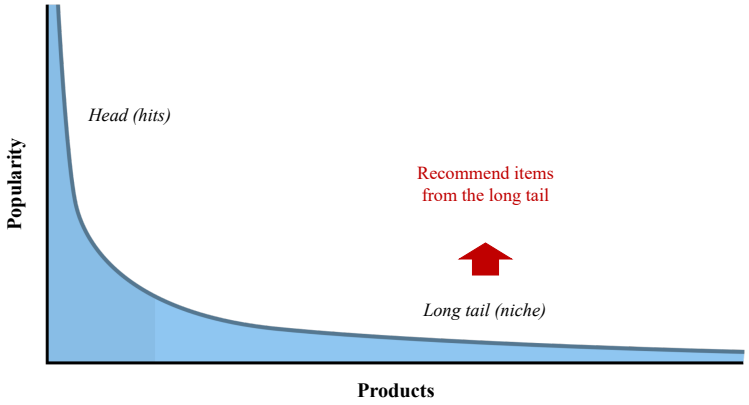

Figure 2.5: Recommendations locked into clusters of similarity

Since the recommendations are locked into clusters of similarity, they are predictable and users find them (such recommendations) uninteresting, non-surprising and boring. Fig. 2.6 are the recommendations for the user who likes "The Beatles" - nothing new, not a surprise and probably boring (the screenshot retrieved January 31, 2018, from Amazon[1]).

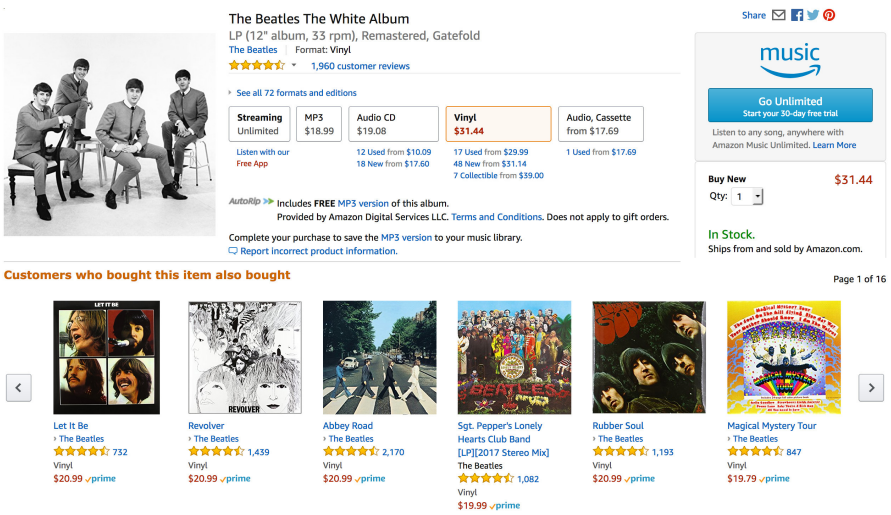

Figure 2.6: Recommendations for user who likes "The Beatles"

[1] https://www.amazon.com/White-Album-Beatles-2009-09-09/dp/B01G99X48K/

2.3 Explanations in Recommender System

Although producing good recommendations is the primary goal of the recommender systems, it is also desirable that such systems provide an explanation accompanying the recommendation. Explainable recommendation was formally introduced in recent years by Zhang et al [75]. However, the basic idea of explainable recommendation was mentioned in some of the earliest work in recommendation research (e.g., [53] noted that recommendations could be explained by other items that the user is familiar with such as *"this product you are looking at is similar to these other products you liked before"*).

Explanations can serve multiple purposes (Table 2.1) [66], one of which is "Transparency". An explanation describing how a recommendation was chosen makes the system transparent to the user as it provides clarity as to how a recommendation was picked for a user when the system shows ambiguous recommendations. *"Customers Who Bought This Item Also Bought..."* on Amazon is a case in point for a transparent recommendation. Explanations can also serve other purposes such as "Effectiveness" by helping users make good decisions and "Efficiency" by helping users make decisions faster. The rest of explanatory goals described in [66] are *scrutability*, *trust*, *persuasiveness*, and *satisfaction*. Some well-known examples of explanations in commercial and academic systems are described in Table 2.2 [65, 46].

Table 2.1: Explanatory goals and their definitions

Purpose	Definition
Transparency	Explain how the system works
Scrutability	Allow users to tell the system it is wrong
Trust	Increase users' confidence in the system
Effectiveness	Help users make good decisions
Persuasiveness	Convince users to try or buy
Efficiency	Help users make decisions faster
Satisfaction	Increase the ease of use or enjoyment

2.3.1 Explanation Approaches and Algorithms

There are two orthogonal dimensions to classify existing explainable recommendation researches [74]: (i) the information source (i.e. relavant user/item, user/item features) or the representation form (i.e. textual sentence, visual image, social explanation, word cluster) of the explana-

Table 2.2: Examples of explanations in commercial and academic systems

System	Item Domain	Explanation
Amazon	e.g., Books, Movies	Content-based
Findory	News	Preference-based
LibraryThing	Books	Collaborative-based
LoveFilm	Movies	Content-based
OkCupid	People to date	Preference-based
Pandora	Music	Preference-based
StumbleUpon	Web pages	Preference-based
Qwikshop [43]	Digital cameras	Preference-based
LIBRA [7]	Books	Content-based, Collaborative-based
News Dude [8]	News	Preference-based
MovieLens [27, 16]	Movies	Collaborative-based
Top Case [44]	Holiday	Preference-based
ACORN [71]	Movies	Preference-based

tions which represents the human-computer interaction perspective of explainable recommendation researches, and (ii) the model or the algorithm (e.g., matrix factorization, graph-based, deep learning, association analysis) to generate such explanations which represents the machine learning perspective of explainable recommendation researches. The examples of explainable recommendation research which are classified into different categories are:

1. Explicit factor model for explainable recommendation [75] developed a MF method which provides an explanation sentence for the recommended items. This explainable recommendation method can be classified as *"matrix factorization with textual explanation"*.

2. Phrase-level sentiment analysis named "Sentires" [76] extracted product aspects and user sentiments from large-scale textual reviews (i.e. in the representation form of "aspect–opinion–sentiment" triplets). For example, given the user reviews about mobile phones, the system can extract triplets such as "noise–high–negative" and "screen–clear–positive". This explainable recommendation method can be classified as *"matrix factorization with word cluster explanation"*.

3. Interpretable convolutional neural network [54] developed a deep convolutional neural network model and displayed item features to users as explanations. This explainable recommendation method can be classified as *"deep learning with user/item feature explanation"*.

4. Visually explainable recommendation [15], developed a deep model to generate image regional-of-interest explanations This explainable recommendation method can be classified as *"deep learning with visual explanation"*.

Although a number of explainable recommendation systems were proposed in the past [74], there has been no work which produces explanations for multiple domains. Moreover, most explanations are presented in forms (Fig. 2.7 modified from [74]) which could possibly lead to a misunderstanding due to their complexity and lack of clarity. It is sometimes difficult to use language in a precise and unambiguous way without making the explanation wordy and difficult to read. Many of the matrix factorization, graph-based, deep learning, and association analysis approaches to explainable recommendation aim to gain an understanding of how the algorithm works or to know the user/item features which are used as an input to the model. In this thesis, relational learning is used to construct rules for recommendation generation along with the explanations which are not only clear but also simple, understandable and unambiguous.

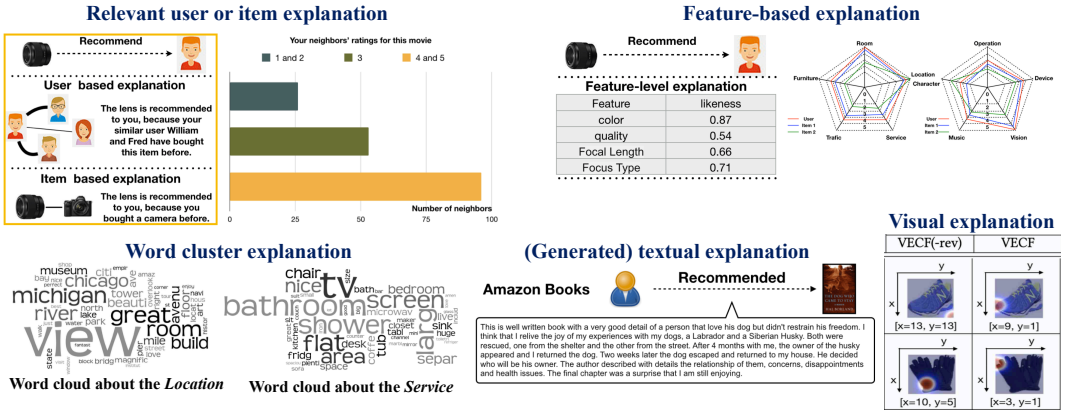

Figure 2.7: Existing explanation forms

2.4 Relational Learning in Recommendations

The success of relational learning has been shown in a number of researches. For example, the recent work of Hoxha and Rettinger [28] and Kouki et al. [35] which applied relational learning to the hybrid recommendations demonstrated that incorporating additional information for users and items was beneficial to the cold-start settings in particular [23]. Another progress in relational learning was presented in Hoxha et al. [28], a probabilistic graphical modeling repre-

sentation using Markov Logic Networks to combine content-based with collaborative filtering. Its recommendation task began with predicting the existence probability of a relation between a particular user and a particular item, and then returning the recommendation for the query predicate by choosing the items with the high probability according to the specified threshold.

In relational learning, first-order logic rule representation is simpler and more concise compared to the rule representation in traditional item-based collaborative filtering algorithms as shown in the following example. The representation of the rule: "Similar items get similar ratings from a user" is "*IF* items i_1 and i_2 are similar, for a given similarity measure, *AND* user u rates item i_1 highly, *THEN* user u will rate item i_2 highly", in traditional item-based collaborative filtering algorithms and (2.4.1) is its representation in first-order logic.

$$similarItems_{sim}(i_1, i_2) \wedge highRating(u, i_1) \implies highRating(u, i_2) \tag{2.4.1}$$

In first-order logic rules, constants represent the objects in a domain of interest. Variable symbols range over the objects. Predicate symbols represent the relationship between objects or features of objects. Variables and constants might be typed, in which case variables only range over objects of the given type.

In 2015, the study by Kouki, et al. [35] showed that relational learning framework could be used to develop a general and extensible hybrid RS framework called Hybrid Probabilistic Extensible Recommender. This framework also provided a mechanism to extend the system by incorporating and reasoning over unspecified types and similarity measures of additional information collected from several sources. A learning method used to appropriately balance the different input signals from many information sources was also discussed.

In 2016, probabilistic program showed its use in single-domain based recommendation by Catherine and Cohen [11]. They proposed three methods that used knowledge graphs for making personalized recommendations. Given that a user liked specific movies and entities in the past, ranked new movies (e.g., The Bridge of Spies) for that user using ProPPR (Programming with Personalized Pagerank) [67]. The methods gave large improvements compared to the state-of-the-art method that used knowledge graphs. The study of the behavior of the methods as rating matrix density was also discussed.

It is evident from the researches by Hoxha et al. [28], Kouki et al. [35] and Catherine and Cohen [11] above that a better recommendation performance is obtained by relational learning with additional information incorporated. As a result, relational learning is used for the proposed framework in this thesis to model and provide a solution to construct rules for recommendations.

2.5 Cross-Domain Recommender System

Instead of performing a recommendation on a single domain and focusing on a certain market because users normally provide feedback for items of different types and express their opinions on different systems, a cross-domain recommendation problem is introduced. With cross domain, prediction accuracy is improved because data sparsity is reduced. Cross domain also offers additional values to the recommendations as it gives diverse, novel and serendipitous predictions [10].

In cross-domain recommendation tasks, items in the target domain are recommended to users in the source domain. Aggregating knowledge and transferring knowledge are the two types of cross domain approaches and the difference between the two approaches lies in how the knowledge from the source domain is exploited.

In the aggregating knowledge approach, user preferences or aggregate models from different domains may be merged, e.g., ratings for items in a book and a movie become a joint matrix and then common single domain approaches can be used to recommend particular items to users in the target domain.

