Context-Aware Recommender Systems for Implicit Data

Xiaohu Liu

Doctor of Philosophy

University of York

Electronics

September 2014

Abstract

Recommender systems are software tools and techniques providing suggestions and recommendations for items to be of use to a user [18, 44, 55, 56]. These suggestions can help users make better decisions on choosing products or services, such as which film to watch, what music to listen to or which travel insurance to buy. When making suggestions, many recommender systems do not consider contextual information, such as location or time [5]. Recommender systems that make use of contextual information are called context-aware recommender systems.

Many context-aware recommender systems can not generate reliable recommendations on sparse data. Besides, in most context-aware recommender systems, the contexts are pre-defined and not personalised. These limitations of existing methods usually lead to inaccurate recommendations.

In this thesis, new context-aware recommendation methods are presented. In these methods, personalised contexts are defined based on users' activity patterns. The underlying associations between contexts are analysed, and similar contexts are combined so that the system can make use of existing data collected in similar contexts. Experimental results from two datasets show that the proposed methods can achieve significantly higher recommendation accuracy than existing context-aware recommendation methods.

Table of contents

List of figures

List of tables

Acknowledgements

I would like to thank my supervisors, Prof. Jon Timmis and Dr. Rogério de Lemos, for their encouragement and guidance.

I would like to thank my thesis advisor, Fiona Polack for her patience.

I also thank my parents, for their love and support.

Special thanks goes to my fiancée, for her patience and company.

My gratitude also goes to the NCR Financial Solutions Group for funding this project.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this thesis are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This thesis is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. All sources are acknowledged as References.

Chapter 1

Introduction

Recommender systems (RS) are software tools and techniques providing suggestions and recommendations for items to be of use to a user [18, 44, 55, 56]. These suggestions can help users make better decisions on choosing products or services, such as which film to watch, which restaurant to have dinner or which travel insurance to buy. Recommender systems were first studied as an independent research area in the 1990s [4]. Much work has been done in industry and academia on developing various new approaches [56].

When making suggestions, many RS do not consider contextual information, such as location, time and the company of other people [5]. However, the context can be of great importance in many cases. For example, the music a user prefers for a romantic dinner can be quite different from the music the user would like to listen to in a party. Recommender systems that make use of contextual information are called context-aware recommender systems (CARS), and the recommendations they make are called context-aware recommendations [35]. To clarify the terminology, in this thesis, the recommender systems that do not take into consideration any contextual information are called traditional recommender systems.

This thesis is focused on context-aware recommender systems. We study two major issues in the field of CARS: non-personalised contexts and data sparsity. Based on user activity data, we identify user's periodic activity patterns. Personalised contexts are defined based on these patterns to improve accuracy of recommendations . We analyse the underlying association between contexts based on item co-occurrence, so that contexts can be clustered and combined into overlapped "topics". Therefore, for a given context, the system can infer the user's preferences based on other different yet similar contexts. Consequently, the system can provide accurate context-aware recommendations on relatively sparse datasets.

In this Chapter, we first present an overview of recommender systems, especially context-aware recommender systems in Section 1.1. Then we discuss the motivations and contributions of the work in Section 1.2 and 1.3. Finally the structure of the thesis is presented in Section 1.4.

1.1 Recommender Systems

1.1.1 Traditional Recommender Systems

Typically, recommender systems provide the user with a ranked list of items (such as books, articles, films, etc.), which the user may be interested in [56]. To generate this list, RS need to predict user's preferences on these items. In order to complete this task, recommender systems must first collect user preferences, either explicitly or implicitly [56]. For example, a film recommender system may ask a user to rate the films the user has watched (explicit data); a music recommender system may consider the user listening to a track repeatedly as the implicit sign of preference for the track (implicit data). In most cases, the task of the RS is predicting the user's preferences on the items that the user has not purchased or used [4]. The items that are predicted to be the most desired are then recommended to the user.

In the field of traditional recommender systems, *collaborative filtering*, or *collaborative* methods, are considered to be the most popular and widely implemented techniques for predicting user's preference [17]. For a given user, collaborative methods recommend the items that users with similar preferences like [41]. Most collaborative methods are based on ratings (explicit data) [39]. The similarity of two users is computed based on their rating histories. Therefore, the collaborative methods are sometimes referred to as "people-to-people correlation" [56]. Collaborative methods only require user-item ratings [41]. No domain knowledge is needed. Therefore, collaborative methods can be easily applied in areas where ratings are available. However, the user usually has to rate a sufficient number of items before the RS with collaborative methods can accurately predict the user's preference [4].

Content-based methods recommend the items which are similar to the ones that the user preferred in the past. The similarity of items is computed based on the features ("content") associated with the items. For example, if a user has positively rated a film that belongs to the romance genre, the system may recommend other films from this genre. The content-based approach has its roots in information retrieval research [58]. Because of the early advancements made by information retrieval, many content-based systems can only recommend items containing textual information [4]. Content-based methods require both "content" of items and user feedbacks (implicit or explicit). However, in some cases, the "content" of is difficult to obtain.

1.1.2 Context-Aware Recommender Systems

What is context?

When we say "take this", "this" can be a cup, a ruler, or anything. It is unclear what "this" refers to, if we do not know anything abut the current context. Context has been defined in many research areas from different angles. In recommender systems, context includes extensive information. Schilit et al. [62] define context as:

Where you are, who you are with, and what resources are nearby.

Chen et al. [23] define context as:

Context is the set of environmental states and settings that either determines an application's behaviour or in which an application event occurs and is interesting to the user.

Abowd et al. [1] give a more specific definition:

Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

In the area of CARS, companion location and time are most widely used [71]. However, other factors such as weather and noise level may also become important information in some cases [5].

How can context be incorporated?

Since there are many existing traditional recommender systems, it is therefore intuitive to simply add a contextual filter to an existing traditional recommender system [4]. That is, we can simply "contextualise" the existing traditional RS, rather than building a new model. Based on which stage the contextual information is used at, context-aware recommender systems can be divided into contextual pre-filtering and contextual post-filtering [5].

The contextual pre-filtering system contains a pre-processing filter, which can contextualise the input data of the traditional RS [50]. That is, the information about the current context is used for selecting the relevant set of data [5]. Then, the preference of the user in the current context can be predicted by a traditional recommender system based on the selected data.

Similar to contextual pre-filtering, post-filtering also makes use of existing traditional RS. The contextual post-filtering methods ignore contextual information in the input data at the beginning, when generating traditional recommendations; then, the contextual post-filtering methods improve the obtained recommendations using contextual information [5]. The improvements can be made by either filtering out irrelevant items for the given context, or adjusting the ranking of items [5].

Sometime, it is not possible to obtain additional contextual information directly. However, we do know what items the user has chosen in the current context (these items are also called seed items). Therefore, if we assume the context remains the same, we can still make context-aware recommendations by [21]:

- recommending items that are similar to the seed items, or
- identifying the characteristics of the current context based on the seed items and then recommending suitable items.

These methods are called session based methods.

1.2 Motivations

In this section, we discuss the limitations of existing context-aware recommender systems, and the motivations for our work.

1.2.1 Data Sparsity and Context Generalisation

In traditional recommender systems, the prediction process usually begin with the specification of the initial rating set. The ratings can be explicitly collected from the users or inferred by the RS implicitly [5]. Then the RS computes the rating function R

$$R : User \times Item \to Rating$$

for all the (user, item) pairs that have not been rated by the users [5].

If the recommender system is based on explicit data, each user has to rate a sufficient number of items before the system can learn the user's preference and present the user with reliable recommendations [4]. However, in reality, most users are reluctant to provide ratings [49]. Typically, users rate only a small proportion of the items that are available [10]. Therefore, the dataset is sparse. In the field of traditional RS, this is usually solved by applying collaborative methods with latent factors [4], such as matrix factorisation [40]. However, in context-aware recommender systems, these methods are not always effective [5]. In CARS, the dataset becomes extremely sparse since the rating function R is extended into three dimensions [5]

$$R : User \times Item \times Context \rightarrow Rating$$

For example, in traditional music recommender systems, the user only needs to provide one general rating for the track that the user has played. However, to make context-aware recommendations, ideally, we would like to ask the user to rate the track for all the possible contexts, e.g. whether the track is suitable to play when driving, before sleeping, or during a party. The RS should be able to estimate the rating function for all the (user, item, context) combinations that have not been rated by the users. Therefore, the corresponding rating set can be much sparser compared to the dataset for traditional recommendation. In these cases, even employing the recommendation techniques that are able to deal with the data sparsity in traditional RS still leads to poor recommendations [11]. In some applications, due to severe data sparsity, for many users, the system is simply not able to provide context-aware recommendations at all [5].

Context generalisation is proposed to solve the problem of data sparsity in CARS [2]. In this method, a more general context is used instead of the exact context. For example, the temporal context can be changed from Saturday to weekend to obtain more data. Although some work has been carried out to improve the performance of context generalisation [36], context generalisation has some intrinsic limitations:

1. Generalisation can lead to inaccurate prediction.

   Using a generalised context means bringing in data collected in different context, in which the user may have different preference. There is a trade-off between accuracy and the level of generalisation.

2. There are too many possible generalisations of the given context [5].

   With highly granular contexts, there can be a very large number of generalisations of the given context. It is impossible to empirically evaluate the predictive performance on each generalised context due to high computational cost.

These problems suggest that a different method is needed to deal with the data sparsity of CARS specifically. Therefore, the first research question of this thesis is: *whether we can develop a CARS that is able to overcome data sparsity, in order to provide more accurate recommendations for more users, compared to existing CARS.*

1.2.2 Non-Personalised Context

Another issue in CARS is that all the temporal contexts are usually predefined based on common sense [5]. For example, in music recommender systems, it is common to define a set of non-personalised temporal contexts, such as morning, afternoon and evening in advance. The boundaries between these temporal contexts are the same for all the users. In reality, this can be problematic:

1. Different preference in the same temporal context.

   Some people may start the day much earlier or later than others. Therefore, different users may have different preferences in the predefined context, such as “evening”. More importantly, for some users, these predefined temporal contexts can be too general. For example, a user listens to rock & roll every morning to wake him up; when he is working in the morning, he listens to classical music. The temporal context “morning” becomes meaningless because it covers such a long period of the day. In the same “morning” the user’s preference can still change significantly. In this case, it might be better to divide the “morning” into two or more shorter contexts. However, it is impossible to know this if we use the same set of temporal contexts for all the users, and do not analyse each user's listening habits.

2. Same preference in different temporal contexts

   In the area of CARS, it is usually reasonable to assume that users have different preferences in different contexts [5]. However, sometimes, in different temporal contexts, a user's preferences could be the same. For example, during working days, a user's listening preference in the context "morning" may be the same as in the context "afternoon". Therefore, it is not necessary to define two different contexts "morning" and "afternoon" for this given user in terms of music listening. Let's also consider the following example of grocery shopping: a housewife goes to the supermarket on Monday afternoon and Thursday morning every week for groceries. The items she needs to purchase in these two temporal contexts are almost the same. If we can combine these two contexts, we can obtain more data for each of them, generating accurate and reliable recommendations in both contexts. Therefore, in this case, it is neither necessary nor appropriate to view these two contexts as different ones. However, in this example, it is almost impossible to discover the similarity between these two contexts based on the temporal information. We also need to analyse what items the user chooses to consume in each context, and whether these chosen items have something in common.

These problems suggest that we should not define a set of non-personalised contexts based on common sense. A new method is needed to discover and define contexts for each user. Therefore, the second research question of this thesis is: *whether we can develop a CARS that is able to define personalised contexts based on users' activity patterns, and to discover the underlying association between contexts, so that the system can achieve higher recommendation accuracy than existing CARS.*

1.2.3 Inability to Separate User Modelling from Context Modelling

Although the session based CARS can generate context-aware recommendations based on seed items, they usually have the following problems:

1. Inability to build a complete user profile.

   In such systems, recommendations are based on the items the user just chose in the current context (seed items). The given user's general preference has little influence on the recommendations. That is, the recommendations are not personalised. The same seed items would lead to the same recommendations.

2. Inability to select different features for user modelling and context modelling.

   In these systems, models with latent factors are employed to reduce the number of features in the data [37]. These new features are selected for either context profiling or user profiling. However, once the features are selected, they are used for both tasks. This can lead to inaccurate recommendations.

Since there are various traditional RS that can be used for user modelling, the key question here is how can we use these traditional methods along with session based methods, achieving accurate user modelling and context modelling at the same time. Therefore, the third research question of this thesis is: *whether we can develop a session based CARS that is able to employ two separate models for user profiling and context profiling respectively, to achieve higher recommendation accuracy than existing session based CARS.*

Although the three research questions focus on different problems of CARS, they are actually the same fundamental question of CARS from different angles: *how can we make better use of the contextual data we have, infer more information about user's preferences in different contexts, and make more accu-*

rate context-aware recommendations. In the following chapters, we review individual research question, along with this fundamental question of CARS.

1.3 Contributions

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

1. The development of a novel contextualisation method that can define personalised temporal contexts and make accurate context-aware recommendation on sparse dataset.

   By working on implicit usage data, users' periodic action patterns can be identified. For every user, personalised temporal contexts are defined based on these periodic action patterns. For each user action, we compute the probability that the action is performed in each context, and use the probability as weight. Consequently, we can make use of data collected in all contexts but with different weights. Therefore, compared to existing contextual filtering methods, our approach can increase recommendation accuracy and provide recommendations for more users.

2. The development of a novel session based contextualisation method that can uncover the underlying association between contexts across all the users, cluster similar contexts into overlapped groups, and apply traditional recommendation methods to generate accurate recommendations.

   We analyse the underlying association and similarity between contexts based on item co-occurrence and item tag co-occurrence. By doing this, we can cluster similar contexts into overlapped groups. For each group, a contextualised dataset is constructed, so that traditional recommendation methods can then be employed. Therefore, in our method, context modelling and user modelling are independent of one another. For a given context, based on the items the user just chose, we can infer the characteristics of the context and identify the corresponding groups of contexts to make accurate and reliable recommendations.

3. The development of a recommendation method that can combine our personalised contextualisation with the context clustering approach.

   The method is developed for implicit data. It combines the personalised contextualisation with our session based clustering method. In this approach, we first employ personalised contextualisation to identify personalised contexts. Then we cluster the contexts into groups based on item distributions. Therefore, the data collected in the same group of contexts can be combined to overcome data sparsity, reduce time complexity, and to generate accurate recommendations.

1.4 Thesis Structure

The thesis is organised as follows:

- Chapter 2 first introduces the field of traditional RS. The basic terms and ideas are outlined, along with the commonly used recommendation methods. Then we focus on the motivation of CARS, the importance of contextual information.
- Chapter 3 reviews the field of context-aware recommender systems. We introduce the fundamental issue and challenge in the field of CARS, discuss existing solutions and their drawbacks, which motivate our work. Besides, the appropriate evaluation methods for CARS are discussed in the chapter.
- Chapter 4 presents a personalised contextualisation method based on implicit data: the idea, the implementation, and the results.
- Chapter 5 presents a session based CARS that can identify and cluster contexts into groups. The experimental results and analysis are also presented.

- Chapter 6 presents a context-aware recommendation approach that combines the methods presented in Chapter 4 and Chapter 5. Results and analysis on different datasets are presented.
- In Chapter 7, the thesis is summarised and the main contributions are presented. A number of opportunities for future work in this area are discussed.

Chapter 2

Recommender Systems

As we introduced in Chapter 1, recommender systems can help users make better decisions on choosing products or services. A well designed RS is able to recommend items that suit the user's needs and wants, therefore improving the user's experience and increasing the number of items sold [56].

In this chapter, we first introduce the fields of traditional recommender systems, the RS that do not take into consideration contextual information. The basic terms and ideas are introduced, along with the commonly used recommendation methods. Then we outline the motivation for context-aware recommender systems.

2.1 Traditional Recommender Systems

The RS is primarily directed towards individuals who lack sufficient personal experience or competence to evaluate the potentially overwhelming number of alternative items that a Web site, for example, may offer [55]. A RS normally focuses on recommending a specific type of item (e.g., CDs, or news) and accordingly its design and the core recommendation technique used to generate the recommendations are customised to provide useful and effective suggestions for that specific type of item [56].

To provide recommendations, RS try to predict what the most suitable items are, based on the user's preferences. In order to complete such a task, RS need to collect users preferences, which are either explicitly expressed by the users (explicit data), e.g., as ratings for products, or inferred by interpreting user actions (implicit data) [61]. The explicit user ratings are usually more accurate since they are directly expressed by the users. Compared to explicit ratings, implicit data has the following characteristics [33]:

1. No negative ratings. By observing the user's behaviour, we can infer which item the user probably likes if the user has purchased or interacted with the item. However, it is hard to reliably infer which items a user does not like. For instance, a user did not purchase an item, simply because he did not know about the item.

2. Implicit data is inherently noisy. While we passively track the users behaviour, we can only guess their preferences and true motives. For example, we may view purchase behaviour for an individual, but this does not necessarily indicate a positive view of the product. The item may have been purchased as a gift, or perhaps the user is disappointed with the product.

3. Evaluation of systems based on implicit data requires appropriate measures. In the traditional setting where a user is specifying a numeric ratings, the difference between the predicted ratings and the true ratings can be computed to measure success in prediction. However, those metrics usually can not be applied to systems based on implicit data since the true ratings are not available.

Therefore, most recommender systems are based on explicit user ratings [4]. In most cases, the task of RS is predicting user's ratings on the items that the user has not purchased or used [4]. The items with highest predicted ratings are recommended to the user.

Formally, the problem of recommendation can be described as follows. Let U be the user (customer) set, and W be the set containing all the items such as books, films or insurance packages. In reality, both sets can be very large, from a few hundreds to millions. Ratings are used to indicate the usefulness of an item to a certain user. Let r_{ij} represent the rating that user u_i has given on item w_j. The rating matrix contains all the user-item ratings. In reality the matrix can be quite sparse. This is because compared to the total number of items in W, the number of items a user has interacted with is usually small, and the user does not rate all the items that he or she has interacted with [49]. Let R be the function that maps the user-item pairs to ratings:

$$R : User \times Item \to Rating$$

The recommender systems estimate the rating function R for all the (user, item) pairs that have not been rated yet by the users [5]. That is, the RS need to compute $\hat{r}_{ij}$, the predicted value of r_{ij} for all the unknown r_{ij}. For every user in U, the goal of the RS is to find the item or items in W that have the highest $\hat{r}_{ij}$. Therefore the central task of recommender systems is to find a function R that can accurately predict the rating for each user-item pair.

There are many ways to predict user-item ratings. Traditional recommendation methods are usually divided into the following three categories [10]:

1. Content-based methods

   These methods apply to items containing textual features, i.e. “content”, such as documents or news articles. The systems compare the contents of the items that the user likes, i.e. the items with relatively higher r_{ij}, and analyse their commonalities. The rating on an item is predicted based on how similar its content is to that of the items the user likes.

2. Collaborative methods

   In these methods, users with similar tastes and preferences are identified by comparing their ratings. A rating on a certain item by the given user is predicted based on similar users’ ratings on this item.

3. Hybrid methods

   These approaches combine collaborative and content based methods in various ways.

2.1.1 Content-Based Methods

Content-based methods make use of the features associated with items. For example, to recommend news articles to a user, the content-based RS tries to find what specific keywords are often used in the articles that the user finds interesting. Then news articles containing these keywords would be recommended [48].

The content-based approach has its roots in information retrieval research [58]. Because of the significant and early advancements made by information retrieval, many content-based systems focus on recommending items containing textual information, so that keywords can be used as item features. The improvement over traditional information retrieval approaches comes from the use of user profile and item profile [4].

Formally, let $Content(w_j)$ be the item profile of item w_j. It is usually built by extracting a set of features from item w_j. Since most content-based systems are designed to recommend items containing text, these extracted features are usually keywords that can be used to describe the characteristics of items. The importance or informativeness of keyword s_i in item profile $Content(w_j)$ is represented by weight v_{ij}. It can be defined in several different ways [48]. One of the most famous measures for specifying keyword weight is the term frequency/inverse document frequency (TF-IDF) [58]. Assume f_{ij} is the number of times keyword s_i appears in $Content(w_j)$. Then TF_{ij}, the term frequency of word s_i in the profile of w_j, is defined as follows [58]:

$$TF_{ij} = \frac{f_{ij}}{f_{max}} \tag{2.1}$$

where f_{max} is the max value of f_{ij} for all i. Usually, the word that appears in many item profiles is not very useful in identifying relevant items. Therefore, the term frequency is used together with the inverse document frequency (IDF). IDF_i, the inverse document frequency of the word s_i is defined as:

$$IDF_i = log\frac{N}{n_i} \tag{2.2}$$

where N is the total number of items (profiles), and n_i is the number of profiles containing word s_i. Then, the TF-IDF weight v_{ij} for word s_i in item profile $Content(w_j)$ is defined as:

$$v_{ij} = TF_{ij} \times IDF_i \tag{2.3}$$

The profile of item w_j can then be defined as a vector of keyword weights:

$$Content(w_j) = (v_{1j}, v_{2j}, ..., v_{Sj}) \tag{2.4}$$

where S is the number of unique keywords in the data.

Since the profile of item w_j can be represented by a vector, the similarity between two item profiles can be calculated by some scoring heuristic defined in terms of vectors, such as the cosine similarity measure [48]. Therefore, the content-based RS can recommend the items that are most similar to the user's favourite items. A user profile can be built by combining the profiles of items that the user likes [53]. Therefore, the similarity between the user profile and the item profile can be computed in a similar way. The candidate items with highest similarities are recommended to the user.

Besides the traditional methods that are based on information retrieval techniques, other model based content methods have also been developed. These techniques differ from information retrieval based approaches in that they predict utility based not on a heuristic formula, such as a cosine similarity measure, but rather on a model learned from the underlying data using statistical and machine learning techniques [43]. For example, in [51], a naive Bayesian classifier is used to determine whether the web page is relevant based on the probability that a given page belongs to a certain class (relevant or irrelevant), given the set of keywords on that page. This method is based on the assumption that all the words are independent. While the keyword independence assumption does not necessarily apply in many applications, experimental results demonstrate that naive Bayesian classifiers still produce high classification accuracy [51].

New Item

When a new item first appears, it has not been rated by any user, and sometimes will not be rated for a period of time [4]. In content-based recommender systems, the similarity between an item profile and a user profile is computed based on their contents. Therefore, a new item can be recommended, provided its associated features are available. That is, the user's rating on a certain item can be predicted even if the item has not been rated by any users [4]. For example, before the release date of the film, no user can rate the film. However, some features of the film, such as genre, cast and even storyline, are usually available before the release date. Therefore, the recommender systems can build the profile of the film, and compare it with the given user's profile to determine whether this film should be recommended to the user.

New User

If a user has not rated a sufficient number of items, a content-based recommender system cannot provide accurate and reliable recommendations (new user problem) [66]. If a new user has only rated a few items, the profile of the user may not contain enough information to represent the user's actual interests and preferences. Therefore, the RS can not generate accurate recommendations.

Overspecialisation

The content-based RS will only recommend items that are similar to those the user rated positively in the past. The recommendations tend to become increasingly overspecialised [10]. For example, if a user who likes Italian food has never been to a Greek restaurant in the past, the content-based RS is not likely to recommend a Greek restaurant to this user, even the best Greek restaurant in the city. This is because compared to Italian restaurants, a Greek restaurant can never be similar enough to be recommended. Moreover, the user may not want a recommendation that is so similar. For example, a user recently purchased a textbook and gave a high rating for the book. This does not mean he wants to buy all other versions of this book. Therefore, in some cases, content-based RS also need to filter out items that are too similar to the items the user has purchased [13]. However, the lack of diversity remains a serious issue of content-based recommender systems [20].

Limited Content Analysis

Content-based methods can be easily applied to items containing textual data. But applying content-based methods to multimedia data, e.g, graphical images, audio streams, is much more difficult [4]. In many applications, it is not practical to assign textual features by hand or obtain them from other sources. Besides, if two items are represented by the same features, it is not possible to discern the difference between the two items. For example, the commonly used TF-IDF technique ignores the order of words. Therefore, it cannot tell the difference between two documents if they have the same word frequencies [66].

Lack of Semantic Intelligence

Content-based methods classify items as positive or negative based on textual features (keywords) associated with the items. Large number of examples of both positive and negative data is required, to ensure reliable “syntactic” evidence of user interests, so that the RS can generate accurate recommendations. A problem with these methods is the “lack of intelligence” [43]. For example, if the user is interested in “French impressionism”, content-based methods will only find items that contain these exact words [43]. That is, an item related to Claude Monet will not be recommended in this case, though it is relevant. Semantic intelligence is needed. However, in reality, most recommender systems do not use semantic analysis techniques due to its high computational cost [4].

2.1.2 Collaborative Methods

Collaborative methods (or collaborative filtering methods) significantly differ from content-based methods. In collaborative methods, users with similar tastes and preferences are identified by comparing their ratings. The user's rating on a certain item is predicted based on his similar peers' ratings on that item. Unlike content-based methods, in collaborative filtering systems, the "contents" of items are not needed. This is convenient since it is not always possible to obtain features of items.

Generally, collaborative methods can be divided into two classes [17]:

1. Memory based methods

   Heuristics that make rating predictions based on the entire collection of items previously rated by the users. That is, the value of the unknown rating for the user-item pair is computed as an aggregate of ratings of some other similar peers of the given user.

2. Model based methods

   These methods try to learn a probabilistic model from the available ratings, and then use the model to predict the ratings.

Memory Based Methods

In memory based collaborative systems, the value of the unknown rating for the given item is usually computed as an aggregate of the ratings of other users for that item. For example, it can be computed as weighted sum of the ratings from the N most similar users on the same item.

Various methods have been used to compute the similarity between users in collaborative RS. Pearson correlation is one of the most popular approaches (also known as user based Pearson) [54]. Let S_{xy} be the set of items rated by both user x and user y. The similarity between the two users, sim_{xy}, is given by

$$sim_{xy} = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r}_x)(r_{ys} - \overline{r}_y)}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r}_x)^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r}_y)^2}} \tag{2.5}$$

where r_{xs} is the rating of item s given by user x, r_{ys} is the rating of item s given by user y, $\overline{r}_x$ is the average value of ratings given by user x, $\overline{r}_y$ is the average value of ratings given by user y. The similarities between any two users are calculated. Then, to predict the rating for the unknown user-item pair, the weighted average of all the ratings on that item are calculated, and the similarity between users is used as weight.

This user based Pearson is intuitive and easy to implement. However, when the number of users is high, computing the similarities between users becomes very time-consuming. In some cases, the number of items is much smaller than that of the users. Therefore, the similarities between items are calculated in a very similar way. [60]. This solves the problem of scalability of user based method.

Cosine based approaches are also used to calculate the similarities between users [17]. In these approaches, two users x and y are treated as two M-dimensional vectors, where M is the number of items that both users have rated, and the mth element of the vector is the user's rating on the mth item. The similarity between user x and y can be measured by the cosine of the angle between them:

$$sim_{xy} = cos(\overrightarrow{x}, \overrightarrow{y}) \tag{2.6}$$

Different recommender systems may use different methods to compute the similarities, either between users or between items. Sometimes, the number of users or items becomes very large. Therefore, computing the similarities can take a long time. In many application, the similarities are computed in advance [4], so when the user asks for recommendations, the RS can respond immediately.

Model Based Methods

In contrast to the memory based method, model based methods try to learn a probabilistic model from the ratings, and then use the model to predict the unknown ratings. This allows the system to learn patterns from the data and make more accurate predictions for the users [39]. Various machine learning methods can be used to predict user-item ratings.

Sometimes, the rating scale can be continuous, e.g. ratings in the Jester joke recommender system [29] can take any value between -10 and 10, such as -1.5. A regression method is therefore more appropriate. This approach is similar to the weighted sum method. But instead of directly using the ratings of similar items, it uses an approximation of the ratings based on regression model [29].

Instead of viewing the recommendation process as a prediction problem, it can also be viewed as a sequential optimization problem and uses a Markov decision process (MDP) model for recommender systems [64]. An MDP is a model for sequential stochastic decision problems, which is often used in applications where an agent is influencing its surrounding environment through actions. Working on an Israeli online book store, Mitos, the deployed MDP recommender system produced a considerably higher profit than the system without using MDP [64].

The restricted Boltzmann machine (RBM) is a stochastic generative model [67]. It is a neural network with one input layer and one hidden layer. For collaborative filtering, the visible units correspond to items [57]. For each user the visible units are activated by the items rated by the user. In [57], the accuracy of RBM applied on collaborative filtering problems is superior compared to memory based methods and linear models, because of its nonlinearity. The computational cost is usually acceptable and the probabilities of the hidden layer need to be pre-calculated [34].

Data Sparsity and Matrix Factorisation

An important issue with collaborative methods is data sparsity [10]. In most applications, the number of ratings obtained is very small, compared to the number of ratings that need to be predicted. Therefore, effective prediction of ratings from a small number of examples is important. Various model based methods have been proposed to solve the problem of data sparsity in recommender systems. In [52], a user profile containing age, gender, employment information is built, in order to identify similar users. This is known as Personality Diagnosis. However, this requires additional information from the user. Moreover, similar personalities can not guarantee similar tastes [52]. A different method for dealing with sparse rating matrix is proposed in [59], where Singular Value Decomposition (SVD) is used to reduce the dimensionality of a sparse ratings matrix. Following the idea of SVD, **matrix factorisation (MF)** is employed to deal with the data sparsity of rating matrix [27]. Matrix factorisation maps both users and items to a joint latent factor space of dimensionality f, such that user-item interactions are modelled as inner products in that space. As the Netflix[1] competition has demonstrated, matrix factorisation models can be superior to classic memory based techniques in terms of recommendation accuracy (top three teams all made use of MF). Especially on sparse dataset, it can generate much more accurate recommendations than most memory based and model based methods [40]. Therefore MF is widely used in many applications [40].

New User Problem

Similar to content-based methods, collaborative methods also require ratings from the user to make accurate recommendations. If a user has not rated enough number of items, the collaborative RS can not make personalised recommendations.

[1]http://www.netflix.com/

New Item Problem

Collaborative systems rely solely on users' ratings to make recommendations. Therefore, until a new item is rated by a substantial number of users, the RS would not be able to recommend the item to any users [61].

2.1.3 Hybrid Methods

To avoid the limitations of pure content-based method and pure collaborative method, many hybrid approaches combine the two. These methods can be classified into the following [4]:

1. Implement separate collaborative and content-based systems, and combine their results.
2. Incorporate some collaborative characteristics into a content-based approach.
3. Incorporate some content-based characteristics into a collaborative approach.
4. Construct a general unifying model that incorporates both content-based and collaborative characteristics.

Combining Results

The most intuitive way to combine content and collaborative methods is to implement two different systems separately, and then combine the results from individual systems. Pazzani proposes a linear combination to combine the ratings from different recommender systems [52]. Similarly, Claypool et al. use a voting scheme to combine the results of the two systems [25]. Instead of combining the results, we can also measure which recommender system is "better" given certain situations. Billsus and Pazzani propose a method in which the recommendation with higher level of confidence is selected [13]. Tran and Cohen employ a different method, in which the RS whose recommendations are more consistent with past ratings of users is chosen [70].

The combination (ensemble) of different types of recommendation algorithms can lead to significant performance improvements over individual algorithms [34]. However, training multiple recommender systems and combining their results can be computationally expensive [63]. Therefore, the criteria used to evaluate the RS, the trade-off between speed and accuracy become very important.

Combining Recommender Systems

Several hybrid methods add collaborative characteristics to content-based models. Usually, a dimensionality reduction technique is applied to a group of content-based profiles. Soboroff and Nicholas use a latent semantic indexing method to create a collaborative view of user profiles, where user profiles are represented by feature vectors [68]. Also, several hybrid methods incorporate some content-based characteristics into a collaborative approach. Pazzani propose a "collaboration via content" method, in which content-based profiles are maintained to compute the similarities together with traditional collaborative filtering [52]. This method can help address the data sparsity problems. Similarly, Melville et al. use a content-method is to obtain additional ratings [47]. Collaborative filtering is then used on both true ratings and predicted ratings.

In some hybrid method, a single unifying recommendation model is employed, rather than combining content-based and collaborative methods. Basu et al. propose a single rule-based hybrid classifier[12]. Ansari et al. use Bayesian mixed-effects regression models for predictions while Markov chain Monte Carlo methods are used to estimate the parameters [8].

2.2 Motivation for Context-Aware Systems

Traditional recommender systems do not take contextual information into account. The recommendations are made solely based on user-item information [5]. However, the context includes information regarding the current situation.

It can be vital in many applications.

In Chapter 1, we presented the definition of context [1]:

Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

This definition covers most aspects of context dealt with in this field. Therefore, it is used in this thesis.

In many cases, a user's preference depends on the given circumstances, such as when, where, and with whom the product will be consumed. For example, when recommending a film, the tradition RS only considers the following:

a. the characteristics of the film and

b. the user who wants to watch the film.

However, contextual information should also be considered. For example, the following information might be used:

c. with whom the film will be seen (e.g. alone, with girlfriend / children);

d. when the film will be seen (e.g. weekday / weekend, morning / afternoon / evening);

e. where the film will be seen (e.g. at home / in the theatre, on the train / plane);

The omission of any of the above information can lead to inappropriate recommendations, e.g. recommending a film that contains intense sequences of violence to a user who is going to watch the film with children. Hence, Adomavicius and Tuzhilin argue that the contextual information does matter in recommender systems and it is important to develop context-aware recommender systems that take contextual information into consideration [3]. Therefore, the rating function R is defined in three-dimension space (as opposed to the traditional two-dimension user-item space):

$$R : User \times Item \times Context \rightarrow Rating$$

In the example of film recommendation we talked about in this section, the $Context$ can contain information c, d and e. Each of these can affect $Rating$.

2.3 Summary

In this chapter, we first introduced the traditional recommender systems: how they work, on what type of data. Then we focused on the motivation for context-aware recommender systems. We described the importance of contextual information, how they can affect user's preferences significantly.

In the following chapter, we present a review of the field of CARS: the challenge of CARS, existing methods and their drawbacks.

Chapter 3

Review of Context-Aware Recommender Systems

In the previous chapter, we provide the necessary background of recommender systems. In this chapter, we review the field of context-aware recommender systems. We first introduce the fundamental issue and challenge in the field of CARS. Then we discuss existing solutions and their drawbacks, which motivate the work in this thesis.

3.1 Lack of Contextual Data

As we discussed in Chapter 1 and Chapter 2, the dataset of RS is usually sparse. In CARS, the rating function R is extended into three dimensions [5]

$$R : User \times Item \times Context \rightarrow Rating$$

A general user-item rating used in traditional RS can not indicate the user's preference in different contexts. To make context-aware recommendations, ideally, we would like to ask the user to rate the item in a context, e.g. whether the track is suitable to play when driving, before sleeping, or for a party. However, in reality it is almost impossible to obtain such rating by constantly asking the user to rate products in different contexts explicitly. This is very different from traditional RS, which rely on user-item ratings.

In recent years, smart mobile phones become increasingly popular. A typical smart phone, such the iphone 5[1], contains a GPS chipset, which can accurately locate the user. Theoretically, companies, such as Apple[2], can easily track the user's location [72]. The location of user can be vital to the study of many context-aware recommender systems. However, Apple has been very protective of its data and relevant research. It has never given public access to any CARS related data. Online music listening platform Spotify[3] has more than half a million users [69]. Spotify is making context-aware recommendation to its users based on time and location, but has never released any user data or published any research work. For researchers, it is difficult to conduct research on CARS without necessary data. The lack of real contextual data was and still is the fundamental issue of CARS.

The same issue is also identified by other researchers. For example, Adomavicius and Tuzhilin argue that most work on context-aware recommender systems has been conceptual, where a certain method has been *developed, tested on some (often limited) data* [5]. Verbert et al. also outline that a big challenge of CARS is *lack of publicly available data sets* [71].

Therefore, the very challenge that all researchers of CARS must face is *how to obtain (enough) contextual data, and effectively learn user's preference from the data.* The lack of contextual data must be considered before designing any context-aware recommender systems.

Since we can not ask the user explicitly, we must collect contextual information in a different way. In general, we can obtain or infer context via the following two types of approaches:

1. Make use of implicit transaction data. For example, we can obtain tem-

[1]http://www.apple.com/shop/buy-iphone/iphone5s
[2]http://www.apple.com
[3]https://www.spotify.com/

poral information from transaction timestamps, and infer user's preference from transactions.

2. Make use of active sessions. If we know what items the user has chosen in an ongoing session (e.g. browsing session, music listening session) we can analyse the characteristics of the session, and make context-aware recommendations.

In the following sections, we introduce these two types of method, discuss their pros and cons respectively.

3.2 Transaction Based Method

If we can obtain context from implicit transaction data, we can include the contextual information into the recommendation process, and make use of existing traditional RS. Based on which stage the contextual information is used at, context-aware systems can be divided into the following groups [5]:

1. Contextual pre-filtering (or contextualisation of recommendation input)

   Contextual pre-filtering system contains a pre-processing filter, which is used to contextualise the input data of the traditional recommender systems [50]. That is, information about the current context is used for selecting or constructing the relevant set of data. Then, preferences can be predicted using a traditional recommender system on the selected data.

2. Contextual post-filtering (or contextualisation of recommendation output)

   Contextual post-filtering also makes use of existing traditional recommender systems. The contextual post-filtering approaches ignore contextual information in the input data when generating a list of traditional recommendations. Then, the contextual post-filtering approaches adjust the obtained recommendations using contextual information [5].

The adjustments can be made by either filtering out irrelevant items for the given context, or adjusting the ranking of items on the list.

These methods are also call contextualisation methods, since they either contextualise the input or the output to generate context-aware recommendations.

3.2.1 Contextual Pre-Filtering

There are various RS that can be used for a traditional recommendation task. It is intuitive to simply add a contextual filter to the existing RS than to develop an entirely new model [4]. A contextual pre-filtering system usually contains a pre-processing filter, which can contextualise the input data for the recommender system [50].

Following this idea, a reduction based method is proposed in [36], which reduces the problem of contextual recommendations to the two dimensional User$\times$Item recommendation space. First, the data selection is carried out: the data that is relevant to the given context is chosen for the next step. Then, traditional recommendation methods are used to to predict users' preferences based on contextualised data. This method is the frequently used as benchmark for context-aware recommender systems [5].

3.2.2 Contextual Post-Filtering

Similarly to contextual pre-filtering, post-filtering also makes use of an existing recommender system. The contextual post-filtering approach ignores context information in the input data when generating recommendations. Then, the contextual post-filtering approach adjusts the obtained recommendation list for each user using contextual information [5]. The recommendation list adjustments can be made by:

- Filtering out recommendations that are irrelevant (to the given context).

- Adjusting the ranking of recommendations on the list (based on the given context).

Post-filtering methods can provide quick response to the user. The traditional recommendation tasks can be done in advance. When context-aware recommendations are required, the system can quickly contextualise (filter or adjust) the traditional recommendations results.

3.2.3 Drawbacks of Contextualisation: Non-Personalised Contexts and Data Sparsity

In this section, we discuss the common issues of the contextualisation methods we have introduced in previous section.

Sometimes, with contextual pre-filtering methods, the exact context can be too narrow. After data selection in contextual pre-filtering, the data sparsity increases. Therefore, the system may not have sufficient data to make accurate recommendations [5]. Consider the film recommendation task: a person wants to see a film with his girlfriend in a theatre, on Friday evening. If the exact context is used, it is likely that results are not accurate and reliable. First, we may not have enough data that can match the exact context. In this case, it is possible we may not get any recommendations at all for this very specific context. Second, it is not necessary to use the exact context. For example, the suitable films for Friday evening may not be very different from the films for Saturday evening. Incorporating a more general context, such as weekend evening in this example, will not reduce the accuracy, but provide diverse and reliable recommendations, since the system can learn from further training data.

Context generalisation is proposed to address this problem [2]. Consider we have context $c =$ (company, place, time). Using the film recommendation example, we have $c =$ (girlfriend, theatre, Friday evening). For every component in the context, a generalised set S, which has a hierarchical structure

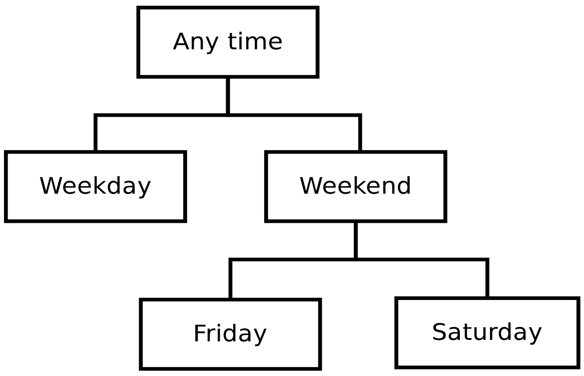

Fig. 3.1 The hierarchical structure of contextual information

of contextual information, can be defined (see Figure 3.1). We can define a different S for each component of the context, and several hierarchies for more general context, as the following example shows:

- Time: Friday evening $\rightarrow$ weekend evening $\rightarrow$ evening $\rightarrow$ any time
- Place: theatre $\rightarrow$ any place
- Company: girlfriend $\rightarrow$ friend $\rightarrow$ any company

Then we can have multiple generalised contexts for $c =$ (girlfriend, theatre, Friday evening). For example:

- $c_1 =$ (girlfriend, any place, Friday evening)
- $c_2 =$ (friend, theatre, Weekend evening)
- $c_3 =$ (friend, any place, any time)

Note that the set of possible contexts here must be pre-defined, or the system is not able to make use of the context. In fact, in most context-aware recommender systems, the contexts are pre-defined. This can be problematic, especially when it comes to temporal contexts. Some people may start the day much earlier or later than others. Therefore, different users may have different preferences in the pre-defined context, such as “evening”. More importantly, these pre-defined temporal contexts can be too general for some users. In a relatively long and general context, the user’s preference can still change significantly. However, if the context is not general enough, the RS can not obtain enough data to learn the user's preference. Beside, there may be many generalised contexts. Therefore, it is very important to determine which context to use and how general the context should be. One option is to use a manual, expert-driven approach [5]; e.g., always generalise specific days of the week into a more general Weekday or Weekend. However, in different applications, the type of context we are interested in may be different. The generalisation rules may not be appropriate in all applications. It is also impossible to manually specify all the generalisation rules in a system that involves a very large number of possible contexts. Another option is to use a more automated approach that could empirically evaluate the predictive performance of the recommender system on contextualised input datasets obtained from each generalised pre-filter. Then the RS chooses the pre-filter with the best performance. In cases of applications with highly granular contexts, there may be a very large number of possible context generalisations. It is not possible to empirically evaluate the predictive performance on each generalised context due to high computational cost.

Contextual post-filtering is faced with the same problems. In fact, the problems of context generalisation are shared by all the contextualisation methods. In these approaches, the contextualisation process requires us to define a set of contexts first, which are not personalised. Exact and detailed context can again lead to data sparsity; while generalised context leads to inaccurate recommendations.

These limitations of contextualisation methods suggest that a new context-aware recommendation method is needed. Therefore, as we introduced in Section 1.2, the following research questions are raised:

- whether we can develop a recommendation method that is able to overcome the data sparsity in context-aware recommender systems.
- whether we can develop a recommendation method that is able to define personalised contexts.

3.3 Session Based Method

Sometimes, it is not possible to obtain additional contextual information. However, we do know what items the user has chosen in the current context (also known as "seed items"). Therefore, if we assume the context remains the same, we can make context-aware recommendations based on these seed items.