The transferring knowledge approach first links the domains, e.g., linked through attributes by transferring function from a book domain to a movie domain, and then the knowledge among domains can be transferred for recommendation. However, as previously mentioned, these two approaches have their own limitations. Aggregating knowledge approach is designed for particular cross-domain scenarios which are quite difficult to generalize while transferring knowledge approach is computationally costly.

Cross-domain caught my attention due to its proven success in a lot of work. The common attributes, semantics and other hidden knowledge can be exploited and leveraged across distinct domains to generate novel or unexpected recommendations. I also succeeded in constructing

the cross-domain recommendation rule using relational learning [57]. In this study, whether the user's preference in one domain, e.g., music, relates to other facets of users' life and preferences in other domains is also investigated.

2.6 Serendipity Recommendations

Serendipity is currently a hot topic in recommendation systems. A number of different approaches to serendipitous recommender systems have been proposed including [55], [29], [31]. However, there are two problems commonly experienced when introducing serendipity into recommender systems. One problem is that the quality of predictions becomes progressively worse, so there is a need to find the right balance between similarity and novelty, and between the immediate surrounding and the periphery [1]. The other problem is that the proposed algorithms mostly require users' interaction to generate unexpected recommendation lists, e.g., [62] propose a music recommendation using MusicGalaxy, an adaptive user interface application for exploring music collections.

2.6.1 Bisociations - Unexpected Interesting Links

Bisociations [6] focus on the discovery of surprising relations in the repositories coming from diverse origins or heterogeneous data sources, forming a different domain [45].

The idea of bisociations was first introduced by Arthur Koestler in 1964. It is defined as an association that can directly connect objects from different domains or connect those of the same domain via an object of another domain (or other domains) (Fig. 2.8) [56]. The promising approach for bisociations data exploration is the graph structure [6].

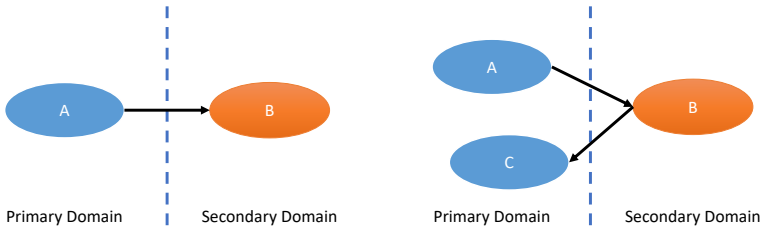

Figure 2.8: Bisociative knowledge discovery concept

As mentioned above, Stober, et al. [62] proposed a serendipitous music recommendation using MusicGalaxy application. They turned their music discovery application into an envi-

ronment that supported bisociative music discovery. The idea was to combine two distinct domain views into one visualization using the secondary focus to highlight connections to the nearest neighbors in a different domain than the one used for projection. The “primary domain" was directly visualized by the projection and contained the displayed tracks connected by neighborhood relations that were implicitly induced between each track and its neighbors in the projection. The “secondary domain", however, was used to identify nearest neighbors for the secondary focus distortion and not directly visible to the user. A bisociation occurred in this setting if two tracks were not neighbors in the projection domain but were connected in the secondary domain.

Nonetheless, Stober, et al. [62] focused only on the field of bisociative music collection exploration using a user interface which integrated the music graph information for each user's interaction and led to the highlighted secondary domain focusing for serendipitous music discovery.

In this research, after thoroughly exploring the dataset, I discovered that properties of the items or those of the users can lead to some relationships which can, subsequently, produce the unexpected recommendation results. My research of bisociations and serendipity recommendations was proposed and published in [56].

2.7 Summary

There are advantages and disadvantages in the explainable RS and the cross domain-based RS. One drawback of most explainable RS is the explanation form. Some examples of explanation forms like textual sentence, tag cloud, visual image could be complicated and require an extra afford to comprehend. In cross domain-based RS, however, the recommendation made is limited to the pre-specified domain. In addition, the learning task is computationally inefficient and quite a challenge to generalize.

The proposed framework in this thesis is a new paradigm which takes an advantage of Inductive Logic Programming (ILP) to provide recommendation rules in first-order logic format to give a clear and concise explanation, unlike most explainable RS. The approach also allows the recommendation to be made in multiple domains, unlike the cross domain-based RS. With

the ability of ILP, the proposed framework is extensible and the re-training is not necessary. The other framework features which differ from the typical explainable and cross domain approaches are discussed in the next chapter.

Chapter 3

Research Methodology

3.1 Preliminaries

3.1.1 Probabilistic Logic Programming and ProbLog

Probabilistic logic programming (PLP), a kind of relational learning in which some of the facts are annotated with probabilities, is deployed since it is suitable for representing and solving recommendation problem. Each rule is represented in relational logic associating with probability. The rule has a condition part (the right-hand side of the rule) and a conclusion part (the left-hand side of the rule). For example, the expected recommendation rule (3.1.1) can be read as the following: If user likes movie genre *western*, then user will also like music genre *country* with a probability of 0.86735. This rule can be represented as Prolog[1] format as (3.1.2).

$$likedMovie(User, Movie) \wedge musicGenre(Music, country) \wedge movieGenre(Movie, western) \implies likedMusic(User, Music)\ with\ a\ probability\ of\ 0.86735 \tag{3.1.1}$$

$$0.86735 :: likedMusic(User, Music) \text{ :- } likedMovie(User, Movie), musicGenre(Music, country),\ movieGenre(Movie, western). \tag{3.1.2}$$

ProbLog [19], a probabilistic Prolog tool, is chosen for my implementation as it helps to build programs that encode complex interactions between large sets of heterogeneous component and inherent uncertainties that are presented in real-life situations. With ProbLog, it

[1] Prolog is a logic programming language.

is also easier to describe the relations between the items in dataset than other representation method such as matrix. ProbLog binaries, source, and documents are available for download at `https://dtai.cs.kuleuven.be/problog/`.

3.1.2 Traditional Recommendation Approaches on Multiple Domains Using Probabilistic Logic Programming

The initial experiments were designed based on traditional CF recommendation approaches (i.e. user-based CF [50] and item-based CF [52]) using PLP to perform deductive reasoning. They were mainly conducted over two datasets, user's music and movie preferences and item attributes. Pearson correlation coefficient was used to find similarities between users, then deduce the certain recommendation rules. For the user-based recommendation, the similar taste on movies would lead to a similar taste on music. For the item-based recommendation, the high confidence of the defined rule would result in a music recommendation. Then, the ProbLog's syntax was used to describe the model. The approach was performed in three steps: define rule, query and select output. Only outputs greater than or equal to the desired threshold would be selected for the recommendations. Some examples are shown in Fig. 3.1.

```
Example: User-based recommendation
%"Music" will be recommended to user2 if user1 likes
%that "Music" and his movie preference influences user2
Defined rule: 0.80000::likedMusic(U2,Music)
:- influencesUser(U1,U2), likedMusic(U1,Music).
Query: query(likedMusic(U2,Music)).

Example: Item-based recommendation
%"Music" will be recommended to user2 if user1 likes
%that "Music", and both of them like the same "Movie"
%and "Movie" influences "Music"
Defined rule: 0.80000::likedMusic(U2,Music)
:- influencesItem(Movie,Music), likedMovie(U2,Movie),
likedMovie(U1,Movie), likedMusic(U1,Music).
Query: query(likedMusic(U2,Music)).
```

Figure 3.1: Traditional recommendation approaches using PLP

User-based and item-based explanations are usually provided based on users' implicit or explicit feedback. In user-based recommendation, the explanation is that the user is similar

to a group of "neighborhood" users, and the recommended item is rated positively by these neighborhood users. In item-based recommendation, the recommended item is similar to some other items the user liked in the past.

3.2 Overview

The main purpose of this research is to develop a new formal framework for RS based on two aspects: include an explanation which explains why the system gives a particular recommendation to a user and provide a novel or an unexpected recommendation to a user. To achieve the goal, the following two questions need to be addressed: (i) what to explain and how? (ii) how to generate a novel and/or an unexpected recommendation?

To answer the first question, what to explain is to provide information which users find helpful and beneficial and help them in their decision making i.e. to find items that they like. Hence, the focus is on the explanation which states clearly why such recommendation is chosen.

To deliver the explanation which is not only clear but also simple, understandable and unambiguous, relational learning is used to construct rules for recommendation generation. As a result, the explanation is represented in the form of logic, the formal language.

Relational learning has received extensive attention in recent years since most of the data available is organized by the relations between entities. Relational learning refers to learning in a context where there may be relationships between learning examples, or where these examples may have a complex internal structure (i.e. consisting of multiple components and there may be relationships among them) [63]. One of the main tasks of relational learning is to make predictions of possible new relations. Relational learning algorithms learn the definition of a new relation in terms of existing relations in the database [49] since items in the world are connected by various relations. In this thesis, Inductive Logic Programming (ILP) in a probabilistic setting is used to perform relational learning from examples.

For the answer to the second question, what is new or what is unexpected to a user, is the item which the user is not familiar with. In other words, in this research, new or unexpected items refer to the suggested items that differ from those recommended based on similarities either between users or items. To generate the new or unexpected items, there is a need to go

beyond the traditional approach since it deals mainly with single dyadic relationships between users and items on a single domain. Adding other domains to the system will broaden the search space and provide more opportunities to discover items which are previously unseen or surprised to a user. To deliver a new or an unexpected item and which would still be useful to a user, it is necessary to relate user's information, i.e. preference, demographic, user's background with some information/knowledge provided in various domains either from other users or items.

Based on the notions discussed above, the proposed framework for a recommendation system is therefore developed using relational learning on multiple domains. The framework is created to have, even not exhaustive, the following features.

1. The framework constructs a general recommendation rule, not an item, unlike a typical RS. The benefits of the rules in general form are that: the rules can be transformed into a clear and understandable explanation to accompany the recommended items, and the rules can possibly be used to manage the cold-start and sparsity problems. More importantly, an explanation is in *if-then* form, which is simpler and more concise compared to most of the explanation forms (e.g. textual sentence, tag cloud, visual image) currently available.

2. The recommendation can be made in any domain led by the learned rule, unlike a typical cross-domain RS. For a single-domain based RS, a recommendation can be made only for that domain. For cross-domain based RS, a recommendation can be made for the pre-specified target domain only. The following are examples of rules constructed using the proposed framework on three different domains; music, movie and book. Each rule can be used to recommend the items in each different domain. Rule (3.2.1) can be used to recommend "Music" composed by *"Ed Sheeran"* to the user if the user likes the book *"Harry Potter"*. Rule (3.2.2) is used to recommend the "Music" according to the conditions on both "Movie" and "Book" domains. Rules (3.2.3) and (3.2.4) can be used to recommend the "Movie" and the "Book", respectively.

$$0.37552 :: likeMusic(User, Music) :- \\ likeBook(User, harrypotter),\ musicArtist(edsheeran, Music). \tag{3.2.1}$$

$$0.13497 :: likeMusic(User, backinblack) \text{ :- } likeMovie(User, Movie), \\ movieProductionStudio(Movie, marvelstudios), \\ likeBook(User, Book),\ bookAuthor(Book, stanlee). \tag{3.2.2}$$

$$0.19461 :: likeMovie(User, Movie) \text{ :- } likeBook(User, Book), \\ bookGenre(Book, science),\ movieGenre(Movie, biography). \tag{3.2.3}$$

$$0.21311 :: likeBook(User, Book) \text{ :- } likeMovie(User, Movie), \\ bookGenre(Book, computer),\ movieGenre(Movie, sci\text{-}fi). \tag{3.2.4}$$

3. The role of source or target domain needs not be pre-defined due to the nature of relational learning, unlike a typical cross-domain RS which the source and target domains must be pre-defined. However, the framework proposed allows a user to specify the particular target and source domains if he/she so desires in the setting at the beginning of the process.

4. A new domain can be added into the framework without requiring a re-training, unlike a typical cross-domain RS which re-training is required for every domain pair. Since number of domains used in the framework are not limited to one or two, any new domain can be added by simply specifying the new domain's predicates and changes in the setting.

5. In this framework, all domains can be processed simultaneously, unlike the typical cross-domain RS which only two domains can be processed each time.

6. The framework can derive negative recommendation rules automatically since ProbFOIL is used in learning the rule, unlike the typical RS. An introduction to ProbFOIL is briefly summarized in the next section. The negative rules can ensure that the item will not be included in the list and can generate negative examples for cold-start problem.