Following this idea, several methods are proposed to make context-aware recommendations. In these methods, a context is usually defined as an active, continuous session with the system, such as a music listening session, or a web browsing session [7, 30].

3.3.1 Non-personalised session based methods

In these methods, recommendations are based on the items that the user just chose in the current context (seed items). The given user's general preference has little influence on the recommendations. That is, the recommendations are not personalised. The same seed items would lead to the same recommendations. For example, Hariri et al. propose a context-aware music recommender system that can infer contextual information based on the most recent sequence of songs liked by the user [30]. Given a sequence of songs in a user's current interaction, the discovered patterns in sequence are used to predict the next track in the playlist. This method can not generate personalised recommendations, because based on the same sequence of songs, the RS always generate the same recommendations, without considering user's general taste reflected in other contexts (sessions).

3.3.2 Personalised session based methods

In these methods, each session is typically viewed as a mixture of various "topics", and each "topic" has unique item distributions. Therefore, the central task of session based methods is to discover the underlying association between contexts, and to identify similar contexts (sessions) based on the topic mixture. Most session based methods are based on some latent factor model. These latent factors determine the characteristics of “topics”, thereby the characteristics of a context. In these methods, the user-item interaction data are replaced with user-topic data. These new features (topics) are selected for either context profiling or user profiling [37]. However, once the features are selected, they are used for both tasks.

The Latent Dirichlet Allcation (LDA) is a widely used latent factor model. It can be used to model the distributions of items in sessions. LDA was originally proposed for modelling text documents, extracting useful features from a set documents[16]. LDA models each document (session) as a mixture of latent topics where each topic is a distribution over the vocabulary (items).

Based on LDA model, Hariri et al. propose the Query-driven context-aware recommendation (QD), a model that integrates user profiles, item representations, and contextual information [31]. In this method, tags or keywords associated with the item are used. Therefore, each item can be represented as a “bag of words”, which is equivalent to a document. LDA model is used to generate the features (topics) of these documents; each topic (feature) is a distribution over tags; user, item and context are all represented as mixture of various features. However, as we discussed earlier in this chapter, matrix factorisation has been proven to be superior in modelling user item interaction. If we could separate the modelling of context, use separate sets of features for contexts and items, we may be able to generate more accurate recommendations.

Similar to QD, Zheleva et al. propose a music-listening session based model (SBM) [73]. This method is also based on LDA model. However, tags and keywords are not used in this method. Therefore, each topic is simply a distribution over items instead of tags. Users and contexts are represented as mixtures of features (topics). If we could separate the modelling of context and user, use different features, we may be able to generate more accurate recommendations.

The above issues of existing session based methods suggest that a new approach is needed. The new approach should be able to model user's general preference, and to model the underlying association between contexts. More importantly, the modelling of users and contexts should be independent of each other, as we discussed in our third research question in Chapter 1. Therefore, accurate user modelling and accurate context modelling can be achieved at the same time.

3.4 Evaluation of Context-Aware Recommender Systems

Most popular recommender systems are based on explicit data. The criteria for evaluating those systems may not be suitable for CARS. In this section, we review commonly used testing and analysis criteria, and choose the appropriate ones for our method.

3.4.1 Rating Prediction Accuracy

The recommender system generates predicted ratings $\hat{r}_{ij}$ for a test set of user-item pairs (user u_i, item w_j) for which the true ratings r_{ij} are known. Root Mean Squared Error (RMSE) between the predicted ratings $\hat{r}_{ij}$ and true ratings r_{ij} is given by [38]:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum\left(\hat{r}_{ij} - r_{ij}\right)^2} \tag{3.1}$$

where N is total number of ratings that need to be predicted.

RMSE is also used to evaluate the recommender systems based on implicit data. For example, Baltrunas and Amatriain [11] convert all the music play counts, p_{ij}, to user-item (user-track) ratings r_{ij} (1 to 5 stars). Then a recommendation method based on ratings is employed. The system is evaluated by RMSE, comparing the difference between the predicted rating and the rating mapped from play counts. In such cases, the ratings mapped from implicit data are used as true ratings. However, the mapping can be inaccurate. An extreme example is that all the user-track play counts are converted to 5-star ratings, no matter how many times the track has been actually played by the user. A zero RMSE can be easily achieved if the system simply predicts all the ratings are 5 stars. Obviously such system would perform poorly in reality. Therefore, RMSE becomes meaningless due to inaccurate mapping from implicit to explicit data. Besides, in most cases, it is impossible to measure whether the mapping is accurate or not, unless both the true rating and its corresponding implicit data are available. Therefore, RMSE should not be used to evaluate the systems built on implicit data.

3.4.2 Relevance Prediction Accuracy

In many applications, we do not care about how accurate the system can predict all the user-item ratings. Instead, we would like to know whether the recommended items are relevant, whether the user would actually use (purchase, listen, or consume) them [22]. Compared to RMSE, this property is far more practical and intuitive.

To employ relevance prediction, typically a data set consisting of the items each user finds relevant and irrelevant is needed. We then select a test user, hide some of his or her selections, and ask the recommender to predict whether the user would find these selected items relevant [65]. Then the recommendations comes in a ranked list of items, ordered by decreasing relevance [22]. We can determine whether a recommended item is in fact relevant or not based on the user's predicted rating on the given item: large value indicates the item is relevant, small value indicates it is not relevant. We then have four possible outcomes, as shown in Table 3.1.

	Recommended	Not Recommended
Relevant	True Positive (TP)	False Negative (FN)
Not Relevant	False Positive (FP)	True Negative(TN)

Table 3.1 Possible results of a recommendation [65]

- True positive (TP): RS recommends an item that is relevant.
- False positive (FP): RS recommends an item that is irrelevant.
- True negative (TN): RS does not recommend an item that is irrelevant.
- False negative (FN): RS does not recommend an item that is relevant.

With implicit data, by observing the user's behaviour, we can infer which items the user finds relevant. For example, if the user repeatedly plays a track, the user must like it. However, it is hard to reliably infer which items the user does not like or finds irrelevant from the smaller values of implicit feedbacks [33]. In some cases, this can be done by randomly selecting the items the given user has never interacted with as irrelevant items. This method can only be used when the dataset contains a significant number of items, and each user is only interested in a relatively small amount of items.

By using this method, we can generate test sets containing both relevant and irrelevant items. Precision measures the proportion of recommended items that are in fact relevant. It is defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.2}$$

Precision can tell us how many irrelevant items are recommended to the user. Precision can also be evaluated at a given cut-off rank, considering only the top-N recommendations. This measure is called precision at N. It is suitable for evaluating CARS on implicit transaction data.

3.4.3 User Space Coverage

User space coverage (or just "coverage") is the proportion of users for whom the system can make recommendations [22].

Although some collaborative methods can predict user's preference on sparse dataset (though prediction accuracy usually drops given less data), not every recommendation method has the ability to work on sparse data. In many applications the system is not able to provide recommendations due to limited amount of data or data sparsity. In such cases we may prefer recommender systems that can provide recommendations for a wider range of users.

In context-aware recommendations, only the data collected in the given context should be used for training. In contextual pre-filtering, data not collected in the given context is filtered out. This filtering step tends to make the dataset far sparser, leading to poor coverage of users. Therefore, it is important to compare the coverage of CARS.

3.4.4 Significance

If a method is non-deterministic, calculating the performance of a single instantiation of the method in order to compare it with other algorithms would not be wise, as that instantiation might be a good or bad one, depending on chance. In order to gain the representative results, we should repeat each experiment involving non-deterministic methods multiple times and take the median of the performance values.

If an experiment involves non-deterministic method, the results can be analysed by the combined use of the Hodges-Lehmann Estimate with associated non-parametric confidence intervals at 95%. The Hodges-Lehmann Estimate is a value representing the differential between two data sets. It is calculated by finding the average difference between all the values in data set A and all the values in data set B. If there are m values in data set A and n values in data set B, then $m*n$ values are calculated. The Hodges-Lehmann Estimate is the median of these values [42]. For example, if the precision of system A was compared with the precision of system B and the analysis is: Hodges-Lehmann Estimate of 20%, with confidence intervals of 15% to 25% at 95% confidence level. This result can be interpreted as: the average difference in precision is a 20% greater result for system B, although that difference is likely to vary between 15% and 25%. Note that all the statistics used to analyse the data are non-parametric. That is, they make no assumptions of a normal distribution in the data, although should a normal distribution be present, these tests will still give correct results.

3.5 Summary

The concept of context-aware recommender system has been proposed and studied for several years [4]. However, as Adomavicius and Tuzhilin stated:

Most of the work on context-aware recommender systems has been conceptual, where a certain method has been developed, tested on some (often limited) data, and shown to perform well in comparison to certain benchmarks [5].

The key issue here is the lack of contextual data.

In this chapter, we focused on context-aware recommender systems. We introduced existing methods for making context-aware recommendations: how they obtain contextual information, how they tackle data sparsity or lack of data. Then we identified the limitations of existing CARS:

- All the possible contexts used in CARS are non-personalised. This can lead to inaccurate recommendations
- Existing CARS can not deal with data sparsity. They can not generate accurate recommendations on sparse data.
- Existing session based CARS can not achieve accurate user modelling and accurate context modelling at the same time.

These limitations correspond to the three research questions that we have described in Chapter 1.

The limitations of CARS motivates for better context-aware recommendation methods. In the following chapters, we propose new methods to improve existing context-aware recommender systems.

Chapter 4

Probabilistic Contextual Filtering

In Chapter 3, we reviewed the field of context-aware recommender systems. Several limitations of existing CARS have been identified. In this chapter, we deal with two of them: data sparsity and non-personalised contexts. We propose a novel context-aware recommendation method in this chapter. Based on implicit data, we analyse and model each user's periodic action patterns, and then define personalised temporal contexts based on these patterns. Our method can avoid context generalisation by assigning "weight" to each user action in each context, so that we do not need to filter out the data collected in different contexts. Consequently, our method can significantly improve recommendation accuracy on sparse datasets.

4.1 Introduction

Recommender systems can rely on different types of input [4]: explicit data and implicit data. Most accurate is the high quality explicit feedback, which includes explicit input by users regarding their preference for items. For example, Amazon[1] usually asks users to rate the product they recently purchased; Netflix[2] collects star ratings for films; TiVo [3] users indicate their preferences for TV shows by hitting thumbs-up/down buttons. The user has to rate a sufficient number of items before a recommender system can capture the user's preferences and present the user with reliable recommendations [4]. Once these initial ratings are obtained, a traditional recommender system tries to estimate the rating function R

$$R : User \times Item \rightarrow Rating$$

Since the users are usually reluctant to provide ratings, the rating dataset is sparse [49]. In the field of traditional recommender systems, this is usually solved by applying collaborative methods with latent factors [4]. In context-aware recommender systems, the rating function is extended into three dimensions

$$R : User \times Item \times Context \rightarrow Rating$$

For example, in traditional music recommender systems, the user only needs to provide one general rating for the track that the user has played. For instances, let us assume the music RS is using binary rating system, like = 1 and dislike = 0. If user x likes track a, then we have $(x,\ a) = 1$. However, to make context-aware recommendations, ideally, we would like to ask the user to rate the track for all the possible contexts, e.g. whether the track is suitable to play before sleeping, or in a party. Let $context_1$ = before sleeping, $context_2$ = in a party. We would like to know the rating for the following:

$(x, a, context_1)$,

$(x, a, context_2)$

Since the users are usually reluctant to provide ratings, the rating dataset becomes so sparse that collaborative methods can not make accurate recom-

[1]http://www.amazon.co.uk/
[2]https://www.netflix.com/
[3]https://www.tivo.com/

mendations and for some users, can not make recommendations at all [5].

A we discussed in Section 3.2, to deal with data sparsity in CARS, context generalisation [2] is commonly used: using a more general context instead of the exact context. For example, the temporal context can be changed from “Saturday 8pm” to “evening” to obtain more training data. However, using a generalised context means using data collected in different context, in which the user may have different preference. In cases of applications with highly granular contexts, there may exist a very large number of possible context generalisations. It is not possible to empirically evaluate the predictive performance on each generalised contexts due to high computational cost [5]. Identifying the suitable generalisation becomes very difficult. That is, there is no consistent and optimal way to find the suitable generalisation. These limitations indicates that context generalisation might not be the best way to deal with data sparsity. Therefore, in Chapter 1, we presented our *first research question: whether we can develop a CARS that is able to overcome data sparsity, in order to provide more accurate recommendations for more users, compared to existing CARS.*

In context-aware recommender systems, all the temporal contextualisation method usually use pre-defined temporal contexts [11]. As we discussed in Chapter 1, in music recommender systems, it is common to define a set of temporal contexts based on common sense, such as morning, afternoon and evening, and the boundaries between these temporal contexts are usually the same for all the users. However, some users may start the day much earlier or later than others. That is, different users may have totally different preferences in the same pre-defined context. More importantly, for some users, these pre-defined temporal contexts can be too general. For example, a user listens to rock & roll every morning to wake him up; when he is working in the morning, he listens to classical music. The temporal context “morning” becomes meaningless, because it covers such a long period of the day. In the same context “morning” the user’s preference still changes significantly. However, in this case it is almost impossible to solve this problem if we use the same set of temporal contexts for all the users, and do not analyse individual user's listening habits. Therefore, it is necessary to discover each user's listening (or other action) patterns and frequencies, and define personalised temporal contexts based on these patterns. Therefore, in Chapter 1 we presented *the second research question: whether we can develop a CARS that is able to define personalised contexts based on users' activity patterns, so that the system can achieve higher recommendation accuracy than existing CARS.*

In this chapter, we propose a novel contextual filtering recommendations technique, Probabilistic Contextual Filtering. Based on implicit data, we analyse and model each user's periodic action patterns based on their listening frequencies at different times of the day. That is, instead of explicit data such as rating, we use log-like implicit data, which can indicate at what time the user is active, which item the user chooses to consume, the frequency of actions (consumptions). We then define personalised temporal contexts based on these identified patterns, classify user actions into these personalised contexts. We calculate the probability that the user action is performed in each personalised user context, and use this probability as the weight of the action in that specific context. Here, the given user's action can be listening to a certain track, or purchasing a certain product.

Our method follows the idea of pre-filtering [5]. That is, the contextual (temporal in our case) information is used to build the contextualised dataset for RS. Our method is different from those contextualisation methods introduced Chapter 2, because we never filter out any data. We don't define a unified set of contexts. Instead, for each individual user, we define personalised temporal contexts as a Gaussian distribution. Therefore all the temporal contexts overlap with each other in our method.

By using probability as weight, our method does not need to generalise the given context, thereby avoiding the limitation of context generalisation. Based on personalised temporal contexts, our method can significantly improve recommendation accuracy on sparse dataset, and provide recommendations for more users, compared to contextual pre-filtering method and traditional collaborative method.

The remainder of this chapter is structured as follows: In Section 4.2 the algorithm of our probabilistic filtering is presented. Section 4.3 describes the testing criteria for context-aware recommender systems. In Section 4.4, experiments results using our probabilistic filtering are given. Final comments are provided in Section 4.5.

4.2 Probabilistic Contextual Filtering

Our Probabilistic Contextual Filtering (PCF) method takes three steps:

1. Capture the user's action patterns based on frequency of actions. The user's action here can be listening to a certain track, watching a certain TV show, or purchasing a certain product. We would like to capture user's temporal patterns, e.g. at round what time of the day the user often listens to music, or at which day of the week the user tends to go to the supermarket. Based on these patterns, we can define personalised temporal contexts.

2. For each user context, build a user-item utility dataset, based on the action frequency patterns. obtained in the first step, for each of the user's contexts. Here, the user-item utility indicate the usefulness of an item to a user. By building a new dataset for each context, we contextualised the data [5].

3. Predict the utility of given user-item pairs. The predicted utility of an item for the given user at a specific time is the weighted sum of its predicted utilities in all contexts. Since we have generate a contextualise dataset for each user context, the predicted utility in a specific context can be computed by a traditional RS.

In the following sections, we describe all of the three steps in detail.

4.2.1 Modelling Action Patterns

In this section, we describe the first step our method.

For explicit data, let r_{ij}^k be the rating provided by user u_i on item w_j in context c_k, where high values mean stronger preference. For implicit data, we use p_{ij}^k to indicate user-item interaction count in context c_k. For example, in music recommender systems, p_{ij}^k is how many times user u_i has played track w_j in context c_k. In this step we use implicit data to model user's action patterns.

Although user's preference might be different in distinct temporal contexts, it can have a periodic pattern [5]. For example, in music recommendation, a user always listens to track w_i while driving to and from work. Therefore, his preference on playing track w_i would have a daily repetition. If we divide the day into 48 non-overlapped periods with the same length (30 minutes), and let p_{ij}^t be the action count of user u_i on item w_j during period t, the value of p_{ij}^t would show a daily pattern: two peaks everyday, one from 8:30am to 9:00am and the other from 5:00pm to 5:30pm, when he is driving to and from work respectively. In this case, based on p_{ij}^t values, we can infer that there might be two meaningful contexts when p_{ij}^t reaches its peaks. Therefore, we would like to focus on predicting the user's preference in such contexts first, instead of predicting this user preferences in each possible temporal context blindly.

Inspired by this, we consider that for each user u_i, the p_{ij}^t values of some items can be used to infer the periodic pattern of the given user' preference. However, inference based on certain items may be problematic. First, it is unreliable. We may obtain different periodic patterns based on different items. Second, no items can be suitable for all contexts. Therefore, it is possible that no item appears in all the contexts. For example, it is likely that a track frequently played in the morning has never been played in the evening. Therefore, we can not discover all the contexts based on a certain item. In our system, instead of p_{ij}^t, the inference is based on p_i^t, the total action counts of user u_i on all items in t. The sharp rises and falls of p_i^t could indicate the boundaries of between contexts.

In our method, we are making the assumption that the total action count of a user would show some fluctuations throughout the day. For every user, we assume that each major peak value of p_i^t is the result of a distinct temporal context. However, to reliably capture user's periodic patterns, we can not rely on the action counts in only one temporal cycle. We would like to study the distribution of user actions in different temporal contexts, rather than counting the user actions in individual short period of time. To make the our model more accurate, we use a continuous variable t to represent the time, instead of dividing each temporal cycle into small periods. Therefore, the number of actions of user u_i is now rewritten as $p_i(t)$, a distribution over time. By using continuous time variable, we no longer need to set boundaries between contexts. Given a specific time point, we can compute the probability that this time point is in a given temporal contexts.

Note that the way we identify meaningful context is very different from context generalisation. Context generalisation expands a pre-defined narrow context to a more general one so that in this general context the RS has enough data to learn the user's preference; while in our case, we model user's action frequency as a distribution, so we can focus on the context in which the user tends to be more active.

The periodic pattern of $p_i(t)$ of each user can be modelled by a mixture of K periodic Gaussians, clustering all actions of user u_i into K overlapped contexts. We select Gaussian mixture model for its simplicity. The statistics of a continuous and periodic variable can be captured either by distributions which are directly defined on the unit circle, like the von Mises distribution, or by "wrapping" the probability density function (pdf) of linear variable t to the circumference of the unit circle [45]. The new distributions obtained by "wrapping" linear Gaussian are called wrapped Gaussian distributions. In fact, the von Mises and the wrapped Gaussian distribution are very similar

[6]. In our method, we use wrapped Gaussian distribution for its simplicity in approximation. The wrapped probability density function $p^W(\theta)$ of the wrapped variable $\theta = \frac{t\times 2\pi}{T} \bmod 2\pi$, defined in the interval $(-\pi, \pi]$, can be generally obtained by tiling different pdfs shifted by multiple of 2π [19, 46]. That is, we combine infinite linear Gaussian models as a new model. The centre of each Gaussian is shifted by 2π. Here, T represents the period of the periodic variable in specific applications. For example, if we are interested in users' daily action patterns, the period of the periodic distribution should be a day. If we are interested in users' weekly or monthly action patterns, the period of the periodic distribution should be a week or a month.

In our model, the distribution of action count in each personalised temporal context is modelled by a wrapped Gaussian distribution (WG). It can be written as follows [19]:

$$\begin{aligned}\mathcal{N}^W(\theta|\theta_0,\sigma) &= \sum_{w=-\infty}^{\infty} \mathcal{N}(\theta - w2\pi|\theta_0,\sigma^2)\\ &= \sum_{w=-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta-w2\pi-\theta_0)^2}{2\sigma^2}}.\end{aligned} \tag{4.1}$$

$\mathcal{N}^W(\theta|\theta_0,\sigma)$ is unimodal with a single local maximum (action count peak) and symmetric about θ_0, with variance σ^2. It is constructed by infinite wrappings of the linear Gaussian pdf inside the interval $(-\pi, \pi]$ [19].

Since we assume there are K major peaks in T, so there are K overlapping contexts. A mixture of WG (MoWG) on a periodic variable θ_0 can be derived as follows [19][4]

[4]In [19], $\boldsymbol{\pi}$ and π_k are used instead of $\boldsymbol{\mu}$ and μ_k

$$\begin{aligned} MoWG(\theta|\boldsymbol{\mu}, \boldsymbol{\theta_0}, \boldsymbol{\sigma}) &= \sum_{k=1}^{K} \mu_k \mathcal{N}^W(\theta|\theta_{0,k}, \sigma_k) \\ &= \sum_{k=1}^{K} \sum_{w=-\infty}^{\infty} \mu_k \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(\theta - w2\pi - \theta_{0,k})^2}{2\sigma_k^2}} \end{aligned} \tag{4.2}$$

where K is the number temporal contexts in a cycle, thus the number of mixture's components. μ_k is the weight of the kth component. Therefore, in each temporal cycle, the total action count is modelled as the weighted sum over action counts in K temporal contexts. Note that, in our model there is no clear boundaries between components. That is, we assume the all the contexts in a temporal cycle overlap with each other.