3.3 ProbFOIL and Relational Learning

In this research, the probabilistic first order rule learner (ProbFOIL) algorithm is adopted to learn the recommendation rule. ProbFoil was first introduced in 2010 by De Raedt and Thon [20] and was upgraded in 2015 [18]. It is a probabilistic extension of the traditional rule-learning system FOIL (First-Order Inductive Learner) that is capable of learning probabilistic rules from probabilistic data.

ProbFOIL supports the probabilistic data by generalizing the concept of true/false, positive/negative to a probabilistic context [4]. Moreover, it performs an additional step of parameter learning to learn rules that express probabilistic relationships. The learning problem considered is defined as follows [18]:

Given:

1. *A set of training examples E, consisting of pairs* (x_i, p_i)*, where* x_i *is a ground fact for the unknown target predicate t and* p_i *is a target probability;*

2. *A background theory B containing information about the examples in the form of a ProbLog [19] program;*

3. *A loss function* $loss(H,B,E)$*, measuring the loss of a hypothesis (set of clauses) H w.r.t. B and E;*

4. *A space of possible clauses* $\mathcal{L}_h$ *specified using a declarative bias;*

Find: *A hypothesis* $H \subseteq \mathcal{L}_h$ *such that* $H = \arg\min_H loss(H,B,E)$ *and the loss function aims at minimizing the standard error of the predictions, that is,*

$$loss(H,B,E) = \sum_{(x_i,p_i)\in E} |P_s(B \cup H \models x_i) - p_i| \tag{3.3.1}$$

This loss function is also used in Kearns and Schapire's probabilistic concept-learning framework [32] and and $P_s(B \cup H \models x_i)$ is the *success probability* of a query x_i.

Algorithm 3.3.0.1 [18] presents ProbFOIL algorithm of which the probabilistic rules are in the form $x :: target \leftarrow body$. ProbFOIL is initialized with the target predicate in the head and it eagerly adds body literals and clauses until no more improvement is observed with respect to a

global scoring function. The clause to be added is obtained by the function LEARNRULE, which greedily searches for the clause with the highest *local scoring function*, using the refinement operator ρ.

Algorithm 3.3.0.1 The ProbFOIL learning algorithm

```
1:  function PROBFOIL(target)
2:      H := ∅
3:      while true do
4:          clause := LEARNRULE(H, target)
5:          if GSCORE(H) < GSCORE(H ∪ {clause}) then
6:              H := H ∪ {clause}
7:          else
8:              return H
9:          end if
10:     end while
11: end function
12: function LEARNRULE(H, target)
13:     candidates := {x :: target ← true}
14:     best := (x :: target ← true)
15:     while candidates ≠ ∅ do
16:         next_cand := ∅
17:         for all x :: target ← body ∈ candidates do
18:             for all refinement ∈ ρ(target ← body) do
19:                 if not REJECT(H, best, x :: target ← body) then
20:                     next_cand := next_cand ∪ {x :: target ← body ∧ refinement}
21:                 end if
22:                 if LSCORE(H, x :: target ← body ∧ refinement) >
                      LSCORE(H, best) then
23:                     best := (x :: target ← body ∧ refinement)
24:                 end if
25:             end for
26:         end for
27:         candidates := next_cand
28:     end while
29:     return best
30: end function
```

The probability x will be determined by the rule learning algorithm. Each call to LSCORE returns the best score that can be achieved for any value of x. The best rule found will be returned with x which is a probability that yields the highest local score. (3.3.2) is a sample recommendation rule learned by ProbFOIL:

$$\begin{aligned} 0.86735 :: likeMusic(A,B) \text{ :- } & musicGenre(B,country), \\ & likeMovie(A,Var\_1), movieGenre(Var\_1,western). \end{aligned} \tag{3.3.2}$$

This rule states that person *A* will like music *B* with a probability of 0.86735 if he likes the western movie and *B* is a country music. In other words, the country music *B* will be recommended to the user with a probability of 0.86735 because of his preference on a western movie. With this rule, it is clear why music *B* is recommended to the user.

The implementation of ProbFOIL algorithm for learning probabilistic clauses from probabilistic data is available at Prob2FOIL repository[2]. Currently, this algorithm will only work on Linux x86_64 and macOS with prerequisite packages as follows:

- Python 2 (https://www.python.org/)
- ProbLog2 (https://dtai.cs.kuleuven.be/problog/)
- YAP Prolog (with tabling) [17]
- GCC 4.9 (https://gcc.gnu.org/gcc-4.9/)
- RAPTOR (https://raptor.martincarlisle.com/)
- GNU Readline Library (http://www.gnu.org/software/readline/)
- The dependencies of above packages

The input of ProbFOIL consists of two parts: settings and data. These are both specified in Prolog (or ProbLog) files, and they can be combined into one. The data consists of (probabilistic) facts and settings which is defined as follows:

- **Target:** the target should be specified by adding a fact 'learn(predicate/arity)'.
- **Modes:** the modes should be specified by adding facts of the form 'mode(predicate(mode1, mode2, ...)', where modeX is the mode specifier for argument X. Possible mode specifiers are:
  (i) +: the variable at this position must already exist when the literal is added.
  (ii) -: the variable at this position does not exist yet in the rule.

[2]https://bitbucket.org/problog/prob2foil/

(iii) c: a constant should be introduced here; possible values are derived automatically from the data.

- **Types:** For each relevant predicate (target and modes) there should be a type specifier. This specifier is of the form 'base(predicate(type1, type2, ...)', where typeX is a type identifier. Type can be identified by arbitrary Prolog atoms.

- Other settings related to the data.

To run the algorithm, run 'probfoil data.pl' where data.pl is the data file. Multiple files can be specified and the information in them is concatenated. Several command line arguments are available. Use '--help' to get more information.

3.4 Data Preparation

In this section, the preparation of the dataset to be used with ProbFOIL is explained. The dataset for RS typically consists of users' information and preferences in the form of attributes and ratings supplied by users. Fig. 3.2 is a portion of an example of Amazon product data. However, this data cannot be directly submitted to ProbFOIL since the input data of ProbFOIL is represented in first order logic format and contains the probabilistic facts. Therefore, before the learning can be performed, the dataset must be converted. The attribute will be turned into predicate and the rating provided in the dataset will be considered as a probability. In the movie domain for example, the predicate likeMovie/2 refers to the movie that the user likes. The constants i.e. drama, action, kids and the predicate movieGenre/2 refer to the genre of movie. The rating 0 - 5 is associated to the probability 0 - 1 to be used in probabilistic logic representation. The example of the Amazon data in Fig. 3.2 could be represented with a probabilistic logic formula for probFOIL as shown in Fig. 3.3.

3.5 Recommendation Relational Rule Learning

The framework for RS proposed is designed with all features previously described in this chapter. The overview of the framework is depicted in Fig. 3.4. ProbFOIL learns the recommendation rules from data and settings. Data is input as facts in Prolog format. The datasets which

{“reviewerID”: “ADZPIG9QOCDG5”, “asin”: “0005019281”, “overall”: 4.0}
{“reviewerID”: “A35947ZP82G7JH”, “asin”: “0005019281”, “overall”: 3.0}
{“reviewerID”: “A3UORV8A9D5L2E”, “asin”: “0005019281”, “overall”: 3.0}
{“reviewerID”: “A1VKW06X1O2X7V”, “asin”: “0005019281”, “overall”: 5.0}
{“reviewerID”: “A3R27T4HADWFFJ”, “asin”: “0005019281”, “overall”: 4.0}
{“reviewerID”: “A2L0G56BNOTX6S”, “asin”: “0005019281”, “overall”: 5.0}
{“reviewerID”: “A5NYUBEKXFLX5”, “asin”: “0005019281”, “overall”: 5.0}
{“reviewerID”: “A2DJ8B8GE4V2VD”, “asin”: “0005019281”, “overall”: 5.0}
{“reviewerID”: “AWF2S3UNW9UA0”, “asin”: “0005019281”, “overall”: 5.0}
{“reviewerID”: “A3O4UUT83DG3OU”, “asin”: “0005019281”, “overall”: 5.0}

{
“asin”: “0005019281”,
“title”: “An American Christmas Carol”,
“genre”: “Kids”,
“director”: “Eric Till”,
“starring”: [“David Wayne”, “Chris Wiggins”, “Henry Winkler”],
“audioLanguage”: “English”,
“categories”: [[“Movies & TV”, “Movies”]]
}

Figure 3.2: An example of one reviewer in Amazon product data format

```
0.8::likeMovie(adzpig9qocdg5, 0005019281).
0.6::likeMovie(a35947zp82g7jh, 0005019281).
0.6::likeMovie(a3uorv8a9d5l2e, 0005019281).
likeMovie(a1vkw06x1o2x7v, 0005019281).
0.8::likeMovie(a3r27t4hadwffj, 0005019281).
likeMovie(a2l0g56bnotx6s, 0005019281).
likeMovie(a5nyubekxflx5, 0005019281).
likeMovie(a2dj8b8ge4v2vd, 0005019281).
likeMovie(awf2s3unw9ua0, 0005019281).
likeMovie(a3o4uut83dg3ou, 0005019281).

movieGenre(0005019281, kids).
movieDirector(0005019281, erictill).
movieStarring(0005019281, davidwayne).
movieStarring(0005019281, chriswiggins).
movieStarring(0005019281, henrywinkler).
movieAudio(0005019281, english).
```

Figure 3.3: A probabilistic logic representation of data in Fig. 3.2

will be described in the next chapter are preprocessed and converted into Prolog format using Problog library loader[3] provided in ProbFOIL package. The settings define target predicate, modes, type information for the predicates and others which relate to the data. Target predicate is the predicate that is needed to learn while mode refers to the predicates that can be added to the rules in the condition part. The specified predicates are from all interested domains. The example settings are shown in Fig. 3.5. In these settings, a definition/rule for learn(likeMusic/2) is induced. The predicates from movie and music domains; likeMovie, movieGenre and musicGenre and their type information are specified. The rules learned by ProbFOIL are represented in FOIL as shown in Fig. 3.6. The first rule states that if the user likes the western movie, then there is 86.735% chance that the user will like the country music. The second rule indicates that the users who like the same genre of movie will also like the same music and the last rule indicates that the users who like two different genres of movie will also like love music. The output learned rules will be ranked based on their probabilities and be used to compute the recommendation item. The top-N items in the list will then be recommended to a particular user.

[3]https://problog.readthedocs.io/en/latest/modeling_advanced.html

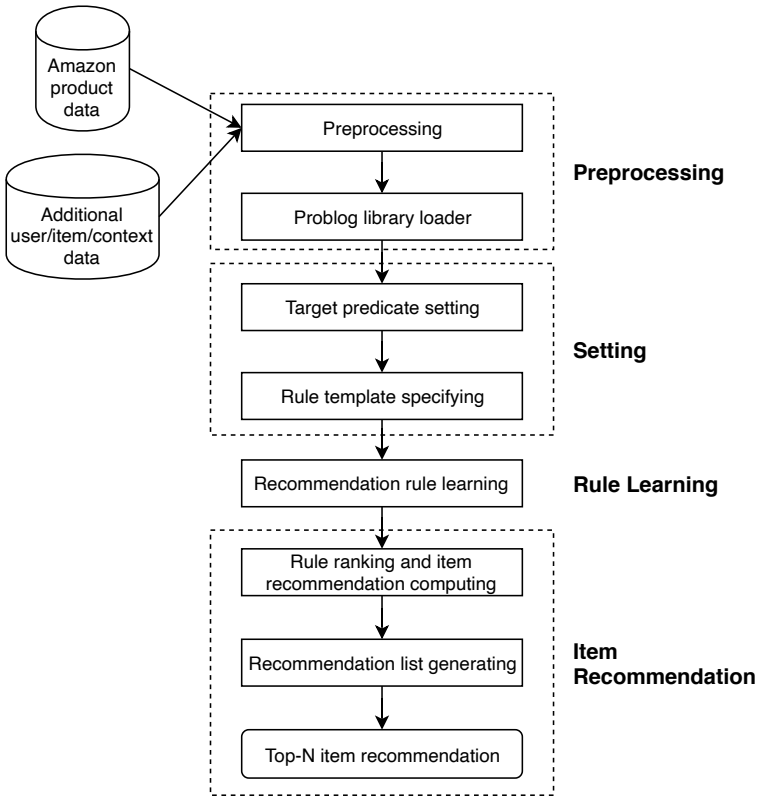

Figure 3.4: The proposed recommendation framework overview

```
%Modes
mode(likeMovie(+, -)).
mode(movieGenre(+, c)).
mode(musicGenre(+, c)).

%Type definitions
base(likeMusic(user, item)).
base(likeMovie(user, item)).
base(musicGenre(item, attribute)).
base(movieGenre(item, attribute)).

%Target
learn(likeMusic/2).
```