The approximation of MoWG include infinite summations [19]. However, in practice, a summation over ± 2 tilings provides a sufficient approximation even for large variances ($\sigma^2 \leqslant 2\pi$), because in that case, the pdf of the truncated Gaussian closely approximates the pdf of the unconstrained Gaussian [19]. Bahlmann demonstrated, the approximation of WG by the most meaningful cycle (wrap) can be used if the variance is small enough ($\sigma^2 < 1$) [9]. Based on Bahlmann's approximation, we can define $\mathcal{N}^{AW}$, the approximated wrapped Gaussian as follows [19]

$$\mathcal{N}^{AW}(\theta|\theta_0, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(\theta - w2\pi - \theta_0)^2}{2\sigma^2}}. \tag{4.3}$$

By applying Bahlmann's approximation, MoWG can be replaced by the mixture of approximated wrapped Gaussian (MoAWG). Therefore we obtain the following [19][5]

[5]In [19], $\boldsymbol{\pi}$ and π_k are used instead of $\boldsymbol{\mu}$ and μ_k

$$\begin{aligned} MoAWG(\theta|\boldsymbol{\mu}, \boldsymbol{\theta_0}, \boldsymbol{\sigma}) &= \sum_{k=1}^{K} \mu_k \mathcal{N}^{AW}(\theta|\theta_{0,k}, \sigma_k^2) \\ &= \sum_{k=1}^{K} \mu_k \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{((\theta-\theta_{0,k}) \bmod 2\pi)^2}{2\sigma_k^2}}. \end{aligned} \tag{4.4}$$

where the operation "mod" represents the remainder of the division. K is the number temporal contexts in a cycle, thus the number of mixture's components as MoWG, and μ_k is the weight of the kth component. Parameters of MoAWG can then be estimated by Expectation-Maximisation (EM) algorithm [26].

EM is an iterative method for finding maximum likelihood of parameters in statistical models, where the model depends on latent variables [14]. Given a Gaussian mixture model, EM maximises the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients). EM initialises the means, covariances of the components and the mixing coefficients randomly. In expectation step, EM evaluates the responsibilities using the current parameter values. In maximisation step, EM re-estimates the parameters using the current responsibilities [14]. After each iteration, EM checks for convergence of either the parameters or the log likelihood. If not converged, EM return to expectation step. The two steps are iterated until convergence [26].

In MoAWG, the basic steps of EM can be summarized as follows [19]:

Expectation Step: The responsibility γ_{xk} of component k for an action x observed at time θ_x, can be estimated using the parameter values of the previous iteration (randomly initialized for the first iteration) as follows:

$$\gamma_{xk} = \frac{\mu_k \mathcal{N}^{AW}(\theta_x|\theta_{0,k}, \sigma_k^2)}{\sum_{y=1}^{K} \mu_y \mathcal{N}^{AW}(\theta_x|\theta_{0,y}, \sigma_y^2)} \tag{4.5}$$

Maximization Step: The M step estimates the new values of the mixture's parameters. They can be estimated for each component k as

$$\mu_k = \frac{1}{n}\sum_{x=1}^{n}\gamma_{xk} \tag{4.6}$$

$$\theta_{0,k} = \arctan\frac{\sum\limits_{x=1}^{n}\gamma_{xk}\sin\theta_x}{\sum\limits_{x=1}^{n}\gamma_{xk}\cos\theta_x} \tag{4.7}$$

$$\sigma_k^2 = \frac{\sum\limits_{x=1}^{n}\gamma_{xk}((\theta_x - \theta_{0,k}) \bmod 2\pi)^2}{\sum\limits_{x=1}^{n}\gamma_{xk}}. \tag{4.8}$$

The two steps are iterated until convergence or until a given number of iterations is reached. Therefore, we obtain a mixture of K Gaussian model, representing K periodic temporal contexts. Given a user action, the probability that the action is performed in a certain context (responsibility of the context) can be computed according to Equation 4.5.

4.2.2 Constructing Datasets

This section describe the second step of our method.

So far, we have been able to model each user's periodic action patterns. Now for each user, we would like to generate a dataset for each of his contexts, based on the MoAWG obtained in the previous step. That is, we would like to build a contextualised dataset as in pre-filtering, for each of the user's contexts. Specifically, we would like to construct a user-item utility matrix for each context of the given user. If a user has never interacted with a specific item, the corresponding user-item utilities in all context are set to zero. If the user has interacted with the item, the user-item utility in each dataset should be able to reflect the corresponding user-item action count.

Formally, we would like to compute user-item utility, A^k_{ij} for user u_i and the item w_j, in kth context first. Note that in this step, we only compute this value if the user has interacted with the given item, otherwise we set it to zero. Let x be an action of user u_i on item w_j at time θ_x. We can calculate γ_{xk}, the probability that x is an action in the kth context of user u_i by Equation 4.5. Let us view this action as K different micro-actions: each in a different context, with a different action count. For each of these micro-actions, we assume the user interacted with the item γ_{xk} time, where k is the corresponding context index. That is, the action count of each micro-action is γ_{xk}. Note that $\gamma_{xk} > 0$ for each context, and $\sum_{k=1}^{K} \gamma_{xk} = 1$. That is, the sum of all the micro-action counts of the same action is one. Here we assume each action is a mixture of micro-action from a different context, with responsibility γ_{xk}.

For dataset d_k, the dataset representing the kth context of user u_i, summing over all the corresponding micro-action counts γ_{xk} of item w_j by user u_i, we obtain A^k_{ij}, the user-item utility of item w_j for user u_i:

$$A^k_{ij} = \sum_{x} \gamma_{xk} \tag{4.9}$$

To generate a user-item utility matrix for user u_i, we also need to compute the utilities of items for other users. This can also be done by Equation 4.5 and 4.9. The only difference is, for action x that is not of user u_i, γ_{xk} is still calculated based on the mixture model of user u_i, since we are building this dataset for user u_i. That is, we are computing the probabilities that the action happens in the contexts of user u_i, even if it is not an action of user u_i.

We compute the user-item utility for all the user-item pairs in all K datasets of user u_i, based on his Gaussian mixture model. Therefore, we obtain K contextualised datasets for user u_i, each based on one of his periodic contexts. Similarly, we can do this for all the users in the system.

4.2.3 Predicting Preferences

Although we have obtained K user-item utility datasets for each user, these datasets can not be directly used by most traditional collaborative recommendation methods, since most of these methods are designed to work on explicit data, such as user-item ratings, while our user-item utilities are based on interaction counts. Some mapping approaches need to be applied to the our datasets, in order to obtain ratings from user-item utilities

$$A_{ij} \to r_{ij}$$

Since most of the users have only interacted with a relatively small number of items, there are many zero-valued ratings in our user-item rating sets. In most applications, we would like to predict the user's preferences on items they have never interacted with. Therefore, after the mapping procedure, a traditional collaborative recommendation method is trained on each dataset to predict each user's ratings on items the user has not interacted with.

Formally, we would like to recommender user u_i some items at time θ_x. Specifically, we would like to know the rating user u_i would give on item w_j, which the user has never interacted with. We apply a traditional collaborative method, such as matrix factorisation, on all K datasets of user u_i. Thus, we obtain K unique predicted ratings, r_{ij}^0 to r_{ij}^K, one from each context. At time θ_x, we can calculate the responsibility of each context γ_{xk}, based on the MoAWG of user u_i by Equation 4.5. We compute the final predicted rating as a weighted sum:

$$r_{ij} = \sum_{k=1}^{K} \gamma_{xk} r_{ij}^k \tag{4.10}$$

In our method, we generate K datasets for each user. Storing these dataset can take large disk space. Modelling users' action patterns and predicting their preferences in all K contexts can be computationally expensive. However, the action modelling step can be done in parallel, and more importantly, in advance. The recommender system can keep updating the parameters of MoAWG, and the value of r_{ij}^k, the predicted rating of user u_i on item w_j in the kth context. When recommendations are requested for time θ_x, the predicted user rating r_{ij} can be calculated as in Equation (4.10). Note in this equation, we only need to calculate γ_{xk}. All the r_{ij}^k has be calculated in advance.

4.3 Experimental Method

4.3.1 Data for Testing PCF

We have obtained two implicit usage datasets: music listening dataset from last.fm[6] and retail transaction dataset from Ta-Feng warehouse. Based on these two datasets, we study the two commonest temporal patterns: daily and weekly patterns respectively.

Music Data

The first dataset we are using was collected from the music website, last.fm. This dataset is available to public[7]. It has been used in several published work [21]. The dataset contains the listening histories of about 1,000 users. Each time when a user listened to a track, a new entry was generated. Each entry includes the user ID, track name, MusicBrainz[8] track ID, artist name, MusicBrainz artist ID and an appropriate timestamp. In this dataset, there are 19,121,228 entries in total, from 992 users, on 176,948 artists.

For simplicity, we only use the tracks that have both MusicBrainz track ID and artist ID. So we clean the data by removing the tracks and artists that have no MusicBrainz ID. After remove tracks and artists without MusicBrainz ID, the dataset still contains 16,984,430 entries (less than 12% of the entries

[6]http://www.last.fm/
[7]http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html
[8]http://musicbrainz.org/

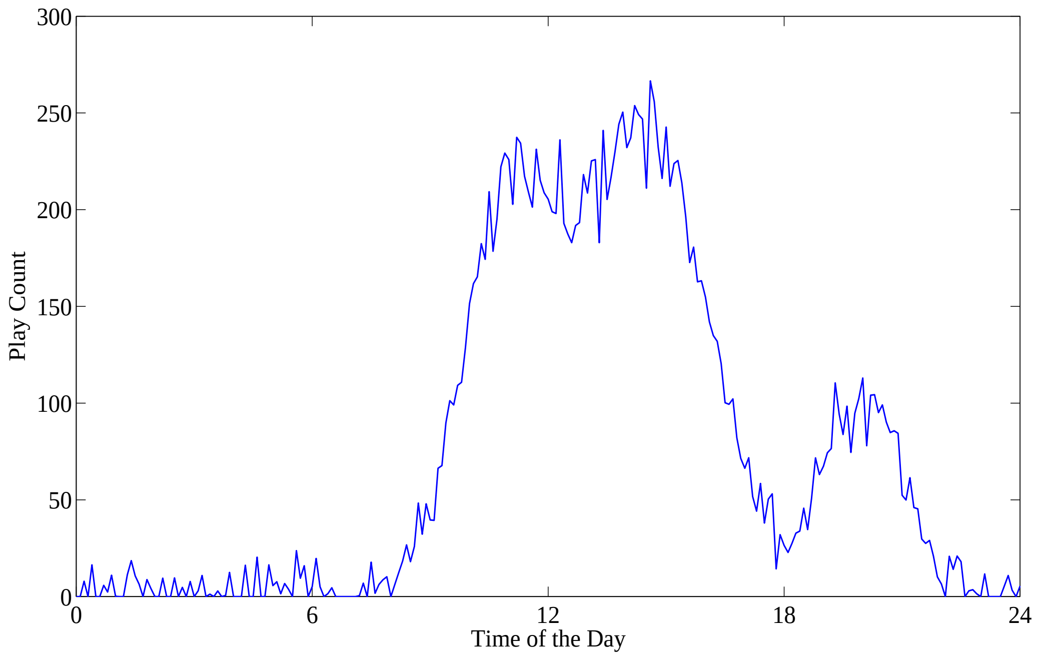

Fig. 4.1 A typical user's total play count at different times of the day

were removed). For each user, we have about 17,000 entries on average.

We need to infer which tracks the user likes based on this implicit dataset. Therefore repeated playing is important. By observing the user's behaviour, calculating the play count of each track, we can infer which tracks the user probably likes more and thus prefers to play. However, people tend to spend most of their time listening to their favourite tracks. Therefore, most tracks that a user has played have very low play counts (usually only once or twice). For simplicity, we assume tracks from the same artist have similar style. Instead of recommending tracks, we recommend artists to the users. Therefore, the dataset become less sparse, and each item (artist) has more action counts.

In this dataset, we find that many users' action counts show very evident daily repetitions. For example, a user tends to listen to music from around 11am to 3pm and at around 8pm everyday (Figure 4.1). In our work, this dataset is used to study users' daily action patterns.

Retail Data

The second dataset we are using is the publicly available Ta-Feng transaction dataset. It is available from ACM Recsys wiki[9]. This dataset has also been used in published work [28]. Ta-Feng is a membership retailer warehouse that sells a wide range of merchandise, from food and grocery to office supplies. This dataset contains transactions data between November 2000 and February 2001 (four months). It contains 119,578 transactions, totalling about 32,000 customers and 24,000 items. Each record in this dataset consists of the following attributes: 1. Transaction date and time; 2. Customer (user) ID; 3. Product (item) ID. On average, a customer purchases 6.8 items in each transaction [32]. This dataset is very sparse.

In this dataset, we find that many customers have weekly shopping patterns. For example, a user purchases many items in one transaction every weekend, while during weekdays the user occasionally goes to the store, and usually purchases only one or two items in one transaction (Figure 4.2 and Figure 4.3). In our work, users' weekly action patterns are studied based on this dataset.

4.3.2 Testing and Analysis Criteria

As we discussed in Chapter 2, most popular recommender systems are based on explicit data. The criteria for evaluating those systems may not be suitable for systems based on implicit data. We select the following methods to evaluate our methods and the baseline approaches.

Relevance Prediction Accuracy

With implicit data, by observing the user's behaviour, we can infer which items the user finds relevant. For example, if the user repeatedly plays a track, the user must like it. However, it is hard to reliably infer which items the user does

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

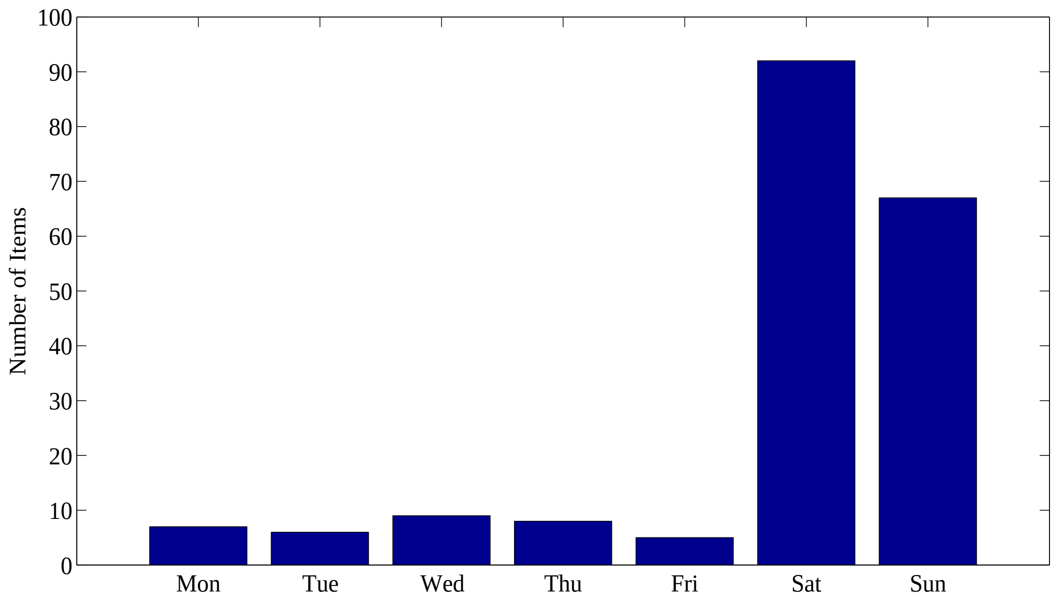

Fig. 4.2 The number of items a typical user purchased on different days of the week

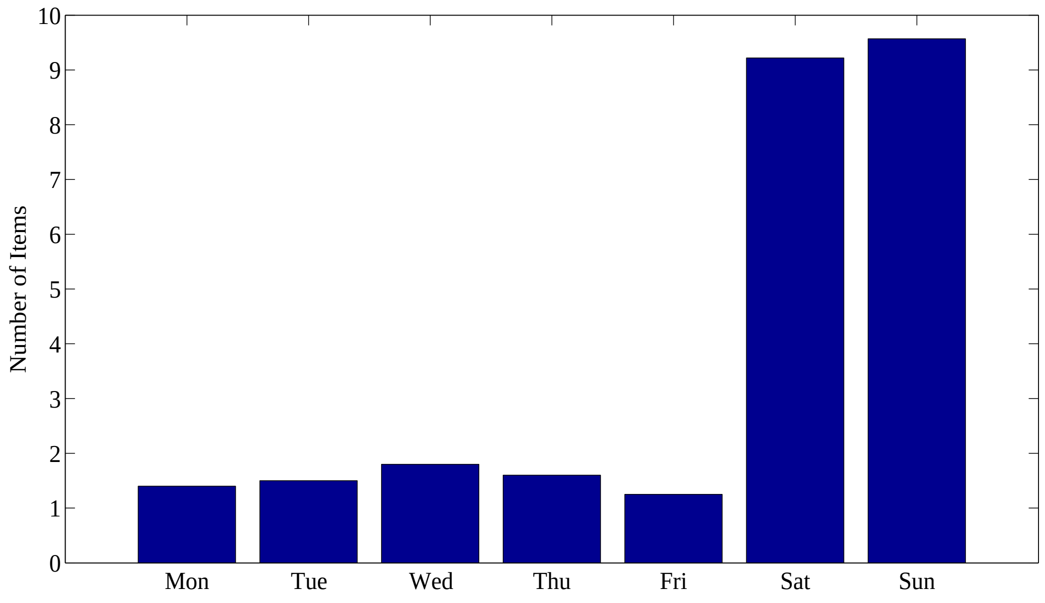

Fig. 4.3 The average number of items a typical user purchased in one transaction

not like or finds irrelevant from the smaller values of implicit feedbacks [33]. In some cases, this can be done by randomly selecting the items the given user has never interacted with as irrelevant items. This method can only be used when the dataset contains a significant number of items, and each user is only interested in a relatively small amount of items. For example, in the last.fm music dataset, there are more than 160,000 artists, and usually the number of artist a user has interacted with is less than 400. The probability of an artist the user has never interacted with being interesting to the user is very small. By using this method, we can generate test sets containing both relevant and irrelevant items.

Precision measures the proportion of recommended items that are in fact relevant. Precision can tell us how many irrelevant items are recommended to the user. In our experiments, we use precision at N to evaluate the prediction accuracy of our RS.

User Space Coverage

In context-aware recommendations, only the data collected in the given context should be used for training. In most contextualisation methods, data not collected in the given context is filtered out. This filtering step tends to make the dataset sparser, leading to poor coverage of users. Therefore, it is important to compare the coverage of CARS.

Significance

The mixture of Gaussian models are non-deterministic. The final positions of components in the mixture model depend in part on randomness for the starting positions during training.

In order to gain the representative results, we repeat each experiment involving non-deterministic methods 100 times and take the median of the performance values. The results are analysed by the combined use of the Hodges-Lehmann Estimate with associated non-parametric confidence intervals at 95%.

4.3.3 Testing Method of PCF

Test Sets

To evaluate our method on music data, we randomly select 200 users for testing (20% for testing). The rest of the users' playing histories are fully observed. For each of the selected users, we generate two testing sets to simulate two different contexts: one at 10am, one at 8pm. To generate relatively more reliable testing sets, we use a one-hour window: in the first context, tracks played between 9:30am to 10:30am will be used for testing, and 7:30pm to 8:30pm for the second context. In each context for each user, we randomly select 10 artists whose tracks have been played at least p_{min} times in this temporal context as relevant artists, and 10 artists whose tracks have never been played in this temporal context as irrelevant artists. So that we have a balanced test set. p_{min} is set to 5, so that each of the selected user has more than 10 relevant artists. The rest of the data (including the data of the users that are not selected for testing) is used for training. The recommender systems will predict the user's preference by ranking these 20 candidates from the most preferred to the least preferred artist. If user u_i played tracks of artist w_j in a testing context, the play count of artist w_j by user u_i is set to zero in the corresponding training set. Therefore, the recommender systems need to predict the user's preference on "new" items, rather than the items the user plays frequently in other different contexts.

A similar approach is used on retail data. 8000 customer are selected randomly for testing. Two testing sets are generated for each testing user: Wednesday shopping set and Saturday shopping set. So we have one testing context for weekdays, one for weekends. In each context for each user, 5 items that the user has purchased are randomly selected as relevant items, and 5 items that the user has never purchased in this context are selected as irrelevant items. The recommender systems will provide a ranked list of these 10 candidates, ordered by decreasing preference. If user u_i purchased item w_j in a testing context, the purchase count of item w_j by user u_i is set to zero in the corresponding training set. Therefore, the recommender systems need to predict the user's preference on "new" items, rather than the items the user purchased frequently in other different contexts.

Baseline Approaches

When evaluating the precisions of our system, a matrix factorisation method [27] with 100 factors is used as baseline approach due to its ability to generate accurate predictions on sparse datasets. In our experiment, MF with more than 100 factors no longer increase prediction accuracy. This is a traditional recommendation method, which does not take into consideration any contextual information. Besides matrix factorisation, we have also implemented the pre-filtering method called micro-profiling (MP) [11], which splits the user profile into morning and evening on music data, weekday and weekend on retail data, a typical pre-filtering method. Matrix factorisation is also used in micro-profiling and our probabilistic contextual filtering. That is, after the contextual filtering step (either by our method or micro-profiling), the traditional recommendation method, matrix factorisation is employed to predict the user's preference. For both datasets, we will compare the precision when recommending only one item (precision at 1), and the precision when recommending the first half of items (precision at 10 on music data and precision at 5 on retail data). In our experiments, we use the implementation of MF from Apache Spark[10].

Matrix factorisation is able to predict user's preference on sparse dataset (though prediction accuracy usually drops given less data). However, not every traditional recommendation method has the ability to work on sparse data. When comparing the user space coverage of different systems, the item based Pearson correlation method [54] is used as the traditional recommendation method of the contextual filtering systems. Compared to matrix factorisation,

[10]http://spark.apache.org/

	$K = 2$	$K = 3$	$K = 4$	$K = 5$
Precision at 10 on Music Data	79.2%	85.0%	86.2%	86.9%
Precision at 5 on Retail Data	72.5%	73.1%	73.4%	73.6%

Table 4.1 Precision of Probabilistic Contextual Filtering as K increases

it is intuitive, easy to implement and its training takes considerably less time. However, this method also requires more training data to predict the user's preference. In our experiment, we compare the user space coverage of our probabilistic contextual filtering and micro-profiling, both using item based Pearson correlation as the traditional recommendation method.

Machine Used

In our experiments, we use a laptop with Intel 2.6 GHz dual core processor and 4GB of RAM.

4.4 Results and Analysis of PCF

In this section, we present the experimental results on probabilistic contextual filtering, and the analysis based on these results.

4.4.1 Number of Gaussians

Before we compare the probabilistic contextual filtering to other methods, we would like to find the optimal value of K, the number of Gaussians used in the mixture model MoAWG. On both datasets, we are comparing the precision at the first half (10 on music data and 5 on retail data) as K increases.

As we can see from Table 4.1, the precision increases as the mixture model consists of more Gaussians. However, the precision increases very slowly after K is greater than 3 on music data, because for most users, there are no more than 3 different contexts in terms of music listening. On retail data, when $K = 2$, the model can already cover most users' shopping patterns. The greater

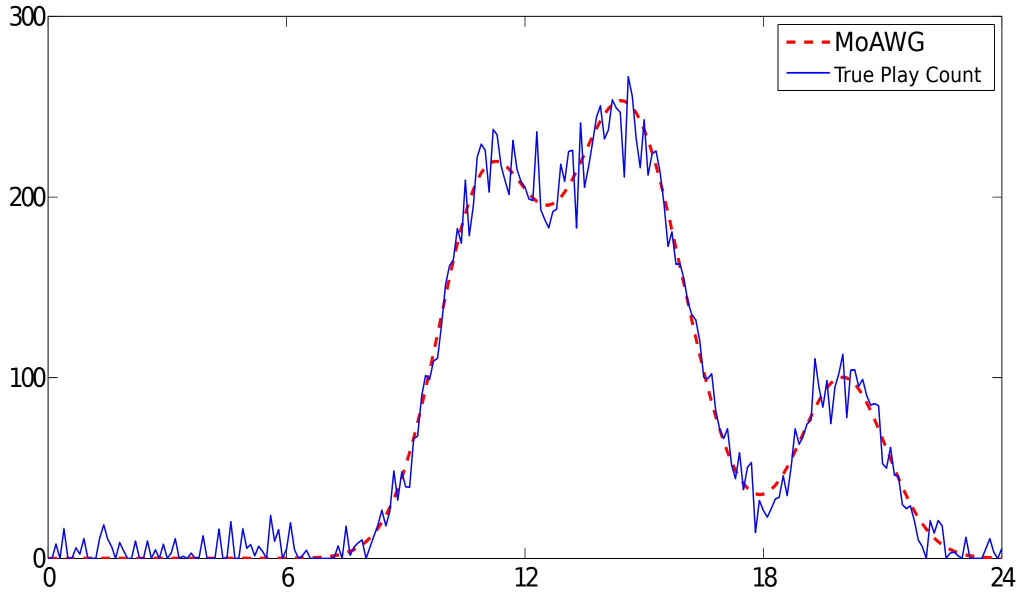

Fig. 4.4 A user's play count modelled by MoAWG.

K is, the longer it takes to train the model. In the following experiments, we use the mixture of 3 Gaussians on music data (an example is presented in Figure 4.4) and the mixture of 2 Gaussians on retail data since bigger K does not lead to significantly higher precision but considerably increases training time.

4.4.2 Comparison with Traditional Recommendation Method

In this section, we compare our PCF with traditional recommendation matrix factorisation.

Hypothesis

$H4.1_0$: There will be no difference in precision between the traditional recommendation method, matrix factorisation (MF) and the combination of probabilistic contextual filtering (PCF) and matrix factorisation (PCF + MF).

Results

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
MF	71.3%	84.6%	65.7%	72.9%
PCF + MF	85.0%	93.9%	72.5%	76.0%

Table 4.2 Precisions of MF and PCF + MF

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
Can the hypothesis be rejected at the 95% confidence interval	Yes	Yes	Yes	Yes
Hodges-Lehmann Estimate	13.7%	9.3%	6.8%	3.1%
Confidence Interval at 95%	(13.1%, 13.9%)	(9.2%, 9.3%)	(6.8%, 6.8%)	(3.1%, 3.1%)

Table 4.3 Summary of analysis between MF and PCF + MF

Table 4.2 shows the precisions of the system using only MF and the system using PCF + MF on both datasets. Table 4.3 shows the results of the Mann–Whitney–Wilcoxon non-parametric test on the distributions of the results from both recommender systems.

The probabilistic contextual filtering substantially increases the precision at 1 on music data by 9.3%, compared to the traditional method, matrix factorisation. Similar results are found on retail data. The system using PCF + MF achieves considerably higher precision at 1 than only using MF on the retail data. On both datasets, when more items are recommended to the user (precision at 10 and at 5) the combination of PCF and MF can find much more relevant items than MF alone. The statistical significance of the results is shown in Table 4.3. On both datasets, using both criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using PCF + MF instead of MF (or vice versa).

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
MF	71.3%	84.6%	65.7%	72.9%
MP + MF	76.3%	90.4%	63.0%	72.7%
PCF + MF	85.0%	93.9%	72.5%	76.0%

Table 4.4 Precisions of MF, MP + MF and PCF + MF

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
Can the hypothesis be rejected at the 95% confidence interval	Yes	Yes	Yes	Yes
Hodges-Lehmann Estimate	8.6%	3.5%	9.5%	3.3%
Confidence Interval at 95%	(8.0%, 9.1%)	(3.3%, 3.6%)	(9.5%, 9.5%)	(3.3%, 3.3%)

Table 4.5 Summary of analysis between MP+ MF and PCF + MF

4.4.3 Comparison with Traditional Contextual Filtering

In this section, we compare our PCF with traditional contextual filtering method.

Hypothesis

$H4.2_0$: There will be no difference in precisions between the micro-profiling (MP) and the probabilistic contextual filtering (PCF), both combined with matrix factorisation (MF).

Results

The results form these experiments are presented in Table 4.4. The results of the system using only the traditional recommendation method MF is also presented in Table 4.4. The statistical analysis between the two contextual filtering methods is given in Table 4.5.

On music data, the combination of micro-profiling and matrix factorisation can achieve higher precision compared to the system only employing matrix factorisation, recommending either 1 or 10 items. However, the precision of micro-profiling is still considerably lower compared to probabilistic contextual filtering. Especially when measured by precision at 10, PCF + MF outperforms MP + MF by 8.6%. These results demonstrate that PCF can predict the user's preference more accurately compared to MP.

On music data, we recommend artists instead of tracks. Therefore, there is more data for each item (artist). Compared to music data, the retail data is far more sparse. In micro-profiling, the retail dataset is further divided into weekday set and weekend set, making the data even more sparse. And data sparsity leads to poor recommendation. As we can see from Table 4.4, applying MP + MF results in even lower precisions compared to the system using only MF. The data sparsity cancels out the advantage of context-aware recommender system.

The statistical significance of the results is shown in table 4.5. On both datasets, using either criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using MP + MF instead of PCF + MF (or vice versa).

Besides precision, the recommender system's ability to deal with sparse data can also be measured by user space coverage.

Hypothesis

$H4.3_0$: There will be no difference in user space coverage between the micro-profiling (MP) and the probabilistic contextual filtering (PCF), both using item based Pearson correlation as the traditional recommendation method.

Results

The results of the experiment are presented in Table 4.6. In most applications, there are always (new) users who have not interact with the system sufficiently, or (new) items that have not been consumed by sufficient number of users.

	On Music Data	On Retail Data
PCF	92.7%	66.0%
MP	83.6%	41.2%

Table 4.6 User space coverage of PCF and MP

Therefore, in most cases, the coverage can never reach 100%. On music data, the coverage of PCF reaches 92.7%, outperforming MP by 9.1%. Since the retail data is more sparse, both systems have relatively low coverage on this dataset. However, PCF still outperforms MP by 24.8%, covering about two thirds of the users.

The user space coverage is deterministic, despite the initialisation of PCF. Therefore, the null hypothesis can be rejected. There is a statistically significant difference in coverage, when using MP instead of PCF (or vice versa).

4.4.4 Discussion

In Section 4.4.1, we observed that the recommendation accuracy increases as the mixture model consists of more Gaussians. However, the accuracy increases very slowly after K, the number of Gaussian, is greater than a certain value. On different datasets, the optimal value of K can be different. In fact, even on the same dataset, the optimal value of K for different users can be different, since K is the number of periodic contexts of individual user. Therefore, in a specific application, and for specific users (if possible), K should be determined based on the recommendation accuracy. Note that the value of K also determines the number of datasets we need to generate for each user, thereby the training time and the disk space required for saving all these datasets. There is a trade-off between recommendation accuracy and computational cost.

We compared our probabilistic contextual filtering with traditional recommendation method and micro-profiling, a contextual pre-filtering approach. Experimental results show that our method can outperform the baseline approaches, in terms of recommendations accuracy and user space coverage. An interesting observation in Section 4.4.3 is that, after applying the micro-profiling method, the recommendation accuracy on retail data actually drops. That is, on retail data, the context-aware recommender system, micro-profiling performs even worse than a traditional RS, since the retail dataset is extremely sparse. The micro-profiling is a contextual pre-filtering method. For a given context, the micro-profiling filters out data collected in different contexts, making the dataset even sparser. This is the intrinsic limitation of traditional pre-filtering. The data sparsity leads to inaccurate recommendations and lower user coverage. This issue was identified in Chapter 1. Based on it, we presented our first research question: whether we can develop a CARS that can overcome data sparsity. In our probabilistic contextual filtering, we assign weight to each data point. Consequently, the dataset is no sparser than that in a traditional RS. Therefore, even on a relatively sparse dataset, after the contextualisation step, we can still apply traditional recommendation methods, such as matrix factorisation, to generate accurate context-aware recommendations. Experimental results show that our method can significantly outperform the baseline approaches, in terms of recommendations accuracy and user space coverage.

In Chapter 1, we also described another issue of existing CARS: non-personalised contexts. In probabilistic contextual filtering, we analyse the periodic patterns in user's activity. We can then define personalised contexts based on these patterns. Our probabilistic contextual filtering is non-deterministic. The converged positions of components in the mixture model depend in part on randomness for the starting positions during training. However, we did not observe a huge variance in terms of recommendation accuracy. Especially on retail data, the mixture model can always converge to the same position. This is because in retail data, the boundaries between contexts are very clear. Most users go to the store at fixed times of the day, e.g. during lunch break. Our method can accurately capture these patterns, define personalised contexts, generate accurate recommendations. In music data, a user may listen to the music all day. Therefore, the user's action count in a day may not show sharp rises and falls. In these cases, it might be slightly more difficult for our model to converge to the same values in repeated experiments. Despite that, our method performs considerably better than the baseline approaches on both datasets, in terms of recommendations accuracy.

In probabilistic contextual filtering, we are looking for periodic action patterns. However, before we applying a mixture model to the data, we already assume a certain type of periodic pattern exists. In music dataset, we are investigating daily patterns; in retail data, we are looking for weekly patterns. However, given a new dataset, we may not know what type of periodic patterns exist. More importantly, we may not know which periodic patterns can affect user's preference. In a dataset where multiple patterns exist, we need to compare the recommendation results generated based on different periodic contexts.

Probabilistic contextual filtering requires huge disk space since we are generating K rating matrices for each user. The training also takes relatively more time since a RS is employed on each dataset. A possible solution to this is combining the datasets that represent similar contexts. For example, in music data, we may identify three temporal contexts for the given user: one in the morning, one in the afternoon and one in the evening. However, the user's listening preference in the morning may not be very different from that in the afternoon. Therefore, the corresponding datasets can be combined into one. We may also combine datasets representing similar contexts but belonging to different users, in order to further reduce the number of datasets. However, to identify similar context, we may need to investigate what items user chooses in each context.

4.5 Summary

In this chapter, the probabilistic contextual filtering is presented. In this method, personalised temporal contexts are identified and modelled by a mixture of periodic Gaussians, based on the user's action count at different time. The probabilistic contextual filtering does not filter out training data like other contextual filtering methods. Instead, the probabilities of an action performed in different contexts are calculated. The user's preference can then be computed as a weighted combination of his preference in these identified contexts. Consequently, we can avoid the problems of context generalisation, and overcome data sparsity in CARS.

The testing criteria and experimental methods used are also presented in this chapter. Experiment results from two datasets have demonstrated that the proposed method can achieve higher prediction accuracy and user space coverage on sparse data, compared to traditional RS and contextual pre-filtering. However, compared to those methods, probabilistic contextual filtering also requires more disk space and training time.

In this chapter, we improve CARS by analysing users' periodic activity patterns. However, these periodic patterns don't always exist. In the next chapter, we focus on improving session based CARS. By modelling item distribution in different contexts, we are able to identify the contexts in which users have similar preferences. We separate context modelling from user modelling, so that accurate context-aware recommendations can be generated.

Chapter 5

Probabilistic Contextual Clustering

5.1 Introduction

In the previous chapter, we identify personalised temporal contexts based on periodic activity patterns. However, these periodic patterns don't always exist. Besides, temporal information is not always available. In this chapter, we are solving this problem from a different angle: instead of incorporating contextual information, we investigate the item distributions in different context.

In many applications, it is usually easy to identify users' continuous activity sessions, based on implicit data [24]. For instance, the music listening log can usually indicate which tracks the user played in each continuous music listening session; the supermarket transaction data can show us what items the user purchased in each shopping session. Moreover, sessions are naturally separate contexts. That is, in each of these activity sessions, the user's context usually stays the same [30]. This is very convenient for our purpose. We can view the items consumed in one session as in the same context.

We believe that the items the user chooses in a session can indicate the characteristics of the context, regardless of the time or the location of the session. Following the idea of contextual pre-filtering [2], to make context-aware recommendations, we would like to identify the sessions that contains similar items first. Specifically, we would like to find all the sessions that have similar item distributions. Thus a contextualised dataset can be constructed based on these similar sessions. A traditional recommendation method can then be applied on the contextualised datasets to generate context-aware recommendations.

There exist session based methods that can make use of item distributions and item co-occurrence. In these methods, a context is defined as an active, continuous session with the system, such as a music listening sessions [30]. Each session (context) is usually viewed as a mixture of various "topics" [15]. Each "topic" has unique item distributions. However, as we discussed in Chapter 2, these methods have either of the following problems:

1. Inability to build a complete user profile.

   In such systems, recommendations are based on the items the user just chose in the current context (seed items). The given user's general preference has little influence on the recommendations. That is, the recommendations are not personalised. Same seed items would lead to same recommendations.

2. Inability to select different features for user modelling and context modelling.

   In these systems, topic models are employed to reduce the number of features in the data. It replaces the user-item interaction data with user-topic features. These new features are selected for either context modelling or user modelling. However, once the features are selected, they are used for both modelling tasks. This can lead to inaccurate recommendations.

Therefore, in Chapter 1 we presented the following research question: whether we can develop a session based CARS that is able to achieve accurate user modelling and accurate context modelling at the same time. In this chapter, we propose a novel context-aware recommendation method: Probabilistic Contextual Clustering (PCC). Based on activity sessions, we employ Latent Dirichlet Allocation topic model to cluster sessions into overlapped groups. Each of these groups is represented by a unique topic from LDA. Each session can be represented as a mixture of topics. Although we don't know the actual context of each session, we can still identify the characteristics of each session by modelling the distribution of items in each session with topics. For each topic, a contextualised dataset is constructed. Therefore, most traditional collaborative methods can then be applied on the new datasets to generate context-aware recommendations. Experimental results show that our method can significantly improve recommendation accuracy, compared to other session based context-aware recommendation methods. In our method, the topic model is only used for session modelling and building the corresponding datasets. This also makes our method highly modular and flexible: it can be easily modified and combined with other traditional recommendation methods in different applications.

The remainder of this chapter is structured as follows: In Section 5.2, the algorithm of our method is presented. In Section 5.3, we improve our probabilistic contextual clustering method by incorporating tags (keywords) associated with items. Section 5.4 describes the testing approaches for the proposed method. The experimental results and analysis are presented in Section 5.5. Finally, a summary of the chapter is provided in Section 5.6.

5.2 Probabilistic Contextual Clustering

Our context-aware recommendation method takes three steps:

1. Apply a topic model (LDA) on the session data to discover and model the topics of sessions. Each topic has a unique item distribution [16].

2. For each topic, construct a user-item utility dataset, based on the user-

item interaction counts and the topic model obtained in the previous step.

3. Predict the item utility by applying a traditional recommendation method on each dataset. For a given session, the predicted utility of an item is the weighted sum of its predicted utilities in all topics.

In this section, we describe the algorithm of our method in detail.

5.2.1 Session Modelling Based on Item Distribution

We choose Latent Dirichlet Allocation to model the item distribution in sessions for its simple but powerful structure. LDA can capture statistical properties of sessions and items. As we discussed in Chapter 2, LDA can provide a compact session representation in terms of underlying topics [16], therefore widely used as a feature extraction model in the field of RS [31].

We first define the following terms [16]:

1. An *item* is the basic unit of discrete data. Let the total number of different items be V. An item is indexed by $\{1,...,V\}$. We represent items using unit-basis vectors that have a single component equal to one and all other components equal to zero. Thus, using superscripts to denote components, the xth item in the item space is represented by a V-vector w such that $w^x = 1$ and $w^y = 0$ for $y \neq x$.

2. A *session* is a sequence of N items denoted by $\mathbf{w} = (w_1, w_2, ..., w_N)$, where w_n is the nth item in the sequence. When modelling item distributions in sessions, we ignore the information about the user of the session. Therefore, all the sessions are anonymous, containing only a sequence of items.

3. A *dataset* is a collection of M sessions denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$.

LDA is a generative probabilistic model. By applying LDA, sessions are represented as mixtures over latent topics (multinomial distribution), where each topic is characterized by a distribution over items.

To apply LDA to our session data, we assumes the following generative process for each session $\mathbf{w}$ in the dataset D [16]:

1. Choose N, the number of items in the session.
2. Choose $\theta \backsim \text{Dir}(\alpha)$, the topic mixture.
3. For each of the N items w_n
   (a) Choose a topic $z_n \backsim \text{Multinomial}(\theta)$
   (b) Choose a item w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

We assume the dimensionality K of the topic variable z (and thus the dimensionality of the Dirichlet distribution) is known and fixed. The item probabilities are parameterised by a $K \times V$ matrix β where $\beta_{ij} = p(w^j = 1|z^i = 1)$. Note that the number of items in a session, N is independent of all the other data generating variables (θ and $\mathbf{z}$). The topic mixture, θ is a K-dimensional Dirichlet random variable. θ can take values in the $(K-1)$-simplex (a K-vector θ lies in the $(K-1)$-simplex if $\theta_i \geq 0, \sum_{i=1}^{K} \theta_i = 1$). It has the following probability density on this simplex [16]:

$$p(\theta|\alpha) = \frac{\Gamma(\sum_{i=1}^{K} \alpha_i)}{\prod_{i=1}^{K} \Gamma(\alpha_i)} \theta_1^{\alpha_1 - 1} \cdots \theta_K^{\alpha_K - 1}, \tag{5.1}$$

where the parameter α is a K-vector with components $\alpha_i > 0$, and where $\Gamma(x)$ is the Gamma function.

Given the parameters α and β, the joint distribution of a topic mixture θ, a set of N topics $\mathbf{z}$, and a set of N items $\mathbf{w}$ generated by $\mathbf{z}$, is given by [16]:

$$p(\theta, \mathbf{z}, \mathbf{w}|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta) \tag{5.2}$$

where $p(z|\theta)$ is θ_i for the unique i such that $z_n^i = 1$. Integrating over θ and summing over z, we obtain the marginal distribution of an activity session [16]:

$$p(\mathbf{w}|\alpha,\beta) = \int p(\theta|\alpha)(\prod_{n=1}^{N}\sum_{z_n} p(z_n|\theta)p(w_n|z_n,\beta))d\theta \tag{5.3}$$

Finally, taking the product of the marginal probabilities of single sessions, we obtain the probability of a dataset [16]:

$$p(D|\alpha,\beta) = \prod_{d=1}^{M}\int p(\theta_d|\alpha)(\prod_{n=1}^{N_d}\sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn},\beta))d\theta_d \tag{5.4}$$

As the exact inference is not possible for learning the parameters of our model, variational message passing is used for approximate inference [16].

5.2.2 Constructing Datasets

So far we have obtained the topic model with K topics. In this step, we would like to construct a contextualised dataset for each topic. Specifically, we would like to construct a user-item utility matrix for each topic. If a given user has never interacted with a specific item, the corresponding user-item utilities in all topics should be zero. If the user has interacted with the item, the user-item utility in each dataset should be able to reflect the corresponding user-item interaction count.

Formally, a session $\mathbf{w}$ of user u_i contains a sequence of N items, $\mathbf{w} = (w_1, w_2, ..., w_N)$. Let w_n be the nth item in $\mathbf{w}$, and the jth item in item space $\{1,...,V\}$ so that $w_n^j = 1$ (superscripts denotes component in unit-basis vector). We would like to compute user-item utility, A_{ij} for user u_i and the jth item in item space $\{1,...,V\}$.

Given the topic model obtained in the previous step, we can compute the probability that session $\mathbf{w}$ is generated by topic z:

$$p(z|\mathbf{w}) = \int_{\theta} p(z|\theta)p(\theta|\mathbf{w})d\theta \tag{5.5}$$

If in session $\mathbf{w}$ the user interacted with item w once, this user-item interaction is viewed as K different micro-interactions: each under a different topic, with a different interaction count. For each of these micro-interactions, we assume the user interacted with item w with micro-interaction count $p(z|\mathbf{w})$, where $w \in \mathbf{w}$, and z is the corresponding topic of this micro-interaction. Note that $p(z|\mathbf{w}) > 0$ for each topic z, and $\sum_{z=1}^{K} p(z|\mathbf{w}) = 1$. That is, the sum of all the micro-interactions counts of the same interaction is still one.