Figure 3.5: The example of ProbFOIL settings

```
%People who like western movie, also like country music
0.86735::likeMusic(User, Music) :- likeMovie(User, Movie),
musicGenre(Music, country), movieGenre(Movie, western).

%People who like the same genre of movie (thriller),
%also like the same music
0.69523::likeMusic(U2, Music) :- likeMovie(U2, Movie1),
likeMovie(U1, Movie2), movieGenre(Movie1, thriller),
movieGenre(Movie2, thriller), likeMusic(U1, Music).

%People who like two different genres of movie
%(drama and comedy), also like love music
0.81232::likeMusic(User, Music) :- likeMovie(User, Movie1),
likeMovie(User, Movie2), movieGenre(Movie1, drama),
movieGenre(Movie2, comedy), musicTheme(Music, love).
```

Figure 3.6: The portions of learned rules likeMusic/2

Chapter 4

Experiments and Results

4.1 Overview

The purpose of the experiments is to answer the two research questions discussed in Chapter 3. More specifically, it is to demonstrate that the proposed framework provides more choices of recommendations to users accompanied by understandable explanations. The experiments were conducted on three different domains, i.e. music, movie and book, which were selected as representative examples. Given background knowledge, positive example facts, and negative example facts, ProbFOIL was used to learn the recommendation rules. The performance of the algorithm was compared to the three well-known baseline methods using $HitRatio@N$ [21] and $SRDP@N$ [24] which were the qualities of the top-N recommendation measurement and the usefulness of the unexpectedness measurement.

4.2 Dataset

"Amazon product dataset" provided by UCSD [42, 26] was used to carry out the experiments in this research. The dataset contained product reviews, metadata and links from "Amazon", including 142.8 million reviews spanning 1996 to 2014. Product reviews included ratings, text and helpfulness votes. Product metadata were descriptions, category information, price, brand, image features. Links were "also viewed" and "also bought graphs".

It is not unusual to find lots of missing values in the user-item preference (rating) matrix **R** and the sparsity of **R** is greater than 90% in "Amazon product dataset". However, it is a true task of the recommender systems to predict ratings for all missing values (unrated items) [40]. There

may also be some noise values from the reviewers in that users might rate an item highly but they do not like and write negative textual review for that item e.g. the reviews of "The World Rose" as shown in Fig. 4.1 (the screenshot retrieved May 18, 2016, from Amazon[1]). In addition, it is noted that the dataset from Amazon and other e-commerce websites may contain fake reviews generated by marketing companies to mislead shoppers to boost their business. The fake review issue is considered as the noise value. To overcome this problem, domain knowledge or domain experts play an important role in the data preprocessing stage to analyze and spot noise reviews.

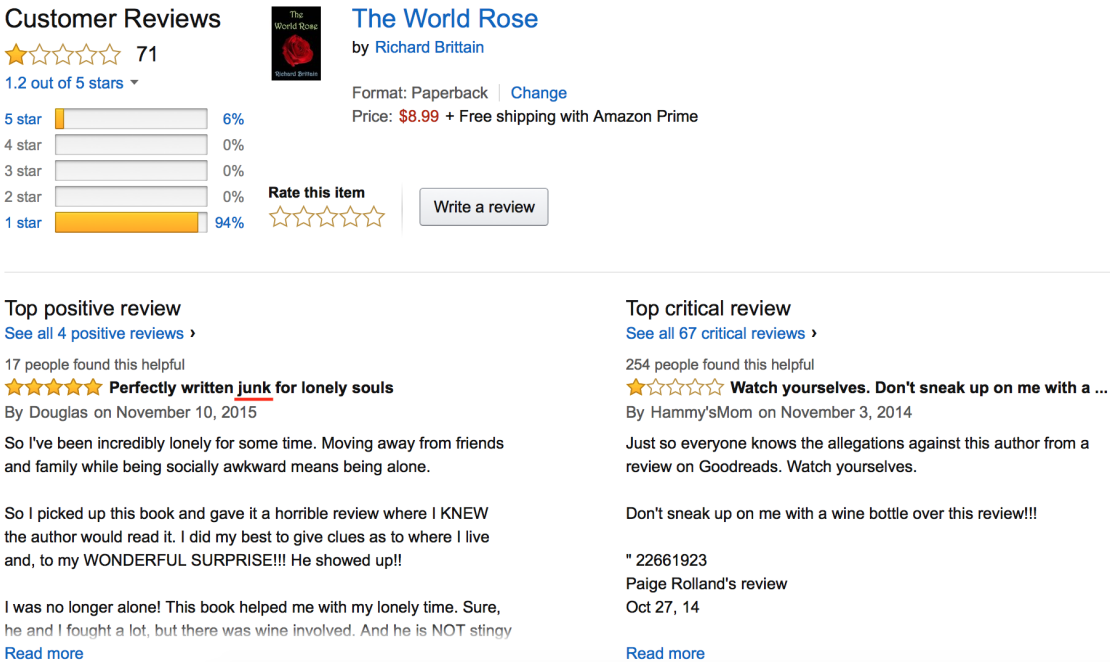

Figure 4.1: The content of top positive review (5-star rating and people found it helpful) for "The World Rose" in Amazon is also negative

Preprocessing of this dataset included the following steps: First, created the user preference dataset by extracting the top 1,000 reviewers from the product review dataset for each domain, music, movie and book. Second, created the item attribute dataset using Amazon's Product Advertising API, Last.fm's API[2], and OMDb API[3] to retrieve the attributes of each item listed in the user preference dataset. Third, created the domain-pair dataset by taking the top 1,000 reviewers from one category (domain) in the user preference dataset. Then, extracted their

[1]https://www.amazon.com/World-Rose-Richard-Brittain/dp/150235974X/
[2]https://www.last.fm/api/
[3]http://www.omdbapi.com/

preference of the other category. For example, the top 1,000 reviewers from music category were taken and their preferences on movie were extracted to form the music-movie dataset.

4.3 Recommendation Rule and Evaluation

To run the experiments on the three selected domains: music, movie and book, the setting in ProbFOIL was properly specified for each domain-pair. Examples of recommendation rules learned by ProbFOIL are presented in Fig. 4.2.

```
%People who like the same genre of movie (thriller),
%also like the same music
0.69523::likeMusic(User2, Music) :- likeMovie(User2, Movie1),
likeMovie(User1, Movie2), movieGenre(Movie1, thriller),
movieGenre(Movie2, thriller), likeMusic(User1, Music).

%People who like movie "Twilight"
%also like music artists whose career started in 1996,
%and they also like these artists' music
0.14434::likeMusic(User, Music) :- likeMovie(User, twilight),
careerStarted(Artist, 1996), musicArtist(Artist, Music).

%People who like book "Harry Potter",
%also like music artist "Ed Sheeran"
0.37552::likeMusic(User, Music) :- likeBook(User, harrypotter),
musicArtist(edsheeran, Music).

%People who like Sci-Fi movie, also like music artist "Aerosmith"
0.45691::likeMusic(User, Music) :- likeMovie(User, Movie),
movieGenre(Movie, sci-fi), musicArtist(aerosmith, Music).
```

Figure 4.2: The unexpected recommendation rules for likesMusic/2

Each rule was represented in relational logic associating with probability. The rule had a condition part (the right-hand side of the rule) and a conclusion part (the left-hand side of the rule). The first rule could be read as the following: If User1 and User2 like the same movie genre thriller, then User2 will like the same music as User1 does with a probability of 0.69523. The second rule could be read as: If user likes the film Twilight then he/she will like the music performed/composed by an artist who started his career in 1996 with a probability of 0.14434. These rules are declarative, that is, easy to understand to human. In addition, this formal lan-

guage is obvious, less redundant and shorter compared to a natural language used in other explanation recommendation systems. The experiment results confirm the answer to the first research question: each rule (in relational learning) states clearly why the items are recommended to a particular user. It can, therefore, be concluded that the generated rules provide explanation facilities for the proposed recommender system.

These rules also demonstrate that user's preference on one domain can be used to predict user's preference on another domain, for example, user's movie taste can be used to predict his/her music taste.

The first and second rules recommend different music to the user based on his/her interest in movie genre. The third and the fourth rules recommend particular music to the user due to his/her preference on different domain objects. Ed Sheeran's music will be recommended to the user whose preference is Harry Potter book while Aerosmith's music will be recommended to the user who likes Sci-Fi movie. These rules return the recommendation items which cannot normally be found in similarity-based recommended system. These experiment results offer the solution to the second research question: the user retrieves the recommendation item which is new and he/she is not familiar with.

Another experiment was also conducted to provide the recommendations in one domain using the information from other different domains: Music, Movie, and Book. Examples of recommendation rules learned by our framework are presented in Fig. 4.3. "Back in Black" song will be recommended to the user whose preferences are Stan Lee's book and movies from Marvel Studios while "Forrest Gump" movie will be recommended to the user who likes "I Really Like You" song and "Catch Me If You Can" book.

In this research, the performance of the proposed framework was evaluated in two aspects: the quality of the top-N recommendations ($HitRatio@N$) and the usefulness of the unexpectedness recommendations ($SRDP@N$).

Deshpande and Karypis [21] suggested that the quality of the top-N recommendations could be measured by the number of hits and their position within the top-N items. For each user u belonging to U, a set of target users, item i was arbitrarily chosen such that it was tagged by user u previously. If i was in the top-N recommendation list, then hit occurred. The HitRatio@N was

```
%People who like movie from "Marvel Studios" and Stan Lee's book,
%also like music "Back in Black"
0.13497::likeMusic(User, backinblack) :- likeMovie(User, Movie),
movieProductionStudio(Movie, marvelstudios),
likeBook(User, Book), bookAuthor(Book, stanlee).

%People who like music "I Really Like You"
%and book "Catch Me If You Can", also like movie "Forrest Gump"
0.21621::likeMovie(User, forrestgump) :- likeMusic(User, ireallylikeyou),
likeBook(User, catchmeifyoucan).
```

Figure 4.3: Examples of music recommendation rules

defined as a total number of hits divided by a total number of users as follows:

$$HitRatio@N = \frac{Number\ of\ Hits}{|U|} \tag{4.3.1}$$

Ge and his colleagues [24] suggested that the serendipitous recommendation could be accurately and precisely measured by considering the two essential aspects of serendipity: unexpectedness and usefulness. The serendipity (*SRDP*) was then defined as follows:

$$SRDP(u) = \frac{|UNEXP(u) \cap USEFUL(u)|}{|UNEXP(u)|} \tag{4.3.2}$$

Where the unexpected ($UNEXP$) set contained elements of RS which did not appear in PM, denoted by:

$$UNEXP = RS \setminus PM \tag{4.3.3}$$

RS was a set of recommendations generated by the proposed framework while *PM* was a set of recommendations generated by a primitive prediction model.

$USEFUL$ was a set of useful items which was determined by a user. For instance, the usefulness of an item could be approximated by the user's feedback. To determine the serendipity of the framework, the average serendipity of all users denoted by $SRDP@N$ was considered as follows:

$$SRDP@N = \frac{1}{|U|} \sum_{u \in U} SRDP(u) \tag{4.3.4}$$

4.4 Evaluation Results

The proposed framework was compared to the four following baseline methods in terms of $HitRatio@N$ and $SRDP@N$:

- UPCC (User-based PCC) [50]: the user-based collaborative filtering algorithm using Pearson correlation coefficient.
- IPCC (Item-based PCC) [21]: the item-based collaborative filtering algorithm using Pearson correlation coefficient.
- FUSE: the cross-domain recommendation which was proposed in Chen et al. [14]. The setting of FUSE method was the same as described in [14] for the model which did not utilize social network information (social network control parameter $\lambda = 10$ and latent factor $R = 50$).
- CIT: the cross-domain RS with consistent information transfer which was proposed in Zhang et al. [73]. The setting of three parameters in the CIT method (the number of user groups; K, the number of user groups; L, and regularization parameter λ) was fixed as described in [73].