Assume z is the kth topic. Let d_k be the constructed dataset of topic z. For this dataset, summing over all the corresponding micro-interaction counts $p(z = k|\mathbf{w})$ of item w by user u_i, we obtain A_{ij}^{k}, the user-item utility of the jth item in the item space, for user u_i in topic k:

$$A_{ij}^{k} = \sum_{\mathbf{w}} p(z = k|\mathbf{w}) \tag{5.6}$$

where session $\mathbf{w}$ is started by user u_i, and contains the item w that satisfies $w^j = 1$. We compute the user-item utility for all the user-item pairs in all K datasets. Therefore, based on item distributions, we obtain a contextualised dataset for each topic.

5.2.3 Predicting Preferences

So far, we have obtained K user-item utility matrices, one for each topic. However, these datasets can not be directly used by most traditional collaborative recommendation methods, since most of these methods can only work on explicit data, such as user-item ratings. Our user-item utilities are based on interaction counts. Depending on the data, the value of A_{ij} can be smaller than one to several hundreds. We need to apply some mapping approaches to the constructed datasets, in order to obtain user-item ratings:

$$A_{ij} \to r_{ij}$$

Usually, a user would only interact with a relatively small number of items from the item space. For example, the online radio last.fm [1] has more than 12 million unique tracks, while most of the users of have only played less than 5,000 of them. Therefore, in the datasets we constructed, the values of many r_{ij} are zeros. In most applications, we would like to know the user's preferences on most of the items before the recommendations are made. Therefore, after the mapping procedure, a traditional recommendation method is applied on each dataset to predict users' ratings in each topic.

Formally, we observe a new session $\mathbf{w}$ of user u_i containing a sequence of N items, $\mathbf{w} = (w_1, w_2, ..., w_N)$. We would like to recommend items to user u_i for the ongoing session $\mathbf{w}$. Specifically, we would like to know the rating user u_i would give on the item w for the current session. Let us assume w is the jth item in the item space, so that $w^j = 1$. Also, let us assume in all the datasets, $r_{ij} = 0$. Now we apply a traditional collaborative method, such as matrix factorisation, on all K datasets. Thus, we obtain K unique predicted ratings, r_{ij}^0 to r_{ij}^K, one from each topic.

Given the topic model obtained in the previous step, we can compute the probability that the current session $\mathbf{w}$ is generated by topic z, according to Equation 5.5. As we discussed at the beginning of this chapter, we can assume the user's context remains unchanged in one session. Therefore the topic mixture of the current session remains the same. The final r_{ij} is computed as a weighted sum of r_{ij}^k:

$$r_{ij} = \sum_{z=1}^{K} (r_{ij}^k \int_\theta p(z|\theta) p(\theta|\mathbf{w}) d\theta) \tag{5.7}$$

[1] http://www.last.fm/

where $p(\theta|\mathbf{w})$ represents the inferred probability of θ given the observed N items in the session $\mathbf{w}$. We can compute the r_{ij} of any given item w in the current session. Finally, r_{ij} is used for ranking items. We can recommend items with greater r_{ij} to user u_i for the current session. Note that as the user continues interacting with more items in the given session, we may want to keep updating the values of $p(\theta|\mathbf{w})$, since now we can observe more items in session $\mathbf{w}$.

Unlike many session based RS, in our method we separate the process of user modelling and the process of context modelling. We can employ different traditional collaborative methods, or an ensemble of traditional collaborative approaches, instead of a specific one, to meet various criteria in different applications. We can also employ different mapping approaches to construct different datasets, in order to improve individual recommendation method in the ensemble.

5.3 Incorporating Item Tags

In Section 5.2, we described the algorithm of Probabilistic Contextual Clustering. We model item distributions, so that sessions containing a sequence of items can be clustered into different topics. In our method, the basic unit is the item. Ideally, each session contains many items, and we don't have to make recommendations until we have observed a sufficient number of items in the current session. However, in reality it is possible that a session only contains a few items, depending on the specific applications. Besides, even if every sessions is long enough, we would like to make context-aware recommendations from the beginning of the session, rather than waiting until the session is about to end. However, at the beginning of the session, with only a few items available, we can not accurately infer the topics of the session based on item distributions.

In many applications, besides the user-item interaction data, we can also obtain additional tags associated with the items. For example, many news agencies label their articles with keywords; digital media service websites, such as Zune[2] and last.fm, ask users to choose tags that can describe the characteristics or the genres of tracks. These item tags can usually provide valuable information about the items, thereby providing more information about the corresponding contexts. In this section, we improve our Probabilistic Contextual Clustering by incorporating item tags. The basic unit of the improved method is the item tag, instead of the item. We model tag distributions so that sessions can still be clustered into topics. In most applications, each item contains multiple tags. Therefore, even if a session contains only a small number of items, we can still obtain a sufficient number of item tags to infer the characteristics of the session. More importantly, compared to Probabilistic Contextual Clustering based on item distributions, we can identify the corresponding topics at the beginning of an ongoing session, with fewer items observed.

Similar to Probabilistic Contextual Clustering without tags, the new method also consists of three steps:

1. Apply a topic model (LDA) on the session data to discover the topics of sessions. Unlike in Section 5.2, each topic here is distribution over tags, instead of items.

2. For each topic, construct a user-item utility dataset, based on the user-item interaction counts and the topic model obtained in the previous step.

3. Predict item utility by applying a traditional recommendation method on each dataset. For a given session, the predicted utility of an item is the weighted sum of its predicted utilities in all topics.

[2]http://www.xbox.com/zune

5.3.1 Session Modelling Based on Tag Distribution

We also use Latent Dirichlet Allocation (LDA) to model the tag distributions in sessions. We define the following terms:

1. A *tag* is the basic unit of the discrete data. Let the total number of different tags be S. A tag is indexed by $\{1,...,S\}$. We represent tags using unit-basis vectors that have a single component equal to one and all other components equal to zero. Thus, using superscripts to denote components, the xth tag in the tag space is represented by a S-vector s such that $s^x = 1$ and $s^y = 0$ for $y \neq x$.

2. An *item* is a sequence of tags denoted by $w = s_1, s_2, ..., s_Q)$. Let the total number of different tags be V. An item is indexed by $\{1,...,V\}$. Note that two different items may have the same sequence of tags. We also represent items using unit-basis vectors that have a single component equal to one and all other components equal to zero. Thus, using superscripts to denote components, the xth item in the item space is represented by a V-vector w such that $w^x = 1$ and $w^y = 0$ for $y \neq x$.

3. A *session* $\mathbf{w}$ is a sequence of items, $\mathbf{w} = (w_1, w_2, ..., w_P)$. However, since we are modelling tag distributions, we would like to use the tags, rather than the items, to represent a session. Therefore, we use all the tags of all the items appeared in the session to represent the session, thus $\mathbf{w} = (w_1, w_2, ..., w_N)$. Note that, the same tag may appear in $\mathbf{w}$ for more than once.

4. A *dataset* is a collection of M sessions denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$.

We model the tag distributions with LDA, so that sessions can be represented as mixtures over latent topics, where each topic is characterized by a distribution over tags. We assumes the following generative process for each session $\mathbf{w}$ in the dataset D [16]:

1. Choose N, the number of tags in the session.

2. Choose $\theta \backsim \mathrm{Dir}(\alpha)$, the topic mixture.
3. For each of the N tags w_n
   (a) Choose a topic $z_n \backsim \mathrm{Multinomial}(\theta)$
   (b) Choose a tag w_n from $p(w_n|z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

This is very similar to the generative process in Section 5.2.1. However, in this generative process, we ignore the concept of item: a session is a mixture of topics and a topic is a distribution over tags. We assume the dimensionality K of the topic variable z is known and fixed. The tag probabilities are parameterised by a $K \times V$ matrix β where $\beta_{ij} = p(w^j = 1|z^i = 1)$. The topic mixture, θ is a K-dimensional Dirichlet random variable.

Given the parameters α and β, we obtain the probability of an activity session [16]:

$$p(\mathbf{w}|\alpha, \beta) = \int p(\theta|\alpha)(\prod_{n=1}^{N} \sum_{z_n} p(z_n|\theta)p(w_n|z_n, \beta))d\theta \tag{5.8}$$

Taking the product of the probabilities of single sessions, we obtain the probability of a dataset [16]:

$$p(D|\alpha, \beta) = \prod_{d=1}^{M} \int p(\theta_d|\alpha)(\prod_{n=1}^{N_d} \sum_{z_{dn}} p(z_{dn}|\theta_d)p(w_{dn}|z_{dn}, \beta))d\theta_d \tag{5.9}$$

Again, variational message passing is used for approximate inference [16]. Thus, we can now represent the sessions as a mixture of topics again. In the next step, for each topic, we construct a user-item utility matrix, which is built in the exact same way as in Section 5.2.2. The constructed datasets can be used to predict user's preferences in different sessions as in Section 5.2.3, since these datasets are also user-item utility matrices. The predicted ratings can be computed by Equation 5.7.

5.4 Experimental Method

5.4.1 Data for Testing PCC

The two implicit datasets used in Chapter 4 are also used in this chapter.

Music Data: the dataset from the music website, last.fm, containing 992 users' listening history in two years' time.

The music data is also used to evaluate our method with item tags. In our system, the set of top tags for each track are retrieved from last.fm. These tags describe various features of the songs including genre, artist name and the era. They also describe users' attitudes toward the songs, including feelings such as sad, nostalgic, upbeat, and calm. Although people may have different and even contradictory opinions about some songs (particularly with tags that are related to "mood"), top tags with frequency above a minimum threshold capture the social opinion about each song. In our system, for each track, the tags we selected must have been used to describe the given track by at least 10 different users.

Retail Data: the dataset from Ta-Feng containing transaction history of 32,000 customers.

The Ta-Feng data is already in the form of transactions. Therefore, we can directly observe which items are bought together in each transaction. However, we are not able to obtain item tags for this dataset. Therefore, this dataset is only used to test the probabilistic contextual clustering on item distributions.

5.4.2 Testing Method of PCC

Test Sets

To evaluate our method on music data, we randomly select 200 users for testing. For the rest of the users, all their sessions are fully observed. For each of these 200 users, we select 10 listening sessions that contains more than 20 items for testing. For simplicity, we only use the first 20 items in each session. The rest of the items in those session are discarded. Of these 20 items in the session, the first 10 items are used as seed items. That is, we have observed that the user chose these 10 items in the current session. The rest 10 items in the selected session are hidden and used for testing. For each testing session, we also randomly select 10 items that the user have played before but not in the given session, for testing. Therefore, for each testing session, the recommender systems need to predict user's preference on the last 10 items that actually appear in the given session, and the 10 items that do not. The recommender systems rank these items from the most preferred to the least preferred. We use precision at 10 and precision at 1 to measure the accuracy of recommendations.

Note that, this is a relatively difficult task for the traditional RS, since all the 20 tracks we selected have been played by the user. It is difficult to determine which items the user prefers if we do not know the 10 items observed in the current context. For context-aware recommender systems, the recommendations are made based on user's playing history and the 10 seed items in the given context.

A similar approach is used on retail data. 8000 users are selected randomly for testing. For the rest of the users, all their transactions are fully observed. For each of these 8000 users, we select 2 transactions that contains more than 8 items. For the simplicity of comparison, we only use the first 8 items in each transaction. The rest of the items in those sessions are discarded. Of these 8 items, the first 4 items are used as seed items. That is, we assume we have observed the user chose these 4 items in the given transaction. The last 4 items in the transaction are hidden and used for testing. For each session, we also randomly add 4 items that the user has purchased, but not in the given transaction, to the testing set. Therefore, in each testing transaction, the recommender systems need to predict user's preference on the last 4 items that the user actually purchased in the transaction, and the 4 items that the user did not purchase in the given transaction. The recommender systems rank the items from the most preferred to the least preferred. We use precision at 4 and precision at 1 to measure the accuracy of recommendations. Similarly, this can be a difficult task for the traditional recommender system.

Baseline Approaches

The first baseline approach we are using is matrix factorisation (MF) [27] with 100 factors. MF is used as baseline approach due to its ability to generate accurate predictions on sparse datasets. This is a traditional recommendation method, which does not take into consideration contextual information. We also use the same matrix factorisation method in our probabilistic contextual clustering (PCC). That is, we apply MF on the contextualised datasets constructed by our PCC. As in Chapter 4, we use the implementation of MF from Apache Spark.

To evaluate PCC on item distribution, we also implemented the session based model (SBM) in [73]. In this method, mood (equivalent to the topic in LDA) is the latent variable of the session model. The model assumes that each user is represented as a distribution over different moods, and for each session, there is a latent mood which guides the choice of items. In this method, the LDA model based on item distributions is used for both user modelling and session modelling. The observed items in each session can be used to determine the user's moods. Then the recommendations are directly generated based on the item distributions of corresponding moods.

To evaluate PCC on item tags, we implemented the Query-Driven (QD) context-aware recommender system [31]. In this method, a LDA topic model is employed to capture the latent factors that determine user's choice. Similarly, the features extracted by LDA based on item tag distributions are used for both user modelling and session modelling. The tags of observed items in a session can be used as a query to determine the topic of current session and search for relevant items. Therefore, recommendations are directly generated based on the tag distributions of corresponding topics. In both QD and PCC, we use the implementation of LDA from Apache Spark.

The probabilistic contextual clustering method is non-deterministic . If the method is initialised several times, we may obtain different results. In order to gain the representative results, we repeat each experiment involving the non-deterministic method 100 times and take the average of the performance values. The results are analysed by the combined use of the Hodges-Lehmann Estimate with associated non-parametric confidence intervals at 95%.

Machine Used

In our experiments, we use a laptop with Intel 2.6 GHz dual core processor and 4GB of RAM.

5.5 Results and Analysis of PCC

In this section, we present the experimental results on probabilistic contextual clustering, and the analysis based on these results.

5.5.1 Comparing PCC on item distribution with traditional recommendation method

In this section, we compare our PCC with traditional recommendation method matrix factorisation.

Hypothesis

$H5.1_0$: There will be no difference in precision between the traditional method, matrix factorisation (MF) and the combination of probabilistic contextual clustering (PCC) on item distributions and matrix factorisation (PCC + MF).

Results

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@4	Precision@1
MF	52.3%	66.6%	51.7%	58.9%
PCC + MF	74.1%	89.2%	67.1%	74.2%

Table 5.1 Precisions of MF and PCC + MF

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@4	Precision@1
Can the hypothesis be rejected at the 95% confidence interval	Yes	Yes	Yes	Yes
Hodges-Lehmann Estimate	21.8%	22.6%	15.4%	15.3%
Confidence Interval at 95%	(21.8%, 21.8%)	(22.6%, 22.6%)	(15.4%, 15.4%)	(15.3%, 15.3%)

Table 5.2 Summary of analysis between MF and PCC + MF

Table 5.1 shows the precisions of the system using only MF and the system using PCC + MF on both datasets. Table 5.2 shows the results of the Mann–Whitney–Wilcoxon non-parametric test on the distributions of the results from both recommender systems.

The probabilistic contextual clustering substantially increases the precision at 10 on music data by 21.8%, compared to the traditional method, matrix factorisation. Similar results are found on retail data: the system using PCC + MF achieves considerably higher precision at 4 than only using MF. On both datasets, when only one item is recommended to the user (precision at 1) the combination of PCF and MF is more likely to find relevant items than MF alone. The statistical significance of the results is shown in Table 5.2. On both datasets, using both criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using PCC + MF instead of MF (or vice versa).

Analysis

As we discussed earlier, the testing task is difficult for the traditional recommender system since the user has interacted with all the items selected for testing. On both dataset, when the system is evaluated by precision at 10 and 4, the traditional RS performs only slightly better than a random recommender system, because based on users' history data alone, the traditional RS is not able to identify the items that the user prefers in a specific context. The precision of traditional RS increases when only one item is recommended. On music data, the reason is that some users tend to listen to their favourite tracks regardless of the context. If such tracks exist for the given user, those tracks are likely to be selected in the testing session. The traditional recommender system is able to identify those frequently played tracks. However, the number of such tracks is usually small. Therefore, when only one item is needed (precision at 1), such items are recommended to the user. In retail data, there are items users tend to purchase in most transactions. For example, it is possible that a user always buys a soft drink, either when he is doing grocery shopping for the whole family, or just buying lunch for himself. Those items can be identified easily by the traditional RS. However, the number of such items is usually small for most users. Therefore, if the RS needs to recommend multiple items, the recommendation accuracy decreases dramatically.

Compared to traditional RS, the combination of PCC and MF leads to significant improvement on both datasets. The PCC can determine the characteristics of the given session based on seed items. MF is then applied on contextualised datasets, making personalised context-aware recommendations. Therefore, the recommendation accuracy increases considerably.

5.5.2 Comparing PCC on item distribution with session based model

In this section, we compare our PCC based on item distribution with SBM

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@4	Precision@1
MF	52.3%	66.6%	51.7%	58.9%
SBM	66.9%	69.0%	60.2%	63.8%
PCC + MF	74.1%	89.2%	67.1%	74.2%

Table 5.3 Precisions of MF, SBM and PCC + MF

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@4	Precision@1
Can the hypothesis be rejected at the 95% confidence interval	Yes	Yes	Yes	Yes
Hodges-Lehmann Estimate	7.2%	20.2%	6.9%	10.4%
Confidence Interval at 95%	(7.2%, 7.2%)	(20.2%, 20.2%)	(6.9%, 6.9%)	(10.4%, 10.4%)

Table 5.4 Summary of analysis between SBM and PCC + MF

Hypothesis

$H5.2_0$: There will be no difference in precision between the session based model (SBM) and the combination of probabilistic contextual clustering (PCC) on item distributions and matrix factorisation (PCC + MF).

Results

The results form these experiments are presented in Table 5.3. The results of the system using only the traditional recommendation method MF is also presented in Table 5.3. The statistical analysis between the two contextual filtering methods is given in Table 5.4.

On both datasets, the session based model can also achieve higher precision compared to the system only employing matrix factorisation, recommending either one item or the first half of items. However, the precision of SBM is still considerably lower compared to probabilistic contextual clustering. Especially when measured by precision at 1 on music data, PCC + MF outperforms SBM by 20.2%. These results demonstrate that PCC can predict the user's preference more accurately compared to SBM.

The statistical significance of the results is shown in table 5.4. On both datasets, using either criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using SBM instead of PCC + MF (or vice versa).

Analysis

In SBM, the topic model selects features for session modelling. This is similar to our PCC model. Therefore, based on seed items, they can both identify the topic mixture of the given context. However, SBM is a unified model. The features selected by LDA are also used to model the user's general preference. RS based on item distributions alone tend to recommend the items preferred by the majority of the users. That is, SBM is less competent in making traditional recommendations, especially on sparse data. In contrast, our PCC model is highly modular. The item distribution is only used to model the sessions. To predict general user preference, we choose MF, one of the most accurate traditional RS. Therefore, the combination of PCC and MF leads to significant rise in recommendation accuracy, compared to SBM.

5.5.3 Comparing PCC on item tag distribution with traditional recommendation method

In this section, we compare our PCC based on tag distribution with matrix factorisation.

Hypothesis

$H5.3_0$: There will be no difference in precision between the traditional method, matrix factorisation (MF) and the combination of probabilistic contextual clus-

	On Music Data	
	Precision at 10	Precision at 1
MF	52.3%	66.6%
PCC with tags + MF	78.5%	90.6%

Table 5.5 Precisions of MF and PCC with tags + MF

	On Music Data	
	Precision at 10	Precision at 1
Can the hypothesis be rejected at the 95% confident interval	Yes	Yes
Hodges-Lehmann Estimate	26.2%	24.0%
Confidence Interval at 95%	(26.2%, 26.2%)	(24.0%, 24.0%)

Table 5.6 Summary of analysis between MF and PCC with tags + MF

tering (PCC) on item tag distribution and matrix factorisation (PCC with tags + MF).

Results

Table 5.5 shows the precisions of the system using only MF and the system using PCC with tags + MF on music data. Table 5.6 shows the results of the Mann–Whitney–Wilcoxon non-parametric test on the distributions of the results from both recommender systems.

The probabilistic contextual clustering based on tag distributions substantially increases the precisions on music data, compared to the traditional method, matrix factorisation. The statistical significance of the results is shown in Table 5.6. Using both criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using PCC with tags + MF instead of MF (or vice versa).

Analysis

As we discussed in Section 5.4.2 and 5.5.1, it is very difficult for the traditional RS to identify the preferred items in the testing contexts, since the user has interacted with all the items in the testing set. Therefore, in this task, tradi-

	On Music Data	
	Precision at 10	Precision at 1
MF	52.3%	66.6%
QD	69.2%	75.5%
PCC with tags + MF	78.5%	90.6%

Table 5.7 Precisions of MF, QD and PCC with tags + MF

	On Music Data	
	Precision at 10	Precision at 1
Can the hypothesis be rejected at the 95% confident interval	Yes	Yes
Hodges-Lehmann Estimate	9.3%	15.1%
Confidence Interval at 95%	(9.3%, 9.3%)	(15.1%, 15.1%)

Table 5.8 Summary of analysis between QD and PCC with tags + MF

tional RS recommends the items that tend to appear in every sessions. Since the number of such items is small, when multiple items are recommended, the precision drops so sharply that the RS is only slightly better than a random RS.

Compared to the traditional RS, our method makes use of common tags of seed items. PCC with items can accurately identify the topic mixture of the current session. Combining with MF, our method can generate personalised recommendations. Consequently, the precision rises considerably.

5.5.4 Comparing PCC on item tag distribution with query driven method

In this section, we compare our PCC based on tag distribution with QD.

Hypothesis

$H5.4_0$: There will be no difference in precision between the query driven method (QD) and the combination of probabilistic contextual clustering (PCC) on item tag distributions and matrix factorisation (PCC with tags + MF).

Results

The results form the experiments are presented in Table 5.7. The results of the system using only the traditional recommendation method MF is also presented in Table 5.7. The statistical analysis between the two contextual filtering methods is given in Table 5.8.

On music data, the query driven method can also achieve higher precision compared to the traditional RS, matrix factorisation, recommending either 1 or 10 items. However, the precision of QD is still considerably lower compared to probabilistic contextual clustering. Especially when measured by precision at 1 PCC with items + MF outperforms QD by 15.1%. These results demonstrate that PCC can predict the user's preference more accurately than QD.

The statistical significance of the results is shown in table 5.8. Using both criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using QD instead of PCC + MF (or vice versa).

Analysis

QD is similar to SBM, but based on tag distribution instead of item distribution. In QD, the topic model selects features for session profiling. This is similar to our PCC model. Therefore, based on the tags of seed items, they can both identify the topic mixture of the given context. However, QD is a unified model. The features selected by LDA are also used to model the user's general preference. RS based on tag distributions is similar to the traditional content-based methods. It recommends the items that contains the keywords of frequently interacted items. In contrast, in PCC model the item distribution is only used to model the sessions. To predict general user preference, we choose MF, which is superior to most traditional recommendation techniques, especially on sparse data [40]. Therefore, the combination of PCC with tags and MF leads to a significant rise in prediction accuracy, compared to QD.

	On Music Data	
	Precision at 10	Precision at 1
PCC + MF	74.1%	89.2%
PCC with tags + MF	78.5%	90.6%%

Table 5.9 Precisions of PCC + MF and PCC with tags + MF

	On Music Data	
	Precision at 10	Precision at 1
Can the hypothesis be rejected at the 95% confident interval	Yes	Yes
Hodges-Lehmann Estimate	4.4%	1.4%
Confidence Interval at 95%	(4.4%, 4.4%)	(1.4%, 1.4%)

Table 5.10 Summary of analysis between PCC + MF and PCC with tags + MF

5.5.5 Comparing PCC on item distribution with PCC on item tag distribution

In this section, we compare our PCC based on item distribution with PCC on item tag distribution.

Hypothesis

$H5.5_0$: There will be no difference in precision between the combination of probabilistic contextual clustering on item distributions and matrix factorisation (PCC + MF) and the combination of probabilistic contextual clustering on item tag distributions and matrix factorisation (PCC with tags + MF).

Results

Table 5.9 shows the precisions of the system using PCC with tags + MF and the system using PCC + MF on music data. Table 5.10 shows the results of the Mann–Whitney–Wilcoxon non-parametric test on the distributions of the results from both recommender systems.

The probabilistic contextual clustering on tag distribution consistently increases the precisions on music data, either evaluated by precision at 10 or

1. The statistical significance of the results is shown in Table 5.10. Using both criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using PCC with tags + MF instead of PCC + MF (or vice versa).

Analysis

In both methods, the topic mixture of the current context is inferred based on the observed seed items. In PCC, the LDA is used to model the item distributions. If a seed item has not been interacted with by sufficient number of users, in sufficient number of contexts, the RS can not accurately infer the characteristics of the context. In contrast, with item tags, the system can model the tag distributions in different contexts. The item is not the basic element of the system. It is replaced by the tag. This substantially reduces the number of basic elements, since the number of tags in most application is much smaller than the number of items. Therefore, we can still infer the topic mixture even if the item has only appeared in a few contexts. More importantly, with tags we can identify the topic mixture of the current context with fewer items, since each item can contain multiple tags, thereby providing more information regarding the ongoing session. This is especially useful at the beginning of a session, when we try to make context-aware recommendations based on only a few seed items. Consequently, this results in a consistent improvement in term of prediction accuracy, as we can see from Table 5.9.

5.6 Summary

In Chapter 1, we introduced the limitation of existing session based CARS: they can not accurately model contexts and general user preferences at the same time, since they usually employ a unified model, select only one set of features for both modelling tasks. Therefore, in Chapter 1, we presented the following research question: whether we can develop a session based CARS that is able to employ two separate models for user profiling and context profiling, to achieve higher recommendation accuracy.

In this chapter, we presented the probabilistic contextual clustering. The testing criteria and experimental results of PCC are also presented in this chapter. By modelling the sessions based on item distributions, we are able to cluster the sessions into topics, and to generate a dataset for each topic. This allows us to employ various traditional recommendation methods to model users' general preferences on the contextualised datasets independently, therefore more accurately predicting users' preferences than unified models. Experiment results have demonstrated that the proposed method can consistently achieve higher prediction accuracy compared to existing session based methods.

In this Chapter, we also presented PCC based on tags. With the incorporation of tags, we can identify the commonality of items consumed in the same session, therefore more accurate model the "topics" of sessions. As in PCC based on items, PCC on tags can also be used together with most traditional recommendation methods. With a separate model dedicated for user modelling, we can achieve higher accuracy, Experiment results of PCC on tags have demonstrated that the proposed method can achieve higher prediction accuracy compared to existing session based methods with tags. Also, the results have demonstrated that PCC on tags can outperform PCC on items in terms of prediction accuracy.

In probabilistic contextual clustering, context-aware recommendations can only be made after we have observed the item(s) the user has chosen in the given context. This can limit the application of our method. The probabilistic contextual filtering method proposed in Chapter 4 only requires temporal information. However, as we discussed at the end of Chapter 4, temporal information can be misleading, and the probabilistic contextual filtering is computationally expensive. In the next chapter, we combine these two methods to deal with their limitations.

Chapter 6

Clustering Based Probabilistic Contextual Filtering

6.1 Introduction

In the previous chapter, we described our probabilistic contextual clustering (PCC) method. In this method, we model the item (tag) distributions in different contexts across all users, and infer the characteristics of the current context based on several seed items. Unlike many existing session based methods, our PCC employs matrix factorisation, a completely separate traditional RS to model the user's general preference, rather than using item (tag) distributions or recommending the items that are similar to the seed items. This allows our system to make not only context-aware but also accurate and personalised recommendations. The collaborative method, matrix factorisation, can also be easily replaced by various traditional recommendation methods in different applications, making our RS highly flexible. However, the system can not work if no seed items can be observed in the given context. For example, in film recommendation, usually the user only watches one film in one sitting. Therefore, there is no seed film that can provide information on the current user context. Besides, even observing seed items is possible, accurate recommendations are still desirable at the very beginning of each context, when the user has not chosen any items yet. In these cases, to make context-aware recommendations, we need to make use of other contextual information, such as temporal information.

In Chapter 4, we presented the probabilistic contextual filtering (PCF). In this method, we investigate the user's periodic activity pattern based on implicit data. Our PCF method defines personalised temporal contexts based on these patterns. Instead of filtering out data collected in different contexts, PCF assigns weight to each user action. Experimental results have demonstrated that compared to traditional contextual pre-filtering, PCF can significantly improve recommendation accuracy, especially on sparse dataset. However, this method also has its limitations. PCF generates a complete user-item utility matrix for each user, then applies a traditional recommendation method on each of them. This is time-consuming and requires considerable disk space. It can be very problematic when applying PCF to system containing large number of users and items. Besides, in some of the personalised contexts based on temporal patterns, the given user's preference can be the same. For example, the user's preferences on music before and after lunch may not be very different, but the user's play count may have a sharp drop around midday since the user does not listen to music at lunch time. Therefore, it is not necessary to create two different datasets for these contexts. However, we can not confirm the similarity if we do not explore the tracks that the user plays in each context. These issues of PCF suggest that improvement can be made by investigating the item distributions in different contexts and combining the contexts with similar item distributions. By doing so, we can obtain more data for each unique context, so that more reliable predictions can be made. More importantly, the computational cost can be substantially reduced.

The limitations of PCF inspire us to add the features of PCC to PCF: if we can identify personalised contexts, cluster and combine these contexts based on item distributions, we are able to address the issues of PCF and PCC at the same time. In this chapter, we present the combined approach, the clustering based probabilistic contextual filtering, the combination of PCC and PCF.

The remainder of this chapter is structured as follows: In Section 6.2 the algorithm of our clustering based probabilistic filtering is presented. The testing criteria and the experiments results of the proposed method are given in Section 6.3 and 6.4 respectively. Final comments are provided in Section 6.5.

6.2 Clustering Based Probabilistic Contextual Filtering

In this section, we present the algorithm of clustering based probabilistic contextual filtering (CBPCF). The idea of our method is simple: modelling the item distributions in the contexts identified by PCF across all the users, so that we can cluster these contexts into topics. The method takes five steps:

1. Capture the user's action patterns based on frequency of actions. Based on these patterns, we define personalised temporal contexts.
2. For each personalised context, compute the interaction counts of the items in the given context, based on the mixture model obtained in step one. Therefore, the context can be represented by a list of items and the corresponding interaction counts.
3. View each personalised context as a session. Apply a topic model on the session data to discover the topics of sessions. Each topic has a unique item distribution.
4. For each topic, construct a user-item utility dataset, based on the user-item interaction counts and the topic model obtained in the previous step.
5. Apply a traditional recommendation method on each dataset. In each

session, the utility of an item is computed as the weighted sum of its predicted utilities in all topics. Then the final utility of the item at a specific time is computed as the weighted sum of its predicted utilities in all sessions.

In this section, we describe the algorithm of our method in detail.

6.2.1 Modelling Action Patterns

In this step, based on the implicit data, we apply a periodic mixture model (MoAWG) to capture the user's action patterns. Then we can define personalised temporal contexts. This modelling step is identical to that in Section 4.2.1. At the end of this step, for each user, we obtain the parameters of MoAWG: μ_k, $\theta_{0,k}$ and σ_k^2, the weight, mean and variance of each component. The responsibility γ_{xk} of component k for an action x observed at time θ_x, can be estimated using the parameter of MoAWG by Equation 4.5. The detailed approximation can be found in Section 4.2.

6.2.2 Generating Sessions

In this step, we would like to calculate the user-item interaction counts in each user context. This step is similar to the step described in Section 4.2.2.

Formally, we would like to compute p_{ij}^k for user u_i and the item w_j, in kth context of user u_i. Let x be an action of user u_i on item w_j at time θ_x. We can calculate γ_{xk}, the probability that x is an action in the kth context of user u_i by Equation (3.4). As in Section 4.2.2, let us view this action as K different micro-actions: each in a different context, with a different action count. For each of these micro-actions, we use γ_{xk} as its action count, where k is the corresponding context index. Note that $\gamma_{xk} > 0$ for each context, and $\sum_{k=1}^{K} \gamma_{xk} = 1$.

In each context, summing over all the corresponding micro-action counts

γ_{xk} of item w_j by user u_i, we obtain p_{ij}^k, the user-item interaction count of item w_j for user u_i:

$$p_{ij}^k = \sum_x \gamma_{xk} \tag{6.1}$$

We compute p_{ij}^k of all the items for user u_i in the kth context. However, note that in Section 4.2.2, for each user, we also compute the interaction count for all other users in order to build a complete user-item utility matrix just for user u_i. In this Chapter, for the context of user u_i, we only compute p_{ij}^k for user u_i.

Based on p_{ij}^k, we can view context c_k of user u_i as an interaction session with the items that have non-zero p_{ij}^k. However, the number of times an item can appear in a session must be an integer, while p_{ij}^k is a real number. Therefore, we round p_{ij}^k up to the nearest integer. We denote this number as $\hat{p}_{ij}^k$.

6.2.3 Modelling Item Distributions

Similarly, we choose Latent Dirichlet Allocation to model the items distributions in personalised contexts.

We define the following terms:

1. An *item* is the basic unit of discrete data. Let the total number of different items be V. An item is indexed by $\{w_1,...,w_V\}$.
2. A *session* is a sequence of N items. We used $\mathbf{w}_k$ to denote the kth session of user u_i, generated from the kth context. The personalised contexts we have obtained can be denoted as a sequence of items, based on all the $\hat{p}_{ij}^k$ in the previous section. This is because, in LDA model, the order of items in a session is ignored [16]. If $\hat{p}_{ij}^k > 1$, we can assume item w_j is interacted multiple times in a row in session $\mathbf{w}_k$.
3. A *dataset* is a collection of M sessions denoted by $D = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_M\}$.

To apply LDA to our session data, we assumes the following generative process for each session $\mathbf{w}$ in the dataset D:

1. Choose N, the number of items in the session.
2. Choose $\theta \backsim \mathrm{Dir}(\alpha)$, the topic mixture.
3. For each of the N items w
   (a) Choose a topic $z \backsim \mathrm{Multinomial}(\theta)$
   (b) Choose a item w from $p(w|z, \beta)$, a multinomial probability conditioned on the topic z.

We assume the dimensionality Φ of the topic variable z (and thus the dimensionality of the Dirichlet distribution) is known and fixed. The item probabilities are parameterised by a $\Phi \times V$ matrix β where $\beta_{ij} = p(w_j|z_i)$. The topic mixture, θ is a Φ-dimensional Dirichlet random variable. The variational message passing is used for learning the parameters. [16].

6.2.4 Constructing Datasets

So far we have obtained the topic model with Φ topics. Next we would like to construct a contextualised dataset for each topic. Similar to Section 5.2.2 we would like to construct a user-item utility matrix for each topic.

Given the topic model obtained in the previous step, we can compute the probability that session $\mathbf{w}$ is generated by topic z:

$$p(z|\mathbf{w}) = \int_\theta p(z|\theta)p(\theta|\mathbf{w})d\theta \tag{6.2}$$

If in session $\mathbf{w}$ the user interacted with item w once, this user-item interaction is viewed as Φ different micro-interactions: each under a different topic, with a different interaction count. For each of these micro-interactions, we assume the user interacted with item w_j with interaction count $p(z|\mathbf{w})$, where $w_j \in \mathbf{w}$, and z is the corresponding topic of this micro-interaction. Note that

$p(z|\mathbf{w}) > 0$ for each topic z, and $\sum_{z=1}^{\Phi} p(z|\mathbf{w}) = 1$. That is, the sum of all the micro-interactions' action counts of the same interaction is one.

Assume z is the ϕth topic. Let d_ϕ be the constructed dataset of topic z. For this dataset, summing over all the corresponding micro-interaction counts $p(z = \phi|\mathbf{w})$ of item w_j by user u_i, we obtain A_{ij}^{ϕ}, the user-item utility of item w_j for user u_i in ϕth topic :

$$A_{ij}^{\phi} = \sum_{\mathbf{w}} p(z = \phi|\mathbf{w}) \tag{6.3}$$

where session $\mathbf{w}$ is a context of user u_i, and contains the item w_j. We compute the user-item utility for all the user-item pairs in all Φ datasets. Therefore, based on item distributions, we obtain a contextualised dataset for each topic.

6.2.5 Predicting Preferences

So far, we have obtained Φ user-item utility matrices, one for each topic. We apply mapping approach to the constructed datasets, in order to convert the user-item utilities to ratings:

$$A_{ij} \rightarrow r_{ij}$$

A traditional recommendation method is applied on each dataset to predict users' ratings on items that they have not interacted with in each topic.

Assume we would like to predict r_{ij}^{x}, user u_i's rating on item w_j at time θ_x. We apply a traditional collaborative method, such as matrix factorisation, on all Φ datasets. Thus, we obtain Φ unique predicted ratings, r_{ij}^{0} to r_{ij}^{Φ}, one from each topic.

Assume user u_i has K contexts, thereby K sessions, $\mathbf{w}_0$ to $\mathbf{w}_K$. At time θ_x, we can calculate γ_{xk}, the responsibility of each personalised context, based on the MoAWG of user u_i by Equation 4.5. Therefore, we obtain γ_{x0} to γ_{xK}, the context mixture at time θ_x. Then based on the topic model, for each context c_k, we can compute the probability that the corresponding session $\mathbf{w}_k$ is generated by topic z, according to Equation 6.2. Therefore, we obtain the topic mixture, $p(z = 1|\mathbf{w}_k)$ to $p(z = \Phi|\mathbf{w}_k)$. The predicted rating is computed as the two-step weighted sum:

$$r_{ij} = \sum_{k=1}^{K} \gamma_{xk} \sum_{z=1}^{\Phi} r_{ij}^{\phi} p(z|\mathbf{w}_k) \tag{6.4}$$

6.3 Experimental Method

6.3.1 Data for Testing CBPCF

We use the same data as in Chapter 4 and 5.

Music Data: the music listening data from last.fm

Retail Data: Ta-Feng transaction dataset.

Detail of the data can be found in Section 4.3.1.

6.3.2 Testing Method of CBPCF

Test Sets

To evaluate our method on music data, we use the same testing set as in Chapter 4: on music data, we generate two testing sets to simulate two different contexts: one at 10am, one at 8pm; two testing set on retail data, Wednesday and Saturday. Detailed information can be found in Section 4.3.3.

Baseline Approaches

A matrix factorisation (MF) method [27] with 100 factors is used as baseline approach due to its ability to generate accurate predictions on sparse datasets.

This is a traditional recommendation method, which does not take into consideration any contextual information. Besides matrix factorisation, we also compare the method to probabilistic contextual filtering (PCF) from Chapter 4. In both PCF and clustering based probabilistic contextual filtering (CBPCF), MF is used. The MF we used is implemented by Apache Spark.

For both datasets, we will compare the precision when recommending only one item (precision at 1), and the precision when recommending the first half of items (precision at 10 and precision at 5 respectively).

Both PCF and CBPCF are non-deterministic. In order to gain the representative results, we repeat each experiment involving non-deterministic methods 100 times and take the average of the performance values. The results are analysed by the combined use of the Hodges-Lehmann Estimate with associated non-parametric confidence intervals at 95%.

Machine Used

In our experiments, we use a laptop with Intel 2.6 GHz dual core processor and 4GB of RAM.

6.4 Results and Analysis of CBPCF

In this section, we present the experimental results on clustering based probabilistic contextual filtering, and the analysis based on these results.

6.4.1 Comparison with traditional recommendation method

In this section, we compare our CBPCF with traditional recommendation method matrix factorisation.

Hypothesis

$H6.1_0$: There will be no difference in precision between the traditional method, matrix factorisation (MF) and the combination of clustering based proba-

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
MF	71.3%	84.6%	65.7%	72.9%
CBPCF + MF	86.2%	94.0%	74.0%	78.7%

Table 6.1 Precisions of MF and CBPCF + MF

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
Can the hypothesis be rejected at the 95% confidence interval	Yes	Yes	Yes	Yes
Hodges-Lehmann Estimate	14.9%	9.3%	8.3%	5.8%
Confidence Interval at 95%	(14.7%, 15.0%)	(9.2%, 9.3%)	(8.3%, 8.3%)	(5.8%, 5.8%)

Table 6.2 Summary of analysis between MF and CBPCF + MF

bilistic contextual filtering (CBPCF) and matrix factorisation (CBPCF + MF).

Results

Table 6.1 shows the precisions of the system using only MF and the system using CBPCF + MF on both datasets. Table 6.2 shows the results of the Mann–Whitney–Wilcoxon non-parametric test on the distributions of the results from both recommender systems.

The CBPCF substantially increases the precision at 1 on music data by 9.3%, compared to the traditional method, matrix factorisation. Similar results are found on retail data. The system using CBPCF + MF achieves considerably higher precision at 1 than only using MF on the retail data. On both datasets, when more items are recommended to the user (precision at 10 and 5) the combination of CBPCF and MF can find more relevant items than

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
MF	71.3%	84.6%	65.7%	72.9%
PCF + MF	85.0%	93.9%	72.5%	76.0%
CBPCF + MF	86.2%	94.0%	74.0%	78.7%

Table 6.3 Precisions of MF, PCF + MF and CBPCF + MF

MF alone. The statistical significance of the results is shown in Table 6.2. On both datasets, using both criteria, the null hypothesis can be rejected at the 95% confidence level, demonstrating that there is a statistically significant difference when using CBPCF + MF instead of MF (or vice versa).

6.4.2 Comparison with probabilistic contextual filtering

In this section, we compare our CBPCF with PCF.

Hypothesis

$H6.2_0$: There will be no difference in precision between the probabilistic contextual filtering (PCF) and the clustering based probabilistic contextual filtering (CBPCF), both combined with matrix factorisation (MF).

Results

The results form these experiments are presented in Table 6.3. The results of the system using only the traditional recommendation method MF are also presented in Table 6.3. The statistical analysis between the two contextual filtering methods is given in Table 6.4.

On muscis data, CBPCF can achieve higher precision compared to PCF when 10 items are recommended. On retail data, CBPCF can considerably increase the precision either at 1 or at 5. Except using precision at 1 on music data, in all other cases, CBPCF can provide consistent improvement.

	On Music Data		On Retail Data	
	Precision@10	Precision@1	Precision@5	Precision@1
Can the hypothesis be rejected at the 95% confidence interval	Yes	No	Yes	Yes
Hodges-Lehmann Estimate	1.2%	0.1%	1.5%	2.7%
Confidence Interval at 95%	(0.8%, 1.9%)	(0.0%, 0.2%)	(1.5%, 1.5%)	(2.7%, 2.7%)

Table 6.4 Summary of analysis between PCF + MF and CBPCF + MF

6.4.3 Analysis of CBPCF

In CBPCF, similar contexts are clustered into topics, based on item distributions. In our experiment, compared to PCF, CBPCF does not lead to lower accuracy. This demonstrates that the clustering of contexts based on item distribution is accurate in general. Reducing the number of contexts does not lead to inaccurate recommendations. That is, we can maintain the same level of accuracy, and substantially reduce the time and space required for training the RS. In music data, we have about 1,000 users. For each user, we have 3 personalised contexts. In PCF, this leads to 3,000 datasets and 3,000 recommender systems. In CBPCF, this number is reduced to 20, which is the number of unique topics. In reality, the RS may need to provide recommendations for more users every day. With CBPCF, the RS does not need to generate new datasets for each new user.