The four baseline methods directly provided the top-N recommendation items for a particular user. To compare with these four baselines, the generated recommendation rules with their probability were used to find the top-N items in the recommendation lists for the users in a test set. Partitioning technique used was hold-out setting. In this setting, test reviews were sampled and hidden from the dataset without partitioning the users (Fig. 4.4). This partitioning technique was suitable to evaluate cross-domain recommendations with the accuracy and the diversity goals [10] and was used in most cross-domain RS researches (e.g., [14, 78, 73]). In the experiments, each target domain data was randomly split into a training set of 80% preference entries (reviews). A test set of 20%preference entries and the average results of 10 random times were then reported. It was noted that the actual user preference entries for the (recommended) items in the test set could be found in the dataset.

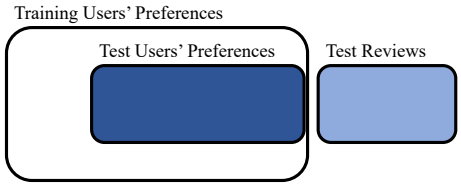

Figure 4.4: Hold-out data partitioning technique

Table 4.1 and Table 4.2 show the $HitRatio@5$ and $HitRatio@10$ for two domain-pairs: Music-Movie and Book-Movie respectively. Even though the quality of the top-5 and top-10 recommendations of the proposed framework was moderate, it did produce recommendations not found in the primitive single-domain based system. Such recommendations were not expected by the users and some of them were found interesting as shown by $SRDP@10$ in Table 4.3. No unexpected recommendation (if any) by UPCC and IPCC on both domain-pairs was found useful to the users.

Table 4.1: Comparison results for the quality of top-N recommendations ($@N$) on Music-Movie data

	UPCC	**IPCC**	**FUSE**	**CIT**	**Proposed Framework**
$HitRatio@5$	0.34	0.23	0.29	0.33	0.32
$HitRatio@10$	0.39	0.30	0.48	0.50	0.53

Table 4.2: Comparison results for the quality of top-N recommendations ($@N$) on Book-Movie data

	UPCC	**IPCC**	**FUSE**	**CIT**	**Proposed Framework**
$HitRatio@5$	0.39	0.27	0.32	0.41	0.38
$HitRatio@10$	0.41	0.34	0.53	0.56	0.57

The framework was also implemented on a single domain using only music dataset and there was no serendipitous recommendation found ($SRDP@10$ is 0 in Table 4.3). Although unexpected recommendations were suggested, $|UNEXP|$ was 2.25 on average out of top 10 recommendations. The output rules basically state that if the user likes music "A", then there is P% chance that the user will like music "B". Some music "B", e.g., "Take Five" and "You Never Know Who Your Friends Are" (Fig. 4.5), are considered as unexpected to the users since they are not found in a set of recommendations generated by a primitive prediction model. However, since only user preferences are available on the single domain, it is inadequate to predict useful

Table 4.3: Comparison results for the serendipity of top-10 recommendations (*SRDP*@10)

Dataset	UPCC	IPCC	FUSE	CIT	Proposed Framework (Single)	Proposed Framework
Music-Movie	0	0	0.15	0.16	0	0.21
Book-Movie	0	0	0.10	0.12	0	0.14

unexpected results which thus make the recommendations unattractive to the users. Hence, this experiment confirms that using user's preference on one domain to predict user's preference on another domain provides more chance in recommending unknown items that are interesting to the user.

```
%People who like music "A Hard Day's Night"
%also like music "Take Five"
0.13934::likeMusic(User, aharddaysnight) :- likeMusic(User, takefive).

%People who like music "Paint It Black"
%also like music "You Never Know Who Your Friends Are"
0.16721::likeMusic(User, paintitblack)
:- likeMusic(User, youneverknowwhoyourfriendsare).
```

Figure 4.5: The unexpected recommendation rules for likesMusic/2 (single domain)

4.5 Discussion

This research aims to affirm that relational learning and multiple domain could be combined to the benefit of RS. Relational learning provides the framework with the ability to make a prediction of possible new relations between users and items and these relations are presented in the form of an *if-then* statement. Incorporating information or knowledge from various or multiple domains will lead to a broader and wider set of recommendations including serendipitous items. The experiments carried out in this research illustrates that the proposed framework generates rules with explanation facilities for the RS and some surprising relations found lead to the unexpected but useful recommended items. The experiment results show that the proposed algorithm is indeed very promising. However, there are a few issues worth discussing in the following sub-sections.

4.5.1 Explainability

Standard measurement can be used to evaluate the quality of the recommendation. However, the method for transparency evaluation for explainability based on historical data (offline evaluation) is proposed in this research. This method is to determine whether the items appearing in the explanation and the items appearing in the user review texts belong to the same domains (i.e. the generated explainable rule can be considered as the generated user review). To perform the evaluation, 100 users were randomly selected. Each user received top 10 items which were recommended by the generated rules from the proposed framework. Then, only the recommended items found useful to the users were selected (based on the actual ratings in the dataset) and the review texts that the users wrote for these (recommended) items in the repository were investigated. If there was a piece of free text in the particular review that corresponded to the generated rules by the proposed RS, e.g., "movie" term was found in the user review text on a music recommended by the generated movie-music rule as shown by the process in Fig. 4.6, the explanation of the rules was transparent and useful to the user. The evaluation results showed 81% relevance for the recommendations from movie-music rules and 73% from book-music rules.

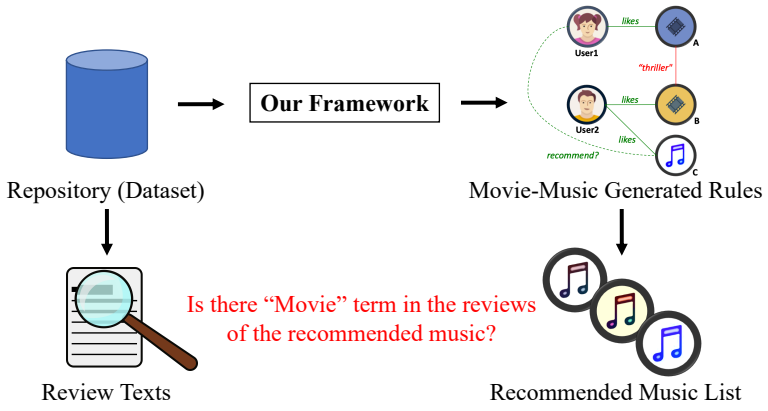

Figure 4.6: Explainability evaluation process

In addition to simplicity and transparency, effectiveness and efficiency also affect user satisfaction [25]. Thus, an evaluation of the effectiveness and the efficiency with four hypotheses should be conducted to ensure the outcomes as shown in Fig. 4.7.

However, it is surprising that even nowadays, the most common way of evaluating expla-

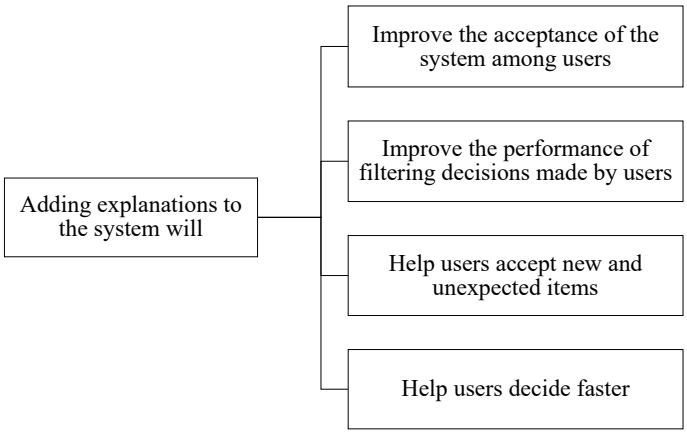

Figure 4.7: The hypotheses of the effectiveness and the efficiency evaluation

nations is by means of real user studies and acceptance testing [46]. The offline experiment of a new form of explanation still lacks a proper evaluation. Developing a reliable and easily usable evaluation will save a lot of efforts for offline evaluation of explainable recommendation systems. A further study on how to evaluate the explainable RS based on historical data (offline experiment) should be considered.

4.5.2 Novel and Unexpected Recommendation Coverage

Merging the results with the recommendations found by traditional methods may be considered to provide more coverage on the recommendation items. For example, merge the music recommendation result with the result from the traditional method on users' music preference only (single domain) shown as the highlighted section in Fig. 4.8.

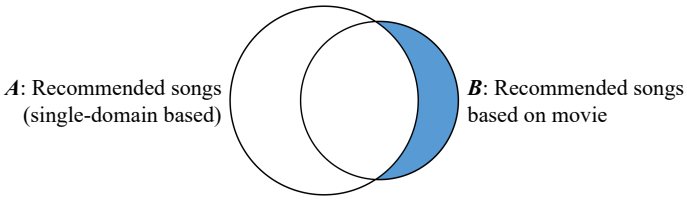

Figure 4.8: The results containing unexpected and useful recommendations - shown as the relative complement of A in B (the colored section) - not found in single-domain based system

The degree of "novelty" can be controlled or increased by incorporating more available user personal property and unexpected item attributes. Some examples of incorporating the unexpected data and the output are as follows:

- Music can be discovered through samples, cover songs and remixes connections (us-

ing data from WhoSampled[4]). Fig. 4.9 shows an unexpected connection between Busta Rhymes and Bernard Herrmann [59].

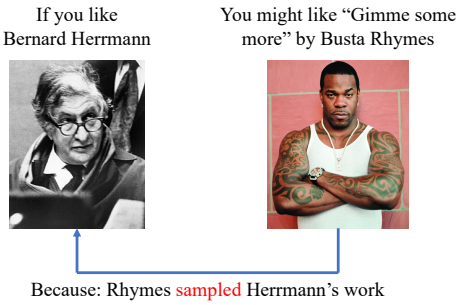

Figure 4.9: An example of unexpected connection via item attributes

- Fig. 4.10 shows examples of association via a user personal property [56]. On the left, the user and "The cup of life" are linked since the user was a soccer fan and the song was a theme song for 1998 FIFA World Cup. On the right, the user is linked to "Candle in the wind" because the user was an admirer of Princess Diana and the song was performed at Princess Diana's funeral.

Figure 4.10: Examples of unexpected connection via user attributes

4.5.3 Scalability

Most ILP methods including ProbFOIL do not scale to very large datasets as the time complexity of inference using grounding is exponential over the size of the dataset [18]. Hence, the execution time of the proposed framework is slower than the baseline methods. However, either minimizing a scalability problem or finding a scalable probabilistic rule learner is a challenge because incorporating additional domains, data, or context can create a scalability problem.

[4]https://www.whosampled.com/

Nonetheless, in this research it is accepted as a trade-off for a more coverage set of recommendations. A further study should be conducted to address the scalability issue, and it is considered as a future work for this research.

4.5.4 Cold-Start Problem

As mentioned earlier, if the recommendation process can be taken out of the black box, some problems could be solved. For example, the cold-start problem can simply be handled by generating the preferences for new users using the learned recommendation rules from this framework. The *item recommendation* module integrates the new user generated preferences with existing user preferences dataset, thus the framework has a better ability to deal with the cold-start problem and provides explainable recommendations.

Chapter 5

Extensibility of ILP Recommender System Framework

5.1 Overview

In this chapter, the extensibility that can enhance the performance of the proposed framework is discussed. With the nature of ILP in using an expressive representation language, it enables the framework to be extended by adding the following functions: context-aware, negative recommendation and constraint-based recommendation. The research of ILP in regard to these functions was published in [58].

The context-aware recommender system (CARS) generates the relevant suggestions to a user by adapting them (the suggestions) to the specific contextual situation of the user. However, in typical CARS, there are some concerns regarding how the information representing the context is obtained and how the contextual user preference is elicited since the user preferences may change over context. With ILP, the new contexts e.g. location, time and mood, can simply be added to the system to generate CARS by only specify predicates and settings.

Another aim of the system is to incorporate the negative recommendations - the items that the system will not suggest to the user. ILP provides recommendation rules in first-order if-then logic form which can indirectly turn into negative recommendation rules. The derived negative rules tell the system not to recommend certain products to the users. The specified constraint can simply be added to the framework as well.

In the following sections, the proposed framework so called, ILP-RS, with extensibilities,

is described. ILP-RS provides or not to provide the recommendation concerning user's requirements and in the right context to the user.

5.2 New Item Domains and Context-Aware

As mentioned earlier, adding a new domain into the system can easily be done by simply specifying the predicate setting. New data (e.g., user/item features and features from review text) and different contexts (e.g., location, time, and mood) can be incorporated as different domains into ILP-RS. It is no doubt that adding more useful features and contextual knowledge (e.g., user's activities collected from wearable devices) to the framework can provide more specific and useful recommendations to a particular user. An example recommendation rule (5.2.1) is the output learned rule that is expected (not from the actual experiment) if the contextual knowledge "user fall asleep" is known to the framework. According to (5.2.1), *music*2 is recommended to the user because he/she falls asleep and his/her preference is *music*1 and *music*1 is similar to *music*2. The "fall asleep" context is considered in suggesting the music.

$$\begin{aligned} &0.80000 :: likeMusic(User, music2) \text{ :- } likeMusic(User, music1), \\ &similarMusic(music1, music2),\ userActivity(User, fallasleep). \end{aligned} \tag{5.2.1}$$

5.3 Negative Recommendations

The negative recommendations refer to the items that will not be suggested to the user. A generated recommendation rule can be transformed into an equivalent negative form of which the conclusion part is negative. The expected rule is shown as follows:

$$0.80000 :: not(likeMusic(User, music1)) \text{ :- } not(userMood(User, good)). \tag{5.3.1}$$

The *music*1 will not be recommended to the user because the user is not feeling good at that moment. The explanation which is transformed from the rule tells the user why not to try out or to purchase a certain item. ILP-RS system can help the user from wasting the time and improve the system's trustworthiness in the user.

The negative recommendation rule also specifies that the users with particular preferences

will not like particular items. These found (by the system) disliked items are located in the tail of the popularity distribution and not previously rated by user. Thus, all disliked items must be discarded from the unexpected recommendation list to ensure that the un-useful outcomes are excluded, resulting in an increase in the serendipity measurement $SRDP@N$. Therefore, negative rules and disliked items could be used to enhance the system performance.

As far as I know, to date, research in negative recommendation has not yet been available. A further experimentation with negative rules may be necessary to confirm the capabilities of this proposed ILP-RS.

5.4 Constraint-Based Recommendations

In some recommendation tasks, there is a variety of constraints relating to user's requirements with item properties that a recommended result must meet. No recommendation that violates these constraints will likely be acceptable.

ILP-RS system, however, can generate a recommendation rule based on the constraints relating to a user's requirements. For example, the *cameraA* will be recommended to the user because the user specifies that he/she accepts a camera with a price of less than 1500 USD and is interested in a good portrait function. The expected rule is shown as follows:

$$\begin{aligned} 0.80000 :: recommendedCamera(User, cameraA)) :- \\ goodFunction(cameraA, portrait),\ price(cameraA, P),\ P < 1500. \end{aligned} \tag{5.4.1}$$

Chapter 6

Conclusion and Future Work

6.1 Summary

In this thesis, a novel framework for recommendation system is proposed. This framework focuses on two main aspects i) provide explanation - a rationale behind the decision to recommend the item which is personalized to the user, ii) offer more varieties and choices, which somehow connect to the user's preference.

For the first aspect, explanation, three important elements: goal, content, and form need to be defined in designing the framework.

Explanation Goal: Explanations for recommendations have been shown in the past to serve multiple purposes. This study focuses on the transparency which is considered the most important. *Transparency* refers to an explanation that clarifies how a recommendation was chosen. Explanation (Transparency) can be beneficial to different groups of people in different ways. To developers, understanding how the recommendation system works can help them tackle some of the problems occurred in RS e.g. cold start problem to improve the performance of their system. To vendors, knowing how the recommendations were chosen can help them better serve their customers which will increase their sales. To users, seeing how the recommended items relate to their preference items, they might pay attention or be influenced to try out or to purchase the items.

Explanation content: In earlier RSs, the content of explanation provided to all users is in the same pattern, based on the type of a particular algorithm. The main challenge encountered in this study is how to produce an explanation for each user that reflects an underlying algorithm

by which the recommendation is retrieved.

In this study, the use of relational learning is proposed to generate the rule for recommendation. The learned rule which is used to produce the recommendation for a particular user is directly transformed into its explanation. Accordingly, the explanation truly presents the reasoning and data behind the recommendation for the individual, no matter what the connection between the user and the chosen item may be e.g. the connections may go through different objects in different domains.

Explanation form: In fact, an explanation is intended for the best benefit of the users, to help persuade them to try or purchase recommended items especially the items that are totally new to them. Unfortunately, the explanation forms available so far mostly require either an extra effort to comprehend or are unclear and verbose, which could possibly be misleading or confusing. Consequently, the items are found uninteresting or overlooked by the users.

In this research, the relational learning is used to induct the explanation in the form of the first-order logic. First-order logic is considered sufficiently expressive to represent the natural language statements in a concise way. It has advantages over the natural language e.g. the English in that it is a powerful language that develops information about the objects in an easier way and can also express the relationship between those objects.

For the second aspect, the proposed framework ensures that the new and unexpected items are also presented to the users since all information and knowledge from multiple domains are utilized using relational learning. The relations between objects, if exist, cannot hide from the relational learning algorithm. I believe that the more knowledge is provided to the framework, the more specific and useful item will be recommended to a particular user. Therefore, the proposed framework is designed to be extendible in that the new domains (e.g., TV show, sport, beauty), new data (e.g., user/item features) and context (e.g., time) can directly be incorporated. All domains can be processed simultaneously, unlike the typical cross-domain RS which only two domains can be processed each time. A further note is that the quality and the degree of novelty can also be adjusted by the numbers of domains used or by incorporating more user personal property and unexpected item attributes available.

The proposed framework is flexible in terms of recommendation domain. The role of source

or target domain needs not be pre-defined due to the nature of relational learning, unlike a typical cross-domain RS which the source and the target domains must be pre-defined. However, this framework allows a user to specify the particular target and source domains if he/she so desires in the setting at the beginning of the process.

It is certain that the items that users dislike will not be presented to the users since the framework can derive negative recommendation rules automatically with the power of ProbFOIL. An introduction to ProbFOIL is briefly summarized in Chapter 3. The negative rules exclude the dislike items from the list and can generate negative examples for cold-start problem.

I hope to have demonstrated the idea of the proposed framework with the experiments on Amazon standard data. The experiment results show that this framework is very promising as it does produce interesting recommendations, not found in the primitive recommender systems, and accompanied with simple and readable explanations.

6.2 Future Research Directions

There are a number of interesting directions of future research that stem from the studies and results reported in this thesis. Further experimenting on context-aware and negative recommendation using ILP is therefore encouraged. In addition, other researchers are starting to implement deep learning (embedding-based recommendation model) [68, 69, 70] in recommendation system. It would be interesting to conduct the studies on deep learning so as to enhance the performance of the proposed framework. It also remains a question as to how to evaluate the explainability without human interaction.

I believe the results of this research will contribute to the current attempts to develop a recommender system which produces a quality recommendation for the user. Nevertheless, I hope to have shown that pursuing further research in this direction is a worthwhile aim.

Bibliography

[1] P. Adamopoulos and A. Tuzhilin. On unexpectedness in recommender systems: Or how to better expect the unexpected. *ACM Trans. Intell. Syst. Technol.*, 5(4):54:1–54:32, Dec. 2014.

[2] G. Adomavicius and A. Tuzhilin. *Context-Aware Recommender Systems*, pages 191–226. Springer US, Boston, MA, 2015.

[3] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol. *Data Mining Methods for Recommender Systems*, pages 39–71. Springer US, Boston, MA, 2011.

[4] L. Antanas, A. Dries, P. Moreno, and L. De Raedt. Relational affordance learning for task-dependent robot grasping. In N. Lachiche and C. Vrain, editors, *Inductive Logic Programming*, pages 1–15, Cham, Switzerland, 2018. Springer International Publishing.

[5] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, Dec. 1992.

[6] M. R. Berthold, editor. *Bisociative Knowledge Discovery: An Introduction to Concept, Algorithms, Tools, and Applications*. Springer-Verlag, Berlin, Heidelberg, 2012.

[7] M. Bilgic and R. J. Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI*, volume 5, page 153, 2005.

[8] D. Billsus and M. J. Pazzani. A personal news agent that talks, learns and explains. In *Proceedings of the Third Annual Conference on Autonomous Agents*, AGENTS '99, pages 268–275, New York, NY, USA, 1999. ACM.

[9] D. Bridge, M. H. Göker, L. McGinty, and B. Smyth. Case-based recommender systems. *Knowl. Eng. Rev.*, 20(3):315–320, Sept. 2005.

[10] I. Cantador, I. Fernández-Tobías, S. Berkovsky, and P. Cremonesi. *Cross-Domain Recommender Systems*, pages 919–959. Springer US, Boston, MA, 2015.

[11] R. Catherine and W. Cohen. Personalized recommendations using knowledge graphs: A probabilistic logic programming approach. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 325–332, New York, NY, USA, 2016. ACM.

[12] Ò. Celma. *Music Recommendation and Discovery in the Long Tail*. PhD thesis, Universitat Pompeu Fabra, Barcelona, 2008.

[13] Ò. Celma. *The Long Tail in Recommender Systems*, pages 87–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[14] W. Chen, W. Hsu, and M. L. Lee. Making recommendations from multiple domains. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, pages 892–900, New York, NY, USA, 2013. ACM.

[15] X. Chen, H. Chen, H. Xu, Y. Zhang, Y. Cao, Z. Qin, and H. Zha. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 765–774, New York, NY, USA, 2019. ACM.

[16] D. Cosley, S. K. Lam, I. Albert, J. A. Konstan, and J. Riedl. Is seeing believing?: How recommender system interfaces affect users' opinions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 585–592, New York, NY, USA, 2003. ACM.

[17] V. S. Costa, R. Rocha, and L. Damas. The yap prolog system. *Theory Pract. Log. Program.*, 12(1-2):5–34, Jan. 2012.

[18] L. De Raedt, A. Dries, I. Thon, G. Van Den Broeck, and M. Verbeke. Inducing probabilistic relational rules from probabilistic examples. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pages 1835–1843. AAAI Press, 2015.

[19] L. De Raedt, A. Kimmig, and H. Toivonen. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, IJCAI'07, pages 2468–2473, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[20] L. De Raedt and I. Thon. Probabilistic rule learning. In *Proceedings of the 20th International Conference on Inductive Logic Programming*, ILP'10, pages 47–58, Berlin, Heidelberg, 2011. Springer-Verlag.

[21] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[22] A. Felfernig and R. Burke. Constraint-based recommender systems: Technologies and research issues. In *Proceedings of the 10th International Conference on Electronic Commerce*, ICEC '08, pages 3:1–3:10, New York, NY, USA, 2008. ACM.

[23] Z. Gantner, L. Drumond, C. Freudenthaler, S. Rendle, and L. Schmidt-Thieme. Learning attribute-to-feature mappings for cold-start recommendations. In *2010 IEEE International Conference on Data Mining*, pages 176–185, Dec 2010.

[24] M. Ge, C. Delgado-Battenfeld, and D. Jannach. Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 257–260, New York, NY, USA, 2010. ACM.

[25] F. Gedikli, D. Jannach, and M. Ge. How should i explain? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367 – 382, 2014.

[26] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 507–517, Republic and Canton of Geneva, Switzerland, 2016. International World Wide Web Conferences Steering Committee.

[27] J. L. Herlocker, J. A. Konstan, and J. Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 241–250, New York, NY, USA, 2000. ACM.

[28] J. Hoxha and A. Rettinger. First-order probabilistic model for hybrid recommendations. In *12th International Conference on Machine Learning and Applications*, volume 2, pages 133–139, Dec 2013.

[29] L. Iaquinta, M. d. Gemmis, P. Lops, G. Semeraro, M. Filannino, and P. Molino. Introducing serendipity in a content-based recommender system. In *2008 Eighth International Conference on Hybrid Intelligent Systems*, pages 168–173, Sep. 2008.

[30] D. Jannach and G. Friedrich. Tutorial: Recommender systems. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI'13, Aug 2013.

[31] N. Kawamae. Serendipitous recommendations via innovators. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 218–225, New York, NY, USA, 2010. ACM.