If a dataset is very sparse, by combining datasets, the RS can obtain more data (action counts) in each topic. Therefore, we can more reliably infer user's preference in the given context. The retail data is much more sparse than the music data, since on the music data we recommend artists rather than tracks. The results on retail data have demonstrated that compared to PCF, CBPCF can provide considerable improvement in terms of recommendation accuracy.

6.5 Summary and Future Work

In Chapter 1, based on the limitations of existing CARS, we presented our first research questions: whether we can develop a CARS that is able to overcome data sparsity. In Chapter 4, we presented PCF. Unlike most pre-filtering methods, PCF does not filter our any data that is collected in different datasets. Instead, it assigns weight to each user-item interaction. Experimental results shows that PCF can outperform traditional RS and contextual pre-filtering system on sparse datasets, in terms of recommendations accuracy and user space coverage. In this chapter, we presented the clustering based probabilistic contextual filtering. This method improves PCF by incorporating the features of PCC to PCF. In CBPCF, we first identify personalised contexts based on user's action patterns. Then based on item distribution, we cluster similar contexts into groups. The data collected in the same group of contexts is combined to reduce the computational cost, and to obtain a more reliable training set. Experimental results have demonstrated that CBPCF can further increase recommendation accuracy on sparse dataset. It can successfully overcome the data sparsity in the field of CARS.

Chapter 7

Conclusions

In this chapter, we first review the research questions we presented at the beginning of this thesis. The proposed solutions are then summarised, and the main contributions are highlighted . Finally, directions for future work are suggested.

7.1 Review of Research Questions

At the beginning of this thesis, we reviewed the area of context-aware recommender systems. We have identified some limitations of existing context-aware recommendation techniques. Based on these limitations, we presented our research questions:

1. Whether we can develop a context-aware recommender system that is able to overcome data sparsity.

In Chapter 4, the probabilistic contextual filtering is presented. PCF does not filter out training data like other contextual pre-filtering methods. Instead, the probabilities of an action performed in different contexts are calculated and used as weights. Therefore, we can also avoid the problems of context generalisation. Experimental results have demonstrated that on sparse dataset, PCF can generate more accurate recommendations, for more users, compared to contextual pre-filtering method. In Chapter 6, we incorporated the features of probabilistic contextual clustering into PCF. The clustering based PCF can cluster similar user contexts to build more reliable datasets. Therefore, we can further improve the recommendation accuracy on sparse data.

Therefore, it can be stated that the goal described in the first research question has been achieved in principle.

2. Whether we can develop a context-aware recommender system that is able to define personalised contexts based on users' action patterns, and to discover the underlying association between contexts, so that the system can achieve high accuracy.

In our PCF method presented in Chapter 4, personalised contexts are defined based on individual users' periodic action patterns. Therefore, we can focus on making recommendations for meaningful and personalised contexts. In Chapter 6, we presented CBPCF. In this method, we model the item distribution in each personalised context identified by PCF, so that the underlying association between contexts can be discovered, and similar contexts can be combined. Therefore, we can make better use of these personalised contexts to make more reliable recommendations. Experimental results have demonstrated that CBPCF can significantly increase recommendation accuracy, compared to traditional RS and contextual pre-filtering.

Therefore, we believe that the goals described in the second research question have been achieved in principle.

3. whether we can develop a session based CARS that can accurately model contexts and general user preferences at the same time.

In Chapter 5, we presented probabilistic contextual clustering. By modelling the item (tag) distributions, we are able to cluster contexts into topics, and to generate a dataset for each topic. This allows us to employ various traditional recommendation methods to independently, thereby accurately model users' general preferences on the contextualised datasets. Therefore, accurate context profiling and accurate user profiling can both be achieved.

Therefore, it can be stated that the goal described in the third research question has been achieved in principle.

7.2 Summary and Contribution

In this section, proceeding chapters are summarised and the main contributions of this thesis are identified.

Chapter 2 and 3 - RS and CARS

This two chapters review the fields of traditional recommender systems and context-aware recommender systems. We focused on context-aware recommender systems, identified the limitations of existing methods, which motivate our work in Chapter 4, 5 and 6.

Contribution - A review of CARS. The review highlighted the limitations of existing context-aware recommender systems, which motivate our work.

Chapter 4 - Probabilistic Contextual Filtering

In this chapter, the probabilistic contextual filtering is presented. In PCF, personalised temporal contexts are identified and modelled by a mixture of periodic Gaussians, based on the user's action patterns. The probabilistic contextual filtering does not filter out training data like other contextual filtering methods. Instead, the probabilities of an action performed in different contexts are calculated. The user's preference can then be computed as a weighted combination of his/her preferences in these identified contexts. The testing criteria and experimental methods used are also presented in this chapter. Experimental results from two datasets have demonstrated that the proposed method can achieve higher prediction accuracy and user space coverage on sparse data, compared to traditional recommendation method and contextual pre-filtering.

Contribution - The development of a new context-aware recommendation method, PCF, which can significantly increase prediction accuracy. Measured by precision, experimental results indicate that PCF can outperform contextual pre-filtering method by at least 8.6% on our testing sets. Measured by user space coverage, PCF can outperform contextual pre-filtering method by at least 9.1%.

Chapter 5 - Probabilistic Contextual Clustering

In this chapter, we presented the probabilistic contextual clustering. By modelling the item (tag) distributions, we are able to cluster the contexts into topics. For a given session, we can compute its topic mixture based on the items we have observed in the current context. The testing criteria and experimental methods used were also presented in this chapter. Experiment results have demonstrated that the proposed method can achieve higher prediction accuracy compared to existing session based methods. Also, the results have demonstrated that when item tags are available, the prediction accuracy can be further improved.

Contribution - The development of a new session based context-aware recommendation method that can achieve higher prediction accuracy than existing methods. Measured by precision, experimental results show that PCC can outperform existing session based methods in [73] and [31] by at least 6.9% on our testing sets.

Chapter 6 - Clustering Based Probabilistic Contextual Clustering

In this chapter, we presented the clustering based probabilistic contextual filtering. This method incorporates the features of PCC to PCF. In our CBPCF, we first identify personalised contexts based on user's action patterns. Then based on item distribution, we cluster similar contexts into groups. The data collected in the same group of contexts is combined to reduce the computational cost, and to obtain more reliable training sets. The predicted preference is computed as a two-step weighted sum. The testing criteria and experimental methods used were also presented in this chapter. The experimental results have demonstrated that CBPCF can reduce computational cost and increase recommendation accuracy on sparse data, compared to PCF.

Contribution - The development of a novel context-aware recommendation method, CBPCF, which can combine PCF and PCC to achieve higher prediction accuracy than existing methods. Compare to PCF, CBPCF can maintain the same level of accuracy and substantially reduce computational cost (from 3000 datasets to 20 datasets on music data).

Based upon these contributions, it is believed that the aims of this thesis have been met.

7.3 Limitation of PCF, PCC and CBPCF

Like all methods, our approaches also have some limitations. In this section, we summarise these limitations.

Our methods can only be applied to temporal context

Through out our thesis, we focused on temporal information. In Chapter 4, we identify personalised contexts by analysing users' temporal patterns. It is difficult to extend PCF to analyse other contextual information, such as location, company of other people. For session based PCC, without temporal information, we can not even identify valid sessions, thereby unable to apply our method. This is a common issue for context-aware recommender systems. For example, in context generalisation, different generalisation rules are needed for different contextual information, and usually these rule can only be generated manually, since domain knowledge is also required.

Our methods are designed for implicit transaction data

In our methods, we infer uses' preference and temporal pattern through transaction-like data. If we are given users' rating, we can not make use of it, even if these ratings are three dimensional (user, item, context). Although it may be possible to identify temporal patters through ratings, it requires far more data than traditional RS, and it's not intuitive how one can transfer rating patterns to consumption patters. Besides, with implicit data, the type of contextual information available can be very limited. Ideally, RS should be able to make use of explicit data when it's available.

Our methods are computationally expensive

All of our methods are computationally heavy. In PCF and CBPCF, we need model each user's action patters. That is to apply a MoAWG model to each individual user. In PCC and CBPCF, we also apply LDA to each user. LDA and MoAWG are iterative methods. Training of these methods can take very long time.

Out methods can not handle data stream

In real world, RS must be updated regularly to keep up with users' changing preferences. That is, if the system can keep collecting new data, RS must be able to make use of the new data, update it's parameters rapidly, ideally without retraining the entire model. Unfortunately, our CARS can not handle a stream of new data. When new data is available, we can only rerun the whole process to obtain updated parameters of the model.

7.4 Future Work

In this section, some potential areas of future work is provided.

7.4.1 Improvements and Extensions to PCF, PCC and CBPCF

- In PCF and CBPCF, the user's action pattern is modelled by Gaussian mixture model. However, the distribution of action count may not always be Gaussian. For example, in some cases, maybe a Poisson distribution is more appropriate.

- In PCF and CBPCF, we need to select the number of Gaussians, thus the number of personalised contexts. In our methods, this number is fixed and the same for all the users. To excel in context personalisation, the number of components should be based on individual user's action patterns.
- In CBPCF, the inferred action count is rounded up to the nearest integer. For user-item pair that has high interaction count, this approximation would not affect the modelling of item distributions. However, for those user-item pairs that have small action counts, this could cause inaccurate predictions. The impact of this approximation should be studied. A topic model that is capable of dealing with non-integer item counts may be employed.
- In CBPCF, we can also study the clustering of contexts based on tag distributions.
- In CBPCF, although we model item distributions, we can not make use of the observed seed items. It might be reasonable to develop a system that can start with a context-aware recommendation method based on available contextual information. Then, after a sufficient number of seed items have been observed, the system switches to session based method.
- In all our methods, we use implicit data. As we discussed in Chapter 2, it is difficult to evaluate which items the user does not like. Therefore, there is no way to explore the performance of our method in terms of false positive/negative. Ideally, we would like to test our method in a application where both implicit data and explicit data are available.
- In all our methods, we only deal with temporal information. However, contextual information can contain much more than just time. We should study how to extend our methods to location-aware or companion-aware

RS.

- All of our methods are computationally expensive. The only way to update our CARS is to retrain the entire model. We should study how to incorporate new data without retraining the entire RS.

7.4.2 Context-Aware Recommender Systems

- It has been demonstrated in Netflix Competition that an ensemble of traditional recommender systems can outperform any existing traditional recommender systems. Therefore, it might be reasonable to consider the same idea in context-aware recommender systems. Multiple CARS can employ different recommendation techniques, and based on different contextual information.
- In many applications, it is difficult to collect users' feedbacks on items, especially in context-aware recommender systems, where the system may ask the user to rate the same item in different contexts. Therefore, it is important to study and to develop CARS based on implicit data.
- Social networks have become a powerful media. Many people choose to express their ideas and opinions via social networks. It is important to study and make use of the powerful yet noisy and unstructured data from social networks.
- More diverse testing criteria may be necessary for evaluating context-aware recommender systems. For example, in some cases, it is more important to determine when the recommendations are needed by the user, than to predict what items are needed. This is especially important for time-aware recommender systems.

7.5 Discussion

In 2007, Netflix held its first million-dollar recommendation competition. It drew talents from all over the world. Researchers from both academia and industry took part in. The competition significantly boosted the advancement of RS. However, Netflix was sued in 2008, because based on the anonymous data it released for the competition, people can still track who the real users are. Ever since that, no major company has ever released any dataset to the public. How can we improve the collaboration between academia and big tech companies like Google and Amazon, is also something worth studying.

Nomenclature

Symbols

A_{ij}	user-item utility for user u_i with item w_j
A_{ij}^k	user-item utility for user u_i with item w_j in kth context
c_k	the kth context
D	a dataset containing a collection of sessions
f_{ij}	the number of times s_i appears in w_j or profile of w_j
$\Gamma(x)$	Gamma function
γ_{xk}	the responsibility of kth component for an action x
IDF_i	the inverse document frequency of s_i
K	the number of components in a mixture model
k	kth component of a mixture model
$\boldsymbol{\mu}$	vector of μ_k
μ_k	the weight of the kth component in a mixture model
$\mathcal{N}^{AW}$	approximated wrapped Gaussian distribution
n_i	the number of items or item profiles containing s_i in TF-IDF

$\mathcal{N}^W$	wrapped Gaussian distribution
p_{ij}^k	user-item interaction count in context c_k
p_{ij}^t	the action count of u_i on w_j in time period t
p_i^k	the total action count of u_i on all items in time period t
$p_i(t)$	the distribution of action count of u_i over time t
R	rating function that maps the user-item pairs to ratings
r_{ij}	the rating given by user u_i on item w_j
$\hat{r}_{ij}$	the predicted value of r_{ij}
r_{ij}^k	the rating given by user u_i on item w_j in context c_k
S	the number of unique keywords in the dataset
s_i	ith keyword/tag in an item or item profile
σ	standard deviation of a distribution
α	hyper parameter of LDA
β	item probability matrix of LDA
TF_{ij}	the term frequency of s_i in w_j or profile of w_j in TF-IDF
θ_0	mean of a distribution
U	user/customer set
u_i	ith user in the user set
$\mathbf{w}$	a session containing a sequence of items
W	item set

w_j	jth item in the item set

Abbreviations

AWG	approximated wrapped Gaussian distribution
CARS	Context-Aware Recommender Systems
CBPCF	Clustering Based Probabilistic Contextual Filtering
EM	Expectation-Maximisation
LDA	Latent Dirichlet Allcation
MDP	Markov Decision Process
MF	Matrix Factorisation
MoAWG	mixture of approximated wrapped Gaussian distribution
MoWG	mixture of wrapped Gaussian distribution
PCC	Probabilistic Contextual Clustering
PCF	Probabilistic Contextual Filtering
pdf	probability density function
QD	Query-Driven Context-Aware Recommendation
RBM	Restricted Boltzmann Machine
RS	Recommender Systems
SBM	Music-Listening Session Based Model
TF-IDF	Term Frequency/Inverse Document Frequency
WG	wrapped Gaussian distribution

References

[1] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a better understanding of context and context-awareness. In *Handheld and ubiquitous computing*, pages 304–307. Springer.

[2] Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145.

[3] Adomavicius, G. and Tuzhilin, A. (2001). Multidimensional recommender systems: a data warehousing approach. In *Electronic commerce*, pages 180–192. Springer.

[4] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749.

[5] Adomavicius, G. and Tuzhilin, A. (2011). Context-aware recommender systems. *Recommender Systems Handbook*, pages 217–253.

[6] Agiomyrgiannakis, Y. and Stylianou, Y. (2009). Wrapped gaussian mixture models for modeling and high-rate quantization of phase data of speech. *Audio, Speech, and Language Processing, IEEE Transactions on*, 17(4):775–786.

[7] Aizenberg, N., Koren, Y., and Somekh, O. (2012). Build your own music recommender by modeling internet radio streams. In *Proceedings of the 21st international conference on World Wide Web*, pages 1–10. ACM.

[8] Ansari, A., Essegaier, S., and Kohli, R. (2000). Internet recommendation systems. *Journal of Marketing research*, 37(3):363–375.

[9] Bahlmann, C. (2006). Directional features in online handwriting recognition. *Pattern Recognition*, 39(1):115–125.

[10] Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72.

[11] Baltrunas, L. and Amatriain, X. (2009). Towards time-dependant recommendation based on implicit feedback. In *Workshop on context-aware recommender systems (CARS'09)*.

[12] Basu, C., Hirsh, H., Cohen, W., et al. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *AAAI/IAAI*, pages 714–720.

[13] Billsus, D. and Pazzani, M. J. (2000). User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180.

[14] Bishop, C. M. (2006). *Pattern recognition and machine learning*. springer.

[15] Blei, D. and Lafferty, J. (2006). Correlated topic models. *Advances in neural information processing systems*, 18:147.

[16] Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022.

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

[18] Burke, R. (2007). Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer.

[19] Calderara, S., Prati, A., and Cucchiara, R. (2011). Mixtures of von mises distributions for people trajectory shape analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(4):457–471.

[20] Cantador, I., Bellogín, A., and Castells, P. (2008). A multilayer ontology-based hybrid recommendation model. *Ai Communications*, 21(2-3):203–210.

[21] Celma, O. (2010). *Music recommendation and discovery*. Springer.

[22] Celma Herrada, Ò. (2009). Music recommendation and discovery in the long tail.

[23] Chen, G., Kotz, D., et al. (2000). A survey of context-aware mobile computing research. Technical report, Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College.

[24] Chen, S., Moore, J. L., Turnbull, D., and Joachims, T. (2012). Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 714–722. ACM.

[25] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., and Sartin, M. (1999). Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR workshop on recommender systems*, volume 60. Citeseer.

[26] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

[27] Funk, S. (2006). Netflix update: Try this at home. http://sifter.org/~simon/journal/20061211.html. Accessed: 12-10-2012.

[28] Gunawardana, A. and Meek, C. (2009). A unified approach to building hybrid recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 117–124. ACM.

[29] Gupta, D., Digiovanni, M., Narita, H., and Goldberg, K. (1999). Jester 2.0 (poster abstract): evaluation of an new linear time collaborative filtering algorithm. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 291–292. ACM.

[30] Hariri, N., Mobasher, B., and Burke, R. (2012). Context-aware music recommendation based on latent topic sequential patterns. In *Proceedings of the sixth ACM conference on Recommender systems*, pages 131–138. ACM.

[31] Hariri, N., Mobasher, B., and Burke, R. (2013). Query-driven context aware recommendation. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 9–16. ACM.

[32] Hsu, C.-N., Chung, H.-H., and Huang, H.-S. (2004). Mining skewed and sparse transaction data for personalized shopping recommendation. *Machine Learning*, 57(1-2):35–59.

[33] Hu, Y., Koren, Y., and Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM '08. Eighth IEEE International Conference on*, pages 263–272.

[34] Jahrer, M., Töscher, A., and Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 693–702. ACM.

[35] Jannach, D., Zanker, M., Felfernig, A., and Friedrich, G. (2010). *Recommender systems: an introduction*. Cambridge University Press.

[36] Jiang, T. and Tuzhilin, A. (2009). Improving personalization solutions through optimal segmentation of customer bases. *Knowledge and Data Engineering, IEEE Transactions on*, 21(3):305–320.

[37] Jin, R., Si, L., and Zhai, C. (2006). A study of mixture models for collaborative filtering. *Information Retrieval*, 9(3):357–382.

[38] Kantor, P. B., Rokach, L., Ricci, F., and Shapira, B. (2011). *Recommender systems handbook*. Springer.

[39] Koren, Y. and Bell, R. (2011). Advances in collaborative filtering. In *Recommender Systems Handbook*, pages 145–186. Springer.

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

[41] Lee, W. S. (2001). Collaborative learning for recommender systems. In *ICML*, pages 314–321.

[42] Lehmann, E. L. and D'Abrera, H. J. (2006). *Nonparametrics: statistical methods based on ranks*. Springer New York.

[43] Lops, P., De Gemmis, M., and Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer.

[44] Mahmood, T. and Ricci, F. (2009). Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM.

[45] Mardia, K. V. (1975). Statistics of directional data. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 349–393.

[46] Mardia, K. V. and Jupp, P. E. (2009). *Directional statistics*, volume 494. John Wiley & Sons.

[47] Melville, P., Mooney, R. J., and Nagarajan, R. (2002). Content-boosted collaborative filtering for improved recommendations. In *AAAI/IAAI*, pages 187–192.

[48] Mooney, R. J. and Roy, L. (2000). Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM.

[49] Oard, D. W., Kim, J., et al. (1998). Implicit feedback for recommender systems. In *Proceedings of the AAAI workshop on recommender systems*, pages 81–83. Wollongong.

[50] Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., and Pedone, A. (2009). Experimental comparison of pre-vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 265–268. ACM.

[51] Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331.

[52] Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408.

[53] Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer.

[54] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM.

[55] Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58.

[56] Ricci, F., Rokach, L., and Shapira, B. (2011). Introduction to recommender systems handbook. In *Recommender Systems Handbook*, pages 1–35. Springer.

[57] Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM.

[58] Salton, G. (1989). *Automatic Text Processing: The Transformation, Analysis, and Retrieval of.* Addison-Wesley.

[59] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document.

[60] Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM.

[61] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

[62] Schilit, B., Adams, N., and Want, R. (1994). Context-aware computing applications. In *Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on*, pages 85–90. IEEE.

[63] Sermanet, P., Kavukcuoglu, K., and LeCun, Y. (2009). Eblearn: Open-source energy-based learning in c++. In *Tools with Artificial Intelligence, 2009. ICTAI'09. 21st International Conference on*, pages 693–697. IEEE.

[64] Shani, G., Brafman, R. I., and Heckerman, D. (2002). An mdp-based recommender system. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 453–460. Morgan Kaufmann Publishers Inc.

[65] Shani, G. and Gunawardana, A. (2011). Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer.

[66] Shardanand, U. and Maes, P. (1995). Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co.

[67] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory.

[68] Soboroff, I. and Nicholas, C. (1999). Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, volume 99, pages 86–91.

[69] Spotify (2015). Information - press - spotify. https://press.spotify.com/nz/information/. Accessed: 20-05-2015.

[70] Tran, T. and Cohen, R. (2000). Hybrid recommender systems for electronic commerce. In *Proc. Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical Report WS-00-04, AAAI Press.*

[71] Verbert, K., Manouselis, N., Ochoa, X., Wolpers, M., Drachsler, H., Bosnic, I., and Duval, E. (2012). Context-aware recommender systems for learning: a survey and future challenges. *Learning Technologies, IEEE Transactions on*, 5(4):318–335.

[72] Williams, C. (2011). Apple iphone tracks users' location in hidden file. http://www.telegraph.co.uk/technology/apple/8464122/Apple-iPhone-tracks-users-location-in-hidden-file.html. Accessed: 20-04-2011.

[73] Zheleva, E., Guiver, J., Mendes Rodrigues, E., and Milić-Frayling, N. (2010). Statistical models of music-listening sessions in social media. In *Proceedings of the 19th international conference on World wide web*, pages 1019–1028. ACM.