[32] M. J. Kearns and R. E. Schapire. Efficient distribution-free learning of probabilistic concepts. *Journal of Computer and System Sciences*, 48(3):464 – 497, 1994.

[33] J. A. Konstan, S. M. McNee, C.-N. Ziegler, R. Torres, N. Kapoor, and J. T. Riedl. Lessons on applying automated recommender systems to information-seeking tasks. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1630–1633. AAAI Press, 2006.

[34] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug 2009.

[35] P. Kouki, S. Fakhraei, J. Foulds, M. Eirinaki, and L. Getoor. Hyper: A flexible and extensible probabilistic framework for hybrid recommender systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 99–106, New York, NY, USA, 2015. ACM.

[36] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, pages 535–541, Cambridge, MA, USA, 2000. MIT Press.

[37] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4, Part 2):2065 – 2073, 2014.

[38] P. Lops, M. de Gemmis, and G. Semeraro. *Content-based Recommender Systems: State of the Art and Trends*, pages 73–105. Springer US, Boston, MA, 2011.

[39] I. MacKenzie, C. Meyer, and S. Noble. How retailers can keep up with consumers. *McKinsey & Company*, 2013.

[40] B. M. Marlin, R. S. Zemel, S. T. Roweis, and M. Slaney. Recommender systems: missing data and statistical model estimation. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2686–2691, 01 2011.

[41] J. Masthoff. *Group Recommender Systems: Aggregation, Satisfaction and Group Attributes*, pages 743–776. Springer US, Boston, MA, 2015.

[42] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52, New York, NY, USA, 2015. ACM.

[43] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth. Thinking positively-explanatory feedback for conversational recommender systems. In *Proceedings of the European Con-*

ference on Case-Based Reasoning (ECCBR-04) Explanation Workshop, pages 115–124, 2004.

[44] D. McSherry. Explanation in recommender systems. *Artif. Intell. Rev.*, 24(2):179–197, Oct. 2005.

[45] U. Nagel, K. Thiel, T. Kötter, D. Piątek, and M. R. Berthold. Bisociative discovery of interesting relations between domains. In J. Gama, E. Bradley, and J. Hollmén, editors, *Advances in Intelligent Data Analysis X*, pages 306–317, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[46] I. Nunes and D. Jannach. A systematic review and taxonomy of explanations in decision support and recommender systems. *User Modeling and User-Adapted Interaction*, 27(3):393–444, Dec 2017.

[47] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. *PolyLens: A Recommender System for Groups of Users*, pages 199–218. Springer Netherlands, Dordrecht, 2001.

[48] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 502–511, Washington, DC, USA, 2008. IEEE Computer Society.

[49] J. Picado, A. Termehchy, A. Fern, and P. Ataei. Schema independent relational learning. In *Proceedings of the 2017 ACM International Conference on Management of Data*, SIGMOD '17, pages 929–944, New York, NY, USA, 2017. ACM.

[50] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, New York, NY, USA, 1994. ACM.

[51] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, pages 1257–1264, USA, 2007. Curran Associates Inc.

[52] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[53] J. B. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, pages 158–166, New York, NY, USA, 1999. ACM.

[54] S. Seo, J. Huang, H. Yang, and Y. Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, pages 297–305, New York, NY, USA, 2017. ACM.

[55] G. Shani and A. Gunawardana. *Evaluating Recommendation Systems*, pages 257–297. Springer US, Boston, MA, 2011.

[56] S. Sopchoke, K. Fukui, and M. Numao. Bisociative serendipity music recommendation. In *Theory and Practice of Computation*, pages 199–210, Singapore, Dec 2018. World Scientific.

[57] S. Sopchoke, K. Fukui, and M. Numao. Explainable cross-domain recommendations through relational learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, AAAI'18, Feb 2018.

[58] S. Sopchoke, K. Fukui, and M. Numao. Ilp recommender system: Explainable and more. In *Proceedings of the 29th International Conference on Inductive Logic Programming*, ILP 2019, Plovdiv, Bulgaria, 09 2019.

[59] S. Sopchoke, K. Fukui, and M. Numao. Explainable and unexpectable recommendations using relational learning on multiple domains. *Intelligent Data Analysis*, 24(6), 2020.

[60] S. Sopchoke and B. Kijsirikul. A step towards high quality one-class collaborative filtering using online social relationships. In *2011 International Conference on Advanced Computer Science and Information Systems*, pages 243–248, Dec 2011.

[61] N. Srebro, J. D. M. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, pages 1329–1336, Cambridge, MA, USA, 2004. MIT Press.

[62] S. Stober, S. Haun, and A. Nürnberger. *Bisociative Music Discovery and Recommendation*, pages 472–483. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[63] J. Struyf and H. Blockeel. *Relational Learning*, pages 851–857. Springer US, Boston, MA, 2010.

[64] D. Sturt and T. Nordstrom. Delight Your Customers By Giving Them What They Didn't Ask For. `https://www.forbes.com/sites/davidsturt/2014/01/03/delight-your-customers-by-giving-them-what-they-didnt-ask-for/`, 2014. [Online; accessed Nov 15, 2019].

[65] N. Tintarev and J. Masthoff. A survey of explanations in recommender systems. In *Proceedings of the 2007 IEEE 23rd International Conference on Data Engineering Workshop*, ICDEW '07, pages 801–810, Washington, DC, USA, 2007. IEEE Computer Society.

[66] N. Tintarev and J. Masthoff. *Explaining Recommendations: Design and Evaluation*, pages 353–382. Springer US, Boston, MA, 2015.

[67] W. Y. Wang, K. Mazaitis, and W. W. Cohen. Programming with personalized pagerank: A locally groundable first-order probabilistic logic. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2129–2138, New York, NY, USA, 2013. ACM.

[68] X. Wang, X. He, Y. Cao, M. Liu, and T. Chua. Kgat: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference*

on Knowledge Discovery & Data Mining, KDD '19, pages 950–958, New York, NY, USA, 2019. ACM.

[69] X. Wang, X. He, F. Feng, L. Nie, and T. Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1543–1552, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[70] X. Wang, X. He, M. Wang, F. Feng, and T. Chua. Neural graph collaborative filtering. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, pages 165–174, New York, NY, USA, 2019. ACM.

[71] P. Wärnestål. User evaluation of a conversational recommender system. In *Proceedings of the 4th IJCAI Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, pages 32–39, 2005.

[72] M. Zanker, M. Jessenitschnig, and W. Schmid. Preference reasoning with soft constraints in constraint-based recommender systems. *Constraints*, 15(4):574–595, Oct. 2010.

[73] Q. Zhang, D. Wu, J. Lu, F. Liu, and G. Zhang. A cross-domain recommender system with consistent information transfer. *Decision Support Systems*, 104:49 – 63, 2017.

[74] Y. Zhang and X. Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192, 2018.

[75] Y. Zhang, G. Lai, M. Zhang, Y. Zhang, Y. Liu, and S. Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '14, pages 83–92, New York, NY, USA, 2014. ACM.

[76] Y. Zhang, H. Zhang, M. Zhang, Y. Liu, and S. Ma. Do users rate or review?: Boost phrase-level sentiment labeling with review-level sentiment classification. In *Proceedings of the*

37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, pages 1027–1030, New York, NY, USA, 2014. ACM.

[77] Y. Zhang, M. Zhang, Y. Liu, S. Ma, and S. Feng. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1511–1520, New York, NY, USA, 2013. ACM.

[78] L. Zhao, S. J. Pan, E. W. Xiang, E. Zhong, Z. Lu, and Q. Yang. Active transfer learning for cross-system recommendation. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, pages 1205–1211. AAAI Press, 2013.

Appendix A

Extended Examples of Generated Recommendation Rules

A.1 50 Examples of Generated Recommendation Rules

- If user likes movie genre *western*, then user will like music genre *country* with a probability of 0.86735.

$$0.86735 :: likedMusic(User, Music) \text{ :- } likedMovie(User, Movie), \\ musicGenre(Music, country), movieGenre(Movie, western). \tag{A.1.1}$$

- If $User1$ and $User2$ like the same movie genre *thriller*, then $User2$ will like the same music as $User1$ does with a probability of 0.69523.

$$0.69523 :: likeMusic(User2, Music) \text{ :- } likeMovie(User2, Movie1), \\ likeMovie(User1, Movie2), movieGenre(Movie1, thriller), \\ movieGenre(Movie2, thriller), likeMusic(User1, Music). \tag{A.1.2}$$

- If user likes book *Harry Potter*, then user will like music by *Ed Sheeran* with a probability of 0.37552.

$$0.37552 :: likeMusic(User, Music) \text{ :-} \\ likeBook(User, harrypotter),\ musicArtist(edsheeran, Music). \tag{A.1.3}$$

- If user likes movie by *Marvel Studios* and book by *Stan Lee*, then user will like music

Back in Black with a probability of 0.13497.

$$0.13497 :: likeMusic(User, backinblack) \text{ :- } likeMovie(User, Movie), \\ movieProductionStudio(Movie, marvelstudios), \\ likeBook(User, Book), bookAuthor(Book, stanlee). \tag{A.1.4}$$

- If user likes book genre *science*, then user will like movie genre *biography* with a probability of 0.19461.

$$0.19461 :: likeMovie(User, Movie) \text{ :- } likeBook(User, Book), \\ bookGenre(Book, science), movieGenre(Movie, biography). \tag{A.1.5}$$

- If user likes movie genre *sci-fi*, then user will like book genre *computer* with a probability of 0.21311.

$$0.21311 :: likeBook(User, Book) \text{ :- } likeMovie(User, Movie), \\ bookGenre(Book, computer), movieGenre(Movie, sci\text{-}fi). \tag{A.1.6}$$

- If user likes movie genre *drama* and movie genre *comedy*, then user will like music genre *love* with a probability of 0.81232.

$$0.81232 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie1), \\ likeMovie(User, Movie2), movieGenre(Movie1, drama), \\ movieGenre(Movie2, comedy), musicTheme(Music, love). \tag{A.1.7}$$

- If user likes movie *Twilight*, then user will like music by an artist who started his career in 1996 with a probability of 0.14434.

$$0.14434 :: likeMusic(User, Music) \text{ :- } likeMovie(User, twilight), \\ careerStarted(Artist, 1996), musicArtist(Artist, Music). \tag{A.1.8}$$

- If user likes music by *Aerosmith*, then user will like movie genre *sci-fi* with a probability of 0.45691.

$$0.45691 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie), \\ movieGenre(Movie, sci\text{-}fi), musicArtist(aerosmith, Music). \tag{A.1.9}$$

- If user likes music *I Really Like You* and book *Catch Me If You Can*, then user will like movie *Forrest Gump* with a probability of 0.21621.

$$0.21621 :: likeMovie(User, forrestgump) \text{ :- } likeMusic(User, ireallylikeyou), \\ likeBook(User, catchmeifyoucan). \tag{A.1.10}$$

- If user likes movie genre *western*, then user will like music by *George Strait* with a probability of 0.45152.

$$0.45152 :: likedMusic(User, Music) \text{ :- } likedMovie(User, Movie), \\ movieGenre(Movie, western), musicArtist(georgestrait, Music). \tag{A.1.11}$$

- If user likes movie genre *family*, then user will like music by *Pink Floyd* with a probability of 0.14672.

$$0.14672 :: likedMusic(User, Music) \text{ :- } likedMovie(User, Movie), \\ movieGenre(Movie, family), musicArtist(pinkfloyd, Music). \tag{A.1.12}$$

- If user likes movie genre *family*, then user will like music genre *country* with a probability of 0.29052.

$$0.29052 :: likedMusic(User, Music) \text{ :- } likedMovie(User, Movie), \\ movieGenre(Movie, family), musicGenre(Music, country). \tag{A.1.13}$$

- If user likes book genre *science*, then user will like music by *Queen* with a probability of 0.26531.

$$0.26531 :: likeMusic(User, Music) \text{ :- } likeBook(User, Book), \\ bookGenre(Book, science),\ musicArtist(queen, Music). \tag{A.1.14}$$

- If user likes music by *queen*, then user will like book by *Marc Eliot* with a probability of 0.32131.

$$0.32131 :: likeBook(User, Book) \text{ :- } bookAuthor(Book, marceliot), \\ likeMusic(User, Music),\ musicArtist(queen, Music). \tag{A.1.15}$$

- If user likes movie by *Marvel Studios* and movie by *DC Films*, then user will like music

by *Breaking Benjamin* with a probability of 0.31586.

$$0.31586 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie1), likeMovie(User, Movie2), movieProductionStudio(Movie1, marvelstudios), movieProductionStudio(Movie2, dcfilms), musicArtist(breakingbenjamin, Music). \quad (A.1.16)$$

- If user likes music by *Deftones* and music by *Jay-Z*, then user will like movie by *DC Films* with a probability of 0.29176.

$$0.29176 :: likeMovie(User, Movie) \text{ :- } likeMusic(User, Music1), likeMusic(User, Music2), movieProductionStudio(Movie, dcfilms), movieArtist(deftones, Music1), musicArtist(jayz, Music2). \quad (A.1.17)$$

- If user likes music by *Lady Gaga* and movie genre *sci-fi*, then user will like music by *Jimi Hendrix* with a probability of 0.19978.

$$0.19978 :: likeMusic(User, Music2) \text{ :- } likeMovie(User, Movie), likeMusic(User, Music1), movieGenre(Movie, sci\text{-}fi), musicArtist(ladygaga, Music1), musicArtist(jimihendrix, Music2). \quad (A.1.18)$$

- If user likes movie *Dirty Dancing* and movie by *Pixar*, then user will like music by *Tim McGraw* with a probability of 0.23157.

$$0.23157 :: likeMusic(User, Music) \text{ :- } likeMovie(User, dirtydancing), likeMovie(User, Movie), musicArtist(timmcgraw, Music), movieProductionStudio(Movie, pixar). \quad (A.1.19)$$

- If user likes movie genre *romance* and movie genre *comedy*, then user will like music by *Celine Dion* with a probability of 0.65349.

$$0.65349 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie1), likeMovie(User, Movie2), movieGenre(Movie1, romance), movieGenre(Movie2, comedy), musicArtist(celinedion, Music). \quad (A.1.20)$$

- If user likes movie with *Oscars* awards, then user will like music by an artist who started his career in 1981 with a probability of 0.27341.

$$0.27341 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie), \\ movieAwards(Movie, oscars), \\ careerStarted(Artist, 1981), musicArtist(Artist, Music). \tag{A.1.21}$$

- If user likes music by *John Williams*, then user will like book *Harry Potter* with a probability of 0.71155.

$$0.71155 :: llikeBook(User, harrypotter) \text{ :-} \\ likeMusic(User, Music),\ musicArtist(johnwilliams, Music). \tag{A.1.22}$$

- If user likes movie genre *romance* and movie genre $family$, then user will like music by *Backstreet Boys* with a probability of 0.36141.

$$0.36141 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie1), \\ likeMovie(User, Movie2), movieGenre(Movie1, romance), \\ movieGenre(Movie2, family), musicArtist(backstreetboys, Music). \tag{A.1.23}$$

- If user likes movie genre *romance* and movie directed by *James Cameron*, then user will like music by *Celine Dion* with a probability of 0.80121.

$$0.80121 :: likeMusic(User, Music) \text{ :- } likeMovie(User, Movie1), \\ likeMovie(User, Movie2), movieGenre(Movie1, romance), \\ movieDirector(jamescameron, Movie2), musicArtist(celinedion, Music). \tag{A.1.24}$$

- If user likes music by *Al Martino*, then user will like book by *Mario Puzo* with a probability of 0.84111.

$$0.84111 :: likeBook(User, Book) \text{ :- } bookWriter(mariopuzo, Book), \\ likeMusic(User, Music),\ musicArtist(almartino, Music). \tag{A.1.25}$$

- If user likes book *The Hunger Games* and movie played by *Jennifer Lawrence*, then user

will like book by *Matthew Quick* with a probability of 0.31343.

$$0.31343 :: likeBook(User, Book) \text{ :- } bookWriter(matthewquick, Book), \\ likeBook(User, thehungergames), likeMovie(User, Movie), \\ movieStarring(jenniferlawrence, Movie). \tag{A.1.26}$$

- If user likes book *The Hunger Games* and movie played by *Jennifer Lawrence*, then user will like movie *Silver Linings Playbook* with a probability of 0.74918.

$$0.74918 :: likeMovie(User, silverliningsplaybook) \text{ :- } likeBook(User, Book), \\ bookWriter(matthewquick, Book), likeBook(User, thehungergames), \\ likeMovie(User, Movie), movieStarring(jenniferlawrence, Movie). \tag{A.1.27}$$

- If $User1$ and $User2$ like movie by *Pixar*, then $User2$ will like the same book as $User1$ does with a probability of 0.41523.

$$0.41523 :: likeBook(User2, Book) \text{ :- } likeMovie(User2, Movie1), \\ likeMovie(User1, Movie2), movieProductionStudio(Movie1, pixar), \\ movieProductionStudio(Movie2, pixar), likeBook(User1, Book). \tag{A.1.28}$$

- If user likes movie played by *Julia Roberts*, then user will like music by *Boyzone* with a probability of 0.39323.

$$0.39323 :: likeMusic(User, Music) \text{ :- } musicArtist(boyzone, Music), \\ likeMovie(User, Movie), movieStarring(juliaroberts, Movie). \tag{A.1.29}$$

- If user likes music by *Elton John* and movie by *Disney*, then user will like music by *Phil Collins* with a probability of 0.37158.

$$0.37158 :: likeMusic(User, Music2) \text{ :- } likeMusic(User, Music1), \\ musicArtist(eltonjohn, Music1), musicArtist(philcollins, Music2), \\ likeMovie(User, Movie), movieProductionStudio(Movie, disney). \tag{A.1.30}$$

- If user likes music album *Mirage* by *Camel*, then user will like book by *J.R.R. Tolkien*

with a probability of 0.37142.

$$0.37142 :: likeBook(User, Book) \text{ :- } bookWriter(jjrtolkien, Book), \\ likeMusic(User, Music), musicArtist(camel, Music), musicAlbum(mirage, camel). \tag{A.1.31}$$

- If user likes book by *George Orwell*, then user will like music by *Pink Floyd* with a probability of 0.12642.

$$0.12642 :: likedMusic(User, Music) \text{ :- } likedBook(User, Book), \\ bookWriter(georgeorwell, Book), musicArtist(pinkfloyd, Music). \tag{A.1.32}$$

- If user likes music by *Barclay James Harvest* and music by *Battlelore*, then user will like book by *J.R.R. Tolkien* with a probability of 0.51132.

$$0.51132 :: likeBook(User, Book) \text{ :- } bookWriter(jjrtolkien, Book), \\ likeMusic(User, Music1), musicArtist(barclayjamesharvestMusic1), \\ likeMusic(User, Music2), musicArtist(battlelore, Music1). \tag{A.1.33}$$

- If user likes music album *Sgt. Pepper's Lonely Hearts Club Band* by *The Beatles*, then user will like book by *Lewis Carroll* with a probability of 0.12998.

$$0.12998 :: likeBook(User, Book) \text{ :- } bookWriter(lewiscarroll, Book), \\ likeMusic(User, Music), musicArtist(thebeatles, Music), \\ musicAlbum(sgtpepperslonelyheartsclubband, thebeatles). \tag{A.1.34}$$

- If user likes music by *Iron Maiden*, then user will like book genre *horror fiction* with a probability of 0.39177.

$$0.39177 :: likeBook(User, Book) \text{ :- } bookGenre(Book, horrorfiction), \\ likeMusic(User, Music), musicArtist(ironmaiden, Music). \tag{A.1.35}$$

- If user likes movie genre *adventure*, then user will like book by *Jules Verne* with a prob-

ability of 0.89127.

$$0.89127 :: likeBook(User, Book) \text{ :- } bookWriter(julesverne, Book), \\ likeMovie(User, Movie), movieGenre(Movie, adventure). \tag{A.1.36}$$

- If user likes movie genre *sci-fi*, then user will like book by *Jules Verne* with a probability of 0.69910.

$$0.69910 :: likeBook(User, Book) \text{ :- } bookWriter(julesverne, Book), \\ likeMovie(User, Movie), movieGenre(Movie, sci\text{-}fi). \tag{A.1.37}$$

- If user likes book by *Jules Verne*, then user will like movie played by *Jackie Chan* with a probability of 0.11915.

$$0.11915 :: likeMovie(User, Movie) \text{ :- } movieStarring(jackiechan, Movie), \\ likeBook(User, Book), bookWriter(julesverne, Book). \tag{A.1.38}$$

- If user likes music *Take Five*, then user will like music *A Hard Day's Night* with a probability of 0.13934.

$$0.13934 :: likeMusic(User, aharddaysnight) \text{ :- } likeMusic(User, takefive). \tag{A.1.39}$$

- If user likes music *You Never Know Who Your Friends Are*, then user will like music *Paint It Black* with a probability of 0.16721.

$$0.16721 :: likeMusic(User, paintitblack) \text{ :-} \\ likeMusic(User, youneverknowwhoyourfriendsare). \tag{A.1.40}$$

- If user likes movie played by *John Travolta*, then user will like music by *Bee Gees* with a probability of 0.21105.

$$0.21105 :: likeMusic(User, Music) \text{ :- } musicArtist(beegees, Music), \\ likeMovie(User, Movie), movieStarring(johntravolta, Movie). \tag{A.1.41}$$

- If user likes book genre *adventurefiction* and book genre *history*, then user will like

movie played by *Harrison Ford* with a probability of 0.33251.

$$0.33251 :: likeMovie(User, Movie) \text{ :- } movieStarring(harrisonford, Movie), \\ likeBook(User, Book1), bookGenre(Book1, adventurefiction), \\ likeBook(User, Book2), bookGenre(Book2, history). \quad \text{(A.1.42)}$$

- If user likes book genre *children's*, then user will like movie by *Disney* with a probability of 0.91241.

$$0.91241 :: likeMovie(User, Movie) \text{ :- } movieProductionStudio(Movie, disney), \\ likeBook(User, Book), bookGenre(Book, childrens). \quad \text{(A.1.43)}$$

- If user likes book genre *Marvel Studios*, then user will like movie by *Disney* with a probability of 0.60012.

$$0.60012 :: likeMovie(User, Movie) \text{ :- } movieProductionStudio(Movie, marvelstudios), \\ likeBook(User, Book), bookGenre(Book, comic). \quad \text{(A.1.44)}$$

- If user likes book genre *romance*, then user will like movie played by *Julia Roberts* with a probability of 0.63451.

$$0.63451 :: likeMovie(User, Movie) \text{ :- } movieStarring(juliaroberts, Movie), \\ likeBook(User, Book), bookGenre(Book, romance). \quad \text{(A.1.45)}$$

- If user likes movie *The Bodyguard*, then user will like music by *Whitney Houston* with a probability of 0.73405.

$$0.73405 :: likeMusic(User, Music) \text{ :- } musicArtist(whitneyhouston, Music), \\ likeMovie(User, thebodyguard). \quad \text{(A.1.46)}$$

- If user likes movie *Titanic*, then user will like music by *Celine Dion* with a probability of 0.55112.

$$0.55112 :: likeMusic(User, Music) \text{ :- } musicArtist(celinedion, Music), \\ likeMovie(User, titanic). \quad \text{(A.1.47)}$$

- If user likes movie *Saturday Night Fever*, then user will like music by *Bee Gees* with a probability of 0.90338.

$$0.90338 :: likeMusic(User, Music) \text{ :- } musicArtist(beegees, Music), \\ likeMovie(User, saturdaynightfever). \tag{A.1.48}$$

- If user likes movie *The Lord of the Rings* and book by *J.R.R. Tolkien*, then user will like music by *Rick Wakeman* with a probability of 0.60883.

$$0.60883 :: likeMusic(User, Music) \text{ :- } musicArtist(rickwakeman, Music), \\ likeMovie(User, thetordoftherings), likeBook(User, Book), \\ bookWriter(jjrtolkien, Book). \tag{A.1.49}$$

- If user likes movie *Star Trek* and book by *J.R.R. Tolkien*, then user will like music by *Leonard Nimoy* with a probability of 0.91732.

$$0.91732 :: likeMusic(User, Music) \text{ :- } musicArtist(leonardnimoy, Music), \\ likeMovie(User, startrek), likeBook(User, Book), \\ bookWriter(jjrtolkien, Book). \tag{A.1.50}$$